基于遗传算法的时态流程挖掘研究与应用

专　　业：计算机软件与理论

硕 士 生：冯树彬

指导教师：李磊 教授

摘　要

目前，大多数的企业管理应用软件都会随着系统运行自动产生大量的日志，这些日志记录了系统的实际业务执行情况等信息。如何合理地对这些日志进行分析，提取有用的知识，成为近年来不少学者研究的热点。

本文首先介绍工作流挖掘的意义及其技术研究的现状，并对流程挖掘技术面临的挑战进行综述，指出当前大部分的研究忽略了日志中的时间因素，并且没有考虑日志的流程增量问题，降低了对业务日志进行流程挖掘的准确性。基于当前流程挖掘研究的不足，本文提出了日志时态分析模型，利用该模型对日志进行预处理，首先对日志中的流程实例和各个任务进行时间分析，利用“时间区间划分法”建立任务间的时态关系，然后提出“时间知识权值法”对日志进行处理，删除一些无效的任务，有效地解决流程增量问题，提高流程挖掘的准确性和挖掘结果的参考价值。

在此基础上，我们提出改进遗传算法的时态流程挖掘框架，该算法在初始种群时引入启发式规则，缩小搜索空间。一个种群包含若干个遗传个体，每个遗传个体对应一个流程模型，遗传个体的适应度函数衡量遗传个体与任务日志的拟合程度。算法的适应度函数加入了微调因子提高流程挖掘的准确性，并且在变异算子中加入启发式规则，加快算法运算速度。在得到最优化个体后，使用合并技术构建时态流程模型。

最后，我们基于上述的讨论，在 Java 平台上实现了基于改进遗传算法的时态流程挖掘框架，通过实验进行检验分析，并与其他算法进行对比，证明算法的有效性，能克服 α 算法和 α++算法在某些结构的不足和限制，并有效地解决一些流程增量问题，挖掘得到合理的时态流程模型。

关键词：流程管理，流程挖掘，时间知识权值法，时态分析，遗传算法，启发式

Research and Application of Temporal Process Mining Based on Genetic Algorithm

Major : Computer Software and Theory
Name : Shu Bin Feng
Supervisor: Prof. Lei Li

Abstract

At present, most enterprise management software can store the execution information of system into log files. How to analyze these log files and abstract the useful knowledge was getting hotter in these few years.

This paper introduces the business process management and status of workflow mining. We point out most of studies ignore the time factor in the log and do not consider the incremental process issues which can reduce the accuracy of process mining. Based on the current lack of process mining, a temporal analysis model based log is proposed. First, this model does some pre-processing on the log, makes the temporal analysis of process instances and various tasks, establish a temporal relationship between the various tasks by using the "Time Interval Division Method". And then a "Time Knowledge Weighted Method" is proposed which can delete some useless tasks and effectively solve the incremental process issues, improve the accuracy of the process mining and excavation results of reference.

On this basis, we propose a framework for process mining based on genetic algorithm. During the initial population stage, the algorithm introduces heuristic rules, which can reduce the search space. A population contains a number of genetic individuals; each genetic individual corresponds to a process model. The fitness function of each genetic individual measures the fitting degree between the genetic individual and the log. This fitness function in algorithm is designed by adding a fine-tuning factor to improve the accuracy of process mining. Meanwhile, the mutation operator with heuristic rules can accelerate the speed of the algorithm.After obtaining the optimal individual; we use the merge technology to build the temporal process model.

Finally, based on the above discussion, we use java platform to implement the framework of temporal process mining based on genetic algorithm. Tested by

experimental analysis and comparison with other algorithms have proved the effectiveness of the algorithm. The algorithm overcomes the deficiencies and limitations of α and α++ algorithm in some structural mining, effectively solved the incremental process issues and rebuild the reasonable temporal process model.

Keywords: Process Management, Process Mining, Time Knowledge Weighted Method, Temporal Analysis, Genetic Algorithm, Heuristic

论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：_____

日　　期：_____

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文。

学位论文作者签名：　　导师签名：

日期：　年　月　日　　日期：　年　月　日

第 1 章 绪论

1.1 研究背景及研究意义

作为实现企业信息化的重要底层支撑技术，流程建模是工作流管理应用于企业资源规划、电子政务、电子商务等信息管理领域中必须解决和无法回避的一个关键问题。近二十年来，也越来越多基于业务系统流程建模技术的出现，包括传统工作流建模技术、基于多 Agent 柔性工作流建模技术等。而目前也越来越多的现代企业使用基于这些建模技术的工作流管理系统(Workflow Management System,WfMS),如 IBM MQSeries、FlowMark、JBPM 等工作流管理系统为结构化业务提供通用的建模和流程定制技术，为业务流程进行图形化展示。还有一些采用基于多 Agent 柔性工作流建模技术、基于 Web Service 动态组合建模技术，如 Dy_{flow}[1]，不需要预先对业务流程进行定制，而是基于规则推导和智能规划技术进行流程路由。

这些建模技术经历了一定时间的发展，但对于真正地应用工作流管理技术仍存在一些问题。一个问题是这些工作流管理系统都需要工作流设计，即流程定制器需要建立详细描述业务流程的精确模型。然而，很多实际应用领域中，为业务流程建立精确的模型是需要具有很高的业务知识水平和深厚的工作流知识，这种业务流程定制工作是领域专家和系统设计人员参与的，这是一个相当不简单的过程，而且在流程定制过程中容易受到流程设计人员的主观因素影响，这样就极大地影响了工作流技术应用的质量。

尽管采用基于多 Agent 工作流建模技术（或 Web Service 动态组合建模技术）的工作流平台不需要预先定义业务流程，但这些平台无法获得真实执行的业务流程模型，无法验证实际的任务执行过程与实际的业务流程需求是否一致，从而降低流程跟踪和优化的效率。

为了实现流程建模的自动化，以及克服建模过程中可能存在的主观因素，美国新墨西哥州立大学的 Joanthan E.Cook 教授于 1995 年提出了过程挖掘的基本思想[2]。过程挖掘旨在对日志数据进行分析，从中抽取执行轨迹信息，这些轨迹信息是不受主观和惯性思维因素影响的，它真实地记录了实际业务系统流程的运

行情况，比起以业务知识人员进行业务流程定制更具有客观性和说服力。通常，领域专家对业务流程模型进行精确定义，然后部署到工作流管理系统上运行，但在实际应用中，系统流程往往是具有易变性的，是随着企业或组织的业务需求而不断变更，这样就容易造成流程丢失。而系统日志中记录了大量的业务流程运行轨迹，通过流程挖掘技术可以较为客观和科学地重构业务流程模型。

流程挖掘还有一个重要的应用意义，那就是为流程监控和流程改进提供支持，一些基于多 Agent 柔性工作流系统，其任务路由技术是基于规则推导和智能规划等技术，而且这些技术提高了平台的柔性的同时，也为系统的业务流程的监控带来不足，流程挖掘通过分析系统的日志，为系统的真实运行情况建立清晰的流程模型，提高了流程跟踪和优化的效率。

1.2 研究内容

流程挖掘的研究工作涉及很多方面，其中控制流挖掘是流程挖掘中重要的领域之一，也是最为活跃的领域[3]。工作流的基本结构有顺序、选择、并行等基本结构，很多工作流模型往往是由这些基本结构进行组合或者复合组成。控制流挖掘可以解决这些基本结构的挖掘及其复合结构的挖掘。此外，重复任务、隐含任务、非自由选择结构等复杂结构的挖掘也是控制流挖掘的主要内容。在实际的工作流模型中，往往存在着由基本结构复合组成的复杂结构，给控制流挖掘工作带来一定的难度。

当前的流程挖掘技术主要是以 α 算法[4]为主，以及基于 α 算法改进的 α+[4]和 α++算法[5]，这些算法主要是针对无噪声日志的，在日志完备的情况下，可以挖掘出合理的 SWF-net，在控制流挖掘方面也取得不错的效果，但是对于重复任务、隐含任务、循环等复杂结构的挖掘的效果不是很理想，而且这些算法处理带噪声日志的能力很弱，所以算法的鲁棒性较差。

本文的研究内容是以基于多 Agent 柔性业务平台日志分析作为背景，这种工作流平台在建模阶段不需要预定义精确的流程模型，存在无法验证系统的实际执行过程是否与企业的业务需求一致，也为流程跟踪和流程改进增加了难度，因此基于平台上构建流程挖掘框架具有重要的意义。本文首先提出流程日志的时态分析模型，在日志挖掘过程中加入时间因素，提出“时间区间划分法”分析得到任

务间的时态关系。此外该模型考虑日志的流程增量问题，创新地提出“时间知识权值法”对日志信息进行处理，有效地解决一些流程增益的问题。在此基础上，提出基于改进遗传算法的流程挖掘。该挖掘框架不仅能有效地解决流程增量问题，而且挖掘出合理的重复任务、循环结构、非自由选择结构等结构的工作流模型，为流程建模人员提供知识参考，提高流程建模、流程跟踪和流程优化的效率。

1.3 论文的组织结构

本文章节的组织结构如下：

第 1 章绪论，首先介绍本文的研究背景，说明流程挖掘在流程建模以及流程优化中的重要性，并指出在挖掘过程中考虑时间因素和流程增量问题在实际应用中的意义，然后引出本文的研究内容——基于改进遗传算法时态流程挖掘框架。

第 2 章相关领域研究。首先详细分析了当今流程挖掘的各种研究方法，并对控制流挖掘技术的发展历程进行了综述。指出当前大部分的流程挖掘研究忽略了时间因素以及流程增量问题，当前以 α 算法为基础的挖掘算法鲁棒性差，并且不能挖掘某些复杂的结构。

第 3 章详细介绍了业务流程挖掘相关知识和技术，包括工作流日志、日志时间、结构化工作流、流程基本结构特征等方面知识的形式化表示。

第 4 章提出了流程日志时态分析模型。首先介绍日志中存在的时态关系。然后讨论日志时态分析模型，提示“时间知识权值法”有效地解决流程增量的问题，通过对任务间的时态关系分析，提出“时间区间划分法”建立任务间时间间隔的时态关系，为时态流程挖掘提供知识支持。

第 5 章在第 3、4 章的讨论基础上，提出基于改进遗传算法的流程挖掘思想。首先对遗传算法的技术特点和应用进行概述，然后对工作流模型中的任务因果矩阵进行讨论，并提出一些启发式规则，在此基础上，构建算法的整体流程，并在算法的种群生成和遗传算子中加入启发式规则，缩小算法的搜索空间。最后通过实验分析，验证挖掘框架的有效性和优势。

第 6 章是总结和展望，对本文进行总结，提出展望和进一步的研究内容。

第2章 相关领域研究

2.1 业务流程挖掘发展历程

1995年,美国新墨西哥州立大学计算机教授Joanthan E.Cook提出了基于软件工程领域的“工作流挖掘”思想，并在其文献[2]中设计了三种工作流挖掘方法：RNet方法、KTail方法和Markovian方法。后两种方法性能和实用性强于前者。然而这3种方法都是基于有限状态自动机(Finite-state Machine,FSM)描述的模型，由于有限自动机描述能力较弱，所以这3种方法的实用性不强。

1998年，美国IBM Almaden研究中心的Rakesh Agrawal首次将工作流挖掘应用到企业工作流建模领域[6]，并提出了两种工作流挖掘算法。与 Joanthan E.Cook不同，Rakesh Agrawal用“有向图”作为工作流模型的描述表示，而且其算法除了能挖掘工作流顺序结构外，还能挖掘工作流模型的循环关系以及任务间的因果关系，却很难挖掘具有同步和选择关系的工作流模型。

近十年来，工作流挖掘越来越得到国内外学者的关注，国外有不少学者（如W.M.P van den Aalst、Joachinem Gerbst等）提出了自己的工作流挖掘算法，并取得不错的效果。荷兰Eindhoven大学教授W.M.P van den Aalst是目前工作流建模领域的最为著名的专家之一，近年来他致力于工作流技术与算法的研究，并提出了几种工作流挖掘算法，如α算法等，并基于这些算法开发出相关的工作流挖掘工具[7]。

随着工作流挖掘技术的发展，工作流挖掘已在多个领域中取得应用：

1）流程监控

传统的工作流管理技术的关注点是业务流程的自动化，而缺乏对业务流程的执行过程进行分析与研究，通过对流程的执行历史数据的分析，利用工作流挖掘技术反映真实执行的流程模型，从而可以帮助企业分析流程的失效和缺陷信息，提高业务流程的质量。

当前一些大的企业利用工作流挖掘技术，构建具有流程监控功能的工作流挖掘平台，如HP公司的流程挖掘引擎(process mining engine)、ARIS PPM(process performance manager)、SPM(staffware process monitor)。

工作流挖掘除了在实际企业中取得应用之外，近十年学术界也基于研

究的算法开发了一些工作流挖掘框架，典型代表包括 Emit[8]、Little Thumb[14]、InWoLve[21]、Process Miner 系统[9]等。

2）流程优化

流程优化是过程管理中的相当重要的功能。工作流挖掘能够对业务过程的真实执行过程分析，重构出业务流程模型，为流程优化工作提供知识参考。FlowMark 是 IBM 公司一个工作流平台，该平台引入了工作流挖掘技术，从日志中挖掘实际执行的模型，然后加以改进优化后，应用到实际的业务系统中。

3）社会关系分析

工作流挖掘在社会关系分析方面取得一定的应用，从工作流日志中构造社会关系网络图，并对工作流参与者之间的关系进行度量，分析他们之间的活动关系和影响程度[10]。具有代表性的挖掘工具有 MiSoN，该工具不仅能挖掘出一个角色在流程中的作用，而且还能挖掘组织结构等信息。

4）其他领域

随着工作流挖掘技术的发展，工作流挖掘在其他一些领域取得应用，如 Web Service 等领域。在文献[11]提出了一种基于工作流挖掘的 Web Service 验证技术，通过对流程日志的分析，验证 Web Services 的行为模式是否合理。

2.2 流程挖掘的研究现状

随着工作流挖掘技术在流程管理、社会网络分析等领域取得不少的应用，近年来工作流挖掘成为研究热点，国外不少学者致力于研究工作流挖掘算法及其应用，并提出相关的算法思想，国内也有少数的学者着眼于这方面的研究。

2.2.1 研究现状

近十五年，工作流挖掘技术取得突破性的发展，越来越多的学者提出新的研究方法。1995 年美国墨西哥州立大学计算机教授 Joanthan E.Cook 提出了过程挖掘的基本思想[2]。工作流挖掘是指在日志信息中分析执行轨迹，挖掘出清晰的流程模型，为业务流程建模提供参考，这样能在一定程度上克服建模阶段人员的主观性。

Joanthan E.Cook 和 Alexander L. Wolf 把过程挖掘应用在软件工程领域中[13]，提出了三种流程发现的方法：神经网络（基于统计技术的）、纯数学方法和混合型的 Markovian 方法。通过一系列的实验证明纯数学方法和 Markovian 方法比神经网络更适合与过程挖掘领域。然而 Joanthan E.Cook 和 Alexander L. Wolf 提出的方法不能生成完整的流程模型，只是针对实际行为和流程模型两方面，提供了一种测量方法进行测量[18]。

随后的很多工作流挖掘研究也是主要集中在控制流的挖掘。德国乌尔姆大学的 Herbst 等人提出的方法具有处理重复任务（所谓重复任务是指在同一流程模型中出现两个任务的名称相同）的能力，并且提出了 3 种算法：MergeSeq、SplitSeq 和 SplitPar。其中，MergeSeq 算法和 SplitSeq 算法能处理顺序过程模型挖掘问题，而 SplitPar 算法则能挖掘并发结构[19][20]。随着工作流模型理论的发展，Petri-net 在工作流领域中得到广泛的推广，这得益于其严格的数学定义，越来越多的工作流技术研究采用 Petri-net 来描述流程模型，J.Herbst 等人基于 Petri-net 理论提出了处理重复任务的归纳挖掘算法，并基于此算法开发了流程挖掘工具 InWoLvE[21]。G.Schimm 等人基于块状结构来考虑流程挖掘问题，并把其算法引入到工作流挖掘工具 Process Miner[9]中。

荷兰埃因霍温理工大学的 Van der Aalst 提出了基于结构化工作流网的 α 算法，α 算法结构简单及容易理解，主要是根据业务流程系统日志中记录的任务间发生的顺序关系来进行挖掘工作，该算法在完备日志中取得较好的挖掘效果，能挖掘出合理的结构化工作流网。此外，很多在 α 算法基础上改进的挖掘算法在控制流挖掘方面也取得不错的效果，这些算法主要是在挖掘过程中加入一些启发式规则，使得算法能挖掘出某些工作流结构，如文献[5]提出了基于 α 算法上改进的 α++算法，在完备日志的前提下，能挖掘出某些非自由选择结构。

随着控制流挖掘得到越来越多的学者关注，近年来出现了一些基于智能算法的工作流挖掘思想，如宋炜等人提出了基于模拟退火法的流程挖掘算法[3]，该算法结合禁忌算法和蚁群算法，在重复任务、隐含任务以及非自由选择等问题上取得不错的效果。De Medeiros 等人在过程挖掘中引入遗传算法思想[22]，通过一系列的选择、交叉、变异等遗传算子运算，最终挖掘出合理的工作流模型。

2.2.2 存在问题与挑战

目前很多流程挖掘研究是基于 Petri-net 理论的，而该领域的研究仍有一些问题不能很好地解决。如 Van der Aalst 提出的基于 WF-net 的 α 算法，虽然被证明在完备日志中挖掘合理的 SWF-net，但其对于工作流模型中一些复杂结构表现出较差的挖掘效果，例如单循环、二重循环、隐藏任务等，而且算法的处理噪声能力弱及鲁棒性差，而在实际的应用中，往往系统日志是不完备或带噪声的。文献[5]基于 α 算法上改进的 α++算法制定了挖掘非自由选择结构的启发式规则，能在无噪日志中挖掘出大多数非自由选择结构，但是该算法在带噪声日志的挖掘效果不能令人满意。文献[3]提出了一种模拟退火过程挖掘算法，该算法在一定程度上挖掘重复任务、隐藏任务等复杂结构，但是对于日志中同时存在重复问题和非自由选择复杂结构，算法的挖掘能力不稳定，特别是对于包含较复杂的非自由选择结构的过程模型，挖掘效果不是很理想。

当前的控制流挖掘研究仍有一些问题不能很好的解决，主要包括重复任务、非自由选择结构、循环（包括单循环、二次循环等）、隐藏任务等，而且这些结构在实际应用中的业务流程模型中是常见的，这就增加了流程挖掘的难度。虽然文献[23]通过对实际应用中可能存在的非自由选择结构进行分析，总结出一系列的启发式规则，能挖掘出大多数的自由选择结构，然而其挖掘部分复杂的非自由选择结构和处理噪声能力仍存在较大的不足。

还有一个重要问题，当前绝大部分的流程挖掘研究都没有考虑日志中时间信息，这对于一些时间敏感度高的应用领域是不合理的。文献[24]基于流程挖掘技术上进行了时间分析，为挖掘得到的任务附上时间标签信息，然而却没有深入地讨论任务之间的时态关系。时间信息的影响主要体现在两方面：首先是日志中任务时间信息，能用于区分流程轨迹中不同的时态任务，使得挖掘得到的流程模型带有时态信息，为流程改进、优化、预测等提供知识支持。其次是流程增量问题，企业随着市场需求的不断变化，其业务流程也不断地变更，这时候系统日志信息的时间跨度可能很大，从系统日志中挖掘出的过程模型不一定能反映近期企业的业务流程模型，这一点对于柔性业务系统（特别是无需预先定义精确的业务流程模型的业务系统）的流程挖掘及监控尤为重要。

第3章 流程挖掘相关技术的数学定义

本章对工作流技术以及工作流挖掘相关概念进行介绍，并对工作流挖掘的各个相关技术进行数学定义描述。

3.1 工作流技术

3.1.1 工作流的相关概念

工作流的概念起源于办公自动化和制造业，它是针对日常工作中具有相对固定程序的活动而提出的一个概念。为了实现组织目标，有关业务活动依时序或逻辑关系相互连接构成业务流程。

工作流管理联盟(WFMC)给出了工作流的定义：工作流(WorkFlow)就是自动运作的业务过程的部分或整体，表现为参与者对文件、信息或任务按照规程采取行动，并令其在参与者之间传递[25]。总的来说，工作流是对工作流程及其各操作步骤之间业务规则的抽象、概括、描述。

WFMC 也给出了工作流管理系统的定义[25]：工作流管理系统通过软件定义、创建工作流，并管理其执行。它运行在一个或多个工作流引擎上，这些引擎通过对过程的定义，与工作流的参与者相互作用。

我们在工作流管理系统的协助下，开发人员遵从一定的编程接口及约定，就可以开发出更具灵活性的事务处理系统，这样使得最终用户无需重新开发事务处理系统，就可以更改工作流程，以达到提高业务系统柔性的效果。工作流管理系统主要的功能分为以下几类[15]：

1）工作流建模

工作流建模是根据企业或组织具体的业务需求，以图形的描述方式定义业务过程，其中包括具体的活动、规则等。

2）工作流运行

工作流运行是指业务流程在工作流管理系统中实施，并以系统功能的形式为企业或组织提供业务功能的服务。

3）系统管理

系统管理指对工作流管理系统中运行的业务流程进行管理，包括任务、工作量等的检察。

4）业务过程管理和分析

工作流管理系统包括了业务过程管理和分析，也就是对业务流程进行监控、跟踪、分析以及改进。

3.1.2 工作流生命周期

90 年代美国 MIT 教授迈克尔.哈默(Miachael Hammer)和 CSC 管理顾问公司董事钱皮(JamesChampy)提出了业务流程再造(Bussiness Process Reproduct,BPR)。在企业发展中，业务过程再造的首要目标是为了获取最有效率和效能的业务过程结构，该过程可以带来成本、质量和服务诸多方面的巨幅改进。

传统的工作流管理所涉及到的生命周期可以分为以下四种：过程定义、过程优化、应用系统开发、工作流管理系统执行[26]。然而，这四阶段在实际应用中出现了一些缺陷，比如过程定义容易渗入了参与人员的主观性，使得定义的流程模型容易出错。此外系统开发和执行过程占用时间过长，使得缺乏对业务过程再造的重视。

2002 年，Van der Aalst 等人将 BPR 生命周期描述为：过程诊断、过程重设计、过程重构和过程运行这四大阶段[27]。如图 3-1 所示，与传统工作流管理生命周期的四个阶段相比，Van der Aalst 提出的工作流管理生命周期更合理，因为该描述注意到过程诊断以及过程重设计阶段对于业务过程改进和过程运行的影响的重要性。

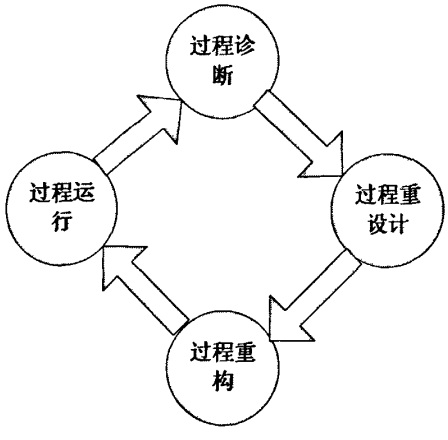

图 3-1 业务流程再造的生命周期四阶段

3.2 工作流挖掘

在本小节，我们将围绕工作流挖掘的基本概念和工作流挖掘问题描述进行介绍。

3.2.1 基本概念

在工作流挖掘出现之前，过程挖掘(process mining)及过程发现(process discovery)已被应用在软件工程中[28]，其目标是利用以后的过程信息，使用过程挖掘技术获得过程模型。近十年来，一些学者把过程挖掘的思想与工作流技术结合，提出了工作流挖掘的研究。

工作流挖掘(Workflow mining)是通过分析业务流程系统的事件日志，自动构造工作流模型的一种分析方法[7][27][29],是把过程挖掘思想应用于工作流管理领域中。在不影响理解的情况下，工作流模型又称为过程模型。

传统的工作流建模方法是根据企业或组织的业务需要，预先定制合适的工作流模型。与这些传统方法不同，工作流挖掘通过收集和分析业务系统日志信息来重构工作流模型，在此阶段是不需要工作流管理系统事先存在的，日志信息可以来源于其它信息系统或者部署任何信息系统前业务过程的手工记录等。当然，在实际应用中，一般的业务系统都会自动把系统执行情况记录到日志文件中。

在一般的情况下，业务流程系统是需要根据实际业务需求预先定义工作流程模型，在工作流管理系统上部署实施，当业务规则或者业务流程变更时，重新定义新的工作流模型。然而，在某些特殊的情况下（如人为原因等），原有工作流模型或者模型的局部分支出现丢失，这时候再通过调研、分析来重构业务流程需要花费企业大量的人力和物力，而且重构得到的流程模型带有一定的个人主观性，降低流程模型的准确性。

