复杂系统建模理论与方法

陈森发　编著

东南大学出版社

内 容 提 要

本书较系统地介绍了复杂系统的理念、复杂系统建模的基本理论方法和途径，初步构建了复杂系统建模的理论体系。书中将复杂系统建模方法划分成基于智能技术的复杂系统建模、离散事件动态系统建模、定性建模、非线性动力学系统建模、其他复杂系统建模方法五大类并对其进行了较深入的研究。本书既展示了作者及其研究生近几年的研究成果，又涵盖了国内外同行的最新资料，有一定的深度，注重理论联系实际。

本书既可作为高等院校理工科系统工程、管理科学与工程、自动控制专业博士研究生、硕士研究生的教学用书或参考书，也可供从事系统理论、系统分析、系统调度、应用数学、智能控制的研究人员以及大专院校理工科系统工程、管理科学与工程、自动控制专业高年级学生参考。

图书在版编目(CIP)数据

复杂系统建模理论与方法/陈森发编著. —南京：东南大学出版社，2005.4

ISBN 7-81089-423-4

Ⅰ.复… Ⅱ.陈… Ⅲ.系统建模 Ⅳ.N945.12

中国版本图书馆 CIP 数据核字(2004)第 104303 号

东南大学出版社出版发行

(南京四牌楼 2 号 邮编 210096)

出版人：宋增民

江苏省新华书店经销 扬中市印刷有限公司印刷

开本：787 mm×1092 mm 1/16 印张：16 字数：410 千字

2005 年 4 月第 1 版 2005 年 4 月第 1 次印刷

印数：1～3000 定价：39.00 元

(凡因印装质量问题，可直接向发行部调换。电话：025-83795801)

前　言

随着社会和经济的迅速发展，系统科学工作者面临的研究对象也越来越复杂，迫切需要复杂系统建模的理论和方法作指导。1990 年，钱学森先生提出用综合集成讨论厅建模法来处理复杂巨系统问题，独树一帜，但这仅仅是万里长征的第一步。复杂系统的特点是：高阶次、多回路、非线性、多时标、层次性、开放性、不确定性、病态结构等。以什么方式来进行综合集成，仍然是一个有待于进一步探讨的难题。对此，国内外的科学工作者，在不同的领域默默地做着不懈的努力，从不同的角度提出了一些方法。从表面上看，这些方法似乎彼此关系不密切。受苗东升教授关于系统整体涌现性的论述的启迪[1]，并以此为线索，作者把不同的关于复杂系统建模的理论和方法联系起来，并引用近几年国内外学者的最新研究成果，加以整理，编写出这本书，希冀为建立处理复杂系统的理论贡献出自己绵薄之力。

本书共 6 章，第 1 章介绍系统的有关概念、复杂系统建模的理念和系统建模的一般方法。第 2 章到第 6 章，介绍复杂系统建模的理论和不同建模方法。本书第 4 章由张文红编写，其余由陈森发编写，全书最后由陈森发统一定稿。

书稿的整理，得到东南大学系统工程研究所博士生胡晓龙、何宽、亓霞、郜振华、吴国富、周振国、韩莹、唐磊、喻瑛、张毅华、孙燕、陈淑燕、陈胜、黄鹃的大力协助，得到东南大学系统工程研究所硕士生林刚、黄怀友、狄娟、华冬冬、周静、丁丽、王红霞、马小勇、顾杰的大力协助，书中图的绘制得到蒋小瑛女士的大力协助，书的出版得到东南大学出版社社长宋增民教授和副编审戴丽女士的大力支持。借此机会，作者对他们一并表示衷心的感谢。

复杂系统建模的理论和方法的研究是 21 世纪国内外一个崭新的研究领域，很多内容都处于国际前沿，在此，作者仅抛砖引玉。由于时间仓促，书中倘有不妥或疏虞，恳请读者不吝赐教。

陈森发　张文红

2005 年 2 月 18 日

目　　录

1 绪论

1.1 概述

系统在自然界和人类社会中普遍存在：太阳系是一个系统，国家是一个系统，学校是一个系统，工厂是一个系统，家庭是一个系统，人体是一个系统。系统这一术语在报纸、杂志、领导的讲话中日益增多。最近几年，在学术刊物和书籍中，出现大系统、巨系统、复杂系统、复杂巨系统等名词，可见复杂系统理念及对其进行建模的理论和方法，已开始受到人们的重视，这方面的研究正逐步深入。

复杂系统的研究是1999年4月2日由美国《科学》(Science)杂志出版的《复杂系统》专辑而兴起的，它的出现与复杂性问题研究有密切关系。复杂性问题的提出起源于奥地利，1928年贝塔朗菲在他写的《生物有机体系统》论文中首次提出复杂性的问题[2]。在此之前的若干年，怀特海(Alfred North Whitehead)在他的《有机体的哲学》论文中，也曾提出类似的观点。此后的20年，在这方面做出较大贡献的有：麦卡洛克(McCulloch)和皮茨(Pitts)的神经网络，冯·诺伊曼的元胞自动机和复杂性，维纳的控制论。中国科学院外籍院士、诺贝尔经济学奖获得者H. A. Simon(中文名：司马贺)1969年在他的著作《人工科学》的书中，用整整一章的内容讨论了复杂性的构造[3]。1984年，美国新墨西哥州洛斯阿拉莫斯国家实验室，汇聚了一批著名的科学家，在距它约55 km的圣菲，建立了圣菲研究所(Senta Fe Institute，简称SFI)。从20世纪90年代开始，SFI致力于复杂性科学的研究工作。

在20世纪80年代初，钱学森先生发表了《系统科学、思维科学与人体科学》的论文。1990年，他及其合作者在《自然》杂志又发表了《一个科学新领域——开放的复杂巨系统及其方法论》的论文。1991年1月，在中国科学院张焘等同志的倡议下，得到一批老科学家的热情支持，由周光召院长主持，举行了“复杂性科学学术讨论会”，引起许多研究者的兴趣。1994年9月，在北京香山又举行题为“开放复杂巨系统方法论”的学术会议。1997年1月，再次在北京香山召开题为“开放复杂巨系统的理论与实践”的讨论会。这3次会议有力地推动了复杂性科学的研究。

到目前为止，中国已经有一些单位专门组织力量，对复杂系统进行探索。例如，中国科学院下设的复杂系统工程学开放研究实验室，进行复杂系统建模理论与方法论研究。上海大学系统分析与集成专业，以复杂系统为对象开展研究。复旦大学智能信息处理开放实验室，以复杂系统的信息处理为核心研究内容，积极地进行探索。

复杂系统和简单系统的主要区别是什么？把多位学者的看法综合起来，就是：高阶次、多回路、非线性、多时标、层次性、开放性、不确定性、病态结构等。本书认为复杂系统与简单系统的本质区别应是复杂系统具有涌现性，并以此为核心内容贯穿全书。

1.2 系统及有关概念

1.2.1 系统

系统是指相互联系又相互作用的元素之间的有机组合。这里所指的系统是广义的，它包含所有的工程系统及非工程系统。电气、机械、机电、声学系统都是工程系统，而经济、交通、管理、生物系统都是非工程系统。

任何系统都存在3个方面需要研究：实体、属性、活动。

实体：组成系统的具体对象元素。

属性：实体的特征（状态和参数）。

活动：对象随时间推移而发生的状态变化。

对于工厂系统而言（图1.1），系统的实体是工厂的部门、订单和产品；其属性是部门类型、订单数量、各部门的设备数量；其活动则是各个部门的计划、采购、装配和销售过程。

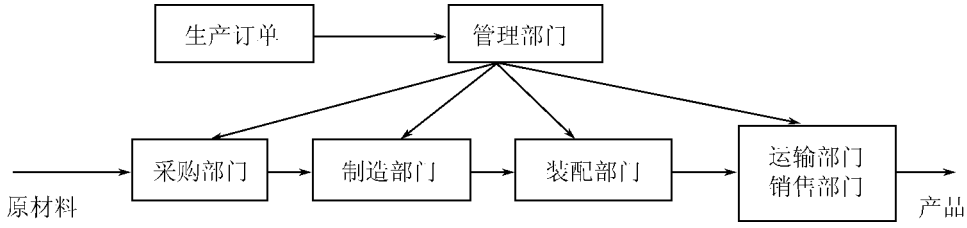


图1.1 工厂系统

由于组成系统的实体之间相互作用而引起实体属性的变化，通常用“状态”的概念来描述。研究系统就是研究系统状态的改变，即系统的演化。

研究系统除了研究系统的实体、属性、活动外，还应研究系统的环境。环境是指对系统的活动结果产生影响的外界因素。自然界的一切事物都存在着相互联系和相互影响，而系统是在外界因素不断变化的环境中演化发展的。因此，环境因素是研究系统时必须考虑的，而对开放的非工程系统更是如此。

1.2.2 系统的分类

由前面论述的定义可知，系统是一切事物存在方式之一，可以说是包罗万象。若讨论所有的系统分类方法，涉及太多的学科。因此，本书仅介绍与后续内容有关的系统的分类方法。

1）按照系统的规模划分

（1）小系统：构成系统的子系统数目若为几个、十几个，则称为小系统。如一台测量仪器可视为一个小系统，这类系统用传统的数学、物理学、化学可以很好地描述。

（2）大系统：构成系统的子系统数目若为几十个、上百个，则称为大系统。如一个仅考虑产品生产的普通工厂可视为一个大系统，它可以用控制论、信息论或运筹学的部分内容加以研究。

（3）巨系统：构成系统的子系统数目若为成千成万个、上百亿个、上万亿个，则称为巨系统。例如激光系统、因特网系统，它可以用耗散结构理论和协同学来研究。

2) 按照系统内的子系统之间的关联程度划分

(1) 简单系统:系统内的子系统间的关联程度比较弱,称为简单系统。例如一个由几个人的松散联合构成的团队,可视为简单系统。又如,贝纳尔流作为物理系统,微观组分的数量级为 10^{23},比人的大脑多 12 个数量级,是一个巨系统,但由于微观组分(子系统)的关联程度低,它仍然属于简单系统。

(2) 复杂系统:系统内的子系统间的关联程度比较高,并产生整体的涌现性,称为复杂系统。核苷酸、氨基酸和碳链分子构成的系统就是复杂系统,因为它们以某种形式排列并聚集起来,就能构成称为生命的东西。中国的人口接近 13 亿,即 1.3×10^9,比贝纳尔流低 14 个数量级,但由于系统内的子系统间的关联程度比较高,它就是一个典型的复杂系统。

3) 按照系统与外界环境的联系情况划分

(1) 孤立系统:孤立系统与外界环境之间没有任何物质、能量和信息的交换。随着时间的推移,其内部状态从有序趋向无序。系统的无序度是用熵来描述的,熵的变化大于零。因此,一个孤立系统的内部熵将随时间不断增加,最终熵达到最大值,此时系统趋向于无序。

(2) 开放系统:开放系统与外界环境之间不断进行着物质、能量和信息的交换。这种交换使它可能从外界环境输入负熵,从而使系统的总熵减小,或控制在某种缓慢的增长速度,其结果是增加了系统的有序性。这种自发的增加系统有序性的性质,称为自组织性,这类系统也称为自组织系统。而复杂系统是开放系统。

1.3 复杂系统的特点

“复杂系统”这一术语于 1999 年 4 月 2 日正式出现在美国《科学》(Science)杂志出版的《复杂系统》专辑上,编者 Richard Callagher 和 Tim Appence 对他们所指的“复杂系统”作了简单的描述:通过对一个系统的分量部分(子系统)的了解,不能对系统的性质作出完全的解释,这样的系统称为复杂系统。他们的意思,用通俗一点的话说:系统的整体性质不等于部分性质之和,这样的系统称为复杂系统。清华大学王正中教授把复杂系统的特点概括为[2]:复杂系统往往具有病态定义的特征,即很难以一种数学形式来对它进行定义及定量分析;复杂系统的另一个难点是病态结构,系统结构很难从空间和时间上加以分割,很难确定系统的边界和水平。国防科技大学瞿继权博士和戴金海教授则认为[4]:复杂系统是由多种类型的子系统组合而成的混合系统,包括连续系统和离散系统、非实时系统和实时系统等。中国科技大学朱六章先生认为[5]:复杂系统通常具有系统结构的多层次性,子系统模型的多样性,相互关联的复杂性,目标的多重性,信息的不确定性,以及由此确定的处理方法的多途径特征。我国的著名科学家钱学森先生把复杂系统的性质概括为[3]:开放性、复杂性、层次性。还有许多学者做过同类性质的论述,在这里就不一一列举。概括起来,这些学者认为复杂系统的特点是:高阶次、多回路、非线性、多时标、层次性、开放性、不确定性、病态结构等。在此基础上,本书将以上特点综合表述为复杂系统的特点就是系统具有涌现性。

1) 北京师范大学教授姜璐、谷可两位学者总结的三类复杂系统[3]

系统科学面对各式各样的系统,采用的方法也略有所不同,从目前研究的情况来看,研究较多,有较多规范方法,已经取得一定成就的有三类系统,它们是非平衡系统、复杂适应性系

统、开放的复杂巨系统。这三类系统都包括大量的子系统，子系统在组成系统时，在系统整体层次都会涌现出新的性质。人们对这三类系统的研究所得到的一些具体结果，就反映了人们从子系统构成的层次关系上研究复杂性的结果。

(1) 非平衡系统

非平衡系统多指由无生命子系统组成的系统，每一个子系统非常简单，多是分子、原子、离子等，子系统之间的相互作用一般也比较简单。人们采取远离平衡的自组织理论(耗散结构、协同学等)，来讨论系统的演化。自组织理论是以物理学的统计物理为基础发展起来的，它要求系统满足统计物理的各态历经、局域平衡等基本假设。自组织理论讨论系统在远离平衡的开放的条件下，由于系统内子系统之间的非线性相互作用，在达到某一阈值时，各子系统独立的运动将会被子系统协同的统一集体运动所代替，系统呈现出有序的结构。对此类系统的讨论表明，系统整体性质的涌现是所有子系统统一协同作用的结果，这种协同作用只有在某阈值条件下才能出现。一旦出现了整体的协同作用，也就标志着系统新性质的涌现。讨论这类系统的理论比较规范，有较强的数理基础，有较好的数学工具，得到了比较可信的成果。

(2) 复杂适应性系统

美国圣菲研究所以生物体为背景建立的复杂适应性系统模型，是近年来研究比较多的另一类系统，它将生物体的适应环境、生长繁殖、遗传变异等生物演化性质条理化、规范化，建立用计算机模拟的子系统的演化机制，使之类似生物体，进而研究由这样子系统组成的系统整体，在一定外界条件下的演化性质。这类系统的子系统与非平衡系统中的子系统(分子、离子等)不同，它具有一定的智能。计算机的使用给此类讨论带来很大方便，用计算机程序描写子系统的局域演化规则，给出对个体遗传、变异、学习、适应等的表述形式，大量子系统组成的系统的性质在计算机中就可显示出来，它可以反映种群的发展、稳定存在、消亡等各种复杂的行为，它在模拟生物种群、生态系统、经济系统等演化中取得较多成果。由于此类系统的子系统具有一定智能，因此不仅在解决实际问题方面有广泛的应用，而且对系统科学的发展也有很重要的意义，可以预计在此方面的研究今后会有更多。

(3) 开放的复杂巨系统

开放的复杂巨系统是由我国研究人员为研究由人组成的最复杂系统演化问题而提出的一类模型系统。此系统中的子系统是人，其演化机制目前还不清楚，研究由大量演化机制还不清楚的子系统组成的系统的性质，当然非常困难。钱学森先生提出从定性到定量的综合集成方法，在方法论的层次给出了解决的办法，具体采用人机对话的科学研讨厅体系。目前在分析、讨论实际问题上取得一定效果，在理论研究上困难还较多，人工智能、思维科学、脑科学等方面的研究将会有助于开放的复杂巨系统问题的研究。

2) 美国学者欧阳莹之举出 3 个复杂系统的案例[6]

(1) 多体系统(Many-body System)

多体系统是由少数几类彼此之间仅有几种关系耦合在一起的大量组分组成的系统。例如，一个由消费者和生产者形成的组织，或者一个由生物体组成的演化物种。组分之间的联系程度可以不同，但它们整体上不会消失。一个多体系统不是一堆沙子，其组分之间存在相互的联系，这样就有足够的整体性使之成为一个更大系统中的个体，就如一个铁架成为建筑物的一部分。另外，在理性选择和交际－行动理论中，社会可以看成是一个多体系统。

(2) 有机系统(Organic System)

有机系统是由许多高度特化的、相互联系紧密的、不同种类的组分组成的系统。有机系统易于进行功能的描述,其组分的定义和刻画是通过在维持系统处于所想状态中的功能作用来进行的,这样这些组分从整体上就从属于这个系统了。例如,生物系统或人体系统。

(3) 控制系统(Cybernetic Systems)

控制系统是把多体系统和有机系统的复杂性结合起来。例如神经网络,称为控制系统,也是复杂系统。

1.4 复杂系统建模的理念

1.4.1 涌现的理念

复杂系统的特点在于它具有涌现性[1]。涌现的英语是 emergence,在中文文献中有的译为涌现,有的译为突现,两种译法各有所长。贝塔朗菲引进 emergence,意在强调非加和的整体特性是在系统形成时突然出现的,"如果我们可以把总和想像为逐渐构成的,那就必须把作为相互关联的部分总体的系统想像为瞬间形成的"(贝塔朗菲,1973),译作"突现"能反映了这一点。但在系统科学领域中,又有突变的概念(托姆,1989),两者仅是一字之差,而突变的意思是:指系统从一种定态到另一种定态变化,是相对于渐变而言。为了避免与系统科学大量使用的突变概念相混淆,笔者认为在这里将 emergence 译为涌现更好些。一方面,把 emergence 译为涌现,能反映从低层次产生高层次特性的现象类似于泉水从地下"涌"出来,生动形象,加深了对"即使不了解低层次的机理,仍可直观考察高层次"这一特点的理解;另一方面,涌现在一定程度上也能反映贝塔朗菲强调的"瞬间形成"的特点。

那什么是涌现性呢? 在客观世界的各个领域,特别是生命、社会、思维领域,普遍存在这样一类现象:诸多部分一旦按照某种方式形成系统,就会产生出系统整体具有而部分或部分总和所不具有的属性、特征、行为、功能等,一旦把整体还原为互不相干的各部分,这些属性、特征、行为、功能便不复存在。系统科学把这种整体具有而部分不具有的东西,称为涌现性(Emergent Property)。从层次结构的角度看,涌现性是指那些高层次具有而还原到低层次就不复存在的属性、特征、行为、功能。当然,新质的涌现不一定都伴随层次的提升,同一层次上一种新结构取代原结构的演化也会伴随不同质的整体涌现性的兴替,单层次的提升必定伴随原层次所没有的新质的涌现。

涌现是一种整体的现象和特性,整体的现象和特性不一定都是涌现的。贝塔朗菲区分了累加性与构成性(非加和性)两种整体特征,把整体分为非系统总和与系统总和两种。整体的那些只需把部分特性累加起来即可得到的特性不是涌现性,只有那些依赖于部分之间特定关系的特征,即所谓构成性特征,才是涌现性。通俗一点讲,就是 "1+1>2"、"整体大于部分之和"、"多来自少"、"复杂来自简单"、"有生于无"等。

氧和氢的结合生成水,是一个化学方面的生动例子:氢元素有易燃性,氧元素有助燃性,化合为 H_2O 分子就失去易燃性和助燃性,这种不易燃性是 H 原子和 O 原子按照特定的化学键相互作用而涌现出来的。在整个人类社会的发展史中,有很多涌现性很好的案例。刘备有了诸葛亮,才能三足鼎立,如果把他们两人分开,则两人都一事无成;刘邦有了韩信、张良、萧何,

才能战胜项羽，如果他们不联合起来，则根本不是项羽的对手。人工生命的倡导者兰顿(1988)对生命的揭示，也是一个很好的说明，他指出："生命是一种形式的性质，而非物质性质，是物质组织的结果，而非物质自身固有的某种东西。尽管核苷酸、氨基酸或碳链分子都不是活的，但是只要以正确的方式把它们聚集起来，由它们的相互作用涌现出来的动力学行为就是被我们称为生命的东西。"这是很有说服力的。组织在细胞中的分子同处于非细胞实体中的分子并无两样，但令人着迷的"生命力"或"活力"只能是物质分子按照细胞这种结构模式进行组织所带来的涌现性。诸如此类，不胜枚举。

1.4.2 复杂系统建模的理念

从前面的分析可知，复杂系统建模的任务就是建立系统的模型，以便描述系统的涌现性，即描述被研究对象何时、何处、出现何种涌现。

由涌现的理念，可以看到，这是一个崭新的理念，对其开展研究具有相当的难度。那么，对系统涌现性进行研究，建立模型，是否超出科学的能力范围？存在认识涌现现象的知识限制吗？在这个问题上西方学术界是有争论的，SFI 曾就此举办关于"知识的限制"研讨会。提出"科学终结论"的人都把涌现看作超越科学能力的现象，霍根甚至把科学界现在提出研究涌现性问题本身当作科学终结的证据。作为辩证唯物论者，相信没有一种涌现性是人类原则上无法认识的，科学终将发现描述涌现性的有效方法。当然，也应看到问题研究的难度，由涌现产生的宇宙奥秘不可能在某一代学者手里全部揭示出来，只要人类存在科学就不会终结。本书根据 SFI 的以上论述以及现有的其他文献，整理和归纳出下面几种复杂系统建模方法。

1）还原论的方法

采用还原论科学在把对象分解为它的微观组分后，进而分析和研究这些组分之间的相互作用，最后，用低层次事物的相互作用去说明高层次的问题。只要能用低层次组织的相互关系来解释高层次现象，就是在一定程度上描述系统的涌现性。