然而由于原有工作流模型的执行轨迹被记录在系统的日志文件中，这些执行轨迹能真实地反映业务流程的需求，通过对日志信息进行分析挖掘，重构一个能反映实际执行情况的流程模型，并在此基础上分析原有模型中存在哪里问题，进行流程改进等方面的工作。

工作流挖掘包括事件日志收集、过程挖掘及过程展现三个步骤[28]，如图 3-2 所示。

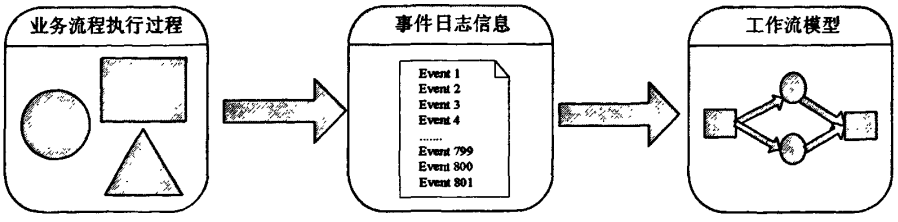

图 3-2 工作流挖掘的三个步骤

总的来说，工作流挖掘思想是利用业务系统日志中的任务的执行情况，对这些执行情况信息进行分析，最后重构出一个合理的工作流模型，并使得日志中记录的所有轨迹都符合重构得到的流程模型的一个实例。

3.2.2 问题描述

工作流挖掘研究的起点是业务系统的日志信息，其一切工作都是以系统日志作为基础，在分析系统日志过程中加入工作流技术，挖掘出与日志相符且符合实际业务需求的工作流模型。

工作流挖掘研究是基于事件日志一般做出以下假设[7]：

1）每一事件指向某一任务（即工作流定义中的活动）。

2）每一事件都归属于某个案例（即工作流中的案例）。

3）事件都按时间排列，或者都有时间记录（即使存在并行任务，事件也是顺序记录的）。

基于以上事件日志的三点假设，工作流挖掘算法着眼于从事件日志中抽离出任务、时间、流程实例等信息，然后经过对这些信息进行归纳、分析等过程，刻画出任务与任务之间的关系（这些关系在工作流模型中体现为工作流的基本结构）。下面我们通过一个示例来阐述工作流挖掘原理。

在日志文件（表 3-1 所示）中包括了五个过程实例的信息，对这五个过程实例进行统计则可以知道共有任务 A、B、C、D、E 五种类型的事件，我们可以根据任务执行时间以及过程实例标识来抽取出这 5 个过程实例信息。从事件日志我们可以观察到，在所有的过程实例中，如果任务 B 被执行，那么任务 C 也会被执行；同时，存在着有些任务 C 被执行，任务 B 也被执行，因此我们可以认为任务 B 和任务 C 是并行执行的。由于事件日志的所有过程实例都是以任务 D 作为结束点，则我们可以认为任务 D 是流程模型的最后任务。假如表 3-1 所表示的事件日志是完备的，按照上述形式类推，得到如图 3-3 所示的 Petri 网工作流模

型。

表 3-1 工作流的事件日志

过程案例标识	任务标识	执行时间
case1	A	2010-03-2 09:08:05
Case2	A	2010-03-2 09:08:36
Case3	A	2010-03-2 09:08:54
Case3	B	2010-03-2 09:09:30
case1	B	2010-03-2 09:09:59
case1	C	2010-03-2 09:11:50
Case2	C	2010-03-2 09:13:50
Case4	A	2010-03-2 09:15:45
Case2	B	2010-03-2 09:16:05
Case2	D	2010-03-2 09:18:10
Case5	A	2010-03-2 09:18:40
Case4	C	2010-03-2 09:19:05
Case1	D	2010-03-2 09:19:55
Case3	C	2010-03-2 09:21:00
Case3	D	2010-03-2 09:23:45
Case4	B	2010-03-2 09:25:10
Case5	E	2010-03-2 09:26:40
Case5	D	2010-03-2 09:28:15
Case4	D	2010-03-2 09:37:55

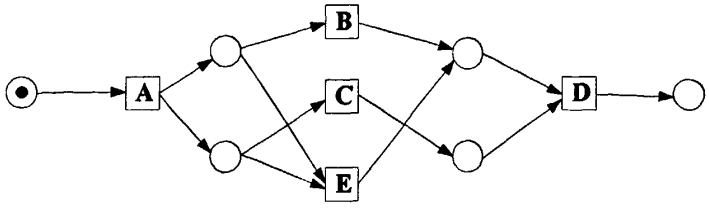

图 3-3 与表 3-1 事件日志相对应的 Petri-net 工作流模型

3.3 业务系统日志

业务系统日志是工作流挖掘的起点，也是过程挖掘的基础。本节主要对业务系统日志特征及其形式化描述进行介绍。

3.3.1 流程日志特征

在一般的业务系统中，业务流程的真实的执行情况会自动的被记录到日志文件中，工作流挖掘就是通过对日志信息进行分析处理，重构得到一个流程模型。也就是说，日志信息是工作流挖掘的数据源。所以在开始工作流挖掘工作之前，需要对日志信息进行预处理，包括收集、转换等。这些日志信息是丰富的，比如人员的登入登出时刻、任务的开始和完成时刻、请假申请单信息和出入库单等一系列与业务功能相关的信息，这些信息能清晰地反映企业的历史或实时的运转情况，它综合了流程、资源、人员等工作流运行状态数据，为进行工作流性能分析提供有效的数据支持。

不同的业务系统，其日志格式可能不同，把各种业务系统的日志内容抽象出来，它们都包括了过程实例（一些柔性工作流平台可能没有过程实例，但能通过一定的规则把一系列任务操作历史关联起来，等价于一个过程实例）、事件名称（任务名称）、执行时间（有些业务系统还包括了任务开始时间、完成事件）三种类型的信息。实际业务系统的流程日志具有以下三个特征[30]：

1）流程日志可能会很大，因为工作流模型包括了一些复杂的结构，如选择、并行路由以及其他基本结构组合成的复杂结构等，流程日志不能确保（在一些实际应用中是不可能）包含所有的可能路径。如工作流模型中有 9 个并行任务，那么可能的路径达到 9!=362880。

2）流程日志可能不完备或者存在噪声，这在实际应用中也是普遍存在的。业务系统的日志可能会有一部分不完整，甚至不正确，如当系统发生异常的时候，一些任务会终止，并且可能会有丢失的可能。

3）流程日志信息丰富多样，日志需要进行一定的预处理。如日志信息包括任务类型、任务名称、任务起始时间等信息，如何充分利用日志信息，来为流程挖掘服务也是工作流挖掘的一个难点。

3.3.2 流程日志的形式化

上一小节介绍了流程日志的内容及其基本特征，为了加深对流程日志的理解和方便下文的流程挖掘讨论的开展，本小节给出流程日志的形式化表示。

定义 3.1 (Workflow log，流程日志[7]) 设 T 代表流程任务的集合，$\sigma \in T^*$ 是

一个流程轨迹，$W \in P(T^*)$是流程日志，其中$P(T^*)$是T^*的幂集,即$W \subseteq T^*$。

在定义 3.1 中，T 表示流程任务的集合，如在 3.2.2 小节的例子中，所有流程实例涉及到的任务包括 A、B、C、D、E，那么集合 T={A，B，C，D，E}。由表 3-1 可以看出实例 1(case1)的包括了任务 A、B、C 及 D，则 ABCD 为一个流程轨迹。在此我们将相同的案例进行合并，得出集合{ABCD，ACBD，AED}为表 3-1 对应的流程日志。

定义 3.2 (Complete workflow log,完整的流程日志[7]) 设三元组 N=(P,T,F)是一个合理的工作流网(WF-net)，即$N \in W$,其中W是 N 的流程日志当且仅当$W \in P(T^*)$且任一路径$\sigma \in W$是 N 的一个执行序列，起始于状态[i]，结束于状态[o]，即(N,[i])[σ>(N,[o])。W 是 N 的完整的流程日志当且仅当：

(1) 对 N 的任一流程日志W'有：$>w' \subseteq > w$。

(2) 任意 t$\in T$,存在一个序列$\sigma \in W$，使$t \in \sigma$。

定义 3.2 给出了完整的流程日志的定义，概括地说，一个合理的工作流网(WF-net)的流程日志只包含能够通过相应流程展示的行为。一个流程日志是完整的，当且仅当所有的直接相连的任务在日志中有某些路径中也是直接相连的。

流程挖掘的目的是通过对流程日志的分析，重构出一个符合业务需求的工作流模型。在进行流程挖掘工作之前，需要对日志中任务之间的关系进行讨论，下面介绍流程日志的一些重要定义。

定义 3.3 (Log-based ordering relations,基于日志顺序关系[7]) 设 W 是任务集合 T 上的工作流日志，如$W \in P(T^*)$，其中W是四元组$(T_w, O_w, t_{wi}, t_{wo})$，$T_w$是任务集合，$t_{wi}$、$t_{wo}$分别对应流程日志的开始任务和结束任务，$O_w$={$>_w, \rightarrow_w, \#_w, \|_w$},设$a,b \in T$，则有：

1）$a >_w b$当且仅当存在这一个轨迹$\sigma = t_1 t_2 t_3 .... t_{n-1}$且有$i \in \{1,.....,n-2\}$，满足$\sigma \in W$、$t_i = a$和$t_{i+1} = b$。

2）$a \rightarrow_w b$当且仅当$a >_w b$和$b \not>_w a$成立。

3）$a \#_w b$当且仅当$a \not>_w b$和$b \not>_w a$成立。

4）$a\|_w b$ 当且仅当 $a>_w b$ 和 $a>_w b$ 成立。

定义 3.3 阐述了日志中的任务之间可能存在的基本的因果关系，给出了任务间的四种关系的定义，分别为顺序关系($>_w$)、直接关系($\rightarrow_w$)、独立关系($\#_w$)以及并行关系($\|_w$)。

这四种基本的关系对于工作流挖掘研究具有十分重要的作用，它们体现日志中任务间最基本、最能直接获取的基本关系，作为挖掘更复杂的结构的基础。下面我们以 3.2.2 小节的工作流日志示例来说明这四种基本关系。

表 3-1 中的日志信息，其具有的流程日志 $W=\{ABCD, ACBD, AED\}$，由表中可得以下关系：$A>_w B$，$A>_w C$，$A>_w E$，$B>_w C$，$B>_w D$，$C>_w D$，$C>_w B$，$E>_w D$，其中关系 $>_w$ 描述了任务间的顺序关系，即两个任务之间相邻执行。而对于直接关系（因果关系），由表可得：$A\rightarrow_w B, A\rightarrow_w C, A\rightarrow_w E, B\rightarrow_w D, C\rightarrow_w D, E\rightarrow_w D$。关系 $\|_w$ 表示潜在的并行关系：$B\|_w C$，$C\|_w B$。关系 $\#_w$ 表示任务之间不直接连接，可能不存在上述关系。

接着，我们再介绍一些符号的定义，便于简化上述的定义。

定义 3.4 $(\in, first, last)$ **[7]** 设 A 是一个集合，$a\in A$，$\sigma=a_1a_2a_3.....a_n\in A^*$ 是集合 A 上长度为 n 的序列，那么 $(\in, first, last)$ 有如下定义:

1. $a\in\sigma$ 当且仅当 $a\in\{a_1, a_2, ..., a_n\}$。

2. $first(\sigma)=a_1$ 当且仅当 $n\geq 1$。

3. $last(\sigma)=a_n$ 当且仅当 $n\geq 1$。

定义 3.4 引入了三种符号：$\in, first, last$，可以对流程日志中的开始任务和结束任务的表述进行简化。

3.3.3 流程日志时间表示

当前绝大部分的业务系统在日志中记录了任务的执行时间（有些系统利用任务类型来标识该执行时间为任务的开始时间还是结束时间），这是时间信息进一步丰富了日志。下面我们就给出关于时间日志的几个定义。

定义 3.5 (Timed Workflow Trace，时间工作流轨迹[31]) 设 T 为日志中的任务集合，D 为时间领域集合（例如，D={……,-3,-2,-1,0,1,2,3,……}或者D={0,1,2,……}形式等），那么$\sigma \in (T \times D)^*$是一个时间工作流轨迹。

定义 3.6 (Timed Workflow Log，时间工作流日志[31]) 设集合 T、 D 为定义 3.5 中描述的集合，$\sigma \in (T \times D)^*$是一个时间工作流轨迹，那么$W \in \mathrm{B}((T \times D)^*)$是一个时间工作流日志。

在定义 3.5 和定义 3.6 中，$T \times D$表示集合 T 和集合 D 的笛卡尔乘积，集合 D 是根据日志时间来定义事件的先后次序（表示方式可以是数字序列或其他基于日志时间上定义的符号等）。在工作流日志信息中，每一个事件都带有一个时间信息来表示该事件的执行时间，一个时间工作流轨迹是事件关于其时间信息的序列，而这些时间工作流轨迹则构成了时间工作流日志。

3.4 Petri 网

工作流模型是对工作流的抽象表示，也就是对经营过程的抽象表示。目前工作流模型理论还是不太成熟，在建模方面没有形成比较有系统的理论体系，下面我们介绍一种有效的工作流模型描述语言—Petri 网。

Petri 网[32]是一种适应于多种系统的图像化、数学化建模工具，其具有严格的数学定义，因此在工作流建模领域中得到广泛应用。作为一种具有数学定义的图形化的工作流建模工具，Petri 网为描述和研究具有并行、异步、分布式和随机性等特征的复杂工作流模型提供支持，并且能用于建立状态方程、代数方程等描述系统行为的模型。

Petri 网是一种有效的模型描述语言，适合于描述具有并行成分的工作流系统。Petri 网在描述工作流中具有以下优势：

1） Petri 网兼顾了严格语义与图形语言两方面。

2） Petri 网在离散化模型描述能力突出。

3） Petri 网是基于状态的工作流模型描述方式。

4） Petri 网具有丰富的模型分析技术与手段。

Petri 网具有较强的工作流模型描述能力，其具有严格的数学定义特征。下面我们给出 Petri 网的相关定义。

定义 3.7 (Petri 网[16]) 设 $PN=(P,T,F,W,M_0)$，其中 P,T,F,W,M_0 满足如下：

1）$P=\{p_1,p_2,.....p_n\}$ 是库所(place)的有限集合。

2）$T=\{t_1,t_2,.....t_n\}$ 是迁移(transition)的有限集合。

3）$F\subseteq(P\times T)\cup(T\times P)$ 是库所与变迁节点间有向弧的集合。

4）W 是弧的权函数。

5）M_0 是初始标记（初始状态）。

在本文不考虑具有权函数的 Petri 网，因此不带初始状态的 Petri 网可表示为三元组 $PN=(P,T,F)$。一个带标记的 Petri 网可以表示为一个有序对 (PN,s)，s 是 P 上的元包，表示 Petri 网的标记。下面我们介绍经典的 Petri 网。

经典 Petri 网是简单的过程模型，由库所(place)和变迁(transition)，有向弧(Connection)，以及令牌(token)等元素组成的。图 3-4 为一个经典的 Petri 网。

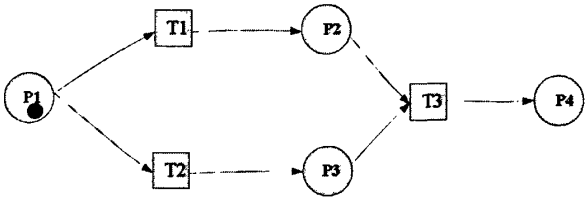

图 3-4 一个经典的 Petri 网

上图由 4 个库所(P1、P2、P3、P4)和 3 个变迁(T1、T2、T3)组成。库所(Place)是用圆形节点表示，变迁(Transition)是用方形节点表示，连接库所和变迁之间的为有向弧(Connection),令牌(Token)是库所中的动态对象，可以从一个库所迁移到另一个库所。

定义 3.8 (Petri 网的实施规则[7]) 设 $PN=(P,T,F)$ 是一个带标记的 Petri 网，那么规则定义如下：

1）变迁$t \in T$是就绪的当且仅当变迁t的每个输入库所都至少包含一个令牌。

2）就绪的变迁可以实施。

3）若变迁t实施，那么t从每个输入库所(input place)p中消耗一个令牌，并为每个输出库所(output place)p产生一个令牌。

定义3.7的含义是：如果一个变迁的每个输入库所都拥有令牌，该变迁即为被允许(enable)。一个变迁被允许时，变迁将发生(fire)，输入库所的标记被消耗，同时输出库所产生令牌。

Petri网的动态性质中，最主要的有有界性（包括安全性）、活性、可达性等，下面对这3种动态性质进行定义。

定义3.9 (可达性[16]) 设(N,M_0)是一个Petri网，其中$N=(P,T,F)$。标号s从M_0是可达的当且仅当存在一个执行序列使得状态M_0经一组中间状态达到状态s。(N,M_0)的可达标记为$(N,M_0>$。

定义3.10(有界性[16]) 设$N=(P,T,F)$是有界的，当且仅当对每个库所p,存在一个自然数n，使得对每个可达状态来讲，p中的标记个数小于n。或者说，可达状态$(N,M>$的数量是有限的。网是安全的当且仅当对于每个库所，标记的最大数目不超过1，或者说对任意的$s' \in [N,s>$和任意$p \in P$，有$s'(p) \le 1$。

定义3.11(活性[16]) 设(N,M_0)是一个Petri网，M_0为初始标识，(N,M_0)称为有活性的，如果在任何状态$M \in R(M_0)$下，都存在$M' \in R(M)$，使得$M'[t>$。

3.5 工作流网 WF-Net

工作流网(WF-net)是基于Petri网的扩展，具有Petri网的优点，又更加适合于对实际业务流程的模拟。工作流网完全支持工作流联盟 WfMC 定义的六种工作流原语：顺序、与分支、与连接、或分支、或连接以及反复关系。本节介绍工作流网 WF-net 的相关概念。

在Petri网基础上，Van der Aalst提出了工作流网(WF-net)的概念：

定义 3.12(工作流网,WF-net[16]) 一个 Petri 网$PN=(P,T,F)$被称为工作流

网，当且仅当它满足如下两个条件：

1．PN 有两个特殊的库所：i 和 o 两个库所。库所 i 是一个起始库所，库所 o 是一个终止库所。

2．如果在 PN 中加入一个新的变迁 t^*，使 t^* 连接库所 o 与 i，即 $\bullet t^* = \{o\}$，$t^* \bullet = \{i\}$，所得到的 PN 是强连接的。

定义 3.11 的含义是：工作流网 WF-net 必须具有一个起始点和一个终止点，从过程实例的生命周期上看，在起始库的 token 代表该过程实例的开始，进入终止库的 token 代表该过程实例的结束。工作流网是存在孤立的状态的，一旦把起始库所和终止库所连接起来，则所得的网是强连通的。

工作流网 WF-net 完全支持工作流联盟 WfMC 定义的六种工作流原语[16]，这些原语在是工作流模型的基础，在工作流模型中普遍存在的。结合 Petri 网，这六种工作流原语向 Petri 网转换的如图 3-5 所示。

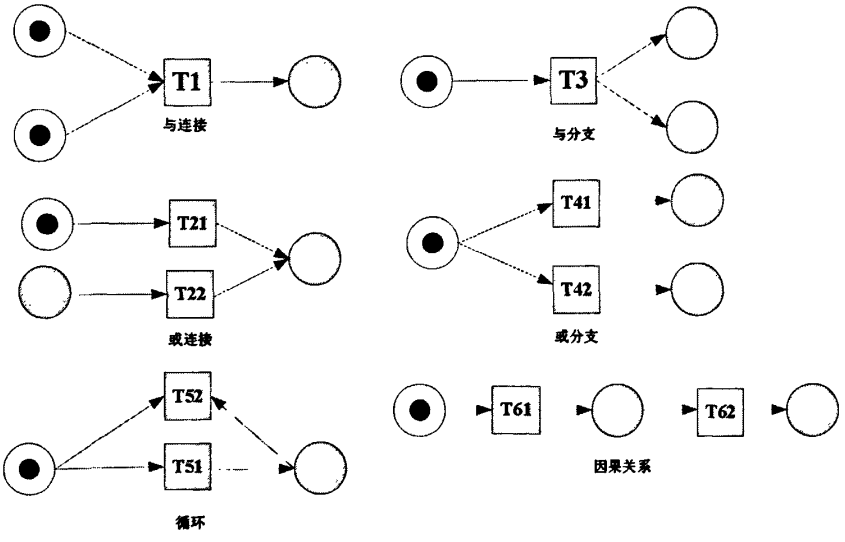

图 3-5 Petri 网表示工作流六种原语

在日志中可以挖掘出库所信息，为了简化工作流网的结构，本文讨论的工作流网 WF-net 是基于不包含隐式库所的前提。因为隐式库所意味着他们不能影响流程的行为，从流程模型的行为角度上看，它们是冗余的。下面我们给出隐式库所的定义。

定义3.13(Implicit place,隐式库所[7]) 设 (PN, s) 是一个Petri网，其中

$PN=(P,T,F)$，s为初始状态。记(N',s)是从(N,s)中删除库所p后所得的网。库所p称为顺序隐式库所(SIP-Sequential Implicit Place)，简称隐式库所。当且仅当$(N,s)=L(N',s)$，其中$L(N',s)$删除p后得到的(N',s)的所有变迁发射序列的集合。

由定义3.13可以得到，隐式库所的删除对原来的Petri网结构及其迁移序列无影响，为了使得工作流挖掘能顺利进行，我们必须保证工作流网WF-net具有清晰的结构，因此要对工作流网中的隐式库所进行排除。

定义3.14(结构化工作流网，SWF网[7]) 设WF-net $N=(P,T,F)$是结构化工作流网(SWF网)，当且仅当：

1. 对所有的$p\in P$且$t\in T$，$(p,t)\in F:|p\bullet|>1$，则$|\bullet t|=1$。
2. 对所有的$p\in P$且$t\in T$，$(p,t)\in F:|\bullet t|>1$，则$|\bullet p|=1$。
3. 不存在隐式库所。

结构化工作流网排除了隐式库所的考虑，从而简化了工作流网，而且完全支持基本的工作流结构：AND-split、AND-join、OR-split以及OR-join。

3.6 流程结构特征

当前大部分的工作流建模技术对于复杂的业务需求往往采用定义复杂的业务流程图方式来解决，所以工作流日志信息中往往包括一些复杂的流程结构，给流程挖掘研究带来一定的难度。下面我们介绍一些复杂的流程结构的特征。

3.6.1 重名任务

重名任务又成为非唯一任务，具体是指在流程模型的任务集合中，存在这两个或者更多的任务具有相同的名字，这些任务虽然具有不同的语义（即每个相同名称的任务是代表不同功能的任务），但是其记录在日志中的任务名称是相同的，所以挖掘算法难以区分重名任务。图3-6为一个重名任务的例子，其中具有两个任务名称为T1。

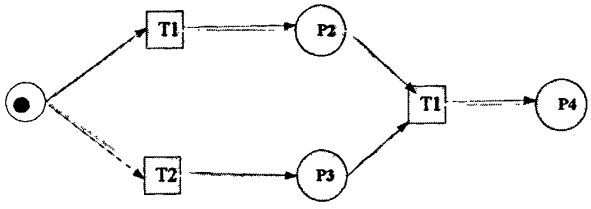

图 3-6 一个重复任务例子

3.6.2 循环结构

带有循环结构的工作流模型在实际应用中是广泛存在的，当前研究的循环结构主要包括两种：单循环结构和双循环结构。设W是由任务集合T构成的业务流程日志，若存在$t \in T$，当且仅当存在$\sigma_1 = ...xy...$，$\sigma_2 = ...xty...$，$\sigma_3 = ...xtty...$且$\sigma_1, \sigma_2, \sigma_3 \in W$，则称$t$为单循环任务。同理，具有双循环结构的日志序列如下：$\sigma_1 = ...xsy...$，$\sigma_2 = ...xss'sy...$，其中$len(s) \geq 1, len(s') \geq 1$，则$s$称为具有双循环任务。

循环结构因为其循环特征，在日志中较难准确地捕捉其循环子序列，图 3-7 为单循环任务和双循环任务的例子。

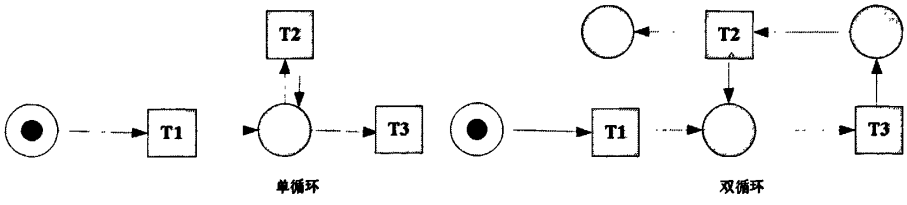

图 3-7 单循环和双循环例子

3.6.3 非自由选择结构

非自由选择结构是控制流挖掘的重点也是难点之一。由于两个任务间可能会存在着间接依赖(Implicit dependencies)，这使得在流程模型中可能存在非自由选择结构。这种结构本质上是任务间的间接依赖关系[17]，其难点表现在：具有非自由选择关系的任务在日志轨迹中不是直接相邻的，那么不相邻的任务间都有可能存在这种关系。图 3-8 为一个自由选择的例子，T1 与 T4、T2 与 T5 是存在非自由选择关系的，因为 T4 的执行，需要 T1 在一开始就执行，同理，T5 也是一样。

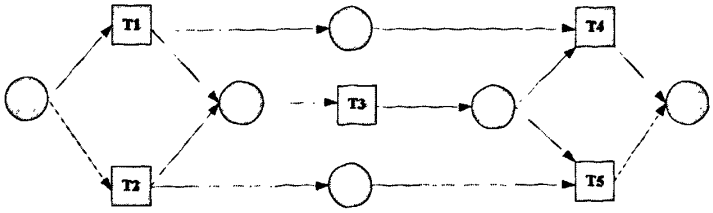

图 3-8 自由选择的例子

3.6.4 隐含任务

隐含任务由于其信息没有被记录在日志中，所以挖掘这类型的任务具有一定的难度。所谓隐含任务，是指某些在建模阶段不存在的任务，由于其不需要被实际执行，所以日志中没有该任务的信息，称为“隐含任务”或者“不可见任务”。

在实际应用中，隐含任务是存在的，比如一些不需要执行任何行为的结点（用于辅助流程建模）等，若在流程挖掘阶段忽略隐含任务，那么挖掘得到的流程模型具有片面性，只体现了日志中反映的行为。

3.7 小结

本章介绍了流程挖掘的相关技术，包括工作流技术、业务系统日志、Petri 网、WF-net 以及挖掘难度大的复杂流程结构的特征。本章首先介绍流程挖掘概念以及其反映的问题，然后给出业务系统日志、Petri 网、WF 网的形式化描述，为后续章节的讨论打下基础，最后基于当前控制流挖掘研究所面临的难点，讨论一些挖掘难度较大的流程结构特征。

第 4 章 流程日志的时态分析

一般业务系统的日志除了包括流程的任务信息外，还有一种重要的信息：日志时间信息。在日志的任务关系分析过程中加入对时间信息的考虑，可以使得挖掘得到的流程模型更符合实际业务需求，并为流程性能分析、改进以及优化提供知识支持。

4.1 日志时态关系

业务系统的日志中一般记录任务自身信息，以及其执行信息：执行开始时间、执行结束时间等。本节讨论日志的相关信息并给出其形式化定义。

4.1.1 时间事件

不同的业务系统其日志格式可能存在较大的差异，然而这些日志都具有一些共性：数据结构语义。日志中都记录了任务的名称、任务时间信息以及任务状态信息，这些信息在语义上是不随日志格式变化而变化的，下面我们给出时间事件的形式化定义。

定义 4.1(Timed Event,时间事件) 设T为日志流程中的任务集合，$t \in T$是其中集合T中的一个任务，$\sigma \in T^*$是一个流程轨迹。那么三元组$E_t = (t, st, et)$表示时间事件，其中st表示任务t在流程轨迹σ中执行的开始时间，et表示任务t在流程轨迹σ中执行的完成时间。

由定义 4.1 定义的时间事件除了具有任务信息外，还包括该任务在其轨迹中的执行开始时间以及完成时间，通过这两个基本的时间信息，能引申出任务的执行花费时间信息。表 4-1 是从系统日志中抽离的时间事件信息。

表 4-1 系统日志中的时间事件信息

流程实例编号	任务名称	开始时间	完成时间	花费时间
case1	A	2010-03-2 09:08:05	2010-03-2 09:08:35	30
case2	A	2010-03-2 09:08:36	2010-03-2 09:08:56	20
case3	A	2010-03-2 09:08:54	2010-03-2 09:08:58	3
case3	B	2010-03-2 09:09:30	2010-03-2 09:09:40	10

case1	B	2010-03-2 09:09:59	2010-03-2 09:10:07	8
case1	C	2010-03-2 09:11:50	2010-03-2 09:12:05	15
case2	C	2010-03-2 09:13:50	2010-03-2 09:14:30	40
case4	A	2010-03-2 09:15:45	2010-03-2 09:16:00	15
case2	B	2010-03-2 09:16:05	2010-03-2 09:16:30	25
case2	D	2010-03-2 09:18:10	2010-03-2 09:18:28	18
case5	A	2010-03-2 09:18:40	2010-03-2 09:18:60	20
case4	C	2010-03-2 09:19:05	2010-03-2 09:19:23	18
case1	D	2010-03-2 09:19:55	2010-03-2 09:20:12	17
case3	C	2010-03-2 09:21:00	2010-03-2 09:22:30	90
case3	D	2010-03-2 09:23:45	2010-03-2 09:24:00	15
case4	B	2010-03-2 09:25:10	2010-03-2 09:25:23	13
case5	E	2010-03-2 09:26:40	2010-03-2 09:27:40	60
case5	D	2010-03-2 09:28:15	2010-03-2 09:29:00	45
case4	D	2010-03-2 09:37:55	2010-03-2 09:38:05	10

上表从日志文件中抽离时间事件信息，并根据时间事件信息计算每个事件的执行花费时间。上表记录了5个过程实例：case1、case2、case3、case4、case5，这5个过程实例涉及到A、B、C、D、E五种类型的事件。如case1的过程实例，其执行轨迹用时间事件表示如下：

$$(A,\text{'2010/3/2 09:08:05'},\text{'2010/3/2 09:08:35'}) \rightarrow (B,\text{'2010/3/2 09:09:59'},\text{'2010/3/2 09:10:07'})$$

$$\rightarrow (C,\text{'2010/3/2 09:11:50'},\text{'2010/3/2 09:12:05'}) \rightarrow (D,\text{'2010/3/2 09:19:55'},\text{'2010/3/2 09:20:12'})^{S}$$

4.1.2 顺序关系

业务系统日志记录业务过程任务的执行时间等信息，而这些时间信息一般表现为时间点，即在时间轴上具有原子性的瞬间时间刻度。

设t_1、t_2为时间轴上的两点，则时间点t_1、t_2一般存在着三种可能的关系：$P=\{>,<,=\}$[35]。这三种关系作用于t_1、t_2可表示为：$t_1>t_2$、$t_1=t_2$、$t_1<t_2$。在日志中，其任务的执行过程其实就是过程实例按照任务的执行时间的排序的任务序列。如表4-1中的过程实例case1，根据其任务A,B,C,D的执行开始时间，任务关系可表示为：$A<B<C<D$，其中$A<B$表示任务A先于任务B执行。在这里，我们不需要考虑任务的执行完成时间：因为前一任务的任务完成时间总是

先于后续任务的执行开始时间的。

基于日志中任务的时间序列，我们给出顺序关系的定义：

定义 4.2(Ordering relations,顺序关系[23]) 设$N=(P,T,F)$是一个工作流网，T是日志中的任务集合，W是一个事件日志，$a,b\in T$，则结合定义 3.3，有如下定义：

1. $a\lhd_w b$当且仅当$a\#_w b$和存在一个任务c，有$c\in T$、$c\rightarrow_w a$和$c\rightarrow_w b$成立。

2. $a\rhd_w b$当且仅当$a\#_w b$和存在一个任务c，有$c\in T$、$a\rightarrow_w c$和$b\rightarrow_w c$成立。

3. $a>>_w b$当且仅当$a\succ_w b$和存在一个轨迹$\sigma=t_1t_2t_3..t_n$以及i，$j\in\{1,...,n\}$，使得$\sigma\in W$，$i<j$，$t_i=a$和$t_j=b$，并且对于所有的$k\in\{i+1,...,j-1\}$，满足$t_k\neq a$，$t_k\neq b$成立，而$t_k\lhd_w a$和$a\rhd_w b$都不成立。

4. $a\succ_w b$当且仅当$a\rightarrow_w b$或$a>>_w b$成立。

定义 4.2 根据日志任务中的执行时间次序来定义任务间的顺序关系，其中$a\lhd_w b$表示a,b具有 XOR-Split 关系；而$a\rhd_w b$表示a,b具有 XOR-Join 的关系。$a>>_w b$表示任务b是非直接方式跟在任务a之后执行的。$a\succ_w b$表示任务b可能是直接跟在任务a之后执行，或者非直接跟在任务a之后执行。定义 4.2 与定义 3.3 共同描述了日志基于执行时间排序的任务顺序关系特征。

4.1.3 时间区间

时间区间在时间轴上表现为无限个连续时间点构成的时段，把时间关系看做一组带定性约束的时段。在业务日志中，一般过程的任务没有附带相关的时间区间属性，然而通过日志的定性时间点信息，可以获取相关的时间区间信息。

定义 4.3(Time interval,时间区间) 设t_i和t_j是时间轴上的不同两点，$t_i\neq t_j$，$I_{ij}=[t_i,t_j]$表示从时间点t_i到时间点t_j的所有连续点构成的空间，即对于时间区间$[t_i,t_j]$的任意时间点t_k,有$t_k\in\{t_{k'}\mid t_i\le t_{k'}\le t_j\}$，则$I_{ij}$称为时间区间。

时间区间信息作为日志中能间接获取得到的时间信息，对流程挖掘研究提供重要的信息依据，因为时间区间信息能反映任务的执行状态信息，包括任务执行

花费时间、任务之间的过渡时间等，这些信息为流程挖掘以及流程改进提供重要的信息支持。

在表4-1是系统日志的一部分片段，该日志中记录了5个过程实例的日志信息，每个日志数据结构包括任务名称、执行开始时间、执行完成时间，从执行开始时间和完成时间可以构成一个时间区间，而这个时间区间的时间跨度代表该任务的执行花费时间。在表4-1中，过程实例case1的执行时间区间表示形式为:[2010-03-2 09:08:05, 2010-03-2 09:08:35]、[2010-03-2 09:08:05, 2010-03-2 09:10:07]、[2010-03-2 09:11:50, 2010-03-2 09:14:30]、[2010-03-2 09:19:55, 2010-03-2 09:20:12]。

4.2 日志时态分析模型

当前绝大部分的流程挖掘研究都没有考虑到日志的一个重要信息：时间。这在一些应用领域中是不合理的，例如在一些服装行业的入库-销售流程，对季度和销售时间间隔等时间因素较为敏感，一些具有相同名称的任务在不同的时期或者不同的执行时间区间，其可能代表不同的意义。同时，流程增量问题也没有被重视。本节针对这两个问题提出了日志时态分析模型。

4.2.1 时态关系测量

一般的日志记录了任务的执行时间（事件能瞬间完成），然而在某些领域中，业务系统的任务不是能瞬间完成的，此时的业务系统日志具有更丰富的时间信息，如执行开始时间（表示任务进入Start状态时间）、执行完成时间（表示任务进入Complete状态的时间）等，也就是说，把任务的开始(Start)和完成(Complete)看作两个原子任务，这些信息在实际应用中都较容易获取并记录下来的。

定义4.4 (Task Status,任务状态) 设T是日志的任务集合，$t \in T$是一个任务，则$S = \{Start, Complete\}$表示日志任务的状态集合，其中$Start$表示当前任务处于开始执行状态，$Complete$表示当前任务处于执行完成状态。集合$E = T \times S = T \times \{Start, Complete\}$表示带状态的事件集合。

上述定义4.4介绍了带状态的任务集合，该任务集合可以进行时间扩展，设ts为一时间戳，则$E = T \times S \times ts = T \times \{Start, Complete\} \times ts$表示带时间状态的事件

集合，其中三元组$(t,Start,ts),(t,Complete,ts)\in E$，前者表示某个带有时间戳$ts$的任务$t$的开始，后者表示某个带有时间戳$ts$的任务$t$的结束。如('经理审核',*Start*,'2009-12-7 15:36:48')表示“经理审核”任务在2009-12-7 15:36:48开始；('经理审核',*Complete*,'2009-12-7 15:37:55')表示“经理审核”任务在2009-12-7 15:36:48结束。

定义 4.5 (Average execution time,平均执行时间) 设T是日志的任务集合，$t\in T$是一个任务，ts_s表示任务t的开始时间戳，ts_c表示任务t的完成时间戳，那么任务t的平均执行时间为$\Delta t_{Exec}=\frac{\sum_{t\in\sigma}(ts_c-ts_s)}{Num_t}$，其中$t\in\sigma$表示在日志轨迹中所有出现的任务$t$，$Num_t$表示任务$t$在日志轨迹中出现的次数。表4-2为对表4-1的工作流日志进行任务的平均执行时间测量。

表4-2 对表4-1的工作流日志中任务的平均执行时间表

任务名称	总次数	平均执行时间（单位：秒）
A	5	17.6
B	4	14
C	4	19.5
D	5	13
E	1	3

表4-2记录了表4-1对应的日志任务的平均执行时间测量，日志中共有5种类型的任务：A、B、C、D、E。通过日志信息，我们可以通过各种任务在日志中出现的总次数以及每次的执行花费时间，能对任务的平均执行时间进行测量。

任务的平均执行时间是流程分析的一个重要指标，为挖掘出的流程模型的任务性能分析提供强而有力的分析依据，并且通过其进行流程任务异常点分析，可以在挖掘过程中能排除一些异常点信息，进而提高流程挖掘的准确性。

定义 4.6 (Average Continuous waiting time,平均连续等待时间) 设T是日志的任务集合，E为日志事件集合，$t_1,t_2\in T$是两个不同的任务，并且$t_1\rightarrow_w t_2$。那么$(t_1,Complete,ts_{1c}),(t_2,Start,ts_{2s})\in E$，其中$(t_1,Complete,ts_{1c})$表示任务$t_1$在时间戳$ts_{1c}$时刻完成，$(t_2,Start,ts_{2s})$表示任务$t_2$在时间戳$ts_{2s}$时刻开始执行。那么平

均连续等待时间 $\Delta TI_{t1\to_w t2}$ 是这样定义的：$\Delta TI_{t1\to_w t2}=\frac{\sum_{t_1\to_w t_2}|ts_{2s}-ts_{1c}|}{Num_{t_1\to_w t_2}}$。其中 $\sum_{t_1\to_w t_2}|ts_{2s}-ts_{1c}|$ 表示日志的所有轨迹中的满足 $t_1\to_w t_2$ 关系的 t_1,t_2 的时间差的总和，$Num_{t_1\to_w t_2}$ 表示日志轨迹中满足 $t_1\to_w t_2$ 关系的 t_1,t_2 的出现次数。

定义 4.6 给出了任务间的平均连续等待时间的定义，表 4-3 是从表 4-1 的系统日志信息抽取出来的平均连续等待时间表。

表 4-3 对表 4-1 的工作流日志中任务的平均等待时间表

任务关系	总次数	平均等待时间（单位：秒）
$A\to_w B$	2	317
$B\to_w C$	2	391.5
$C\to_w D$	2	272.5
$B\to_w D$	2	426
$A\to_w C$	2	239.5
$A\to_w E$	1	460
$C\to_w B$	2	221
$E\to_w D$	1	35

从日志表 4-1 中抽取得到任务间的 8 个 $\to_w$ 关系，如 $A\to_w B$ 关系在日志中出现的次数是 2 次，共花费的总时间是 634 秒，任务 A 与任务 B 平均连续等待时间为 317 秒。

任务间的平均连续等待时间是流程挖掘过程中的重要时间指标，也为流程的任务预测提供时间知识支持。在当前的绝大多数的流程挖掘中，都忽略了日志中的时间信息，这在一些应用领域中是不合理，甚至是不允许的。如一些产品销售流程等，其流程上的行为可能对时间敏感度高：客户在购买汽车后，在一个月内继续购买与汽车相关的配件产品，可以享受 6.5 折的折扣优惠；而在一个月后、半年内继续购买与汽车相关的配件产品，只能享受 9.5 折的折扣优惠。虽然购买汽车和购买汽车配件是属于两个相连的任务，设 A:购买汽车，B:购买配件，那么对于两个同样的 $A\to_w B$ 关系，加入了时间信息（一个是一个月以内，另一个是一个月后、半年之内）后，这两个在日志中的 $A\to_w B$ 关系可以体现为两个不

同的事件处理过程。这样在挖掘出来的流程模型中，任务与任务之间的关系体现为带有时间度量的关系，系统建模人员或业务人员可以根据这些信息很清晰地把握系统实际执行的业务过程，以便作出相关的市场策略。

此外，平均连续等待时间也是流程模型中任务时间预测的重要指标，在流程挖掘中加入任务的时间预测，可以为实际应用中的用户提供宝贵的信息。

定义 4.7 (Process Instance，流程实例) 设 T 为日志中的任务集合，D 为时间领域集合，$\sigma \in (T \times D)^*$是一个时间工作流轨迹，那么$(caseId, \sigma)$则为一个流程实例，$PI = \{(caseId, p) \mid p \in (T \times D)^*\}$为流程实例的集合，其中 *caseId* 为流程实例编号。

一个流程实例在日志文件中表现为具有相同的流程实例编号的一组工作流执行轨迹，流程实例编号在业务日志中是全局唯一的，标识了一个唯一的业务流程实例。在表 4-1 中，共有 5 个流程实例，其流程实例编号依次为:case1、case2、case3、case4 和 case5。表 4-4 为表 4-1 日志信息的 5 个流程实例。

表 4-4 在表 4-1 的工作流日志中 5 个流程实例

流程实例编号	任务轨迹
case1	$A \rightarrow B \rightarrow C \rightarrow D$
case2	$A \rightarrow C \rightarrow B \rightarrow D$
case3	$A \rightarrow B \rightarrow C \rightarrow D$
case4	$A \rightarrow C \rightarrow B \rightarrow D$
case5	$A \rightarrow E \rightarrow D$

在日志中，通过流程实例编号信息，可以将在日志中离散的任务执行记录串联起来，组成流程实例，从而度量出任务之间的各种关系。

然而，在一些柔性的工作流平台中，如基于多 Agent 的柔性工作流平台、基于 Web Service 的灵活组合平台[1]等,其建模方式不需要预先定义一个完整的业务流程模型，而是通过定义一些业务规则来完成建模工作。这样的业务平台就没有体现了流程的概念，所以其产生的日志信息中也没有流程编号等相关信息，而是一些离散的任务操作历史记录，然而这些业务平台记录任务的前置信息，这样使得能将此类格式的日志抽象出具有流程实例形式的日志轨迹信息。下面以本文研究的应用背景：基于多 Agent 的柔性流程平台为例，该平台是中山大学软件研究所设计的柔性的业务流程平台，其在业务流程建模不需要预先定义完整的业务流程模型，其产生的业务日志信息形式如下：

基于多Agent柔性业务平台日志格式

```
<Logs>
  <Agent AgentName="" InstanceId="">
    <EventType></EventType>
    <ActionDate></ActionDate>
    <PreAgentIns></PreAgentIns>
    <Operator></Operator>
  </Agent>
</Logs>
```

图 4-1 基于多 Agent 柔性业务平台日志格式

由图 4-1 可知，这种平台的日志数据结构与传统的工作流平台的日志是可以相互转化的。日志格式中的 PreAgentIns 节点记录了当前 Agent 实例的前置 Agent 实例，可以通过类似动态链表方式将离散的任务串联成一个轨迹，组成一个具有唯一流程编号（虚拟的）的流程实例。

定义 4.8 (实例开始时间戳，完成时间戳) 设 T 为日志中的任务集合，σ 是一个时间工作流轨迹，而 $(caseId,\sigma)$ 则为一个流程实例，其中 $\sigma = t_0 t_1 t_2 .. t_n$，$t_0$ 表示流程轨迹的开始点，t_n 表示流程轨迹的结束点，ts_{0s} 表示任务 t_0 的开始时间，ts_{nc} 表示任务 t_n 的完成时间，那么时间戳 ts_{0s} 称为实例开始时间戳，ts_{nc} 称为实例完成时间戳。

在定义 4.8 之前，我们对日志的时间信息讨论只考虑了日志中任务的执行时间信息。在把日志中的任务信息抽取出来组成具体的流程实例后，流程实例自身也具有时间信息，如流程的开始时间戳、完成时间戳等信息，这些信息能为解决一些实际问题提供时间信息帮助。表 4-5 为表 4-1 日志中流程实例的时间信息。

表 4-5 在表 4-1 日志中流程实例的时间信息

流程实例编号	开始时间戳	完成时间戳
Case1	2010-03-2 09:08:05	2010-03-2 09:20:12
Case2	2010-03-2 09:08:36	2010-03-2 09:18:28
Case3	2010-03-2 09:08:54	2010-03-2 09:24:00
Case4	2010-03-2 09:15:45	2010-03-2 09:38:05
Case5	2010-03-2 09:18:40	2010-03-2 09:29:00

表 4-5 说明了日志中 5 个流程实例的每个实例的开始时间戳和完成时间戳，从流程实例的开始时间戳和完成时间戳可以获取整个实例的花费时间。下面我们

再给出流程实例的耗费时间的定义。

定义 4.9 (流程实例生命周期) $p=(caseId,\sigma)$ 则为一个流程实例，其中 $\sigma=t_0t_1t_2..t_n$ 表示一个日志轨迹，ts_{0s} 为流程实例 p 的开始时间戳，ts_{nc} 为流程实例 p 完成时间戳，那么 $[ts_{0s},ts_{nc}]$ 称为流程实例 p 的生命周期，集合 $C=\{[ts_{0s},ts_{nc}]\}$ 为流程实例的生命周期集合，$Cp=|ts_{nc}-ts_{0s}|$ 称为流程实例耗费时间。

表 4-6 是对表 4-1 的扩展，表示流程实例的生命周期信息。

表 4-6 日志中 5 个流程实例的生命周期信息

流程实例编号	生命期	耗费时间（单位：秒）
Case1	[2010-03-2 09:08:05,2010-03-2 09:20:12]	727
Case2	[2010-03-2 09:08:36, 2010-03-2 09:18:28]	592
Case3	[2010-03-2 09:08:54, 2010-03-2 09:24:00]	906
Case4	[2010-03-2 09:15:45, 2010-03-2 09:38:05]	1340
Case5	[2010-03-2 09:18:40, 2010-03-2 09:29:00]	620

上面统计了日志中各个流程实例的生命周期信息，并获取得到各个流程实例的耗费时间，这样可以有助于开展流程性能分析的工作。

4.2.2 流程实例时间分析

上小节讨论了时态关系测量，包括了流程实例的定义、流程实例开始时间戳、流程实例完成时间戳以及流程实例的生命周期等信息。这些信息为流程实例的时态信息，为进行流程实例时间分析提供重要的依据。

当前绝大部分的流程挖掘研究中，都没有考虑到流程增量的问题，所谓流程增量，是指随着企业业务的发展，市场需求的不断变迁，越来越多新的业务规则应用到企业的业务系统中，而有些旧的业务规则不再被使用，所以在系统日志中出现了流程信息的增量情况。

业务系统经过长期的运行后，产生了大量的业务执行日志信息，这些日志信息的时间跨度很大（可能是一年，也可以是五年等），若直接对这些日志信息进行流程挖掘处理，则挖掘出来的流程覆盖的流程模型信息很可能出现混乱的情况，即挖掘得到的流程模型不但不能为业务建模或者业务流程改进提供支持，而且有可能存在错误的信息，对企业的建模人员或者流程优化人员产生信息误导，

这种影响在实际应用领域中不允许的。

当前大部分的业务系统，特别是一些柔性的业务平台（如基于多 Agent 的柔性业务平台等），都需要通过对日志文件进行分析，重构当前有效的（或能代表最近的业务需求的）实际执行的业务流程模型，以便对流程模型进行监控、改进等工作。然而，仅仅通过直接对日志文件进行挖掘分析，所挖掘出的流程模型很可能不符合（或者反映）当前实际的业务发展需要，一些很旧的（对业务需求无效的）任务（或任务的关系）被挖掘出来，而一些最新引入的任务（或任务的关系）却未被挖掘出来，同时一些以前出现概率很高的任务（或任务关系）可能被忽略。

基于此问题，我们通过对日志中流程实例的时态分析，提出了“时间知识权值法”来处理一些流程增量的问题。在确保挖掘出能反映当前业务需求的流程模型的前提下，充分地利用日志信息来挖掘出一些可能对当前业务有用的流程知识（任务及任务间的关系）。