2）多变量的联立方程组描述法

一般来说，反映系统宏观变量的方程都是描述整体涌现性的模型，多变量的联立方程组尤其有效。由庞加莱开创的非线性动力学揭示，动态系统的质的规定性是由它的稳定定态或吸引子表示的。系统的宏观整体特性可以通过研究它的稳定定态来把握。稳定定态就是系统的整体涌现性，只有形成系统整体时才会出现，一旦还原为部分这种定态就不复存在。系统定态不可能由元素或子系统性质推测出来。贝塔朗菲已意识到这一点，坚持用联立微分方程组定义他的一般系统，并据此讨论生长、竞争、机构化、中心化、果决性这些"大于部分之和"的涌现现象，得出一些启发性意见。因而，研究非线性动力学，也将是研究整体涌现性的一种方法，也是复杂系统建模的重要方法之一。

3）基于演绎逻辑的方法

数学方法基于演绎逻辑处理涌现性，本质上是生成论的，也是研究整体涌现性的一种方法。这里的关键是建立数学模型，只要有了数学模型，就可以用演绎方法严格逻辑地预测系统可能发生的涌现行为。如果系统比较简单，能够建立它的演化方程，即可通过研究这种方程来了解它的涌现性。对于那些无法建立演化方程的系统，只要它们是在现实世界生存发展的，就一定存在稳定定态，通过定性的或半定性半定量方法，也可以对整体涌现性有所了解。因此，

在本书中安排了一章的定性建模的内容，作为复杂系统建模的理论和方法之一来讲解。

4）简单巨系统的描述法

钱学森先生把现在通用的系统科学方法概括为两种：一是处理简单系统的方法，首先根据基本科学原理（通常是力学的或电磁的）建立各个环节的数学方程，再按照不同环节耦合为系统的方式进行综合，便得到描述系统整体行为的数学方程。钱学森先生称其为直接综合法，也就是前面介绍的还原论方法。它也包括唯象方法，即着眼于对系统宏观行为的观测结果，提出某种猜想，直接用适当的数学方程表示出来。控制理论和运筹学用的就是这种方法，几十年的发展证明其是成功的；二是简单巨系统，构成元素数量巨大，但“花色品种”很少，例如前面曾提到的贝纳尔流，微观组分的数量级为 10^{23}，就是典型的简单巨系统。这种系统的元素间相互关系简单，允许忽略其间的差异，把微观元素的相互作用简化为类似物质分子之间的随机碰撞。由于元素数量巨大，大数定律保证这种系统具有宏观整体的统计特性。统计特性是一种特殊的整体涌现性，在微观层次是看不到。基于概率论的统计综合法是把握简单巨系统整体涌现性的有效方法。自组织理论特别是协同学提供了成功应用的范例。

5）从定性到定量综合集成法[7]

当对象属于复杂巨系统时，直接综合法（即还原论方法）和统计综合法均告失效。钱学森先生提出研究复杂巨系统的从定性到定量综合集成法，这是一种把握系统整体涌现性的方法。面对一个现实的开放复杂巨系统，不可能把它从环境中暂时孤立出来进行可控实验，无法把它分为许多部分分别加以研究，然后进行综合。但社会分工、学科划分本身就是对这个复杂巨系统的一种分解或还原。工作在复杂巨系统不同局部的人们、特别是相关领域的专家分别掌握了不同部门的科学知识，积累了大量局部感性经验，加上关于复杂巨系统的各种统计资料、数据，构成关于复杂巨系统整体定量认识的微观基础。按照实践论的原理，理性认识是从感性认识中涌现出来的，个体认识如此，群体认识也如此。一切有关开放的复杂巨系统宏观整体行为特性的理性认识，只能从上述零碎的感性认识中涌现出来。问题是如何使这种向来只靠人类大脑进行的认识系统的涌现行为，变成有现代科学知识体系和高新技术支撑的可操作的过程，以保证有关开放复杂巨系统行为特性的整体定量认识在此过程中最充分最有效的涌现出来。从定性到定量综合集成法提供了实现这种涌现行为的一种可行方法，所建立的体系被称为“从定性到定量综合集成讨论厅体系”。本书的后续内容，对该方法会做一定的介绍。

6）基于主体（Agent）的复杂系统智能建模法[8]

对于系统的整体涌现性，SFI 学派倡导的是生成论的研究方法。他们把能够呈现出涌现性的系统分为两类：一类是规则支配的系统，相信这类系统的涌现性可以用科学方法描述。简单而典型的例子是各种棋盘式游戏，如下棋，根据很少几条关于棋子合法行走的局部规则，可以下出（涌现出）无穷多种不可能从这些规则中预测的棋盘局势，一切有规则支配的系统都如此。SFI 的思路是，选择适当的主体（Agent）作为构件（Building Block），用计算机程序设计少数支配主体相互作用的规则，通过计算机仿真考察该模型的涌现行为。目的是建立一种概念框架，以便能够预测何时、何处出现涌现，出现什么样的涌现。生成论的涌现研究方法建立在归纳逻辑之上，所用模型是由计算机程序表示的，根据模型让系统在计算机上产生、演化，让宏观整体行为由下而上、自然而然的涌现出来，使研究者能够直接观察系统的生成、演化过程，从观察现象中发现规律，提炼概念，形成洞见，建立理论。这就是在复杂系统建模中，本书将智能

建模列入其中的原因。至于另一类系统，范围非常广阔，如伦理、诗歌等，虽然可以明显的观察到涌现现象，但不属于规则支配的系统，能否成为科学的对象，SFI 学者们仍然还在研究。

7）复杂系统建模的途径[9]

从方法论的角度来看，建立复杂系统的模型时，可以采用 3 种不同的基本方法，即演绎法、归纳法和演绎-归纳法，从而形成了两种基本的建模途径：

（1）演绎-归纳建模

① 演绎建模。

根据有关系统的一般原理、定律、系统结构和参数的具体信息和数据，进行从一般到特殊的演绎推理和论证，建立面向组分（子系统）的模型。这样的方法，称为演绎法，所建立的模型，称为机理模型或解析模型。

演绎法适于组分（子系统）为白箱的情况，通常所建立的模型有唯一性。

② 归纳建模。

利用对实际系统的输入和输出的观测与统计数据，运用记录或实验资料，进行从特殊到一般的归纳推理和总结，建立系统的外部等效模型。这样的方法，称为归纳法，所建立的模型称为经验模型或外部模型。

归纳法适于系统为黑箱的情况，通常所建立的模型不是唯一的。

③ 演绎-归纳建模。

演绎-归纳建模法也称混合法。通常，它采用演绎法或专家经验，确定模型类别、维数及结构，然后用归纳法辨识模型参数。混合法适用于系统为灰箱的情况。

（2）分解-联合建模

这种方法适于简单大系统建模。采用的方法是：首先，将系统分解为若干个子系统；其次，不计各子系统间的关联，分别用演绎、归纳法建立各子系统的模型，有的文献称之为分解建模；最后，建立各子系统间的关系模型，利用该关系模型将各子系统有机的组装起来，形成系统的总模型。这种建模方法也称为结构建模法。

8）复杂系统建模的基本原则[16]

（1）模型众多性原则

1795 年，年仅 21 岁的德国青年数学家高斯（K. F. Gauss）解决了一个著名的问题，就是根据非精确观测的试验数据来确定行星椭圆轨道的参数。椭圆轨道在诸观测点中的位置这样选定，即使诸观测数据偏离椭圆轨道的偏差平方和达到极小值。这种数据处理方法就称为最小二乘法。在高斯以后相当漫长的一段岁月里，许多学者遵循高斯的思路，试图将最小二乘法推广应用于解决各种更加复杂的问题，都没有取得重大的进展和实质性的突破。

直到 20 世纪中叶，英国科学家 Y. R. 艾西比指出：“简单模型的时代已经过去了，对研制能够表现高级神经功能的装置发生浓厚兴趣的时代来到了。C. E. 申农于 1938 年对继电器网络以及马克·卡洛克和彼特斯于 1943 年对简单的类神经元组成的网络证明了，需要描述的机器的任何行为不是只能由某一个机器来实现，而是可以由无限多个机器来实现。从那个时候起，新的唯一模型的研究仅仅表明，模型的创建者还没掌握申农、马克·卡洛克以及彼特斯的研究工作的意义。”

1971 年，苏联科学家 A. т. 依瓦赫连柯提出了模型众多性原则。1978 年，另一位苏联科学

B. N. 别涅耶夫在建立复杂地质系统理论时也提出了同样的原则。1982 年，B. C. 佛列依斯曼认为，模型众多性原则是系统学中的一个基本原则。

模型众多性原则断言，根据实验数据原则上不能找到唯一的模型。例如，在插值问题中，可以选取任何形式和任意次数的多项式作为对象的数学模型，由回归分析就可以确定其中每一个多项式系数的数值，这些数学模型中的许多模型都在同等程度上对应于原始数据。由此可见，对于被看作黑箱的每一个对象，都不是只能建立唯一的模型，而是可以建立具有相同或几乎相同外部表现的无限多模型。事实上，即使在像现代物理学这样严密的知识领域里，也同样表现出模型的众多性，尤其是从不同的原则出发解释同一个被观测现象的量子物理理论就有好几个。

只有外部增补才能判断和评价各种彼此矛盾或相互不一致的理论。由所有观测点上得到的误差不是外部增补。

既然由观测数据确定的模型不是一个，而是有复杂性程度不同的无限多个，那么，是否存在复杂性最优的唯一模型呢？要回答复杂性最优的唯一模型的选择问题，还必须利用外部增补原则。

在数理逻辑中，德国著名数学家 K. 哥德尔于 1931 年证明了哥德尔非完备定理，从而使外部增补概念得到了重大的发展。这个定理指出，任何公理体系都不可能是逻辑上封闭的：总是可以找到这样的一个定理，它的证明需要外部增补，即是要求扩充原来的公理体系。

例如，在自动调节系统中，外部增补就是被调节量的设定值。自动调节系统镇定反馈回路中的所有变量（设定值除外），为了使设定值恒定，就要求附加镇定回路，而镇定回路同样又需要新的外部增补，即新的设定值，如此等等。多级系统就是利用许多设定值来作为外部增补。

必须指出的是，尽管哥德尔非完备定理的证实已有 70 多年的历史，但是现代的系统分析、控制对象的辨识理论以及回归分析仍然利用内部准则，并没有考虑哥德尔非完备定理的要求。根据哥德尔的思想，模型的综合应该建立在利用外部准则的基础上，然而在数字模拟中至今尚未注意到这一点。

根据哥德尔非完备定理，可以将选择模型的准则划分成内部准则和外部准则。内部准则指的是这样的准则，确定该准则时利用的信息或数据就是得到模型本身所利用的信息或数据。反之，如果确定准则时，利用的是模型综合时未曾利用过的新信息或新数据，则该准则就称为外部准则。