我们在上一小节中给出了流程实例、流程实例开始时间戳和完成时间戳、流程实例生命周期的形式化定义，基于这些定义，下面给出日志时间跨度的形式化定义。

定义 4.10 (日志时间跨度) 设T代表流程任务的集合，$\sigma \in T^*$是一个流程轨迹，$W \in P(T^*)$是流程日志，PI为日志的流程实例集合，C_ts_{os}为流程实例开始时间戳集合，C_ts_{nc}为流程实例完成时间戳集合，那么$L_{st} = \{\min(ts_{0s}) \mid ts_{0s} \in C_ts_{os}\}$日志开始时间戳，$L_{ct} = \{\min(ts_{nc}) \mid ts_{nc} \in C_ts_{nc}\}$日志结束时间戳，区间$I_{\log} = [L_{st}, L_{ct}]$表示日志的生命期，$Sp_{\log} = \mid L_{ct} - L_{st} \mid$表示日志时间跨度。

由定义 4.10 可知，表 4-1 的记录的日志设计到 5 个流程实例，集合C_ts_{os} = {2010-03-2 09:08:05, 2010-03-2 09:08:36, 2010-03-2 09:08:54, 2010-03-2 09:15:45, 2010-03-2 09:18:40} 为流程实例开始时间戳集合；集合C_ts_{nc} = {2010-03-2 09:20:12, 2010-03-2 09:18:28, 2010-03-2 09:24:00, 2010-03-2 09:38:05, 2010-03-2 09:29:00} 为流程实例完成时间戳集合，那么我们可得到：

$I_{log}=[2010-03-2\,09:08:05,2010-03-2\,09:38:05]$ 为日志的生命期，$Sp_{log}=|2010-03-2\,09:38:05-2010-03-2\,09:08:05|=1800$秒。日志的开始时间戳为流程实例 case1 的开始时间戳，日志的结束时间戳为流程实例 case4 的完成时间戳。

业务系统日志记录了一系列的历史执行任务，这些带有流程实例编号的执行任务具有执行的相关时间戳信息。根据流程实例编号和时间戳信息对这些任务进行重新组合及排序，得到一系列的流程实例。在日志文件中，可以抽取出成千上万个流程实例信息，这些流程实例信息可以表示为 $p=(caseId,[ts_{0s},ts_{nc}])$，其中 $caseId$ 表示为流程实例的编号，ts_{0s} 表示该流程实例的开始时间戳，ts_{nc} 表示该流程实例的完成时间戳。表 4-6 是表示在表 4-1 的日志中抽取的流程实例的生命周期信息。

下面考虑表 4-1 中的日志信息，从日志记录中，我们可以抽取出 5 个流程实例的信息：case1、case2、case3、case4、case5。这 5 个流程实例根据其开始时间戳排序得到如下队列：

($case1$,[2010-03-2 09:08:05,2010-03-2 09:20:12])

($case2$,[2010-03-2 09:08:36,2010-03-2 09:18:28])

($case3$,[2010-03-2 09:08:54,2010-03-2 09:24:00])

($case4$,[2010-03-2 09:15:45,2010-03-2 09:38:05])

($case5$,[2010-03-2 09:18:40,2010-03-2 09:29:00])

根据上述的讨论知，$I_{log}=[2010-03-2\,09:08:05,2010-03-2\,09:38:05]$，$Sp_{log}=|2010-03-2\,09:38:05-2010-03-2\,09:08:05|=1800$ 秒。表 4-1 中的日志跨度为 1800 秒，即 30 分钟。

定义 4.11 (时间知识等级) 设 T 代表流程任务的集合，$W\in P(T^*)$ 是流程日志，区间 $I_{log}=[L_{st},L_{ct}]$ 表示日志的生命期，把区间 I_{log} 根据时间递增划分 Gn 份，并且每份的时间跨度相等，那么 Gn 称为时间知识等级。

上述定义把日志的生命期区间 $I_{log}=[L_{st},L_{ct}]$ 划分为 n 等份，也即产生了 Gn

个相同大小的区间，在各自的区间中所有时间点，我们称这些时间点具有相同的时间知识等级。

在日志文件中，其日志信息的时间跨度可能很长，所以在很多应用领域中，对业务流程日志进行流程挖掘的工作，就不能不考虑日志信息的时效性，如需要根据日志信息挖掘出能代表当前业务发展需要的流程模型，以便为企业业务流程优化提供帮助等。在此，我们把整个日志的生命期划分为 Gn 个相同的区间，即 Gn 个时间知识等级。时间知识等级序列标记为：$Se=\{1,2,3,.....,Gn\}$，从 1 等级开始，到 Gn 等级结束，等级的数值越大，表示在该等级的元素的时间知识值越高（对流程挖掘的重要度越高）。

下图为时间知识等级图。

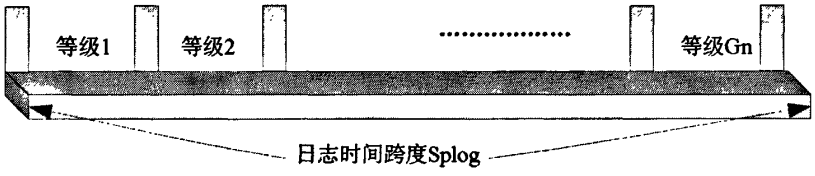

图 4-2 时间知识等级图

图 4-2 把日志时间跨度 $Sp_{\log}$ 划分为 Gn 个时间知识等级，从左到右的时间知识等级序列标记为 $Se=\{1,2,3,.....,Gn\}$。如在最左边的等级 1 内，我们把在这等级内的所有元素的时间知识值是相同的。

定义 4.12 (时间知识跨度) 设区间 $I_{\log}=[L_{st},L_{ct}]$ 表示日志的生命期，Gn 是日志的时间知识等级，$Sp_{\log}=|L_{ct}-L_{st}|$ 表示日志时间跨度，那么 $Sp_{Gn}=\dfrac{Sp_{\log}}{Gn}$ 称为时间知识跨度。

由定义 4.11 得到的 Sp_{Gn} 表示被时间知识等级 Gn 划分的等份区间的时间长度大小，如在表 4-1 中的日志信息中，设时间知识等级 Gn=5，我们由定义可以计算出时间知识跨度：

$$Sp_{\log}=|2010\text{-}03\text{-}2\,09:38:05-2010\text{-}03\text{-}2\,09:08:05|=1800秒=30分钟$$

$$Sp_{Gn}=\frac{1800}{5}=360秒=6分钟$$

上述计算表示等份的5个等级区间的时间长度为6分钟。

定义 4.13 (时间知识区间) 设区间$I_{log}=[L_{st},L_{ct}]$表示日志的生命期，其中L_{st}表示日志的开始时间戳，L_{ct}表示日志的完成时间戳，Gn是日志的时间知识等级，Sp_{Gn}为时间知识跨度，那么区间$I_i=[L_{st}+(i-1)*Sp_{Gn},L_{st}+i*Sp_{Gn}]$称为日志时间知识区间，其中$i\in\{1,2,3,....Gn\}$。

由定义 4.13 得到日志时间知识区间$I_i=[L_{st}+(i-1)*Sp_{Gn},L_{st}+i*Sp_{Gn}]$，由公式看起来$I_i$与$L_{ct}$没有关系，其实在公式$I_i$中隐含了$L_{ct}=L_{st}+Gn*Sp_{Gn}$的关系，也就是说日志时间知识区间$I_i$关于$L_{st},L_{ct}$的表示形式为：

$$I_i=[L_{st}+(i-1)*Sp_{Gn},L_{ct}-(Gn-i)*Sp_{Gn}]$$

定义 4.11、4.12 和 4.13 把日志的生命期划分为Gn个时间跨度相同的时间知识区间，如上述讨论的Gn=5，$Sp_{Gn}=\frac{1800}{5}=360秒=6分钟$，日志的生命期为$I_{log}=[2010-03-2\,09:08:05,2010-03-2\,09:38:05]$，那么划分出5个时间知识区间：$I_1,I_2,I_3,I_4,I_5$。图 4-3 是对图 4-2 的扩展。

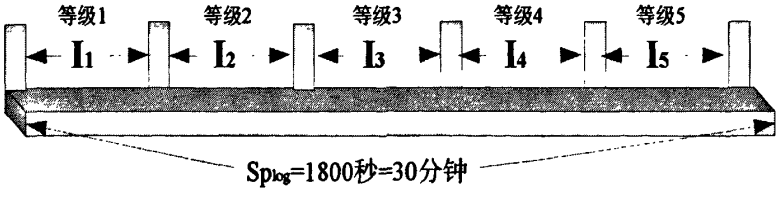

图 4-3 时间知识区间图

图 4-3 展示了在表 4-1 日志信息中的5个时间知识的区间以及每个区间代表的时间知识等级。表 4-7 为这5个时间知识区间的具体时间信息。

表 4-7 日志中5个时间知识区间信息

区间名称	区间值	区间跨度（分钟）
I_1	[2010-03-2 09:08:05,2010-03-2 09:14:05]	6
I_2	[2010-03-2 09:14:05, 2010-03-2 09:20:05]	6
I_3	[2010-03-2 09:20:05, 2010-03-2 09:26:05]	6
I_4	[2010-03-2 09:26:05, 2010-03-2 09:32:05]	6

I_5	[2010-03-2 09:32:05, 2010-03-2 09:38:05]	6

表4-7中具有5个时间知识区间，每个区间的跨度为6分钟，区间与区间之间的关系是相连的，除了区间的边界点之外，各个区间没有交集。在区间内部的所有时间点中，其具有的时间知识相同。

定义4.14 (时间知识值) 设Gn是日志的时间知识等级，时间知识等级序列为1,2,3,....Gn，那么对于任意的一个时间知识区间I_i，我们称$IV_i = \frac{i}{\sum_{k=1}^{Gn} k}$为区间$I_i$的时间知识值。

由表4-7和定义4.14可得表4-1日志中5个时间知识区间的知识值信息，如下表4-8。

表4-8 日志中5个时间知识区间的知识值

区间名称	区间值	时间知识值
I_1	[2010-03-2 09:08:05,2010-03-2 09:14:05]	1/15=0.067
I_2	[2010-03-2 09:14:05, 2010-03-2 09:20:05]	2/15=0.133
I_3	[2010-03-2 09:20:05, 2010-03-2 09:26:05]	1/5=0.2
I_4	[2010-03-2 09:26:05, 2010-03-2 09:32:05]	4/15=0.267
I_5	[2010-03-2 09:32:05, 2010-03-2 09:38:05]	1/3=0.333

表4-8中有5个时间知识区间及其相应的时间知识值，从区间值和知识值可知，时间区间所对应的时间等级越小，其时间知识值越小，其区间对应的时间点在日志的生命期中排得越前。也就是说，在时间等级越小的时间区间中产生的业务系统日志信息，其时效性越小，对挖掘出能反映当前业务流程模型的贡献越小。

通过在日志分析过程中加入对日志的时间知识区间的考虑，能有效地降低这种误差的可能性，然而仅仅通过区间时间知识值来确定日志的时效性的策略存在着一定的不足：如果在整个日志生命期中，日志的时效性差别不大（即业务规则随时间变化不是很大，一些距离现在比较久远的业务规则仍在使用等），由于当前的日志区间时间知识值是只受到时间等级数量多少影响，所以不能体现到在不同应用领域中日志信息具有不同程度的时效性区别。

下面我们给出区间时间知识值的调整因子的定义，并阐述其能动态地调整不

同日志的时效性差别的证明。

定义 4.15 (等级调整因子) 设Gn是日志的时间知识等级，时间知识等级序列为 1,2,3,....Gn，对于任意的一个时间知识区间I_i，定义ω为时间的知识等级调整因子，那么区间I_i的时间知识值$IV_i = \frac{i^{\omega}}{\sum_{k=1}^{Gn} k^{\omega}}$。

由定义 4.15 给出的等级调整因子能合理的反映出日志的时间等级因素在日志信息的时效性中的重要程度，等级调整因子默认值设置为 1，而当$\omega > 1$的时候，说明时间知识等级间的知识值差异明显，较$\omega = 1$的时候大。相反，当$\omega < 1$的时候，时间知识等级间的知识值差异则不是很明显。

定义 4.16 (区间知识值微调因子) 设Gn是日志的时间知识等级，时间知识等级序列为 1,2,3,....Gn，ω为时间的知识等级调整因子，对于任意的一个时间知识区间I_i，定义区间时间知识微调因子ρ，那么区间I_i的时间知识值$IV_i = \frac{i^{\omega} - \rho}{\sum_{k=1}^{Gn} (k^{\omega} - \rho)}$。

定义 4.15 和 4.16 给出了日志区间IV_i在引入等级调整因子以及区间知识值微调因子后的定义。从区间时间知识值公式$IV_i = \frac{i^{\omega} - \rho}{\sum_{k=1}^{Gn} (k^{\omega} - \rho)}$可知，等级调整因子能对区间时间知识值产生较为明显的调整效果，而区间时间知识值微调因子采用与时间等级信息相减的方式，其对区间时间知识值产生的调整效果是不大的。

通过在日志时间区间的知识值IV_i中引入等级调整因子和区间知识值微调因子，可以有效地动态配置日志时间区间之间的日志信息的时效性的差异，从而能灵活地根据不同实际应用（或业务系统实际情况）来调整日志信息的时效性的差异。

定理 4.1 设区间时间知识微调因子为ρ，那么随着微调因子ρ越大，日志时间区间之间的日志信息时效性差异越大。反之，随着微调因子ρ越小，日志信息的时效性差异越小。

证明：设I_i, I_{i-1}为日志的两个时间区间，其区间的时间知识值分别为：$IV_i = \frac{i^{\varpi} - \rho}{\sum_{k=1}^{Gn}(k^{\varpi} - \rho)}$和$IV_{i-1} = \frac{(i-1)^{\varpi} - \rho}{\sum_{k=1}^{Gn}(k^{\varpi} - \rho)}$。那么这两个连续的时间区间的差异值为：$D_{i,i-1} = IV_i - IV_{i-1} = \frac{i^{\varpi} - \rho}{\sum_{k=1}^{Gn}(k^{\varpi} - \rho)} - \frac{(i-1)^{\varpi} - \rho}{\sum_{k=1}^{Gn}(k^{\varpi} - \rho)} = \frac{i^{\varpi} - (i-1)^{\varpi}}{\sum_{k=1}^{Gn}(k^{\varpi} - \rho)} = \frac{i^{\varpi} - (i-1)^{\varpi}}{\sum_{k=1}^{Gn}(k^{\varpi}) - Gn \bullet \rho}$。由公式$D_{i,i-1}$可知，两个连续的时间区间的差异值是随着$\rho$的增大而增大，证毕。

而对于等级调整因子ω，我们在实际应用领域中控制在$0 \le \omega \le 3$的范围内，因为对于ω过大会造成对相连的两个日志时间区间的日志时效性差异过大，对于某些应用领域中是不合理的。在$0 \le \omega \le 1$的范围内，两个连续的时间区间的差异值会随着ω的增大而增大。

在一些情况下，我们可能不需要考虑日志的时效性问题，即日志文件中不同时期的事件日志信息都对流程挖掘的影响是等同的，这时我们可以设置$\rho = 0$和等级调整因子$\omega = 0$。这时设置的日志时间等级Gn可以为任意，假设 $Gn=5$,那么$IV_1 = IV_2 = IV_{31} = IV_4 = IV_5 = \frac{1-0}{\sum_{k=1}^{5}(1-0)} = \frac{1}{5}$，这样就代表这 5 个时间区间的时间知识值相同。

在进行后续讨论之前，我们先回顾本节的问题：流程增量问题。流程增量问题主要是由于企业的业务需求变动，业务系统执行的业务规则随着时间变迁而产生较大的变化，如果不预先对日志信息进行处理，直接进行流程挖掘会使得挖掘得到的模型不能反映当前或者近期的业务需要。产生这样的原因主要日志中的流程实例信息在流程挖掘中存在着时效性，一些历史比较久远的流程对当前实际执行（或者近年实际执行）的业务过程没有参考价值。而可能由于时间关系，近期执行的流程实例数目远远没有那些陈旧的流程实例的多，对这样的日志进行流程挖掘可能会存在着准确性问题以及降低了对流程建模或改进的参考价值。

下面我们基于本节的讨论，提出“时间知识权值法”来衡量日志中各个流程实例信息的时效性，并在日志文件中合理地定量选取的流程实例信息，为下一节的流程挖掘工作提供合理的日志信息。

定义 4.9 给出了流程实例生命周期的形式化表示，流程实例生命周期是一个时间区间的表现形式：$C_{plns}=[ts_{0s},ts_{nc}]$，其中$ts_{0s}$表示该流程实例的开始时间戳，$ts_{nc}$表示该流程实例的完成时间戳。而定义 4.13 给出了日志时间知识区间的形式化表示：$I_i=[L_{st}+(i-1)*Sp_{Gn},L_{st}+i*Sp_{Gn}]$，其中$L_{st}$表示该日志的开始时间戳，$Sp_{Gn}$表示日志的时间区间的跨度。在同一个日志时间知识区间上的所有时间点具有的时间知识值是相同的。然而对于非时间点类型（而是时间区间类型）的流程实例生命周期而言，其具有的时间知识值的怎样衡量呢？

一般的业务系统日志都会存在日志的生命期，从而可以计算出日志的时间区间，这些日志的时间区间与流程实例具有一定的时间上的关系，设I_i为一日志时间区间，C_{plns}为某一流程实例的生命期，I_i与C_{plns}的交集多于一个时间点，那么I_i与C_{plns}一般具有以下 3 种：

- ✓ 交叉重叠关系
- ✓ 包含关系
- ✓ 被包含关系

交叉重叠关系是指I_i与C_{plns}具有重叠的集合，即$ts_{nc}>(L_{st}+i*Sp_{Gn})>ts_{0s}$且$ts_{0s}>L_{st}+(i-1)*Sp_{Gn}$，或者$(L_{st}+i*Sp_{Gn})>ts_{nc}>L_{st}+(i-1)*Sp_{Gn}$。图 4-4 为交叉重叠关系的展示。

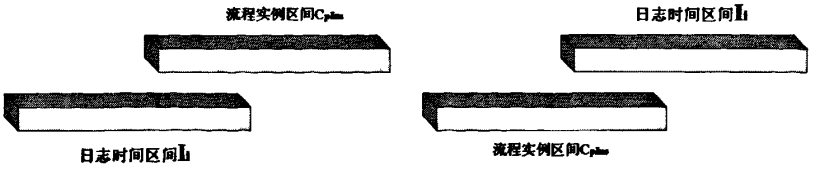

图 4-4 交叉重叠关系

包含关系是指I_i包含C_{plns}，即$(L_{st}+i*Sp_{Gn})>ts_{nc}>ts_{0s}>L_{st}+(i-1)*Sp_{Gn}$。图 4-5 为包含关系的展示。

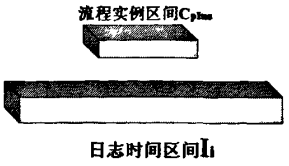

图 4-5 包含关系

被包含关系是指C_{plns}包含I_i，即$ts_{mc} > (L_{st} + i * Sp_{Gn}) > L_{st} + (i-1) * Sp_{Gn} > ts_{0s}$。图 4-6 为被包含关系的展示。

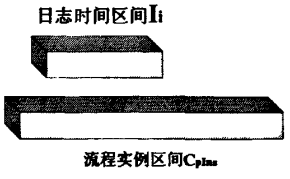

图 4-6 被包含关系

在一般的实际应用领域中，日志的时间区间的跨度一般是远大于任何一个流程实例区间的时间跨度的。所以在上述讨论的 3 种区间之间的关系中，包含关系是在日志信息中最经常出现的，同时也会出现交叉重叠关系，但其出现的频率远没有包含关系多，被包含关系出现的频率很低，只要日志时间等级Gn设置合理，基本上不会出现被包含关系。

当一个流程实例的C_{plns}被日志时间区间I_i包含时，该流程实例的时间知识值等于该时间区间I_i的时间知识值，而当流程实例C_{plns}与日志的时间区间I_i发生交叉重叠关系或者I_i被C_{plns}包含时候时，该流程实例的时间知识值则需要综合流程实例所涉及到的日志时间区间，具体形式如下：

定义 4.17 (流程实例时间知识值) 定义V_{plns}表示流程实例时间知识值，I_i表示一个日志时间区间，IV_i表示I_i的时间知识值，C_{plns}为一个流程实例的生命周期，那么流程实例时间知识值的可表示如下：

$$V_{plns}=\begin{cases} IV_i \quad \text{.....当} I_i \text{包含} C_{plns} \\ \dfrac{\sum_{I_i \in I'}(L_i \times IV_i)}{Sp_{Gn}} \quad \text{.....当} I_i \text{与} C_{plns} \text{交叉重叠或} I_i \text{被} C_{plns} \text{包含} \end{cases}$$

其中，I' 表示与区间 C_{plns} 发生交叉重叠或者被 C_{plns} 包含的所有日志时间区间的集合，L_i 表示这些日志时间区间与 C_{plns} 的重叠时间区间的长度。

通过流程实例时间知识值 V_{plns} 的定义，可以计算出日志中每个流程实例的时间知识值，从而可以定量地衡量该流程实例对流程挖掘工作的时效性。下面通过一个简单的例子来说明流程实例的时间知识值的计算过程。

在表 4-1 的日志信息中，包含了 5 个流程实例：case1、case2、case3、case4、case5。表 4-6 给出了这 5 个流程实例的生命周期信息，设 $Gn=5$，则产生了 5 个日志时间区间，表 4-8 给出了这个 5 个日志时间区间的知识值。

表 4-9 流程实例与日志时间区间的关系表

实例编号	生命期	涉及的日志时间区间
Case1	[2010-03-2 09:08:05,2010-03-2 09:20:12]	I_1，I_2，I_3
Case2	[2010-03-2 09:08:36, 2010-03-2 09:18:28]	I_1，I_2
Case3	[2010-03-2 09:08:54, 2010-03-2 09:24:00]	I_1，I_2，I_3
Case4	[2010-03-2 09:15:45, 2010-03-2 09:38:05]	I_2，I_3，I_4，I_5
Case5	[2010-03-2 09:18:40, 2010-03-2 09:29:00]	I_2，I_3，I_4

由于流程实例时间知识值 V_{plns} 公式中的当 I_i 包含 C_{plns} 情况的计算比较简单，所以当前例子着重于当 I_i 与 C_{plns} 交叉重叠或 I_i 被 C_{plns} 包含情况下的流程实例时间知识值 V_{plns} 的计算。

如流程实例 case1，与三个日志时间区间有交叉重叠关系或者包含关系，其中 case1 包含了 I_1，I_2，case1 与 I_3 有交叉重叠关系。根据公式 V_{plns} 以及表 4-8 的区间时间知识值，有如下计算过程：

$$V_{plns}=\frac{360\times 0.067+360\times 0.133+7\times 0.2}{727}=0.1001$$

那么流程实例 case1 的时间知识值为 0.0845。同理可得，可以计算出其他流程实例的时间知识值。表 4-10 为 5 个流程实例的时间知识值表。

表 4-10 5 个流程实例的时间知识值表

实例编号	涉及的日志时间区间	实例时间跨度（秒）	时间知识值
case1	I_1, I_2, I_3	727	0.1001
case2	I_1, I_2	592	0.0963
case3	I_1, I_2, I_3	906	0.1278
case4	I_2, I_3, I_4, I_5	1340	0.2407
case5	I_2, I_3, I_4	620	0.214

表 4-10 给出了 5 个流程实例的时间知识值，时间知识值越大，代表该流程实例对流程挖掘的贡献越大(这里的贡献大是指该流程实例的在日志信息中的时效性越强)。

4.2.3 任务时态关系构建

在一些实际应用中，尤其是对时间敏感度高的行业，如 4.2.1 小节提到的例子：客户在购买汽车后，在一个月内继续购买与汽车相关的配件产品，可以享受 6.5 折的折扣优惠；而在一个月后、半年内继续购买与汽车相关的配件产品，只能享受 9.5 折的折扣优惠。虽然购买汽车和购买汽车配件都可以属于两个相连的过程，设 A:购买汽车，B:购买配件，那么对于两个同样的 $A\rightarrow_w B$ 关系，加入了时间信息（一个是一个月以内，另一个是一个月后、半年之内）后，这两个在日志中的 $A\rightarrow_w B$ 关系可以体现为两个不同的事件处理过程。

在流程挖掘中引入了任务之间的时间信息，可以使得挖掘出来的流程模型能为业务人员或者系统建模人员提供更丰富的信息，流程模型中任务与任务之间的过渡过程带有时间信息，使得业务人员能根据这些信息很清晰地把握系统实际执行的业务过程，以便市场人员作出相关市场策略，应对市场的不断变化发展。

日志信息中一个流程实例的任务序列是表现为带有完成时间戳的任务排序，

这样的一个序列可表示为：(t_1,ts_{1c})，(t_2,ts_{2c})，(t_3,ts_{3c})，……(t_n,ts_{nc})。其中$t_i \in T$，T为日志中的任务集合，$i \in \{1,2,3,.....,n\}$。ts_{ic}表示任务t_i的完成时间戳。那么在流程实例中，对于两个相连的任务t_i与任务t_{i+1}，我们给出如下定义：

定义 4.18 (任务时间间隔) 设T 为日志信息中的任务集合，任务t_i，$t_j \in T$。任务t_i与任务t_j之间的时间间隔定义为$Sp_{i,j} = |ts_{js} - ts_{ic}|$，其中$ts_{js}$表示任务$t_j$的开始时间戳，$ts_{ic}$表示任务$t_i$的完成时间戳。

定义 4.8 给出了任务间的时间间隔定义，在一个流程实例中，对于相连的两个任务t_i与任务t_{i+1}，其任务间的时间间隔$Sp_{i-1,i} = |ts_{is} - ts_{(i-1)c}|$。在日志中的流程实例中，我们一般只需要考虑流程实例中相连任务之间的时间间隔。

设$P=(caseId,\sigma)$为日志中的一个流程实例，$\sigma = t_1t_2t_3.....t_n$为流程实例的任务轨迹，对于任务$t_i \in T$，让$(t_i,Sp_{i+1,i})$表示带有时间间隔的任务$t_i$的二元组，那么该流程实例的时间间隔任务序列可表示为如下方式：

$(t_1,ts_{2s}-ts_{1c})$，$(t_2,ts_{3s}-ts_{2c})$，$(t_3,ts_{4s}-ts_{3c})$，……$(t_{n-1},ts_{ns}-ts_{(n-1)c})$

即：$(t_1,Sp_{2,1})$，$(t_2,Sp_{3,2})$，$(t_3,Sp_{4,3})$，……$(t_{n-1},Sp_{n,n-1})$

在各个流程实例中，其相连的的两个任务的时间间隔可能会存在这较大的差异，如在表 4-1 中的日志信息中，case1 的任务轨迹为$A \rightarrow B \rightarrow C \rightarrow D$,该流程实例的时间间隔任务序列可表示为如下：

$$case\,1:(A,84),\ (B,103),\ (C,470),\ (D)$$

同理可得，可以获取得到其他流程实例的时间间隔任务序列：

$$case\,2:(A,294),\ (C,95),\ (B,100),\ (D)$$

$$case\,3:(A,32),\ (B,680),\ (C,75),\ (D)$$

$$case\,4:(A,185),\ (C,347),\ (B,752),\ (D)$$

$$case\,5:(A,460),\ (E,35),\ (D)$$

在上面所述的形式中，任务间的时间间隔单位为秒，从流程实例 case1 可知，任务 A 执行完后，经过 84 秒间隔后，任务 B 进入开始执行状态，待任务 B 执行完后，经过 103 秒后，任务 C 进入开始执行状态，待任务 C 执行完后，经过 470 秒后，任务 D 进入开始执行状态，待任务 D 执行完毕后，流程结束。

定义 4.19 (最大时间间隔，最小时间间隔) 设 SpT 为任务间时间间隔的集合，$SpT=\{Sp_{i,j} \mid t_i,t_j \in T,\exists \sigma,t_i t_j \in \sigma\}$ 。那么我们定义 $Max\_sp=\max_{Sp_{i,j} \in SpT}\{Sp_{i,j}\}$ 为最大时间间隔，$Min\_sp=\min_{Sp_{i,j} \in SpT}\{Sp_{i,j}\}$ 为最小时间间隔。

由定义 4.19 可知，表 4-1 日志信息中的任务间最大时间间隔 $Max\_sp=752$，所属流程实例为 case4，是任务 B 到任务 D 的时间间隔；任务间最大时间间隔 $Min\_sp=32$，所属流程实例为 case3，是任务 A 到任务 B 的时间间隔。

定义 4.20 (时间间隔差异范围) 设 SpT 为任务间时间间隔的集合，且 Max_sp 表示最大时间间隔，Min_sp 表示最小时间间隔，那么我们定义 $\mathrm{Re}gion\_sp=\mid Max\_sp-Min\_sp \mid$ 为时间间隔差异范围。

时间间隔差异范围表现为日志中任务间的最大时间间隔与最小时间间隔的差距，差异范围值越大，表示在当前日志中任务间时间间隔的分布有可能比较分散，事件的过渡时间可能有不同范围的取值。

在表 4-1 的日志信息中，任务间的时间间隔差异范围 $\mathrm{Re}gion\_sp$=752-32=720 秒=12 分钟。

定义 4.21 (间隔均差) 设 $\mathrm{Re}gion\_sp$ 为时间间隔差异范围，定义参数 k，则称 $I\_\mathrm{Re}=\frac{\mathrm{Re}gion\_sp}{k}$ 为任务时间间隔均差，参数 k 表示把时间间隔差异范围等分为 k 等份。

定义 4.21 把任务时间间隔差异范围等分为 k 等份，得到任务间隔均差，下面我们讨论任务间隔均差的意义及其应用。

在日志信息中，可以获取任务的相关时间信息：最大时间间隔 Max_sp、最小时间间隔 Min_sp 以及时间间隔差异范围 $\mathrm{Re}gion\_sp$。时间间隔差异范围

$\mathrm{Re}gion_sp$体现了系统日志任务间的等待时间的差异性程度，参数k表示为业务日志信息中任务间的间隔时间类型的种类。

如设置参数$k=3$表示任务间的时间间隔可划分为3种类型，$\mathrm{Re}gion_sp=720$秒，间隔均差$I_\mathrm{Re}=\frac{720}{3}=240$秒=4分钟。那么可根据最大时间间隔$Max_sp$、最小时间间隔$Min_sp$和间隔均差$I_\mathrm{Re}$来划分如下区间：

- $I_1:[Min\_sp, Min\_sp+I\_\mathrm{Re}]$，即$\forall t\in I_1, Min_sp\le t\le Min_sp+I_\mathrm{Re}$
- $I_2:(Min\_sp+I\_\mathrm{Re}, Min\_sp+2\times I\_\mathrm{Re}]$，即
  $\forall t\in I_2, Min\_sp+I\_\mathrm{Re}<t\le Min\_sp+2\times I\_\mathrm{Re}$
- $I_3:(Min\_sp+2\times I\_\mathrm{Re}, Max\_sp]$，即
  $\forall t\in I_3, Min\_sp+2\times I\_\mathrm{Re}<t\le Max\_sp$

上述讨论把最大时间间隔Max_sp划分3等份，每等份的间隔均差为4分钟，参数$k=3$表示会产生3个时间区间：I_1，I_2，I_3。进一步把划分的区间进行形式化描述为如下：

设$IC=\{I_1, I_2, I_3, \ldots. I_k\}$表示任务间隔区间集合，那么对于任意的任务间隔区间$I_i$，其中$i\in\{1,2,..k\}$。那么$\forall t\in I_i$，$Min_sp$为最小时间间隔，下面关系式成立：$Min\_sp+(i-1)\times I\_\mathrm{Re}<t\le Min\_sp+i\times I\_\mathrm{Re}$。

以表4-1的日志信息作为基础，参数$k=3$，$\mathrm{Re}gion_sp=720$秒，间隔均差$I_\mathrm{Re}=240$秒，最大时间间隔$Max_sp=752$、最小时间间隔$Min_sp=32$。那么可以得到如下时间间隔时间区间信息。

$$I_1:[32,272]\qquad I_2:(272,512]\qquad I_3:(512,752]$$

区间I_1，I_2，I_3没有交集，所以流程实例中的任意的一个任务（除了终止任务）都可以找到其相连的任务，并计算出时间间隔$Sp_{i,j}$，在区间集合IC存在一个区间I_i，使得$Min_sp+(i-1)\times I_\mathrm{Re}<Sp_{i,j}\le Min_sp+i\times I_\mathrm{Re}$。

表4-11为表4-1的日志信息的任务间隔时间区间信息。

表4-11 表4-1的日志信息中任务间隔时间区间表

任务关系	实例编号	任务间隔时间区间
$A \rightarrow_w B$	case1;case3	I_1
$B \rightarrow_w C$	case1;case3	$I_1; I_3$
$C \rightarrow_w D$	case1;case3	$I_1; I_2$
$B \rightarrow_w D$	case2;case4	$I_1; I_3$
$A \rightarrow_w C$	case2;case4	$I_1; I_2$
$A \rightarrow_w E$	case5	I_2
$C \rightarrow_w B$	case2;case4	$I_1; I_2$
$E \rightarrow_w D$	case5	I_1

表 4-11 第 1 列给出了表 4-1 的日志信息的任务关系，第 2 列说明了该任务关系在哪些流程实例中出现，第 3 列说明了该任务关系在日志信息中对应的的任务间隔时间区间信息。

4.2.4 时间日志处理

在本节之前，我们讨论了日志的时态关系、日志时态关系测量以及流程实例和任务时态关系的分析，本节在此基础上提出了日志时间处理模型，对原始日志进行预处理，并抽取出对流程挖掘有价值的时间知识。

日志时间处理模型主要包括如下两方面：

- ◆ 基于“时间知识权值法”生成待挖掘的日志信息
- ◆ 日志任务时间关系信息抽取与分析

1. 基于“时间知识权值法”生成待挖掘日志信息

在一些应用领域中，由于原有的业务流程模型缺失，或者如基于多 Agent 的柔性业务平台一样：在系统建模阶段都不需要预先定义完整的业务流程模型等，需要对业务系统的日志信息进行挖掘，重构日志反映的业务流程模型。

然而，业务系统日志信息的时间跨度往往很大，有可能出现几年前的业务系统执行记录，这样的日志信息可能不适合于需要挖掘能反映当前（或者近期）业务执行过程的应用，因为日志信息存在一定的时效性。在本节中，基于以前章节的讨论，提出“时间知识权值法”，考虑到日志信息的时效性对流程挖掘工作的

影响，合理地生成日志信息，以提供给流程挖掘使用。

设$W \in B((T \times D)^*)$是一个时间工作流日志，$\sigma \in (T \times D)^*$是一个时间工作流轨迹，$PI = \{(caseId, p) \mid p \in (T \times D)^*\}$为流程实例的集合，$V_{plns}$为一个流程实例时间知识值，那么对于流程实例集合中所有的流程实例，根据定义 4.17 计算其对应的流程实例时间知识值。表 4-12 为表 4-1 的日志信息的 5 个流程实例对应的时间知识值表。

表 4-12 5 个流程实例的时间知识值表

实例编号	涉及的日志时间区间	时间知识值
case1	I_1, I_2, I_3	0.1001
case2	I_1, I_2	0.0963
case3	I_1, I_2, I_3	0.1278
case4	I_2, I_3, I_4, I_5	0.2407
case5	I_2, I_3, I_4	0.214

流程实例时间知识值是衡量该流程实例的时效性的指标，若时间知识值越大，表示该流程实例所包含的任务执行信息越符合近期或者当前的业务流程模型，对流程挖掘工作的贡献越大，那么该流程实例被用于进行流程挖掘工作的可能性越高。

本文讨论的流程挖掘研究主要包括以下两方面的工作：日志信息预处理、流程挖掘与分析。日志信息预处理阶段对流程挖掘研究异常重要，该阶段影响着流程挖掘工作的准确性及信息的有效性。当前的流程挖掘研究没有考虑流程增量的问题，使得挖掘出的流程模型不能正确反映当前或近期的实际业务流程模型。在此，我们通过考虑日志信息的时效性问题，来有效地降低流程增量问题对流程挖掘带来的影响，提高挖掘出来的流程模型的准确性和信息的参考价值。

定义 4.22 (时间知识权值) 设$W \in B((T \times D)^*)$是一个时间工作流日志，$PI = \{(caseId, \sigma) \mid \sigma \in (T \times D)^*\}$为日志$W$中的所有流程实例的集合，$\forall p_i \in PI = \{p_1, p_2, p_3, ..., p_n\}$，$i \in \{1,2,3,.....,n\}$，$V_{plns}(p_i)$表示流程实例$p_i$的时

间知识值，那么定义 $w(p_i)=\frac{V_{plns}(p_i)}{\sum_{j=1}^{n}V_{plns}(p_j)}$ 为流程实例 p_i 的时间知识权值。

流程实例的时间知识权值衡量了该流程实例在当前日志中的重要程度，时间知识权值越高，表示该流程实例的信息对流程挖掘工作的正确性支持越大。在待挖掘日志中其出现的频率应该越高。

表 4-13 为日志表 4-1 中所有流程实例的时间知识权值信息。

表 4-13 日志表 4-1 中所有流程实例的时间知识权值

实例编号	时间知识值	时间知识权值
case1	0.1001	0.1285
case2	0.0963	0.1236
case3	0.1278	0.1641
case4	0.2407	0.3090
case5	0.214	0.2748

由表 4-13 可知，case4 和 case5 的时间知识权值较大，证明它们的时效性较高，较能反映当前业务执行过程，而 case1、case2、case3 的时间知识权值都非常接近。在实际应用中，日志文件存在着成千上万的个任务执行记录，这些任务执行记录能组织成许多流程执行实例。假如我们为流程挖掘工作提供 1000 个流程实例日志信息，通过从原始日志抽取所有的流程实例信息，并计算每个流程实例的时间知识权值，以确定为日志流程实例候选集生成该类型的流程实例的数目。下面以表 4-13 的例子来说明日志流程实例候选集的生成过程。

日志流程实例候选集的实例数目为 1000 个，那么在表 4-1 中的 5 个流程实例各自需要生成的数量为 $N(p_i)=N*w(p_i)$，其中 N 为日志流程实例候选集的实例数目，$w(p_i)$ 为流程实例 p_i 的时间知识权值。那么可计算各个流程实例的需生成的数量：

$N(case\ 1)=1000\times0.1285\approx128$　　$N(case\ 2)=1000\times0.1236\approx124$

$N(case\ 3)=1000\times0.1641\approx164$　　$N(case\ 4)=1000\times0.3090=309$

$N(case\ 5)=1000\times0.2748\approx275$

也就是说，候选集中包括了 128 个流程实例 case1 类型的实例信息，124 个流程实例 case2 类型的实例信息，164 个流程实例 case3 类型的实例信息，309 个流程实例 case4 类型的实例信息，275 个流程实例 case5 类型的实例信息。

这 1000 个流程实例构成了日志流程实例候选集。然而在实际应用中，业务流程会随着时间的推移而变更，可能一些以前使用的任务已经被取消，而一些新增的任务由于运行的时间过短，在日志中记录的信息非常有限，导致新增的任务与任务之间的关系难以被挖掘，而被取消的任务由于历史执行信息过多，却被挖掘出来，导致挖掘得到的流程模型不能较准确地体现当前（或近期）的业务实际执行过程，不仅不能为流程模型改进提供知识支持，而且可能产生误导业务人员的模型信息。因此我们需要消除日志候选集中价值不大的信息。

定义 4.23 (任务消除规则) 设 CPI 为候选集流程实例集合，T 为 CPI 中涉及到的任务集合，$\forall t_i \in T$，设 $N(t_i)$ 表示任务 t_i 在流程实例集合 CPI 中出现的次数，$N(T)$ 表示各个任务在 CPI 中出现的总次数，那么任务 t_i 的频率为 $Frequency(t_i)=\frac{N(t_i)}{N(T)}$，设 ∂ 为任务消除率，表示在所有的任务中，需要被选取的任务占总任务的比率。

在大部分的业务系统中，存在这某类型任务是不随时间推移而变更的（也就是说其时间敏感性不高），如流程模型的异常处理任务等，这些任务在业务流程中都是很通用的，它基本不随时间推移而变更，然而由于任务的性质，其执行的概率是很低的，所以这类型任务在日志中出现的频率很低，但是不能因为其频率低而把这类型任务在流程中消除，所以我们需要引入一种机制来确保这类型任务不被消除：先验知识验证机制，也就是说流程挖掘工作者（如建模人员）可以在流程工作开始之前，以参数的形式配置哪些任务为必要任务。这样，在日志时态分析工作中，即使该任务的概率很低，系统仍然不会将其消除。以下为任务消除算法的伪代码：

输入:候选流程实例集 CPI,必要任务列表 TList，任务消除率∂
输出:候选处理日志 CW
算法过程:
1. 获取 CPI 中的任务集合 T
2. T←T/TList,计算集合 CPI 中所有属于 T 的任务的出现综次数 N(T)
3. 对于任务集合 T 中的每个任务 t，计算任务的频率：N(t)/ N(T)
4. 对任务集 T 按任务的频率进行升序排序，Sort({N(t)/ N(T)});
5. 计算待消除任务数 n=|T|*∂。
6. 获取待消除任务集合 DelT 为任务集合 T 的前 n 个任务。
7. 对于候选日志的每个轨迹ρ，判断是否含有 DelT 集合中的元素，若有则在ρ执行删除该任务。
8. 重构候选日志信息为 CW。
9. 结束，返回 CW。

通过任务消除算法得到有效的任务集合，这些任务集合所在的流程实例构成候选日志集合。

2. 日志任务时间关系信息抽取与分析

系统业务日志中包括了成千上万的任务执行记录，这些记录包括了任务名称、执行时间等信息。通过对日志中任务的时态关系进行分析，可以挖掘出任务的相关时间信息（如任务平均执行时间、任务之间的等待时间等信息）。

在一些应用领域的业务流程模型中，任务之间的关系需要考虑时间因素，即对于两个任务之间的关系，其在不同的时期（或者在不同的时间间隔）上对实际的业务人员来说，可能具有不同的意义。所以在挖掘过程中，我们需要对任务间的时间关系进行考虑。在 4.2.3 小节中讨论了日志任务时态关系构建，得到日志的任务间隔时间区间信息，如表 4-11 所示，该表列举了日志中所有的任务之间的关系以及其任务时间间隔区间信息。如任务间的关系$A \rightarrow_w B$，出现在流程实例 case1 和 case3 中，任务隔间时间区间为I_1。

通过对日志中的任务间关系进行时间隔间分析，构建任务间时态关系，可以挖掘出任务间具有时间间隔关系的流程模型。图 4-7 为对日志表 4.1 挖掘得到的流程模型，任务间的关系具有时间区间特征。

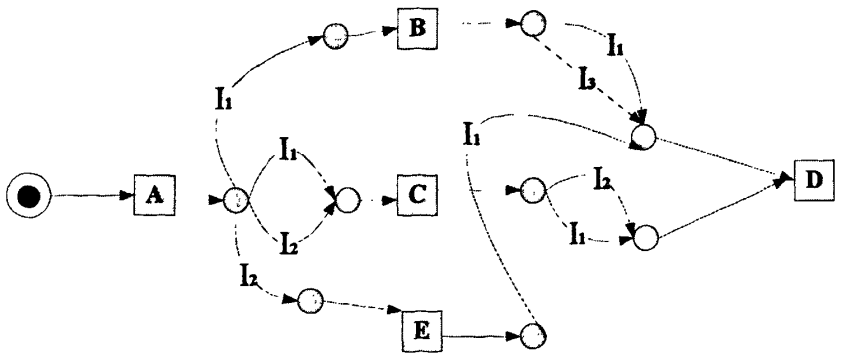

图 4-7 具有时间区间特征的流程模型

上图是一个具有任务时间间隔关系的流程模型，从 4.2.3 小节的讨论可知，$I_1:[32,272]$，$I_2:(272,512]$，$I_3:(512,752]$。任务 A 与任务 B 之间的关系是具有时间间隔区间I_1的特征，也就是说任务 A 与任务 B 的时间间隔是在 30 秒到 272 秒之间。而任务 A 与任务 C 之间的关系是具有时间间隔区间I_1和I_2两种特征，即这种事件具有两种时态特征，一种是时间间隔在 30 秒到 272 秒之间，另一种是在 272 秒到 512 秒之间。

通过在流程挖掘中引入对任务间的时态关系分析以及任务自身的时间分析，可以对任务执行进行预测，为流程性能分析或者（业务上的特殊服务，如电子政务的投诉单，投诉人可以获取投诉单处理的时间范围信息等）提供信息支持。由上图可知，任务的开始执行时间预测可以通过其前置任务的时间间隔以及平均执行时间进行累加得到。

3. 时间日志处理模型

基于上述对日志生成和任务时间关系分析的讨论，我们构建时间日志处理模型，对日志进行预处理和分析，时间日志处理模型有三部分组成，如图 4-8。

- ◆ 候选挖掘日志生成模块
- ◆ 日志任务时间关系分析模块
- ◆ 模型持久化模块

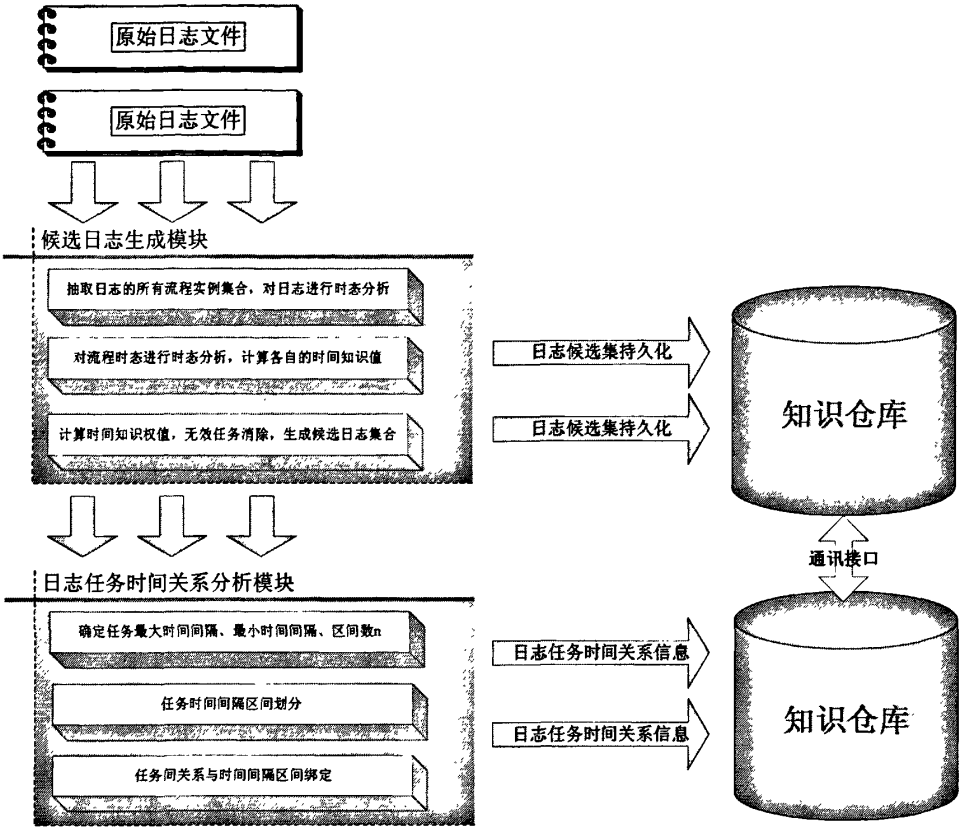

图 4-8 时间日志处理模型

图 4-8 为时间日志处理模型的结构示意图，该模型包括了 3 个主要模块，其中候选日志生成模块的主要功能是对原始日志文件（即业务系统生成的日志文件）进行预处理，包括抽取日志的所有流程实例信息，构成流程实例集合，并对日志进行时态分析等工作，最后将产生的日志候选集信息持久化到知识仓库中，同时也将该日志候选集合作为日志任务时间关系分析模块的输入，为该模块提供分析对象。日志任务时间关系分析模块基于生成的日志候选集，从集合中确定任务的最大时间间隔和最小时间间隔信息，然后对任务时间间隔区间进行划分和分析，最后得到日志任务时间关系信息，并把关系信息持久化到知识仓库中。模型持久化模块主要是将其他模块的信息持久化到知识仓库中去，为后续的流程挖掘工作提供丰富的知识支持。

4.3 小结

本章详细地介绍流程日志信息的时态特征，包括日志时态关系（时间事件、顺序关系、时间区间）等信息。在此基础上，本文提出了日志时态分析模型，首先给出了日志时态信息测量项的相关定义以及测量方法，然后详细地对流程实例和任务间关系进行时态分析，创新地提出“时间知识权值法”为原始日志文件生成日志候选集，有效地解决一些流程增量问题。同时，本章提出的任务时间间隔区间划分法，挖掘任务间关系具有的时间区间特征，使得挖掘的流程模型信息更丰富，并且为流程中任务的时间预测提供帮助。