在不划分原始数据以获得外部准则的情况下，用极大似然法或最小二乘法确定的数值就是利用内部准则的例子。但是，当使用内部准则时，研究者就会得到“模型越复杂越精确”这个结论，显然，这是不符合事实的。因此，选择模型时应该采用的仅是外部准则。

当模型的复杂性逐步增加时，外部准则通过极小点，因而根据某个外部准则就可以找出最优复杂性的唯一模型。极小值的大小可以作为所求模型的置信度指标。

如果模型结构的复杂性低于最优复杂性，则该模型称为欠复杂性模型；反之，如果模型结构的复杂性超过了最优复杂性，则该模型就称为超复杂性模型。与欠复杂性模型一样，超复杂性模型也不是我们所期望的。

因此，在数据处理组合法（Group Method of Data Handling，简记为 GMDH）中，把原始数据划分成两部分，即拟合数据和检验数据，而且由检验数据确定的均方差 $\delta^2(B)$ 就是选择用拟合数据综合的模型结构的一个准则。这样就建立了第一外部增补。如果再提出，还要把原

始数据最优地划分成拟合数据和检验数据，那就还要有一个（第二）外部增补，这就必须把原始数据划分成 3 部分，即拟合数据、检验数据和校核数据。由校核数据确定的均方差 $\delta^2(C)$ 就是第二外部增补。如果又要把原始数据最优地划分成水 3 组数据，就还需要有一个（第三）外部增补 $\delta^2(D)$。如此类推，这样就能够完全遵循哥德尔非完备定理的基本思想。

还要指出的是，哥德尔非完备定理仅对某种形式的逻辑体系才成立，并不是对所有的逻辑体系都成立。事实上，存在着这样的封闭逻辑体系，对它们来说，哥德尔非完备定理是不适用的。自动调制器就是这种逻辑体系的一个例子。自动调制器中不存在外部的给定基准，而是利用系统元件的非线性特性来镇定系统的状态。Y. R. 艾西比的内稳态装置就是自动调制器的一个著名的例子。

【例 1-1】 一个独立的联立方程组，当方程的个数等于自变量的个数时，例如：

$$x_1 = f(x_2) \tag{1-1}$$

$$x_2 = f(x_3) \tag{1-2}$$

$$x_3 = f(x_1) \tag{1-3}$$

就是一个非哥德尔体系，其中的每一个方程都可以由其余的两个方程来求解。

一个非独立的联立方程组，例如：

$$x_1 = f_1(x_2, \varphi) \tag{1-4}$$

$$x_2 = f_2(x_3) \tag{1-5}$$

$$x_3 = f_3(x_1) \tag{1-6}$$

则是一个哥德尔体系，其中的任何一个方程都不能由其余的两个方程来求解。

（2）非最终决策原则

根据 1971 年度诺贝尔物理奖获得者、英国学者 D. 伽柏（D. Gabor）在一部论著中所作的证实，可以确定自组织算法，该算法规定了“选择后续决策的自由”。

数据处理组合法的基本思想之一就是 D. 伽柏的非最终决策原则。根据这个原则，只有在对前面每一层的若干最优决策保证充分的选择自由的条件下，所有的单层次过程才可以用要求较少计算时间的多层次过程来代替。

F. 诺森布拉特（F. Rosenblat）的仿脑机就是非最终决策的一个典型的应用例子，其中由一个层次到另一个层次产生的不是唯一的最优决策，而是许多接近最优的决策。

育种家进行的植物选种是应用非最终决策原则的另一个例子。对于每一个后续的选择，提供的是前面选定的许多决策，并不是一个决策，这就保证了选择的自由。育种家早就发现了选择自由的重要性。对于每一代，选择的不是一对而是一定数量的最确切满足要求的植物种子。此外，育种家认为过分自由和没有自由一样，都是有害的。每一代中选出的植物种子数量都存在一个最优值。

对于模式识别、规划和控制问题，如果每次的选择是唯一的，就称为刚性的。如果在每次决策时，选出的是（根据正则性准则）接近最优的解的某个集合时，这样的控制就相应于自组织原则。

根据非最终决策原则建立的系统或算法，其基本特征就是选解自由的存在和多层次性。这种系统或算法是与刚性确定型系统或算法性质完全不同的一种系统或算法。

（3）模型自组织原则

模型自组织原则可以这样来陈述：当逐步增加模型的复杂性时，例如：线性模型、二次型模型、三次型模型等等，内部准则值（噪声存在时）单调下降。在同样的条件下，所有外部准则都

通过自己的极小值(极值),这就为确定唯一的(对每个外部准则来说)最优复杂性模型提供了可能性。通常可以编制出计算机程序,以便寻找这样的模型结构的复杂性程度,这时外部准则的极小值是唯一的。

【例 1-2】 表 1.1 介绍了二元二次型多项式表达的模型逐步复杂化的一个例子,表中模型的位置是根据对所有试验数据计算出来的均方差单调减少的顺序来排列的。

从最简单模型到最复杂模型,或者反过来,从最复杂模型到最简单模型,由计算机逐个搜索,就可以选出唯一的(对每个外部准则来说)最优复杂性模型。为了圆满地解决这个问题,应该根据所要解决问题的类型选择合适的外部准则。

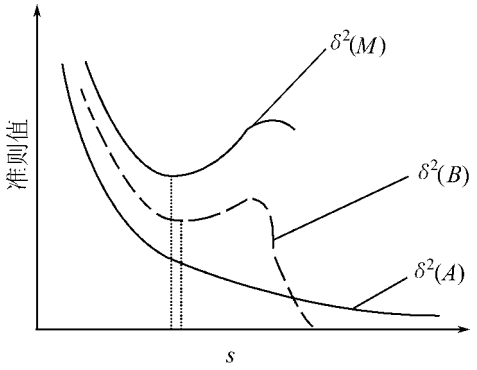


图 1.2 根据外部准则值求最优复杂模型

当模型的复杂性逐步增加时,内部准则、外部准则和混合准则的数值变化情况如图 1.2 所示。作为一个例子,图中外部准则选取正则性准则 $\delta^2(B)\rightarrow\min$;内部准则选取出最小二乘法确定的均方差 $\delta^2(A)\rightarrow\min$。当多项式的项数等于原始数据表中的点数时 $\delta^2(A)=0$,即误差等于零。图 1.2 中,s 为多项式的次数。该图中虚线表示混合准则:

$$\delta^2(M)=\lambda\delta^2(B)+(1-\lambda)\delta^2(A) \tag{1-7}$$

式中:$0\leqslant\lambda\leqslant1$。

表 1.1 二元二次型多项式表达的模型逐步复杂化的过程

x_1	x_1^2	x_2	x_1x_2	x_2^2
$y_1=a_0+a_1x_1$	$y_2=a_0+a_1x_1^2$ $y_3=a_0+a_1x_1+a_2x_1^2$	$y_4=a_0+a_1x_2$ $y_5=a_0+a_1x_1+a_3x_2$ $y_6=a_0+a_1x_1^2+a_2x_2$ $y_7=a_0+a_1x_1+a_2x_1^2+a_3x_2$	$y_8=a_0+a_1x_1x_2$ $y_9=a_0+a_1x_1+a_2x_1x_2$ $y_{10}=a_0+a_1x_1^2+a_2x_1x_2$ $y_{11}=a_0+a_1x_1+a_2x_1^2+a_3x_1x_2$ $y_{12}=a_0+a_1x_2+a_2x_1x_2$ $y_{13}=a_0+a_1x_1+a_2x_2+a_3x_1x_2$ $y_{14}=a_0+a_1x_1^2+a_2x_2+a_3x_1x_2$ $y_{15}=a_0+a_1x_1+a_2x_2^2+a_3x_2+a_4x_1x_2$	$y_{16}=a_0+a_1x_2^2$ $y_{17}=a_0+a_1x_1+a_2x_2^2$ $y_{18}=a_0+a_1x_1^2+a_2x_2^2$ $y_{19}=a_0+a_1x_1+a_2x_2^2+a_3x_2^2$ $y_{20}=a_0+a_1x_2+a_2x_2^2$ $y_{21}=a_0+a_1x_1+a_2x_2+a_3x_2^2$ $y_{22}=a_0+a_1x_1+a_2x_2+a_3x_2^2$ $y_{23}=a_0+a_1x_1+a_2x_1^2+a_1x_2+a_4x_2^2$ $y_{24}=a_0+a_1x_1x_2+a_2x_2^2$ $y_{25}=a_0+a_1x_1+a_2x_1x_2+a_3x_2^2$ $y_{26}=a_0+a_1x_1^2+a_2x_1x_2+a_3x_2^2$ $y_{27}=a_0+a_1x_1+a_2x_2^2+a_3x_1x_2+a_4x_2^2$ $y_{28}=a_0+a_1x_2+a_2x_1x_2+a_3x_2^2$ $y_{29}=a_0+a_1x_1+a_2x_2+a_3x_1x_2+a_4x_2^2$ $y_{30}=a_0+a_1x_1^2+a_2x_2+a_3x_1x_2+a_4x_2^2$ $y_{31}=a_0+a_1x_1+a_2x_2^2+a_1x_2+a_4x_1x_2+a_5x_2^2$

由图 1.2 可见，叠加曲线亦可指出唯一模型，但由于选择系数 λ 的随意性，使得这种选择方法没有多大意义。此外，根据混合准则得到的模型复杂性超过根据外部准则得到的模型复杂性。

1.5 广义模型的概念[9,10]

在大多数的有关著作中，模型通常分为物理模型和数学模型。但随着科学技术的发展和研究对象的复杂性的提高，以及出现不确定性和不确知性等问题，有必要将数学模型的概念广义化，称之为广义模型。

1.5.1 集成模型

广义模型是一种集成模型，它是知识模型、数学模型和关系模型的综合与集成，具体见图 1.3。

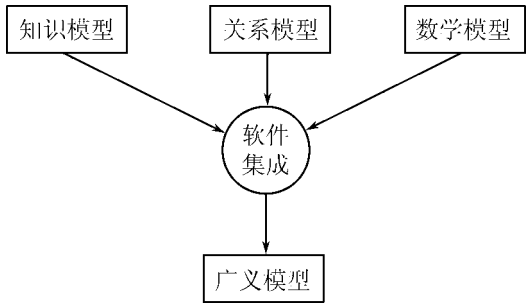


图 1.3 广义模型：集成模型

1）知识模型——KM (Knowledge Model)

运用人工智能和知识工程的方法与技术，例如：知识表达方法（产生式规则、语义网络、框架等）、知识获取技术（人工移植、机器感知、机器学习等）所建立的知识模型，主要用于表达人们关于事物定性知识和经验知识，以便利用知识进行定性分析和逻辑推理，求解有关问题。

2）数学模型——MM (Mathematical Model)

运用控制理论、系统辨识或运筹学及其他数学的方法和技术，所建立的数学模型，例如：状态方程、传递函数、代数方程等，主要用于定量地描述事物的有关动态过程与静态特性，对问题进行定量分析和数值计算。