第 5 章 遗传算法的时态流程挖掘

当前大部分的流程挖掘研究主要集中在控制流挖掘上，具代表性的挖掘算法有 α 算法和 α++算法。本章提出基于改进遗传算法的流程挖掘思想，通过在初始种群中加入启发式规则，并且在变异算子中加入启发式规则，加快算法运算速度，有效地解决 α 算法和 α++算法在某些结构上挖掘的不足。

5.1 遗传算法的概述

遗传算法距今已经有三十多年的发展，并在不少的领域中取得不错的应用效果，本节将对遗传算法的概念及其应用进行介绍。

5.1.1 遗传算法基本概念

1975 年，美国的 Michigan 大学 J.Hollan 教授提出了一种模拟自然进化过程的搜索算法，该算法被称为简单的遗传算法(SGA)[34]。随着遗传算法的三十多年的研究与发展，遗传算法在实际应用领域中取得不错的应用效果，越来越多的学者将遗传算法应用到不同的领域中。

遗传算法是将实际领域问题的解集模拟为一个种群(population)，然后通过对种群中的个体(invidual)按照一定的规则进行编码，其中个体代表领域问题的一个解，在算法的开始，需要初始化种群，然后按照适者生存和优胜劣汰的原理，逐代沉淀出最优的个体，即是问题的最优解。算法使用适应度来衡量种群中的个体的优劣程度，并借助遗传学的遗传算子来对个体进行处理，以便对其他不同类型的个体进行搜索。具体的遗传算子有：组合交叉、变异，通过遗传算子对个体的处理，产生出代表新的解集的种群，新的解集的种群就像自然界的进化一样从原来个体进化得到更优的个体的集合，表示后生代的种群比前代更加适应环境。

遗传算法是具有鲁棒性的全局搜索算法，具有如下特点[36]：

- 遗传算法模拟自然界生物优化劣汰的进化过程，借鉴生物学中的染色体、基因和遗传进化等概念。
- 遗传算法使用适应度函数衡量当前解的质量。

- 遗传算法使用概率搜索技术，并在搜索过程中使用多个点的信息。

5.1.2 遗传算法应用

经过三十多年的发展，遗传算法在很多应用领域中取得不错的应用效果。由于遗传算法是具有很强的鲁棒性算法，所以它不依赖于问题的具体领域（与领域无关），越来越多研究将遗传算法应用到各个领域中解决一些实际问题。下面是遗传算法的主要研究领域。

- 组合优化。组合优化问题面对的最大难题就是问题的解空间（搜索空间）过大，甚至一些组合优化问题在目前计算机的计算能力上是很难枚举得到最优解。较具代表性的组合优化问题有旅行商问题、背包问题、图形划分问题等NP难题，遗传算法能较好地解决这类型的组合优化问题。
- 自动控制[36]。自动控制问题包括航空控制系统优化、智能决策系统、模糊控制规则学习等方面，这些方面都涉及到一定的优化问题，目前很多研究将遗传算法应用到自动控制问题，并取得良好的效果。
- 图像处理。遗传算法在图像处理领域中取得不错的应用效果，尤其图像恢复、图像识别（模型识别）等方面，遗传算法的应用可以降低图像处理过程中的误差，或将误差减少到最小。
- 数据挖掘。数据挖掘一直是近年来计算机研究的热点，最初是应用在从大型数据库中提取隐含的、有潜在应用价值的知识，现在数据挖掘技术取得广泛的应用。数据挖掘主要是基于空间搜索策略的，把数据库看作搜索空间，从搜索空间中挖掘出相关的知识。遗传算法能解决这方面的搜索问题，首先是随机产生一组规则，在数据库这个大的搜索空间上检查规则的价值并进行优化，直到数据库的知识能对该组规则覆盖。

遗传算法在很多实际应用领域中取得应用，如医疗行业的医疗评估系统、智能决策系统等，这些系统通过把遗传算法进行改进，加入与领域相关的规则，使得遗传算法在该领域中取得不错的应用效果。

5.2 遗传算法与流程挖掘

基于遗传算法的流程挖掘思想把流程模型当作一个遗传个体，由若干个遗传个体组成种群，根据适应度函数来衡量个体的优劣。在不断地对遗传个体进行改进优化（交叉、变异等），使得遗传个体不断地向最优解（即最接近实际业务执行过程的模型）收敛。遗传算法应用于流程挖掘主要有以下三方面难点要考虑[37]：

◆ 内部表示与语义。

把遗传算法应用在流程挖掘问题中，需要对日志信息以及日志的关系进行形式化描述，如日志中任务之间的关系等信息，在此本文引入因果矩阵(Causal Matrix)。此外，遗传算法的个体代表问题的一个候选解，所以基于遗传算法的流程挖掘中一个遗传个体代表着一个流程模型，需要把流程模型进行形式化描述，以及把因果矩阵与流程模型之间的转换。

◆ 适应度函数。

适应度函数是遗传算法的大脑，它是衡量遗传个体的优劣的最重要指标，适应度函数的设计与遗传算法的问题求解质量密切相关。把遗传算法应用与流程挖掘问题上，需要根据业务流程的相关背景知识以及流程挖掘问题来设计合理的适应度函数。

◆ 遗传算子

遗传算子是遗传算法的重要组成部分，一般的遗传算子包括选择、交叉、变异。通过遗传算子可以避免一些无用的空间搜索，使得个体（当前解）不断地向问题最优解（或满意解）收敛。在流程挖掘的问题中，如何通过遗传算子为当前的种群（一系列流程模型的集合）生成更有价值的新个体（流程模型）也是需要考虑的。

基于上述的讨论，下面我们介绍因果矩阵及其与流程模型的转化过程。出于对流程模型描述的规范化考虑，本文主要使用 Petri 网对流程模型进行描述，遗传算法挖掘得到的遗传个体最终也会转化成 Petri 网表示。

定义 5.1 (输入函数，输出函数) 设T是任务的有限集合，$a_i \in T$ 为一任务，

那么定义$I(a_i)$为任务a_i的输入函数，$O(a_i)$为任务a_i的输出函数，其中$I(a_i)$为日志中所有以a_i为后续的任务的相关集合，$O(a_i)$为日志中所有以a_i为前序的任务的相关集合。

输入函数和输出函数用于描述任务与其前置和后续任务的关系，表5-1为图3-3的Petri网模型任务输入输出函数信息。

表5-1 图3-3的Petri网模型任务输入输出函数信息

任务名称a_i	输入函数$I(a_i)$	输出函数$O(a_i)$
A	ϕ	$\{\{B,E\},\{C,E\}\}$
B	$\{\{A\}\}$	$\{\{D\}\}$
C	$\{\{A\}\}$	$\{\{D\}\}$
D	$\{\{B,E\},\{C\}\}$	ϕ
E	$\{\{A\}\}$	$\{\{D\}\}$

在输入函数表示的集合中，包括了若干个子集合，子集合之间是表示任务的And-Split关系，子集的内部元素之间是Or-Split关系。在输出函数表示的集合中，子集合之间是表示任务的And-Join关系，子集的内部元素之间是Or-Join关系。

定义5.2(Causal Matrix,因果矩阵[22]) 设三元组$CM=(T,C,I,O)$，其中T是任务的有限集合，$C\subseteq T\times T$是任务间的因果关系，$I\in T\rightarrow P(P(T))$是输入条件函数，$O\in T\rightarrow P(P(T))$是输出条件函数。三元组$CM$满足如下条件：

- ✓ $C=\{(a_1,a_2)\in T\times T \mid a_1\in \cup I(a_2)\}$
- ✓ $C=\{(a_1,a_2)\in T\times T \mid a_2\in \cup O(a_1)\}$
- ✓ $C\cup\{(a_0,a_i)\in T\times T \mid a_0\overset{c}{\bullet}=\phi\wedge\overset{c}{\bullet}a_i=\phi\}$是一个强连通图。

这样的三元组CM被称为因果矩阵。

因果矩阵CM形式化地描述了任务之间的关系，以及任务的前置任务（输入函数产生）和后续任务（输出函数产生）信息。通过因果矩阵可以描述出流程模型的任务关系，所以因果矩阵能转化为基于petri网描述的流程模型。因此基于遗传算法的流程挖掘可通过构造任务的因果关系作为遗传个体，经过不断地改进

优化，最后得到适应度最大的个体（即代表一个因果矩阵），转化为基于 Petri 网描述的流程模型。

定义 5.3 ($\Pi_{PN\to CM}$,Petri 网转因果矩阵[22]) 设 $PN=(P,T,F)$ 是一个 Petri 网，$\Pi_{PN\to CM}(PN)=(A,C,I,O)$ 是 Petri 网 PN 到因果矩阵的映射，其中 $A=T$ 和 $C=\{(t_1,t_2)\in T\times T \mid t_1\bullet\cap\bullet t_2\neq\phi\}$，$\Pi_{PN\to CM}$ 被称为 Petri 网到因果关系的映射。

定义 5.4 ($\Pi^N{}_{CM\to PN}$,因果矩阵转 Petri 网[22]) 设 $CM=(A,C,I,O)$ 是一个因果矩阵，$\Pi^N{}_{CM\to PN}=(P,T,F)$ 是因果矩阵 CM 到 Petri 网的映射，其中：

- ✓ $P=\{i,o\}\cup\{i_{t,s}\mid t\in A\wedge s\in I(t)\}\cup\{o_{t,s}\mid t\in A\wedge s\in O(t)\}$
- ✓ $T=A\cup\{m_{t_1,t_2}\mid(t_1,t_2)\in C\}$
- ✓ $F=\{(i,t)\mid t\in A\wedge \overset{c}{\bullet}t=\phi\}\cup\{(t,o)\mid t\in A\wedge t\overset{c}{\bullet}=\phi\}\cup\{(i_{t,s},t)\mid t\in A\wedge s\in I(t)\}$

  $\cup\{(t,o_{t,s})\mid t\in A\wedge s\in O(t)\}\cup\{(o_{t_1,s},m_{t_1,t_2})\mid(t_1,t_2)\in C\wedge s\in O(t_1)\wedge t_2\in s\}\cup$

  $\{(m_{t_1,t_2},i_{t_2,s})\mid(t_1,t_2)\in C\wedge s\in I(t_2)\wedge t_1\in s\}$

在定义 5.4 中，转化而成的 Petri 网中集合 P,T,F 是由因果矩阵 CM 中任务间的关系产生的。库所集合 P 包括了开始库所和结束库所，以及任务的输入库所和输出库所；变迁集合 T 包括了所有存在因果关系的任务；有向弧 F 是由因果矩阵中任务之间的各种关系构成（包括 And-Split、Or-Split、And-Join、Or-Join 等结构）。通过因果矩阵到 Petri 的映射关系，可以把遗传算法求得的最优个体转化为与其等价的 Petri 网，从而挖掘出最优化的业务流程模型。

5.3 启发式规则

遗传算法具有天生的鲁棒性，这是得益于其全局搜索策略。然而，由于这种全局搜索策略，使得在求解过程需要大量的搜索，造成算法的较大时间复杂度。在遗传算法中加入一些与领域相关的启发式规则，可以有效地为减少算法的搜索空间，使得算法在设定的迭代代数中，加快问题最优解的收敛速度。本文中，在算法中引入启发式规则的主要有如下几方面：

- ✓ 种群初始化。在种群初始化阶段引入启发式规则，生成具有较高适应度函数的个体，而且初始种群对问题解空间的覆盖面广，有助于提高算法的求解质量。
- ✓ 适应度函数。适应度函数是衡量当前个体的优劣程度的数量标准，并对后续运算具有指导性意义，所以适应度函数需要根据当前个体反映的流程模型进行评估，包括适合程度、精确性等因素。
- ✓ 遗传算子。在遗传算子中加入启发式规则，不仅能有效地提高问题的求解效率，而且能克服遗传个体对日志中出现频率较大的流程实例过分拟合，从而忽略了其他流程模式。

5.4 遗传算法流程挖掘

当前很多的业务系统产生的日志具有信息不完整和存在噪声（即存在错误的信息）等特征，这就给流程挖掘工作带有一定的难度。遗传算法具有天生的鲁棒性，能较好地处理噪声问题，把遗传个体作为流程挖掘的问题的解的表示方式，通过对遗传个体的适应度函数计算，评估当前个体的优劣程度，然后通过遗传算子对个体进行选择、交叉、变异等操作，产生新的更优化的种群。下面本节详细地介绍遗传算法的流程挖掘方法。

5.4.1 种群初始化

遗传个体(Genetic Indivival)是遗传算法的基本运算对象，一个遗传个体对应着一个流程模型，表示流程挖掘问题的一个解。种群(Population)是遗传个体(Indivival)的集合，在集合中包含了若干个遗传个体，相当于流程挖掘问题的一组解。对于同一个业务系统日志，由于个体表示因果矩阵，因此在所有的种群中的个体都含有相同的任务集合。该任务集合包含了日志中的所有任务。

遗传算法是从包含若干遗传个体的初始种群开始，以初始种群的各个遗传个体进行遗传运算，进而不断地改进优化个体。所以，初始化的种群影响着遗传算法流程挖掘的质量以及挖掘运算的速度。本文在种群初始化阶段引用启发式规则，使得初始化种群中的个体具有较高的质量，提高遗传算法的运算效率。

定义 5.5 (任务直接依赖) 设$W \in P(T^*)$是一个业务日志，集合T是日志中的任务集合，$t_1,t_2 \in T$且$t_1 \neq t_2$，若任务序列模式t_1t_2或$t_1t_2t_1$在日志中以一定的概率ρ存在，则称任务t_1与t_2存在着直接依赖关系。

定义 5.5 的含义是对于任意的两个不相同的任务，其组成关联任务序列t_1t_2或$t_1t_2t_1$的频繁程度决定这两个任务之间是否存在着直接依赖关系。

定义 5.6 (任务依赖值) 设$W \in P(T^*)$是一个业务日志，集合T是日志中的任务集合，$t_1,t_2 \in T$，$N_f(t_1,t_2)$表示任务序列模式t_1t_2在日志中出现的次数，$N_l(t_1,t_2)$表示任务序列模式$t_1t_2t_1$在日志中出现的次数，$N_l(t_1)$表示任务序列模式t_1t_1在日志中出现的次数，那么定义这两个任务的依赖值为$DV(t_1,t_2)$：

$$DV(t_1,t_2)=\begin{cases}\dfrac{N_f(t_1,t_2)+N_f(t_2,t_1)}{N_f(t_1,t_2)+N_f(t_2,t_1)+1} & \text{当}t_1 \neq t_2\text{且}N_l(t_1,t_2)>0\\[2ex] \dfrac{N_f(t_1,t_2)-N_f(t_2,t_1)}{N_f(t_1,t_2)+N_f(t_2,t_1)+1} & \text{当}t_1 \neq t_2\text{且}N_l(t_1,t_2)=0\\[2ex] \dfrac{N_l(t_1)}{N_l(t_1)+1} & \text{当}t_1=t_2\end{cases}$$

对于在日志中两个任务，它们之间的关系主要体现为：直接相连、短循环和不相关。如两个不相关的任务（这里的不相关是指没有直接相关的关系），其任务间的依赖值为 0；在定义 5.6 中的任务间依赖值公式中，短循环情况比直接相连情况具有更高的优先级，当两个任务之间可能存在短循环关系，即$N_l(t_1,t_2)>0$时，其依赖值随着模式t_1t_2或t_2t_1在日志中出现的总次数的增大而越来越接近 1，而且$DV(t_1,t_2)=DV(t_2,t_1)$；当两个任务之间可能存在着直接相连的关系时，即$N_l(t_1,t_2)=0$，任务t_1,t_2之间的依赖值随着日志中的t_1t_2模式出现的次数与t_2t_1模式出现的次数之间的差异的增大而越来越大。

任务依赖值是参数有序的，即$DV(t_1,t_2)$不一定与$DV(t_2,t_1)$相等，而且在很多情况下是不相等的。任务依赖值$DV(t_1,t_2)$描述了任务t_2对任务t_1的依赖程度，即任务t_1的执行后，然后执行任务t_2的可能性。

定义 5.7 (Genetic Indivial,遗传个体) 设T是日志中的任务集合，$\forall t_i \in T$，$I(t_i)$表示任务t_i的输入函数，$O(t_i)$表示任务t_i的输出函数，定义$GI = \{(t_i, I(t_i), O(t_i)) \mid \forall t_i \in T\}$为遗传个体。

基于上述讨论，我们引入基于任务依赖值的启发式规则来生成初始种群，以提高遗传算法的运算效率，下面为启发式初始种群算法的伪代码：

```
输入：日志信息W
输出：初始种群（遗传个体集合）{GI}
算法过程：
1.提出日志中任务集合T。
2.计算任务间依赖值矩阵DVM。|DVM|=|T|*|T|。
3.获取开始节点集合S={ti| 对于任意tj属于T,有DV(tj,ti)=0}
   获取结束节点集合S={ti| 对于任意tj属于T,有DV(ti,tj)=0}
4.   ran:random      n=|T|    CM为因果矩阵
      for   i=1:n
         for j=1:n {ran=random(1);
               if(ran<DV(ti,tj))CM(l,j)=1; else   CM(l,j)=0
          }
5. 因果矩阵封装在个体 GI 中,返回{GI}。
```

5.4.2 适应度函数

适合度函数是遗传算法的大脑，它是衡量遗传个体对挖掘问题的优劣程度，所有适应度函数的设计对流程挖掘非常重要。

流程挖掘的目标是从任务日志中挖掘出流程模型，该流程模型应该能反映实际的业务执行情况，即能体现日志中的任务。也就是说，通过挖掘过程得到的流程模型应用具有两个特征：完整性和准确性。

完整性体现了挖掘的流程模型对日志中的轨迹的覆盖程度，即一个流程模型具有完整性是指该模型能对日志中的所有事件轨迹解析，日志信息中的所有流程实例都可以看作该流程模式的执行轨迹。而一个流程模型的准确性是指该流程模型不能解析没有在日志中出现的流程实例信息。完整性和准确性对于被挖掘出的流程模型是重要的，因为它们衡量了该流程模型对日志信息的拟合程度，流程模型是否存在一些不合理的结构或者出现多余的任务等。

适应度函数的设计需要考虑挖掘得到的流程模型的完整性和准确性问题，对这两方面进行定量设计，下面介绍流程模型的完整性和准确性要求。

1. 遗传个体的完整性

流程模型的完整性体现了该模型对日志的执行轨迹（流程实例）的覆盖程度，也就是说，挖掘得到的流程模型能对日志中的所有执行轨迹解析，通过该流程模型，可以重新生成日志中的所有流程实例信息。在表 4-1 中的日志信息，图 5-1 的流程模型图能对该日志来说是具有完整性的。该流程模型能解析日志的所有执行轨迹（流程实例 case1、case2、case3、case4、case5）。

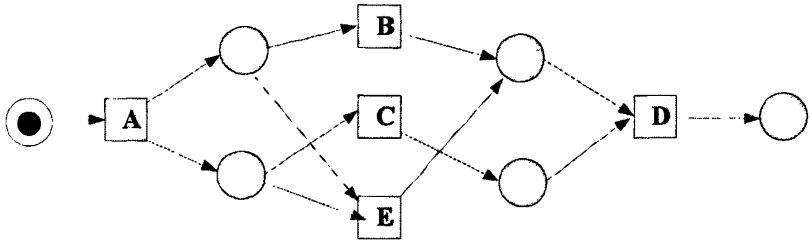

图 5-1 具有完整性的流程模型

由于流程模型的完整性体现了流程模型对日志执行轨迹的覆盖程度，我们很自然地感觉应该用流程模型能解析的流程实例数量除以日志中所有的流程实例数量来衡量流程模型的完整性。然而，这种设计是过于粗糙的，它没有把衡量完整性的粒度细化到具体的任务中，除了需要知道日志中哪些流程实例不能被挖掘得到的流程模型解析之外，还需要知道该流程模型（即最优个体）不能对某一流程实例解析时，模型中的有哪些部分及有多少部分是正确的等。

定义 5.8 (完整性适应度,$PF_{complete}$) 设$W \in P(T^*)$是一个业务日志，CM是一个因果矩阵，那么定义$PF_{complete}: \varpi \times CM \to (-\infty,1]$为完整性适应度函数：

$$PF_{complete}(W,CM) = \frac{allTasks\_Parsed(W,CM) - punishment - adjustment}{numOfTasks\ InLog(W)}$$，其中 $$punishment = \frac{allMissedTokens(W,CM)}{numOfTraces\ InLog(W) - numOfTraces\ MissedToken(W,CM)+1}$$和

$$adjustment = \frac{allExtraTokens(W,CM)}{numOfTraces\ InLog(W) - numOfTraces\ ExtraToken(W,CM)+1}$$。

给定一个业务流程日志和因果矩阵，完整性适应度函数综合考虑了日志中有多少流程实例能被该因果矩阵对应的流程模型解析，有多少任务能被流程模型解析，有多少任务能被流程模型解析等因素。在定义 5.8 的公式中，$allTasks\_Parsed(W,CM)$ 表示日志中能被因果矩阵 CM 解析的任务数量；$numOfTasksInLog(W)$ 表示日志中任务的总数量；$punishment$ 表示对该流程模型存在丢失标记的惩罚，在该公式中 $allMissedTokens(W,CM)$ 表示所有丢失的标记的数量，$numOfTracesMissedToken(W,CM)$ 表示日志中所有丢失标记所在的流程实例的数量，$numOfTracesInLog(W)$ 表示日志中流程实例的数量；$adjustment$ 表示对该流程模型存在多余的标记的调整，在该公式中 $allExtraTokens(W,CM)$ 表示多余的标记的数量，$numOfTracesExtraToken(W,CM)$ 表示日志中所有多余标记所在的流程实例的数量。

从定义 5.8 可以知，完整性适应度函数既考虑能被解析的任务数量，也对流程模型中丢失标记数量的惩罚以及多余标记的调整。通过完整性适应度函数，挖掘得到的因果矩阵可以定量地评估其对应的流程模型的质量。

2. 遗传个体的准确性

流程模型的准确性是指该流程模型不能解析没有在日志中出现的流程实例信息，准确性要求是基于找出有多少任务在日志中是被支持的，有多少任务是不存在日志的任务集合中。由于在挖掘过程中，我们没有反面的例子（日志中反映的都是正面的例子）来指导算法的搜索，所以较难设计出相关的准确性适应度函数。

图 5-1 为基于日志表 4-1 的一个流程模型，该模型能够解析日志中的所有轨迹，也就是说其完整性是相当高的。然而，该流程模型明显就不符合实际的业务应用需要，因为其存在着多余的任务序列，这些任务序列是没有在日志表 4-1 中反映得到的。在此，流程模型的准确性表示挖掘得到的流程模型能解析多余任务序列的程度（数量），由此可见，流程模型的准确性当然是越小越好的，因为由图 5-2 可知，多余的任务序列是有害的。

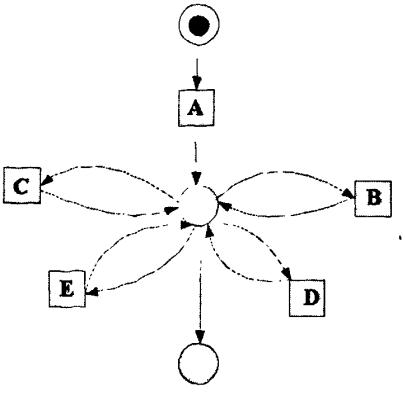

图 5-2 日志表 4-1 对应得到的一个流程模型

定义 5.9 (准确性适应度,$PF_{precise}$[22]) 设$W \in P(T^*)$是一个业务日志，CM是一个因果矩阵，SCM是所有因果矩阵的集合$SCM=\{CM\}$，那么定义$PF_{precise}:\varpi \times CM \times SCM \to (-\infty,1]$为准确性适应度函数：

$$PF_{precise}(W,CM,SCM)=\frac{numOfTasks\_Enabled(W,CM)}{\max\limits_{\forall CM_i \in SCM}\{numOfTasks\_Enabled(W,CM_i)\}}$$

在挖掘过程中，提供一些反面例子来指导算法搜索是不现实的，在此我们利用就绪的可见任务的数量来衡量流程模型的准确性，因为在一般挖掘得到的具有额外任务序列的流程模型跟其有更多的就绪可见任务相关，所以在挖掘算法的运算过程中，用就绪可见任务数量来指导算法的搜索，尽量搜索出具有尽可能少的就绪可见任务，从而可以挖掘出具有较少的多余任务序列的个体（即流程模型）。在定义 5.9 中的准确性公式中，$numOfTasks_Enabled(W,CM)$表示当前的因果矩阵CM就绪的可见任务的数量，$\max\limits_{\forall CM_i \in SCM}\{numOfTasks_Enabled(W,CM_i)\}$表示在因果矩阵集$SCM$中的所有因果矩阵的就绪可见任务的数量。由此可见，流程模型的准确性适应度越小，表示该流程模型的价值也大。

3. 遗传个体适应度

基于遗传个体的完整性和准确性的讨论，下面定义遗传个体的适应度函数，适应度函数需要综合考虑遗传个体的完整性、准确性及其平衡关系。

定义 5.10 (遗传个体适应度,$PF_{fitness}$) 设$W \in P(T^*)$是一个业务日志，GI是一个遗传个体,CM是该遗传个体GI对应的因果矩阵，SCM是所有因果矩阵的集合

$SCM=\{CM\}$，那么定义$PF_{fitness}:\varpi\times CM\times SCM\rightarrow(-\infty,1]$为遗传个体$GI$的适应度函数：

$$PF_{fitness}=PF_{complete}(W,CM)-\upsilon\times PF_{precise}(W,CM,SCM)$$

其中υ为遗传个体准确性的微调因子，$0\leq\upsilon\leq1$。

遗传个体适应度体现了挖掘得到的流程模型应该能解析日志中的所有任务序列，同时应该避免产生一系列在日志中不存在的任务序列。准确性的微调因子υ的设置一般是根据具体的应用需要，衡量精确性的重要程度来决定。

5.4.3 遗传算子

遗传算子是遗传算法的重要的模块，通过遗传算子可以为种群产生新的遗传个体，而且这些新的遗传个体具有更接近问题解的特性。遗传算子包括选择、交叉、变异三方面。我们通过精英选择、交叉、变异为下一代的流程挖掘过程生成新的种群，下面我们介绍本文算法中的遗传算子的机制原理。

1. 选择

选择操作是遗传操作的基础，选择操作采取的策略直接影响交叉操作。在本小节讨论的选择操作包括个体选择和复制两方面，个体选择采用锦标赛策略，而个体复制则采用精英选择策略。个体选择是在当前种群中选择满足一定策略的一个个体，加入到下一代遗传操作的新的种群中，而选择复制是在当前的种群中按一定的比例选择一批个体，直接加入到新的种群中。下面我们介绍个体选择和复制的过程：

- ✓ 个体选择。个体选择采用锦标赛的策略，即随机地从当前种群中选择两个个体，对这两个个体进行适应度比较，以一定的概率λ选择适应度最优的个体加入到新的种群中，以1-λ的概率选择适应度最差的个体加入到新的种群。下面为个体选择算法的伪代码：

输入:当前种群 {GI}、优化率λ
输出:被选择的个体 GI
算法过程:
1. 随机选择两个不相同的个体 GI_1, GI_2
2. $ran \leftarrow Random(1)$
3. if$(ran < \lambda)$ return Max($PF_{fitness}(GI_1), PF_{fitness}(GI_2)$)
   else return Min($PF_{fitness}(GI_1), PF_{fitness}(GI_2)$)

✓ 精英选择。精英选择是指在当前种群中，以精英率*Elistisn*选择具有最优适应度的个体，选择的个数为$|\{GI\}|*Elistism$，然后把这些个体重复到新的种群中。下面为精英选择的伪代码：

输入:当前种群 {GI}、精英率*Elistisn*
输出:精英个体集合{newGI}
算法过程:
1. 种群按适应度降序排序，计算精英个数$n = |\{GI\}|*Elistism$
2. 获取$\{GI\}$中的前 n 个个体，组成精英个体集{newGI}

2. 交叉算子

交叉算子是根据交叉率将种群的两个个体随机地交换某些元素，从而产生新的基因组合。应用于流程挖掘的遗传算法的交叉算子有点不同，由于一个遗传个体对应着一个流程模型，所以交叉操作将首先为两个母体随机地选择一个任务作为一个交叉点，一个任务对应着一个输入集 Input 和一个输出集 OutPut。输入输出集合又包含了若干各子集，同时在 Input 和 OutPut 集中随机选择一个子集作为交换点，接着可以根据预先设定的交叉率对子集进行交换：将交叉点开始的集合到最后一个子集作为交换内容以交叉率进行交换。

在经过两母体的交换操作后，需要检查交换得到的输入集 Input 和输出集 OutPut 是否合理，即不能使得所在个体的因果矩阵存在不一致性问题。

下面为交叉算子的运行机制的伪代码：

输入:两个母体 Parent1,Parent2,交叉率 $CRate$
输出:新生成的两个个体 Child1,Child2
算法过程:
1. Child1← Parent1, Child2← Parent2
2. ran←Random(1)
3. 若 ran≤$CRate$，那么执行步骤 4-10 步骤
4. 在两个个体 Child1,Child2 随机地选择一个任务作为交换点
5. 获取这个交换点的任务 t 的输入集合 Input1(t),Input2(t)
6. 随机选择 Input1(t)、Input2(t)的子集交换点 p1,p2。
7. 构造集合:Input1_Sub1(t)表示集合 Input1(t)中第一个子集到第 p1-1 个子集构成的集合;Input1_Sub2(t)表示集合 Input1(t)中第一个子集到第 p1-1 个子集构成的集合。同样道理，Input2_Sub1(t)和 Input2_Sub2(t)表示类似意义。
8. ran←Random(1),对于集合 Input1_Sub2(t)中的每个子集 sub，根据概率选择以下执行方式:
   1) 若 ran<=1/3, Input1_Sub1(t) = Input1_Sub1(t)∪sub
   2) 若 1/3<ran<=2/3, 随机选择 Input1_Sub1(t)的一个子集 sub11,删除 $sub \cap sub11$中的元素，然后Input1_Sub1(t) = Input1_Sub1(t)∪sub
   3) 若 2/3<ran<=1, 随机选择 Input1_Sub1(t)的一个子集 sub11, sub11= sub∪sub11
9. 对 Input2_Sub2(t)执行类似 Input1_Sub2(t)的过程。
10. Input1(t)← Input1_Sub1(t)　Input2(t)← Input2_Sub1(t)
11. 个体的输入集合交叉完毕，结束
12. 对于个体的输出集合交叉，过程类似步骤 5-10 步骤。
13. 返回 Child1,Child2

在上面的算法伪代码中的步骤 8 可知，交叉算子过程中的子集交换的主要策略有子集合并到输入/输出集合中，或者在一个输入/输出集合中删除某两个子集的交集元素，引入新的子集等。这些策略都可以一定程度上改变交叉个体的特征，增加种群个体的多样化。

3. 变异算子

变异算子在遗传算法框架中能有效地调整种群个体的多样性，其目标是为当前的个体加入一些新的特征。应用于流程挖掘的遗传算法的变异算子与一般的应用不同，由于其个体中以任务作为基本的元素，所以变异算子的作用对象是任务、任务的输入 Input 集和 OutPut 集、以及任务所在的子集。下面为变异算子的运行机制的伪代码:

输入:遗传个体 GI、变异率 $MRat$
输出:变异后新的个体 newGI
算法过程:
1. ran←Random(1)，若 ran≤ $MRate$，执行步骤 2。
2. 对于个体 GI 中的任一个任务 t，获取其对应的 Input 和 Output 集合。
3. 在 Input 集合中随机选取一个子集 sub, ran←Random(1),执行如下操作。
   1）0<=ran<0.25,则在子集中删除该任务。
   2）0.25<=ran<0.5,则在子集中随机地加入任务集合中的任意一个任务。
   3）0.75<=ran<1,则在利用种群初始化过程，利用启发式初始化该个体。
4. 对 Output 集合采用类似步骤 3 的过程。
5. 完毕，返回 newGI。

由上面的伪代码可知，变异算子采用的策略主要是针对任务的输入/输出集的的子集进行变异操作，包括在子集中增加、删除一个任务，或者利用 5.4.1 小节介绍的启发式规则生成新的遗传个体。

5.4.4 时间信息合并

在遗传运算过程中，种群中的个体不断地进化，最优个体逐渐接近挖掘问题的最优解。当遗传算法完成其运算后，得到具有最优适应度的遗传个体，其对应着一个挖掘的到的流程模型。该模型通过与日志时态分析模型得到的任务间时态关系信息结合，构成一个能体现系统真实执行情况的时态流程模型，下面为时间信息合并算法的伪代码：

输入:最优遗传个体 GI、任务间时态关系矩阵 TTM
输出:最优的时态遗传个体 TGI
算法过程:
1. 因果矩阵 CM←CM(GI)。
2. 遍历任务集合 T 中的每个任务 ti,对于每个 ti，遍历任务集合 T 中的每个 tj。
3. 从 TTM 中获取两个任务间的时态关系 R(i, j);
4. 设 Arc(i, j)为任务 i 到任务 j 的有向弧集合,那么遍历 R(i, j)中的每种时间区间关系 I′；
5. $Arc(i,j) = Arc(i,j) \cup I'$，返回步骤 2。
6. 封装 Arc 矩阵到时态遗传个体 TGI。
7. 完毕，返回 TGI。

5.4.5 算法思想

基于上述对遗传算法的讨论，应用于流程挖掘的遗传算法思想主要由以下几方面组成：种群初始化、适应度函数设计、遗传算子运行机制（个体选择、交叉、

变异)。图 5-3 我们对算法的整个流程进行介绍。

算法首先读取由时态分析模型处理得到的日志信息，然后进行这些日志信息进行预处理，读取日志信息中的任务集合，为这些任务集合构建任务依赖值矩阵，这有助于种群的初始化以及个体变异操作。在开始遗传算法运算前，需要设置各类型的参数：优化率、变异率、精英率、交叉率、算法迭代代数、种群大小等。然后通过启发式来生成初始种群，计算种群中个体的适应度函数，然后判断当前运算满足结束条件（如迭代到最大代数或适应度满足结束条件等），若不满足结束条件，则进行遗传算子操作，生成新的种群，然后再进行新的迭代运算，直到满足结束条件为止。

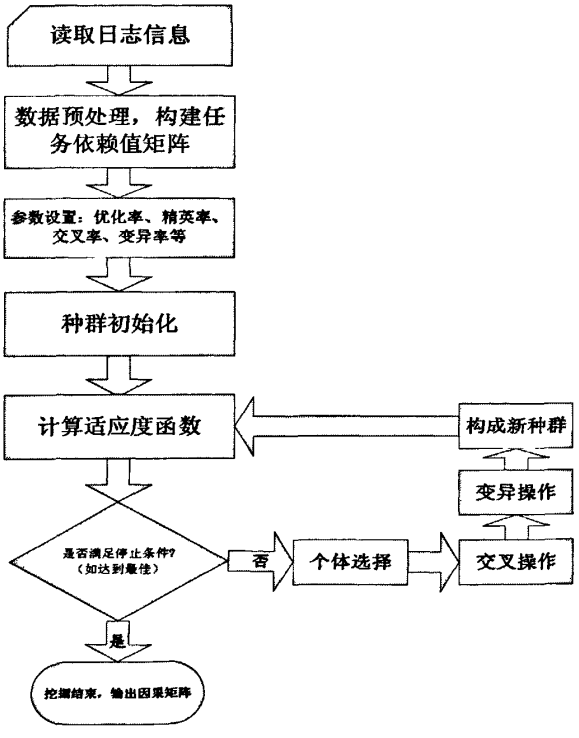

图 5-3 遗传算法流程挖掘思想

5.5 框架的实现与实验分析

本文针对流程挖掘问题提出了基于改进遗传算法的时态流程挖掘框架，并利用 Java 实现了遗传算法的时态流程挖掘的工具，本节主要介绍该框架的实现原理以及对算法的实验分析。

5.5.1 框架实现

本文提出的流程挖掘框架中，首先对原始日志进行分析，利用本文提出的日志时态分析模型从日志中提取出日志的时态信息，并考虑到日志信息对业务流程的时效性问题，通过时间日志处理模块的一系列处理，生成合理的日志的候选集合以及日志的任务时间间隔关系信息，为后续的流程挖掘工作提供支持。

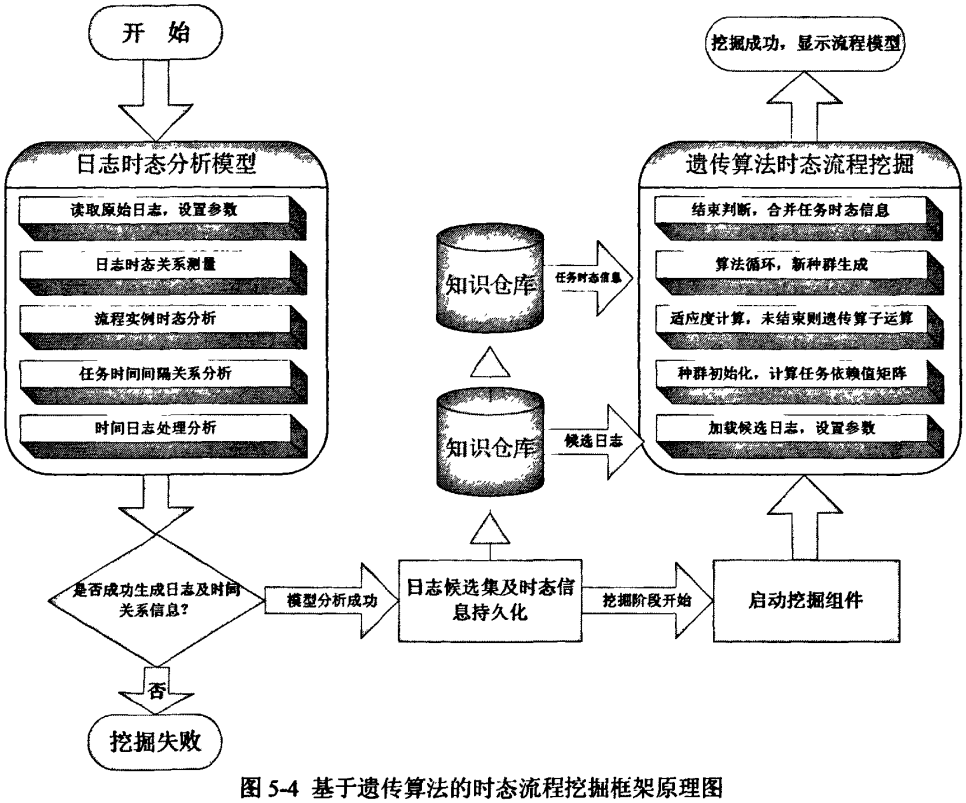

图 5-4 基于遗传算法的时态流程挖掘框架原理图

在流程挖掘运算阶段，通过录入由日志时态分析模型产生的候选日志，然后经过一系列遗传算法运算，得到具有最大适应度的遗传个体，再利用合并技术在该遗传个体中加入时态模型产生的任务时间间隔关系信息，由于遗传个体中包含了因果矩阵的信息，通过因果矩阵到 Petri 网的映射关系，最后重构得到一个 Petri 网描述的流程模型。图 5-4 为整个遗传算法的时态流程挖掘框架的原理图。

基于本文讨论的日志时态分析模型和遗传算法流程挖掘思想，通过 Java 平台实现了基于遗传算法的时态流程挖掘框架，该框架已经成功的应用在实际的业务平台中，如中山大学软件研究所的业务流程集成平台 GBPIP2.2 中，并成功地

利用该平台产生的业务日志信息，重构出能反映业务实际执行过程的流程模型。同时，本文的流程挖掘框架也正在应用到基于多 Agent 的动态业务平台中，并取得不错的应用效果。图 5-5 和图 5-6 为该挖掘框架的效果图。

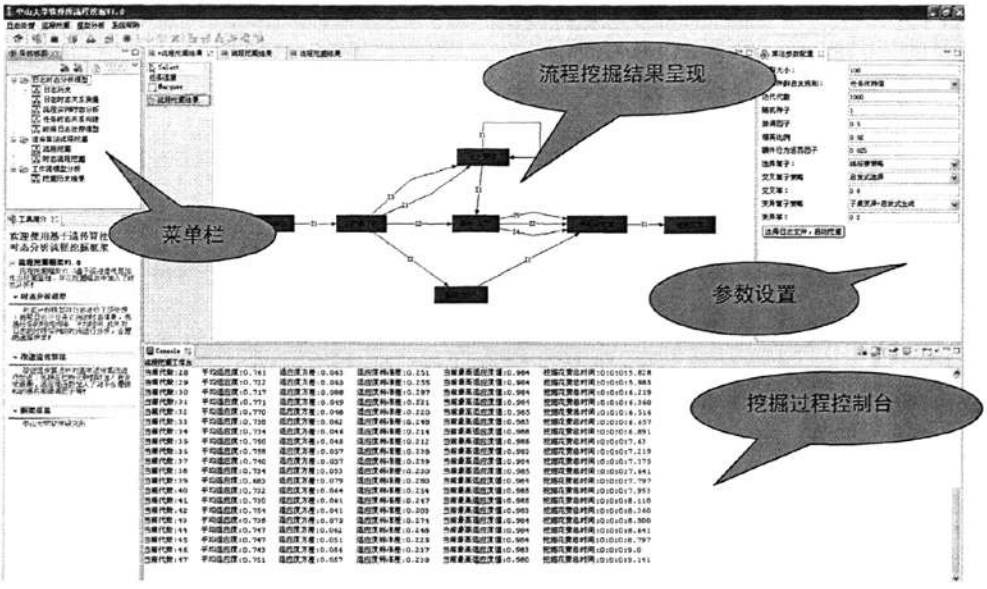

图 5-5 基于遗传算法的时态流程挖掘框架效果图

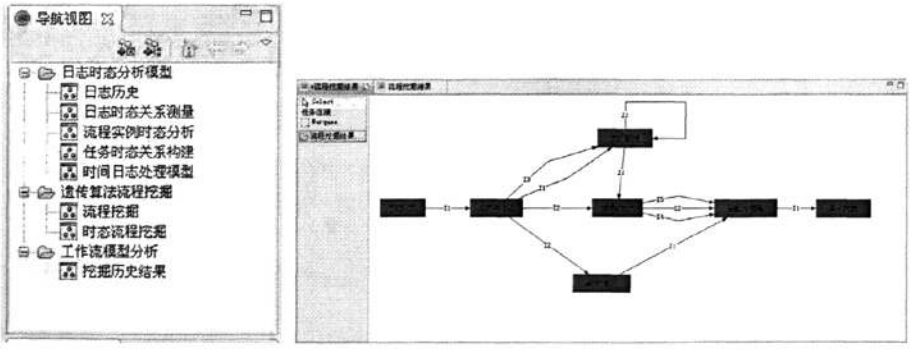

图 5-6 基于遗传算法的时态流程挖掘框架效果图

5.5.2 算法实验分析

在上一小节介绍了框架的整体实现以及运行效果，下面通过几个实际问题的实验来验证本文讨论的模型和算法的有效性和优势。

1. 流程增量问题

由于实际的日志数据较难获取，本文通过模拟软件 CPNTools[38][39][40]生成实验数据。主要步骤为以下两步：

1) 首先设计一个特定的流程模型 PModel，然后利用 CPNTools 模拟流程模型运行，生成 100 个流程实例，这些流程实例的时间跨度为半个年。图 5-7 为 PModel 的 Petri-net 表示。

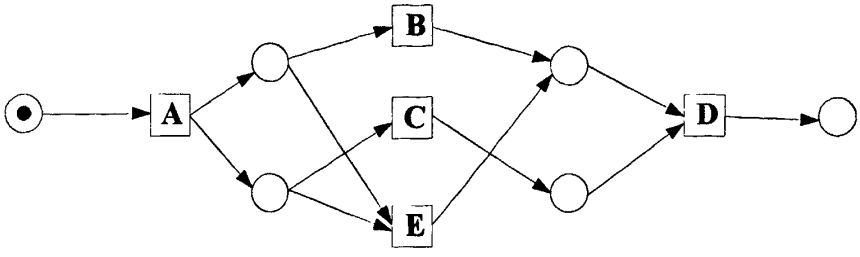

图 5-7 PModel 的 Petri-net 表示

2) 基于步骤一设计的流程模型 PModel，我们模拟真实的业务执行过程，删除任务 E，并新增一个新任务 F，得到新的流程模型 PModel'，然后利用 CPNTools 模拟流程模型运行，生成 80 个流程实例，这些流程实例的时间跨度为 4 个月。图 5-8 为 PModel'的 Petri-net 表示。

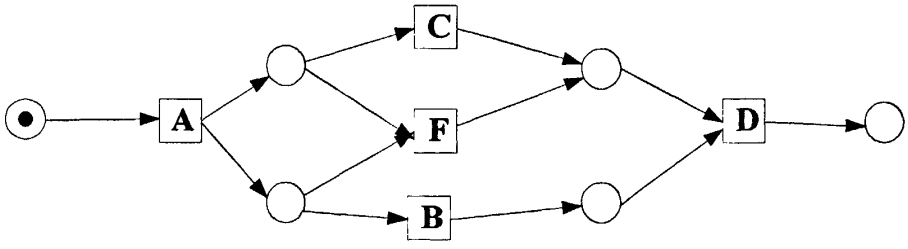

图 5-8 PModel'的 Petri-net 表示

3) 将模型 PModel 和 PModel'产生的流程实例信息汇合成系统的原始日志文件，该日志文件共有 180 个流程实例，时间跨度为 10 个月。

基于产生的原始日志文件，设置时间知识等级 $Gn=2$，那么时间知识跨度 $Sp_{Gn}=\frac{10}{2}=5$个月，划分的时间区间 $I_1=[1,5], I_2=[6,10]$，其对应的时间知识值分别为：0.333,0.667。设置时间等级调整因子 $\omega=2$，微调因子 $\rho=1$，任务消除率为 0.2，那么经过模型分析后得到日志各个任务的频率信息如图 5-9。

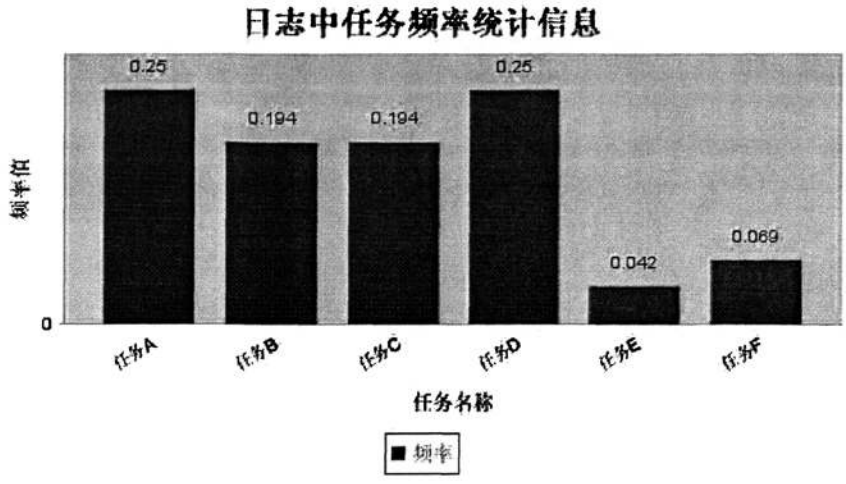

图 5-9 日志中任务频率统计信息

如图 5-9 所示，任务 E 的频率值为最小 0.042，需要消除的任务数量为 $0.2\times6=1.2\approx1$个，那么模型中的任务消除算法会将任务 E 消除。最后挖掘得到的模型如图 5-10，新增的任务 F 虽然其执行频率没有任务 E 高，然后考虑到时间知识因素，任务 F 及与 F 相关的变迁被挖掘出来，而任务 E 由于其时间知识值较低，并且在近期的日志信息中出现较少，因为被消除。本例子证明日志时态分析模型能有效地解决一些流程增益问题。

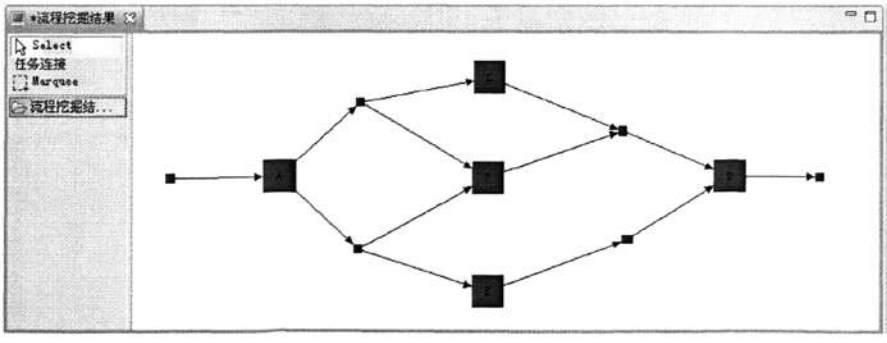

图 5-10 流程挖掘结果

2. 时间流程挖掘

通过本文实现的流程挖掘框架，可以挖掘出具有时态信息的流程模型，下面我们使用本文的流程挖掘框架对一个实际应用的日志文件进行时态流程挖掘，该日志文件包括了 300 个流程实例，7 种任务类型，每个任务类型代表着流程的一个环节，这些信息经过流程日志的时态分析模型处理后，持久化到知识库中。该日志所对应的业务流程为企业的生产管理流程的一个子流程：业务部员工接收客