3）关系模型——RM (Relation Model)[11]

运用“图论”方法(Graphic Theory)和逻辑学，所建立的各种关系模型，例如：赋值的有向图、无向图、树图、网络图、逻辑式等，主要用于定性或定量地描述人们或事物之间的各种关系。例如：组织结构、工艺流程、信息传递、时序关系等。用于进行系统的结构分析与结构综合，也可称之为“结构模型”。

在复杂系统模型化的实际工作中，应根据具体任务的需求和环境条件的可能，灵活地运用上述 3 种模型。在知识模型、数学模型、关系模型相结合的基础上，利用计算机软件进行集成，建立适用于实际系统的“集成模型”，即广义模型。

广义模型可以全面地（定性、定量、静态、动态）描述实际系统的结构、参数、功能和特性。

1.5.2 控制论模型

从控制论观点出发，任何复杂控制系统都可以用方框图（图 1.4）表示，称为控制论模型。

1）控制者

控制者产生控制作用，接受反馈信息，它可以是人，例如控制、管理、调度、指挥、决策人员等；也可以是机器，例如控制器、调节器、计算机等。

2）控制对象

被控制对象接受控制作用，提供反馈信息，它通常是机器或设备，也可以是人或人群。例如社会经济管理系统。

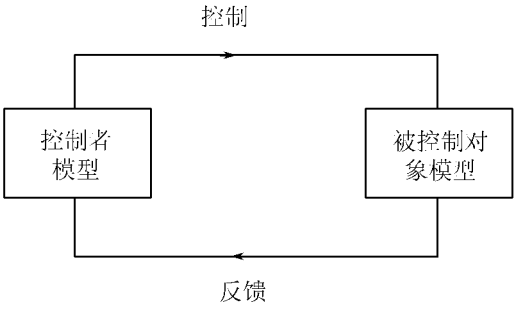


图 1.4 广义模型：控制论模型

通常，在控制理论方面，主要致力于建立被控制对象的数学模型，例如系统辨识方法和技术。而较少研究控制者的模型化问题。所以，在控制系统模型化的方法上存在局限性。

对于复杂系统的模型化，应采取控制者模型与被控制对象模型并行处理方法，建立控制论模型。例如，对于主动系统，可以利用人工智能专家系统的方法和技术，建立控制者的知识模型，以有经验的操作、调度、指挥人员为领域专家，建立相应的知识模型，以及基于知识库的、用于控制和管理的各种人工智能专家系统。这样，逐步地实施钱学森先生提出的综合集成厅体系。

1.5.3 分层模型

由于实际复杂系统可以分解为若干“子系统”，而子系统又可以再细分为若干“子子系统”。因此，可以采用分层模型化方法，分别地对于宏观、中观、微观，建立相应的模型，它们的量化单位，可称为粗粒度、中粒度、细粒度；再考虑层与层之间的关联变量，将它们组织起来，构成复杂系统的分层模型，如图 1.5 所示。

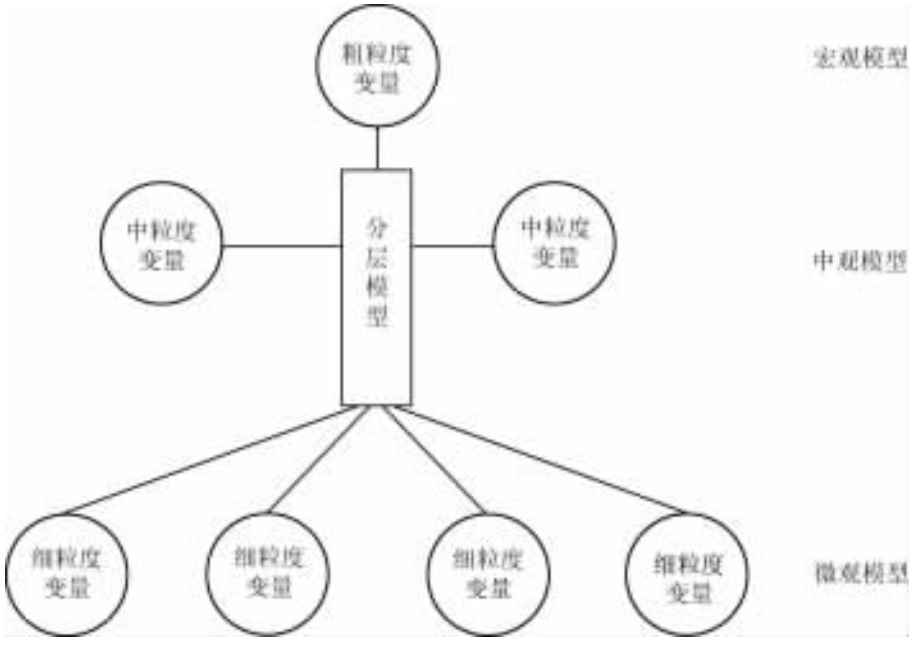


图 1.5 广义模型：分层模型

1）宏观模型

宏观模型就是用粗大的量化单位或知识基元去描绘复杂系统的模型。例如国家级的经济模型，从全国的宏观水平，用国民经济总产值、全国人均年收入等粗粒度的经济指标，描述国民经济的管理系统。

2）中观模型

即用中等规模的量化单位或知识基元描述分系统的模型。例如省级经济模型，按照本省观点，用本省经济总产值、本省人均月收入等指标，描述本省的经济管理系统。

3）微观模型

即用细小的量化单位或知识基元描述分系统的模型。例如县级经济模型，从本县角度，用本县经济总产值、本县人均日收入等指标，描述本县经济管理系统。

利用粗粒度、中粒度、细粒度的量化单位、知识基元（例如国家级、省级、县级的经济指标）之间的关系，可将上述宏观、中观、微观模型组织起来，建立描述复杂系统的广义模型。

1.5.4 智能模型

为了建立发展中系统、不确定系统、不确知系统、主动系统的模型，建议用人工智能、控制理论、运筹学相结合的方法，开发各种智能模型，例如，自学习模型、自适应模型、自组织模型等。如图 1.6 所示。

图 1.6 广义模型：智能模型

1）自学习模型

利用人工智能的自学习方法与控制理论的系统辨识技术相结合，发展智能辨识技术，建立具有自学习性能的数学模型。在系统运行过程中，通过示教学习或非示教学习，获取知识，积累经验、自动辨识修改模型参数，提高模型的精度，以跟踪实际系统的变化和发展。

2）自适应模型

利用在线动态系统辨识技术和自校正方法，可以建立具有自适应性能的模型。当系统的环境条件发生变化或对象特性漂移时，能够自动校正模型参数，以适应系统的变化和漂移。

3）自组织模型

利用模型库和模型库管理系统，可以根据用户需求和解题规划，进行模型构造和组织。在知识模型、数学模型、关系模型相结合的基础上，组成适应实际系统的广义模型。

综上所述，关于“广义化模型”的概念是：从控制论观点，用人工智能、系统辨识、图论等多学科的方法和技术，灵活运用知识模型、数学模型、关系模型，并行处理控制者模型、被控制对象模型，建立宏观、中观、微观相结合的，具有自学习、自适应、自组织性能的、不同粒度、智能化的复杂系统广义模型。各种不同模型（知识、数学、关系模型）可在计算机中相互结合，构成面向对象的综合集成的软件模型。

2 基于智能技术的复杂系统建模

2.1 概述

对复杂的系统，如社会系统、经济系统或复杂的工程系统，其特征是：高阶次、多回路、非线性、多时标、层次性、开放性、不确定性、病态结构等，有时还存在时延，此时采用分解-联合建模方法或演绎-归纳建模方法就难以解决。往往采用人工智能，控制专家的知识、经验和技巧，建立系统的模型，用如神经网络(Neural Network，简记 NN)建模，通常就很有效，这也是建模的新的研究热点和前沿课题。

从方法学的角度分析，基于智能技术的复杂系统建模方法可以分为功能派和结构派两类模型化的方法。

1) *功能派方法*

主要研究和模拟人的智能的宏观表现和功能，利用心理学方法和计算机软件技术，建立人工智能的物理符号演算系统，对人的思维活动与认知过程进行智能模拟、延伸或扩展。可称之为心理学派、计算机学派或“符号主义”学派，本书称该种方法为符号推理与集成建模方法。其典型的具有代表性的成果如下：

(1) 启发式程序(Heuristic Programming)。具有类似于人类求解问题的启发能力(如窍门、技巧、策略等)的计算机程序。有可能在信息不完备、数据不精确、知识不充分等不确定、不确知的环境下，进行各种问题的求解，通过试探和搜索，求得问题合理解或满意解。

例如，能够求解 11 种不同类型问题的通用解题者(General Problem Solver)程序、具有学习功能的下跳棋启发式程序等。

(2) 专家系统(Expert System)。具有类似于专家本人知识水平和求解专门问题的能力，能利用专业知识求解专门问题的计算机软件系统。它是基于知识表达、知识推理方法和技术，拥有知识库与推理机的知识工程系统。例如，化学专家系统 DENDRAL，可分析质谱仪的测试结果，识别化合物的分子结构式。

2) *结构派方法*

从人脑的生理结构原型出发，利用仿生学和微电子技术，研究神经细胞模型及其联结机制，建立神经网络模型及脑模型。对人脑的智能活动进行模拟、延伸或扩展，可称之为生理学派、仿生学派或“联结主义”学派，其典型代表结果如下：

(1) 神经细胞模型。如 M-P(McCulloch and Pitts)神经细胞模型，可以模拟神经细胞的兴奋、抑制状态，阀值效应，时空整合作用和突触连接机制。

(2) 神经网络模型。如“感知机”(Perception)，应用神经网络突触连接的可塑性，通过学习，改变突触连接强度，具有分类、识别等感知功能。“联想机”(Association)，根据神经网络分布式记忆特性和突触的柔性连接，可以通过学习，进行联想记忆和联想识别。能在信息缺损的情况下，提高识别率、减少误识率与拒识率。

上述两种方法，各有长短，可以按照具体情况灵活运用。功能是结构的表现，结构是功能的基础，两种方法相互结合，可以取长补短，设计和建立更完善的系统的模型。由于功能派方法，在建模过程中涉及知识的表示、知识的获取、解决组合爆炸等方面遇到的困难越来越大，因此，本书重点在结构派方法。另外从生物演化过程的“适者生存”来看，演化不是均衡，不是最优，而是不断适应，适应的结果就是不断优化和不断演化。属于这一思想的“遗传算法”、“基于 Agent 的建模方法”、“基于 CGP 建模方法”等，归到“基于智能技术的建模方法”中。还有一类模型和算法，如蚁群优化算法、粒子群优化算法是群体智能的模型和算法，也将其放到这一章中。

2.2 神经网络建模

神经网络是由许许多多的神经元构成的。关于神经元的模型有很多种，限于篇幅，下面仅介绍神经元的数理模型、分层网络模型、B－P 学习算法和 Hopfield 模型的建模方法。

2.2.1 神经元的数理模型[12]