户的生产订单后，启动生产流程，首先是下达生产单，分派给相关的 3 个部门并行处理，分别是配件生产部门、部件一生产部门、部件 2 生产部门，在生产配配件部门生产完毕后，把配件发给部件 1 生产部门进行组装，待部件 1 和部件 2 生产完毕，将进入装配与包装环节，待该环节结束后，整个生产流程就结束。经过日志时态分析模型分析后，可以从日志文件抽取出日志中任务的以下信息：

任务信息：开始状态、生产单下达、生产配件、部件 1 生产、部件 2 生产、装配与包装、结束状态。各个任务执行间隔的最小时间间隔：2 天；各个任务执行间隔的最大时间间隔：37 天。所以时间间隔差异范围为 37-2=35 天。若我们设置参数 $k=5$，则时间间隔均差 $I\_\mathrm{Re}=\frac{35}{5}=7(\text{天})$，那么可以把生产过程中的各个环节的时间间隔划分为如下 5 种类型的时间区间：

$I_1:[2,9]$，$I_2:(9,16]$，$I_3:(16,23]$，$I_4:(23,30]$，$I_5:(30,37]$

经过遗传算法挖掘后，重构出一个能反映业务系统实际执行的具有时态特征的生产流程模型，如图 5-11 所示。

设置遗传算法参数：种群大小 100；初始种群类型为启发式；最大迭代数量为 100；随机因子为 1；精英因子为 0.02；交叉率为 0.8；变异率为：0.2；适应度微调因子为 1。经过遗传算法 100 次迭代后得到最好的个体的适应度为 0.986。

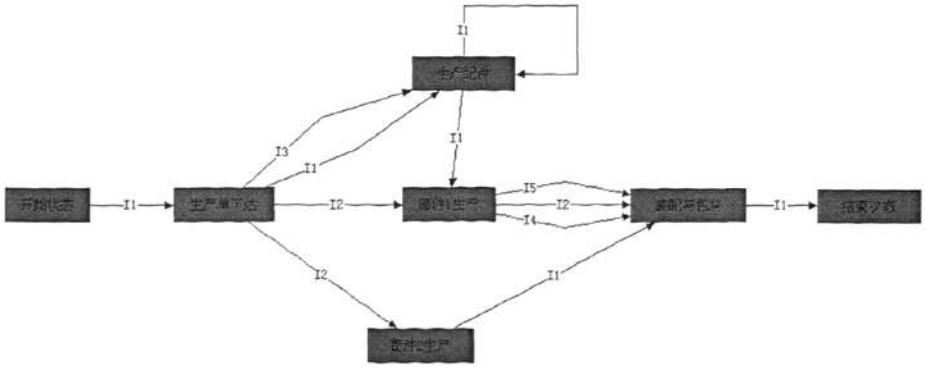

图 5-11 具有时态特征的生产流程模型

图 5-11 展示了挖掘得到的具有时态特征的生产流程，该流程中每个任务之间都建立了以时态区间表示的边，如生产单下达完成后，到生产配件完毕，这两个任务之间的时间间隔为 I_1 或 I_3。这就说明生产单下达后，到生产配件完成操作

的时间间隔主要集中在两个时间段:2到9天、16到23天。任务“部件1生产”到任务“装配与包装”的时间间隔包括三种类型区间：$I_2:(9,16]$，$I_4:(23,30]$，$I_5:(30,37]$，这说明这两个任务之间的生产的流畅度不高，有时候可以在较短时间处理，而有时候则需要很长时候才能完成任务。通过这种时态流程模型，企业决策者或者分析者可以很好地把握到业务流程的瓶颈在哪里，为业务流程的改进提供知识支持。

此外，挖掘得到的时态流程提供任务时间预测的帮助，如当前的流程实例的状态为：部件1生产完毕，那么可以对距离完成该订单的生产时间进行预测：

最短时间:$2+9=11$天 $<$ min_ *FinishTime* $\leq 9+16=25$天

最长时间：:$2+30=32$天 $<$ min_ *FinishTime* $\leq 9+37=46$天

3. 算法性能实验

下面我们通过几方面对本框架的遗传算法进行性能分析，具体包括处理噪声日志、具有并行、选择、循环结构的日志等，并与当前流程挖掘研究的较具代表性的挖掘算法α算法和α++算法进行对比。

1) 噪声日志挖掘

由于在研究阶段难以获取实际应用中的噪声日志，即使能获取噪声日志，也很难衡量日志存在的噪声的量，所以本实验通过使用模拟软件CPNTools来生成特定任务数量的日志文件，然后采用三种规则来破坏日志中的信息，生成存在一定噪声比例的日志文件。本实例使用CPNTools生成任务数量为12的日志，然后通过对一个日志轨迹进行以下四种类型的操作来产生噪声日志：删除头序列中的子任务序列（头序列表示日志轨迹中的前1/3的任务序列）、删除中间部分序列中的子任务序列（中间部分表示日志轨迹等分3份后，中间部分的任务序列）、交换两个任务、删除尾序列中的子任务序列（尾任务序列表示日志轨迹等分3份后，最后一份的任务序列）。如在表4-1日志中的case1日志轨迹，其执行序列为：$A\rightarrow B\rightarrow C\rightarrow D$，那么删除开始任务，则变为$S\rightarrow B\rightarrow C\rightarrow D$，其中$S$为特殊的标识，标记被删去的开始点；删除中间部分序列后，则变为$A\rightarrow C\rightarrow D$；交换两个任务后，则变为$A\rightarrow C\rightarrow B\rightarrow D$；删除结束任务后，则变为$A\rightarrow B\rightarrow C$。

根据CPNTools生成的12个任务的日志信息，采用删除头序列中的子任务

序列操作，得到带有 5%噪声的日志，设置遗传算法参数：种群大小 100；初始种群类型为启发式；最大迭代数量为 100；随机因子为 1；精英因子为 0.02；交叉率为 0.8；变异率为：0.2；适应度微调因子为 1。经过遗传算法 100 次迭代后得到最好的个体的适应度为 0.98612。挖掘得到的流程模型如图 5-12 所示，其中图。

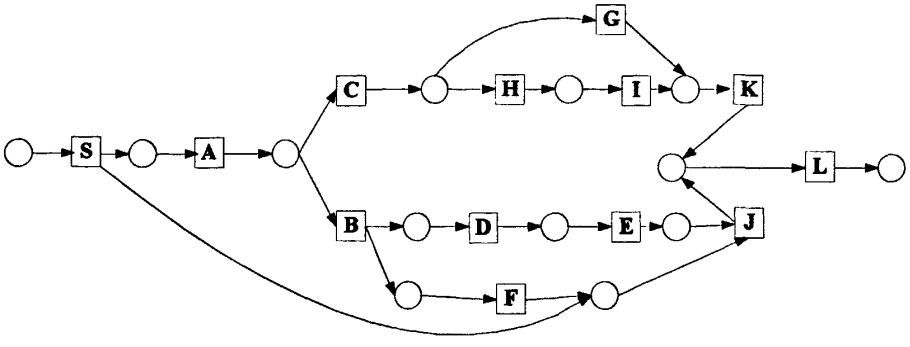

图 5-12 带有 5%的噪声（删除头任务序列）的日志对应的 Petri-net

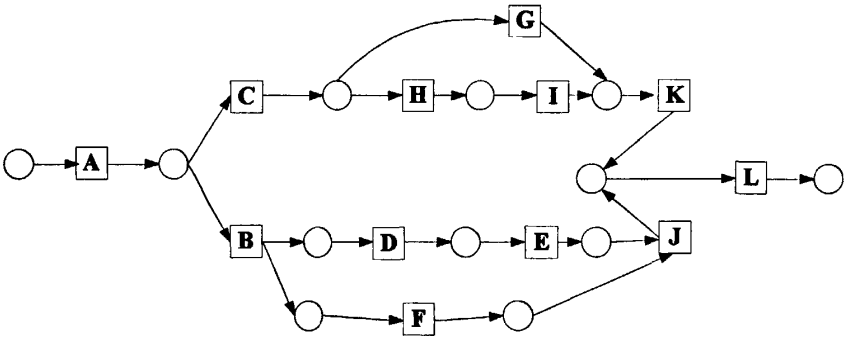

图 5-13 原始流程模型

图 5-12 为本文框架挖掘对带有 5%噪声的日志进行挖掘得到的 Petri-net 模型，该模型与原设计模型一致，表示本文的算法对含有噪声的日志是有效的。图 5-14 为使用 α++算法挖掘带有 5%噪声的日志得到的流程模型。

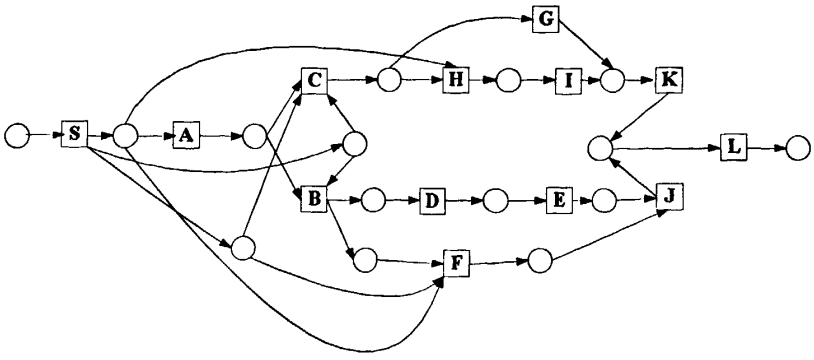

图 5-14 使用 α++挖掘得到的流程模型

对比图 5-12、5-13、5-14 可知，遗传算法对噪声日志具有很好的挖掘能力，能挖掘出与原始流程基本上一致的流程模型，而 α++算法具有较差的噪声处理能力，挖掘得到的流程模型与原始模型偏差较大，加入了很多活动到库所，库所到任务的弧。同样，基于 CPNTools 生成的 12 个任务数量的日志文件，我们采用其他 3 种生成噪声的方式（删除中间任务序列、删除尾部任务序列、交换任务）产生 5%噪声的日志，并进行流程挖掘实验，如表 5-2 为其他类型噪声挖掘对比。

表 5-2 其他噪声类型的 5%噪声日志挖掘对比

噪声类型(5%)	本文框架	α++算法
删除中间任务序列	√	×
删除尾部任务序列	√	×
交换两个任务	√	×

对于同样是含噪率为 5%的噪声日志，遗传算法能很好地处理其他噪声类型的日志挖掘，而 α++算法则不能挖掘出与原始日志一致的流程模型。虽然随着含噪率的上升，本文算法对噪声的鲁棒性会降低，但是挖掘相同类型的噪声问题，挖掘效果会比 α 算法和 α++算法的效果要明显好。

2) 无噪声日志挖掘

通过模拟软件 CPNTools 设计具有并发、长度为 1 的循环的流程模型，然后模拟现实生成 50 个流程实例的日志，通过本文的挖掘框架，设置参数：种群大小 100；初始种群类型为启发式；最大迭代数量为 100；随机因子为 1；精英因子为 0.02；交叉率为 0.8；变异率为：0.2；适应度微调因子为 1，最优个体的适应度值是 0.99238。挖掘得到的流程模型如图 5-15。

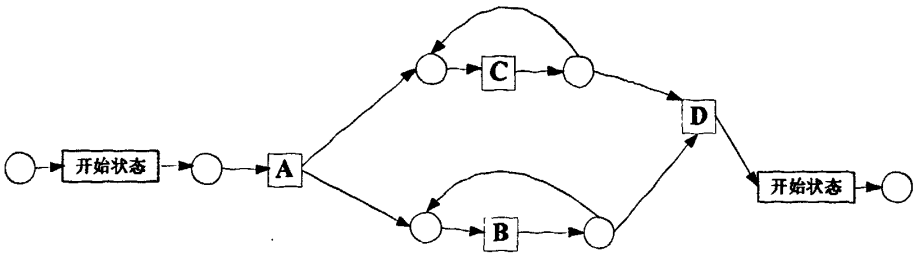

图 5-15 并发、长度为 1 的循环结构流程模型

对同一日志文件，使用 α 算法进行流程挖掘，挖掘得到的结果如图 5-16。α 算法不能挖掘得到两个长度为 1 的循环，任务 A、C、B、D 之间的关系挖掘不出来。本文的遗传算法框架能挖掘出长度为 1 的短循环、并发等结构，这些是 α 算法不能挖掘得到的。

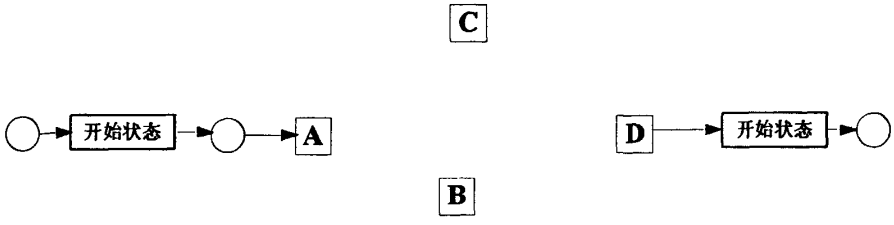

图 5-16 α 算法挖掘得到的结果

5.6 小结

当前大部分的流程挖掘研究主要集中在控制流挖掘上。本章提出改进遗传算法的时态流程挖掘思想，通过在初始种群中加入启发式规则，并且在变异算子中加入启发式规则，加快算法运算速度，有效地解决 α 算法和 α++算法在某些结构上挖掘的不足。通过实验进行检验分析，并与其他算法进行对比，证明算法的有效性，能克服 α 算法和 α++算法在某些结构的不足和限制，并有效地解决一些流程增量问题，挖掘得到合理的时态流程模型。

第6章 总结和展望

6.1 总结

本文介绍了流程挖掘在过程管理方面的应用，指出当前大部分的研究缺乏对日志时间因素的考虑，并且没有考虑流程增量问题，这些在业务需求变动大和对分析要求高的业务系统都是不合理的。在详细地对流程挖掘的相关知识进行介绍后，本文提出日志的时态分析模型，该模型首先对日志中各个任务进行时间分析，然后提出“时间知识权值法”对日志进行处理，有效地解决日志知识增量问题，此外该模型通过为任务构建时态关系，提出“任务时间间隔区间划分法”来对任务进行时态分析，抽取出有价值的任务时态信息，提高流程挖掘的准确性和挖掘结果的参考价值。在此基础上，我们提出改进遗传算法的时态流程挖掘框架，该算法在初始种群时引入了启发式规则，缩小了搜索空间，通过在适应度函数加入微调因子来提高流程挖掘的准确性，并且在变异算子中加入启发式规则，加快算法运算速度。在得到最优化个体后，使用合并技术构建时态流程模型。最后本文利用 Java 语言实现本文讨论的框架，并成功应用到实际的业务流程平台中。总的来说，本文具有如下的创新点：

1) 提出流程日志的时态分析模型，在该模型中创新地提出“时间知识权值法”来衡量日志中流程实例的时效性，并提出任务消除算法消除日志中无效的任务，从而有效地解决一些流程增量的问题。
2) 在流程挖掘中加入了对时间因素的考虑，提出任务时间间隔区间划分法，使得挖掘得到的流程模型含有时间区间信息，为流程改进、优化及改进提供有效的知识支持。
3) 在遗传算法挖掘过程中，在初始种群时引入了启发式规则，缩小了搜索空间，在个体变异算子中加入启发式规则，解决了挖掘循环长度为 1 的循环问题，并加快算法运算速度。

6.2 下一步工作

本文的研究在下列几方面尚有一些不足及提升空间：

1) 本文提出的日志时态分析模型在分析之前需要配置一些模型参数，而这些参数的设置会影响模型产生的知识的质量，后续的工作需要在该模型中引入参数学习方法，即利用日志的时态分析历史记录信息，来自动生成合理的参数。

2) 时态分析模型中的流程实例时态分析需要考虑流程模型的基本结构，如选择结构等，因为这些结构会降低任务在日志中出现的频率，可能会降低流程挖掘的质量。

3) 遗传算法面临着最大的瓶颈：运算速度较慢，下一步的工作将从遗传算法的遗传算子进行改进，研究更多的启发式规则，来提供算法的运算速度。

参考文献

[1] Liangzhao Zeng, Boualem Benatallah, Hui Lei,et al. Flexible Composition of Enterprise Web Services. Electron. Mark .Int. J.Electron. Commer. Bus. Media 13(2) (2003).

[2] Jonathan E.Cook. Process Discoverying, and Validation Through Event-data Analysis. Boulder, University of Colorado, Technical Report: CU-CS-817-96, 1996-11.

[3] 宋炜,高佃芳,刘强.复杂工作流结构的研究.软件学报.2008, 19:104-123, supplement.

[4] W.M.P Van der Aalst, de Medeiros AKA, and Weijters AJMM. Process mining:a research agenda. Computers in Industry, 2004, 53(3):231-244.

[5] Wen Lijie, Wang Jianmin, and Sun Jiaguang. Detecting Implicit Dependencies Between Tasks from Event Logs. Berlin, Germany: Springer Verlag, 2006.

[6] Agrawal R, Gunopulos D, and Leymann F. Mining Process Models from Workflow Logs.Proceedings of the 6^{th} Internationnal Conference on Extending Database Technology.1998:469-483.

[7] W.M.P Van der Aalst, Weijters A J M M, and Maruster L. Workflow Mining: Discovering Process Models from Event Logs.IEEE Transactions on knowledge and Data Engineering, 2004, 9(12):369-378.

[8] B.F. van Dongen and W.M.P Van der Aalst. EMiT: A process mining tool. In J. Cortadelle and W. Reisig, editors, International Conference on Applications and Theory of Petri Nets(ATPN 2004),volume 3099 of Lecture Notes in Computer Science, pages 545-463.Springer-Verlag, Berlin,2004.

[9] W.M.P Van der Aalst, Reuers Ha, Weu Ters AJ MM, et al.Business process mining: an industrial application.Information Systems, 2007, 32(5):713-732.

[10] 赵卫东,范力.工作流挖掘研究的现状与发展. 计算机集成制造系统.2008, 14(12):2289-2296.

[11] Rouached M, Gaaloul W, W.M.P Van der Aalst, et al.Web service mining and verification of properties: an approach based on event calculus. On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE. Springer Berlin / Heidelberg. 2006, Vol. 4275:408-425.

[12] 李小白,陈攸跻,张赛红.工作流挖掘技术研究综述.电脑知识与技术.2009.5(6):1283-1286.

[13] Jonathan E.Cook, and Alexander L. Wolf. Software process validation: Quantitatively measuring the correspondence of a process to a model. ACM Transactions on Software Engineering and Methodology, 1999, 8(2):147-176.

[14] A.J.M.M. Weijters, and W.M.P Van der Aalst.Rediscovering workflow models from event-based data using little thumb.Integrated Computer-Aided Engineering, vol.10, 151-162, 2003.

[15] 范玉顺，罗海滨，林慧苹，赵虹.工作流管理技术基础.北京:清华大学出版社,2001.

[16] W.M.P Van der Aalst , and Kees van Hee. 工作流管理---模型、方法和系统.北京:清华大学出版社,2004.

[17] Jonathan E.Cook, and Alexander L. Wolf. Automating process discovery through event-data analysis.Proceedings of the 17th international conference on Software engineering, p.73-82, 24-28, 1995.

[18] W.M.P Van der Aalst, J Desel, and A Oberweis, Editors. Bussiness Process Management: Model, Techniques, and Empirical Studies, Volume 1806 of Lecture Notes in Computer Science.Berlin:Springer-Verlag,2000.

[19] Hammori M, Herbst J, and Kleiner N. Interactive workflow mining.In:Proc. Of the 2nd Int'l Conf.on business Process Management.2004.211-226.

[20] Hammori M, Herbst J, and Kleiner N. Interactive workflow mining requirements, concepts and implementations. Data and Knowledge Engineering.2006.56:41-63.

[21] J.Herbst and D.Karagiannis. Workflow Mining with InWoLve.Computers in Industry, 53(3):245-264, 2004.

[22] A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P Van der Aalst. Genetic Process Mining: A Basic Approach and Its Challenges. In Business Process Management 2005 Workshops, volume 3812 of Lecture Notes in Computer Science,page 203-215.Springer Verlag.2006.

[23] L. Wen, W.M.P Van der Aalst, J. Wang, and J.Sun. Mining Process Models with Non-Free Choice Constructs,Data Min.Knowl.Discov.,15-2(2007),pp.145-180.

[24] M.Berlingerio, F.Pinelli, M.Nanni, and F.Giannotti. Temporal mining for interactive workflow data analysis. In 15th ACM SIGKDD Int.Conf.on knowledge Discovery and Data Mining (KDD'09), pages 109-118, 2009.

[25] Meilin, S., Guangxin, Y., Yong, X. and Shangguang, W. (1998): Workflow Management Systems: A survery. In Proceedings of IEEE International Conference on Communication Technology. Beijing, China.

[26] 陈亮, 石美红.基于工作流挖掘的业务过程持续改进研究.企业管理与信息化.2008, 37(11), 17-22.

[27] W.M.P Van der Aalst, and B.F.van Dongen. Workflow Mining:a Survey of Issues and Approaches.Working Paper 74, Beta:Research School for Operations Management and Logistics.2002.

[28] Jonathan E.Cook and Alexander L. Wolf. Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology, 7(3):1998.215-249.

[29] Rakesh, Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs, the Sixth International Conference on Extending Database Technology, 1998.

[30] W.M.P Van der Aalst, and Kees van Hee. Workflow management: models, methods, and systems. Cambrige, Massachusetts: The MIT Press, 2002:118-133.

[31] W.M.P Van der Aalst, and B.F. van Dongen. Discovering Workflow Performance Models from Timed Logs.in:Y.Han,S.Tai,D.Wikarski(Eds), International Conference on Engineering and Deployment of Cooperative Information Systems(EDCIS2002),Lecture Notes in Computer Science,vol.2480,Springer-Verlag.Berlin,2002,pp.45-63.

[32] Lin C, and Y Qu. Temporal inference of workflow systems based on time Petri-net: Quantitative and qualitative analysis. International Journal of Intelligent Systems, 2004, 19(5):417-442.

[33] 宋贤钧,王炳鹏,郭佳.基于 WF-net 的工作流建模技术及应用.计算机科学.2006, 33(4):134-136.

[34] 李敏强,寇纪淞,林丹等.遗传算法的基本理论与应用.北京:科技出版社,2004.

[35] 加拉卜,特拉韦尔索著,姜云飞 译.自动规划:理论和实践.清华大学出版社.2008-03.264-273.

[36] 杜文丽,原亮.遗传算法的特点及应用领域研究.科技信息（科学教研）.2008.10:31-32.

[37] A.k. Alves de Medeiros. Genetic process mining.Ph.D. Dissertation, Eindhoven Technical University,2006.

[38] Ratzer A,Wells L, Laursen M, et al. CPN tools for editing,simulating, and analysing

coloured Petri nets.Lect Notes Comput Sc 2003;2679:450-62.

[39] CPN Tools:http://wiki.daimi.au.dk/cpntools/.

[40] Alves de Medeiros AK, Guenther CW. Process mining: using cpn tools to create test logs for mining algorithms. In: Jensen K(ed) Proceedings of the sixth workshop on the practical use of coloured Petri net and CPN tools. 2005-10.Vol. 576, pp 177-190.

致 谢

本文行文至此，亦是我研究生生涯即将划上圆满句号之时。在这里，我要衷心感谢尊敬的导师李磊教授。感谢李老师在这两年的悉心关怀和指导，在研究生的学习阶段，他无私地向我们传授他的学术研究方法，教导我们如何学会解决学习、工作中遇到的困难，鞭策我们通过端正的科研态度、合理的途径来获取学术成果。李老师和蔼可亲、风趣幽默、学识广博和态度严谨的治学态度一直感染着实验室的每一个人，鞭策着我们不断地前进。感谢姜云飞教授，姜老师性格和蔼、为人谦虚、知识渊博，有着儒雅的学者风范，一直是我学习的榜样。

感谢软件工作组的各位师兄师姐。感谢陈冰川师兄，他在学术上引领着我们不断的前进。感谢夏兰亭师兄，他不但从来不吝与我分享学术思想上的火花，而且教会了我很多为人处事的方法。感谢黄泽龙、林春泉、刘生寒等同学，在这两年来，我们一起学习，一起搞平台研究，结下了深厚的友谊。

感谢我的室友，在这两年的研究生生涯中，大家相互勉励。他们带给我的一切快乐和帮助我将永远珍惜和铭记，他们是梁焯佳、廖定锋、罗达。

感谢答辩委员会的老师，在百忙中抽出时间审阅我的论文。

最后，向其他关心、帮助过我的老师、同学和各位朋友表示深深的谢意。