专家学者经过研究，发现人类的神经网络系统是通过感受各种刺激信号，引起不同感觉，产生并传递神经冲动信息，从而完成各种活动的。

目前所提出的神经元模型是多输入、单输出的元件。根据输出值与神经元内部状态的关系，有以下几种模型。

1）阈值元件模型

1943 年由 McCulloch 和 Pitts 首先提出的最早和最简单的神经元模型，如图 2.1 所示。

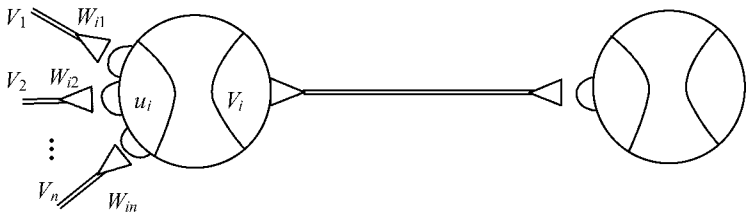


图 2.1 神经元模型

（1）离散输出模型

每一神经元都是从数十甚至数百个其他神经元接收信息产生神经兴奋和冲动，并符合一个称为“全或无定律”的特征。即在其他条件不变的情况下，不论何种刺激，只要达到阈值以上，就能产生一个动作电位，并以最快速度作非衰减的等幅传递，但如果输入总和低于阈值，则不能引起任何可见反应。在图 2.1 中，i 神经元输入的总和为

$$u_i = \sum_{j=1}^{n} W_{ji} V_j - \theta_i \tag{2-1}$$

i 神经元的输出为

$$V_i = 1\left(\sum_{j=1}^{n} W_{ji} V_j - \theta_i\right) \tag{2-2}$$

式中：W_{ji} 表示 i 神经元和 j 神经元的结合权值；V_j 表示 j 神经元的输出（也是 i 神经元的输入）；

θ_i 表示 i 神经元的阈值。

$$\text{阶跃函数为 } 1[Z]=\begin{cases}1, & Z>0\\ 0, & Z\leqslant 0\end{cases} \tag{2-3}$$

(2) 连续输出模型

这种模型采用 S 形函数来表征神经元的非线性输入输出特性，如图 2.2 所示。

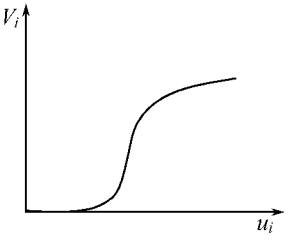


图 2.2　S 形非线性函数

$$V_i=f(u_i)=\frac{1}{1+\exp(-u_i)} \tag{2-4}$$

(3) 微/差分方程模型

这种模型将神经元状态随时间变化的特性用微/差分方程式表示。所采用的方程式各种各样，较具代表性的有

$$\frac{\mathrm{d}v}{\mathrm{d}t}=-\frac{1}{\tau}v+u \tag{2-5}$$

$$V=f(u) \tag{2-6}$$

式中：v 是神经元内部状态值；V 是输出值；u 是神经元输入的总和；f 是上述 S 形函数。

(4) 概率模型

这种模型借助于统计物理学的概念和方法，神经元的动作采用了概率的状态变化规律，例如，被称为玻耳兹曼机的神经网络模型就是采用这种方法，这时

$$P(V_i=1)=\frac{1}{1+c^{-\Delta E_i/T}} \tag{2-7}$$

式中：P 是第 i 神经元状态更新时新状态为 1 的概率。

2) 神经网络模型

上面介绍的是单个神经元模型，如果将很多个神经元组合成一个网络，如图 2.3 所示，并将神经元之间的相互作用关系模型化就构成神经网络模型。

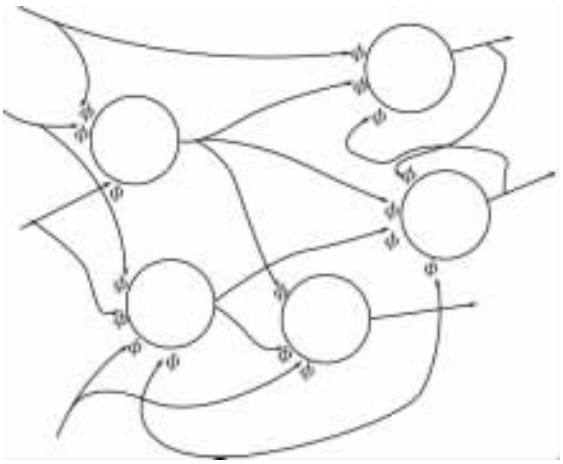


图 2.3　神经网络

根据神经元之间的相互结合关系和作用方式，神经网络可有如下几种典型的结合形式：

(1) 分层网络模型

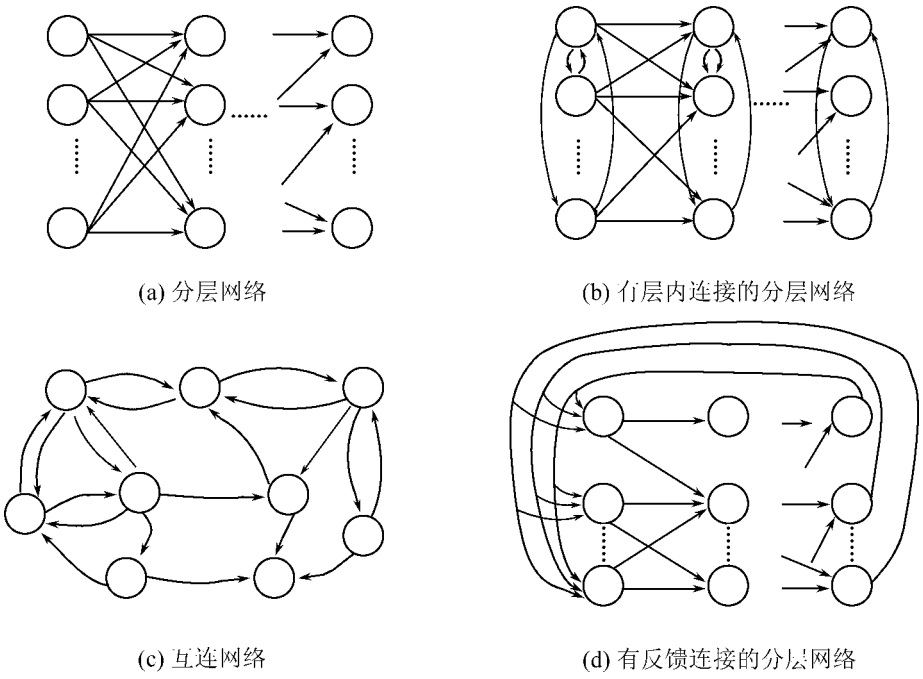
这种模型将众多神经元分成若干层顺序连接，在第一层(输入层) 加上输入样本，通过中间各层进行变换，到达最终层(输出层)，则完成一次动作。

这种分层网络还可以有如下几种结合方式：

① 简单的分层网络，如图 2.4(a) 所示。

② 层内相互之间有结合关系的分层网络，如图 2.4(b) 所示。

③ 有反馈连接的分层网络，如图 2.4(d) 所示。



(a) 分层网络　(b) 有层内连接的分层网络

(c) 互连网络　(d) 有反馈连接的分层网络

图 2.4　神经网络模型

(2) Kolmogorov 定理[13~15]

令 $\phi(x)$ 为非常量、有界单调递增的一元连续函数，M 为 $\mathbf{R}^n$ 的紧致子集，$f(x)=f(x_1,x_2,\cdots,x_n)$ 为 M 上的连续实值函数，则对于任意 $\varepsilon>0$，存在正整数 N 和实数 $C_i,\theta_i(i=1,2,\cdots,N),W_{ij}(i=1,2,\cdots,N;j=1,2,\cdots,n)$，使得

$$\hat{f}(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{N}C_i\phi\Big(\sum_{j=1}^{n}W_{ij}-\theta_j\Big) \tag{2-8}$$

满足

$$\max\left|\hat{f}(x_1,x_2,\cdots,x_n)-f(x_1,x_2,\cdots,x_n)\right|<\varepsilon \tag{2-9}$$

该定理有这样的含意：存在一个三层神经网络，其隐层单元输出函数为 $\phi(x)$，输入层和输出层单元的输出函数为线性的，则该三层神经网络的总的输入输出关系 $\hat{f}(x_1,x_2,\cdots,x_n)$ 可以任意精度逼近 $f(x_1,x_2,\cdots,x_n)$。

比较常用的是图 2.4(a) 所示的分层网络，称为前馈式三层网络，具体在 B-P 学习算法中介绍。这里略作说明：前馈式三层神经网络的输入层的节点数等于输入变量的个数，输出层的节点数等于输出变量的个数，中间隐层的节点数 H 的选择本身就是一种艺术，如果样本数目为 L，一般应满足 $L\leqslant 2^H$。例如，用于交通流预测的模型。由于交通流量的变化是居民出行目的、出行方式、季节、节假日、天气、突发交通事件等因素的综合反映，这些因素与交通量的关系具有高度的非线性，有些因素有很强的随机性。因此，城市交通流预测是一个复杂系统的建模问题。实际建模中，可以换一个角度考虑问题，直接用交通流的测量值进行预测，使建模工作量得到简化。笔者和学生在完成课题的实践中，用 1 个三层神经网络，6 个输入节点、5 个隐节点、1 个输出节点，预测城市道路交叉口的交通流，预测误差小于 5%。

(3) 互连网络模型

这是一种在任意两个神经元之间有相互连接的网络模型，如图 2.4(c) 所示，网络的动作采用动态分析方法，即由某一初始状态出发，根据网络的结构和神经元的特性进行网络的能量最小优化计算，最后达到稳定状态。例如，后面将要介绍的 Hopfield 模型就是采用这种形式。

2.2.2　分层网络模型和 B－P 学习算法[12]

1986 年，Rumelhart 提出了反向传播(Back-Propagation，简记为 B－P) 学习算法，这个算法不仅考虑正向传播时网络中各层权值参数的调整，还考虑反向传播时网络中各层权值参数的调整，使得算法适用于多层网络，因此是目前广泛应用的神经网络学习算法之一。

多层网络模型如图 2.5 所示，它是在输入与输出层之间增加若干层(一层或多层) 神经元，这些神经元称为隐单元，它们与外界没有直接联系，但其状态的改变，则能影响输入与输出之间的关系。

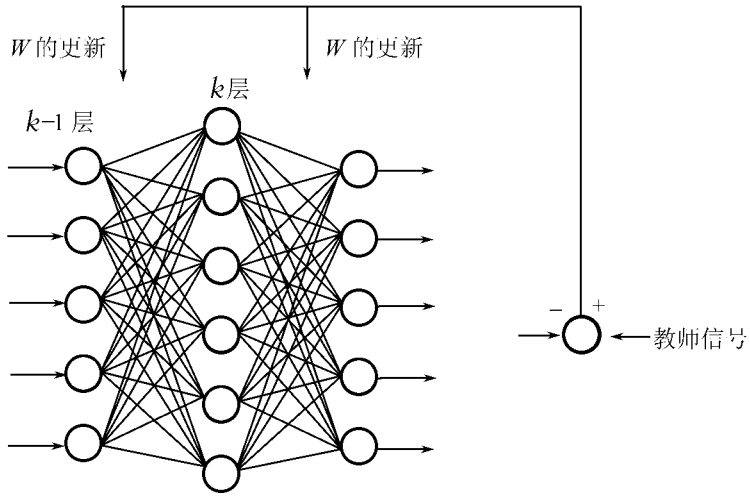


图 2.5　反向传播学习算法

设有 m 层神经网络，在输入层加上输入模式 P，并设第 k 层 i 单元输入的总和为 u_i^k，输出为 V_i^k，由 $k-1$ 层的第 j 个神经元到 k 层的第 i 个神经元的结合权值为 W_{ij}。各个神经元的输入与输出关系函数是 f，则各变量之间的关系为

$$V_i^k = f(u_i^k) \tag{2-10}$$

$$u_i^k = \sum_j W_{ij} V_j^{k-1} \tag{2-11}$$

这个算法的学习过程，由正向传播和反向传播组成。正向传播过程，输入模式从输入层经隐单元层逐层处理，并传向输出层，每一层神经元的状态只影响下一层神经元的状态。如果在输出层不能得到期望的输出，则转入反向传播，将误差信号沿原来的连接通路返回，通过修改各神经元的权值，使得误差信号最小。

首先，定义误差函数 r 为期望输出与实际输出之差的平方和：

$$r = \frac{1}{2} \sum_j (V_j^m - y_j)^2 \tag{2-12}$$

式中：y_j 是输出单元的期望输出，在这里作为教师信号，因此，这个算法是一种有教师示教的

学习算法；V_j^m是实际的输出，它是输入模式 P 和权值 W 的函数。

这种学习算法实际上是求误差函数的极小值。可利用非线性规划中的最快下降法，使权值沿着误差函数的负梯度方向改变，其权值 W_{ij} 的更新量 ΔW_{ij} 可由下式表示：

$$\Delta W_{ij} \propto \frac{-\partial r}{\partial W_{ij}} \tag{2-13}$$

或写成

$$\Delta W_{ij} = -\varepsilon \frac{\partial r}{\partial W_{ij}} \tag{2-14}$$

下面求$\frac{\partial r}{\partial W_{ij}}$，由于

$$\frac{\partial u_i^k}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}}\left(\sum_j W_{ij} V_j^{k-1}\right) = V_j^{k-1} \tag{2-15}$$

则

$$\frac{\partial r}{\partial W_{ij}} = \frac{\partial r}{\partial u_j^k}\frac{\partial u_j^k}{\partial W_{ij}} = \frac{\partial r}{\partial u_j^k} V_j^{k-1} \tag{2-16}$$

得

$$\Delta W_{ij} = -\varepsilon \frac{\partial r}{\partial u_j^k} V_j^{k-1} \tag{2-17}$$

设

$$d_j^k = \frac{\partial r}{\partial u_j^k} \tag{2-18}$$

则得此时的学习公式为

$$\Delta W_{ij} = -\varepsilon d_j^k V_j^{k-1} \tag{2-19}$$

式中：ε 是学习步长；为正的参数。下面讨论求 d_j^k的计算式：

$$d_j^k = \frac{\partial r}{\partial u_j^k} = \frac{\partial r}{\partial V_j^k}\frac{\partial V_j^k}{\partial u_j^k} \tag{2-20}$$

由式(2-10)可得，式(2-20)等号右边第二项为

$$\frac{\partial V_j^k}{\partial u_j^k} = f'(u_j^k) \tag{2-21}$$

为了便于求导数，取 $f(u_j^k)$ 为连续函数。例如取 $f(u_j^k)$ 为非线性 S 形函数：

$$f(u_j^k) = \frac{1}{1+\exp(-u_j^k)} \tag{2-22}$$

则

$$f'(u_j^k) = V_j^k(1-V_j^k) \tag{2-23}$$

再求式(2-20)等号右边第一项为$\frac{\partial r}{\partial V_j^k}$，对于式(2-20)可以有以下两种情况：

(1) 如果 j 是输出层(第 m 层)的神经元，$k=m$，则 y_j 是整个网络的期望输出，为定值，则由式(2-12)可得

$$\frac{\partial r}{\partial V_j^k} = (V_j^m - y_j) \tag{2-24}$$

则

$$d_j^m = V_j^m(1-V_j^m)(V_j^m - y_j) \tag{2-25}$$

(2) 如果 j 不是在输出层，而是在中间的隐单元层 k，则有

$$\frac{\partial r}{\partial V_j^k}=\sum_i \frac{\partial r}{\partial u_i^{k+1}}\cdot\frac{\partial u_i^{k+1}}{\partial V_j^k}=\sum_i W_{ji}\cdot d_i^{k+1} \tag{2-26}$$

则

$$d_j^k=V_j^k(1-V_j^k)\cdot\sum_i W_{ji}\cdot d_i^{k+1} \tag{2-27}$$

可以看出，k 层的误差信号 d_j^k 正比于下一层的误差信号 d_j^{k+1}。

综上所述，多层网络的训练方法是将某一样本加到输入层，这时，按前传法则，它将逐个影响下一层状态，最终得到一个输出 V_j^m，如果这个输出与期望值不符，就产生误差信号，然后通过如下公式改变权值：

$$\Delta W_{ij}=-\varepsilon\cdot d_j^k\cdot V_j^{k-1} \tag{2-28}$$

其中

$$d_j^m=V_j^m(1-V_j^m)(V_j^m-y_j) \tag{2-29}$$

$$d_j^k=V_j^k(1-V_j^k)\sum_l W_{jl}\cdot d_l^{k+1} \tag{2-30}$$

以上公式说明求本层 d_j^k 必需用到下一层的 d_l^{k+1}，因此，误差函数的求取是一个始于输出层的反向传播的递归过程。通过多个样本的反复训练并朝减少偏差的方向修改权值，最后达到满意的结果，故称为反向传播学习算法。

反向传播学习算法是解决多层网络的有效算法，但如果网络的层次较多时，计算量很大，收敛较慢，原则上也存在局部最小问题。

为改善收敛特性，可采用权值更新量 ΔW_{ij} 的修正公式，称为惯性校正法：

$$\Delta W_{ij}(t+1)=-\varepsilon d_j^k V_j^{k-1}+\alpha\Delta W_{ij}(t) \tag{2-31}$$

即后一次的权值更新适当考虑到上一次的权值更新值，其中 α 为调整变化量的参数，当学习步长取 ε 为 0.1 ～ 0.4 时，α 可取 0.7 ～ 0.9，视实际情况而定。有的文献[16] 称式(2-31) 右边第二项为惯性项，α 为惯性系数。

2.2.3 神经网络在城市交通信号灯系统建模和控制中的应用

1) 控制原理

城市交通系统是一个非线性、时变的复杂系统，城市交通系统的控制问题是一个复杂系统的建模和控制问题。

分析表明，城市交通系统的控制主要是对交通信号灯系统的控制。纵观国内外现有城市交通信号灯系统的控制方法，基本上都是通过建立精确的数学模型，确定色灯的信号的配时方案来实施控制的。20 多年的实践证明，其效果不尽如人意。原因是路口车辆到达的数学模型难以精确建立，学者们采用过均匀分布、泊松分布、正态分布、截尾正态分布、爱尔兰分布等，控制效果都不满意。

多次实践证明：一个有经验的警察站在路口指挥，其控制效果胜过英国的 SCOOT 动态交通控制系统。然而采用人工指挥方法警察容易疲劳，他们也不可能 24 小时都站在路中央，尤其是在炎炎夏日、三九严寒、大雨倾盆、风雪交加的时候，更是如此。分析警察指挥交通的过程，可概括为下面几步：首先用眼睛观察路口 4 个方向的车辆的排队情况；接着根据经验确定放行的方向和放行时间；当某个方向车辆放行时间超过某一最大值或者没有车辆到达，则立刻重新调整放行的方向和放行时间。笔者就是根据这一过程进行分析，实现交叉路口交通的智

能控制的。做法是:利用路口信号检测器代替警察的眼睛,将警察的经验存入计算机并建立知识库,由根据检测器检测的到达车辆数,确定信号灯的配时方案。具体流程见图 2.6。

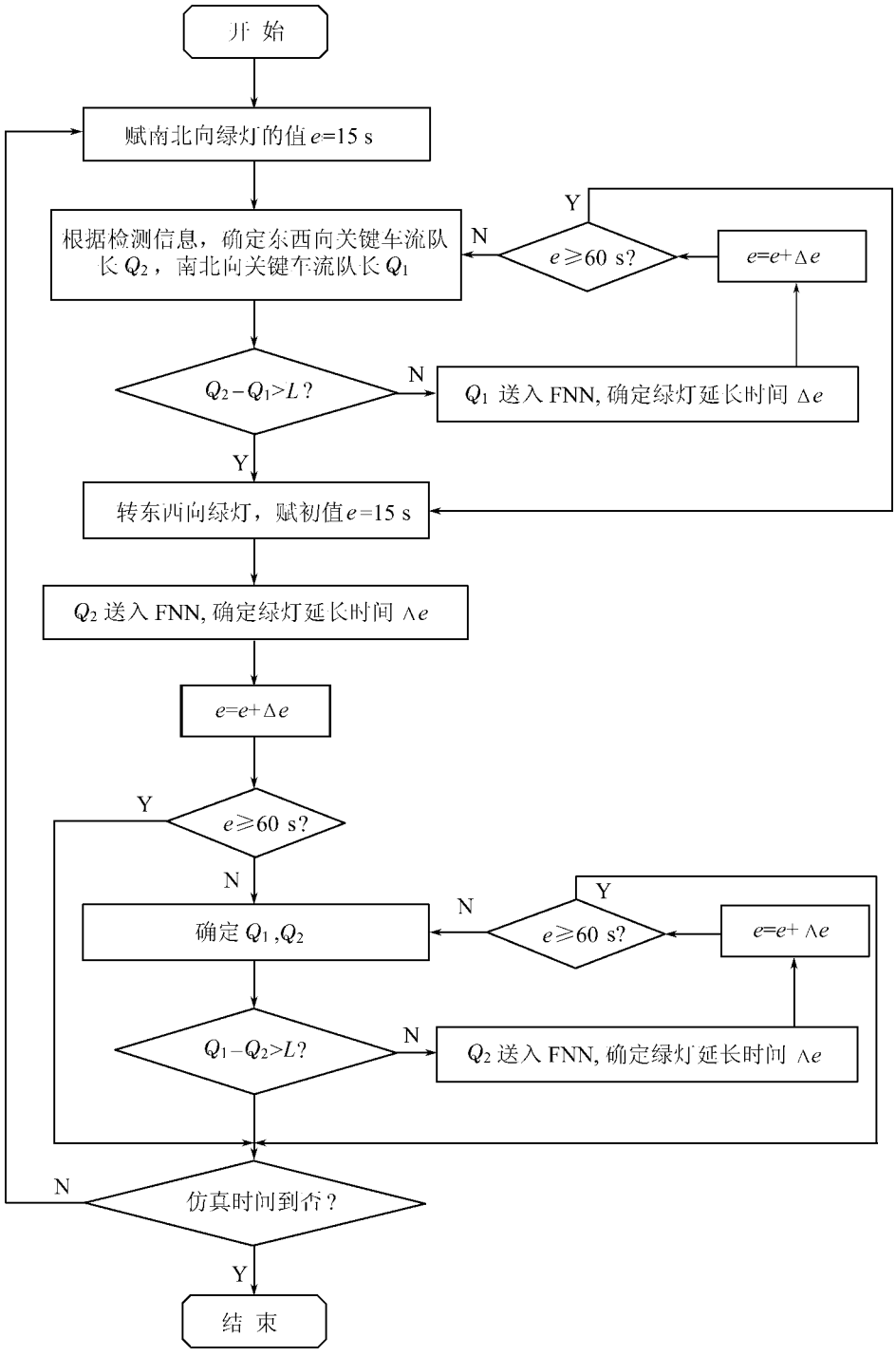


图 2.6　城市孤立交岔路口信号灯神经网络控制流程

图中有几点需要说明的是：

（1）FNN(Fuzzy Neural Network)，表示模糊神经网络。这里指用神经网络实现的模糊控制。

（2）关键车流，指路口某个方向（南北向或者东西向）到达路口的两股车流中，在路口排队长度最长的那股车流。

（3）色灯转换的原则是由东西方向和南北方向的关键车流的队长的差来确定。

2）FNN 控制器的设计

FNN 控制器的系统结构，采用模糊控制器的一般结构，它包括模糊化、模糊控制规则、模糊判别，每一模块用一个三层神经网络来实现。

在该控制方案中，需要模糊化的量是停车线前面排队车辆数。从模糊数学方面看，它相当于确定车队长度 Q 在不同语言值（很短、短、较短、中等、较长、长、很长）下的模糊隶属度。根据交通实际情况，使用表 2.1 所示的隶属度函数。

表 2.1　车队长度的模糊隶属度

模糊子集	语言值	论域 Q						
		1	4	7	10	13	16	19
Q_1	很短	1.0	0.7	0.3	0.0	0.0	0.0	0.0
Q_2	短	0.7	1.0	0.7	0.3	0.0	0.0	0.0
Q_3	较短	0.3	0.7	1.0	0.7	0.3	0.0	0.0
Q_4	中等	0.0	0.3	0.7	1.0	0.7	0.3	0.0
Q_5	较长	0.0	0.0	0.3	0.7	1.0	0.7	0.3
Q_6	长	0.0	0.0	0.0	0.3	0.7	1.0	0.7
Q_7	很长	0.0	0.0	0.0	0.0	0.3	0.7	1.0
—	—	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7

（1）在 FNN 的控制器中，模糊化模块用 1 个三层神经网络来实现，其中输入节点 1 个，隐层节点 4 个，输出节点 7 个，具体见图 2.7。

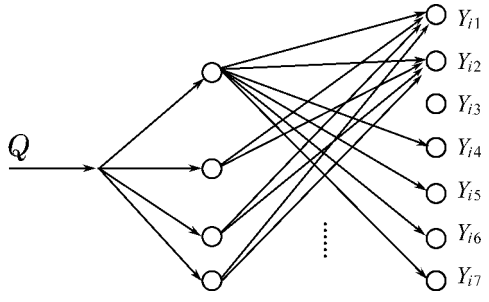


图 2.7　模糊化的三层神经网络结构

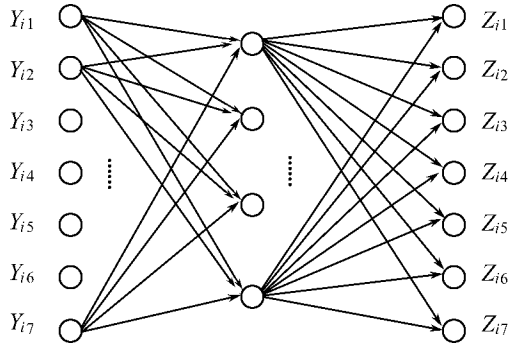


图 2.8　模糊控制规则的三层神经网络结构

网络的输入为车辆排队长度 Q，输出为相应于 Q 的 7 个不同语言值的模糊隶属度；用改进的 B－P 算法，在奔腾 133 微机上经过 1 200 次学习，运行 5 s，可获得各节点间的连接权，误差为 10^{-2}。

(2) 在 FNN 控制器中,模糊控制规则模块也用 1 个三层神经网络来实现,其中输入节点 7 个,隐层节点 4 个,输出节点 7 个,具体见图 2.8。

模糊控制规则,采用如下 7 条规则:

IF Q={很短}　　THEN G={很短}
IF Q={短}　　THEN G={短}
IF Q={较短}　　THEN G={较短}
IF Q={中等}　　THEN G={中等}
IF Q={较长}　　THEN G={较长}
IF Q={长}　　THEN G={长}
IF Q={很长}　　THEN G={很长}

以这些控制规则作为样本,用改进的 B-P 算法,在奔腾 133 微机上,经过 3 万次学习,可获得神经网络节点间的连接权,计算时间为 30 s,精度可达 10^{-2}级。

(3) 在 FNN 控制器中,模糊判别(也称清晰化、解模糊或反模糊化)模块也用 1 个三层神经网络来实现,其中输入节点 7 个,隐层节点 5 个,输出节点 1 个,具体如图 2.9 所示。

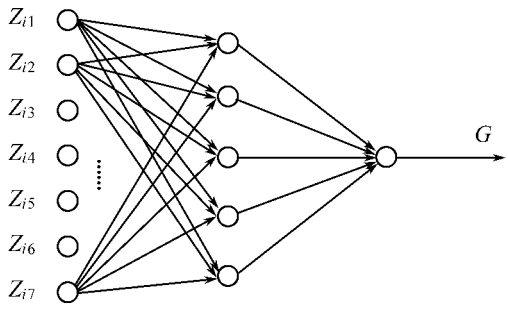


图 2.9　模糊判别的三层神经网络结构

绿灯延长时间的模糊隶属度,根据交通的实际情况,采用表 2.2 所示的隶属度函数。

表 2.2　绿灯延长时间的模糊隶属度

语　言　值	绿灯延长时间/s						
	2	8	14	20	26	32	38
很短	1.0	0.5	0.2	0.0	0.0	0.0	0.0
短	0.5	1.0	0.5	0.2	0.0	0.0	0.0
较短	0.2	0.5	1.0	0.5	0.2	0.0	0.0
中等	0.0	0.2	0.5	1.0	0.5	0.2	0.0
较长	0.0	0.0	0.2	0.5	1.0	0.5	0.2
长	0.0	0.0	0.0	0.2	0.5	1.0	0.5
很长	0.0	0.0	0.0	0.0	0.2	0.5	1.0
—	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7

以表 2.2 的数据为样本,用改进的 B-P 算法,在奔腾 133 微机上,经过 3 万次学习,计算时间为 25 s,可获得图 2.9 中节点间的连接权,精度可达 10^{-2}级。

将模糊化、模糊控制规则、模糊判别 3 个模块的 3 个三层神经网络连接起来,就构成了 FNN 控制器。

上述控制方案已经用 C 语言编成计算机仿真程序,在奔腾 133 微机上仿真 20 min,部分结果如表 2.3 所示。

表 2.3 双关键车流队长度 FNN 控制方法的部分仿真结果

项 目	周 期 号											
	1	2	3	4	5	6	7	8	9	10	11	12
南北向绿灯/s	15	15	36	15	15	42	39	15	40	39	45	46
东西向绿灯/s	15	15	15	15	38	43	46	38	41	45	47	15
周期/s	30	30	51	30	53	85	85	53	81	84	92	61
每周期延误/s	100	97	47	30	58	50	40	105	70	67	50	52
每车平均延误/s	3.29											

1998 年 12 月,该控制方案在南京市进香河路—珠江路交叉口实际应用。和定周期控制比较,该控制方案使车辆平均排队长度缩短 13 %,停车率下降 4.2 %,路口每车时间延误减少 4.8 %,效果令人满意。

2.2.4 Hopfield 模型[12]

1982 年,生物物理学家 Hopfield 提出一种新的神经网络模型,它实际上是一种离散的随机模型,由 N 个神经元构成互联网络,如图 2.10 所示。

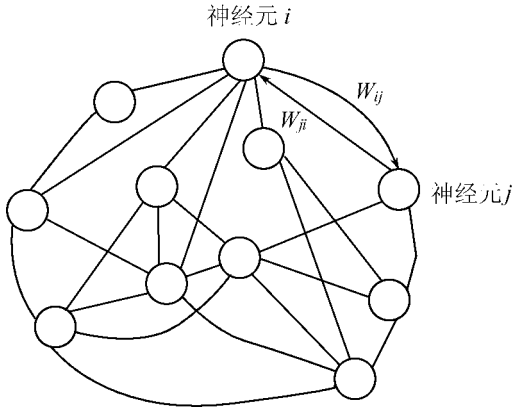


图 2.10 Hopfield 模型

神经元的输出 $V_i(t+1)$ 取离散值 1 或 0,每个神经元可按下述规则改变状态:

$$V_i(t+1)=\begin{cases}1, & \text{若} \sum_{i\neq j} W_{ij}V_j(t)+\theta_i>0\\ 0, & \text{若} \sum_{i\neq j} W_{ij}V_j(t)+\theta_i\leqslant 0\end{cases} \tag{2-32}$$

该模型可以工作在如下异步和同步两种工作方式:

(1) 串行(异步) 工作方式

在任一时刻 t,只有某一个神经元 i 的状态发生变化,而其他神经元的状态不变。

(2) 并行(同步) 工作方式

在任一时刻 t,部分神经元或全部神经元都同时改变状态。

如果网络从某一时刻以后,状态不再发生变化,则称该网络处于稳定状态,此时

$$V(t+\Delta t)=V(t),\quad \Delta t>0 \tag{2-33}$$

模型的条件是对称连接,$W_{ij}=W_{ji}$,并且无自身的反馈 $W_{ii}=0$,Hopfield模型的特点是具有联想功能。

对于 $W_{ij}=W_{ji}$ 的对称连接,Hopfield 引入“能量” 函数

$$E=-\frac{1}{2}\sum_i\sum_{j\neq i}W_{ij}V_iV_j-\sum_i\theta_iV_i \tag{2-34}$$

式中:V_i,V_j 是各个神经元的输出。神经网络的状态是各个单位神经元输出的组合,设第 m 个

神经元的输出由 0 变为 1，由式(2－32) 有

$$\sum_{j\neq m} W_{mj}V_j + \theta_m > 0 \tag{2-35}$$

下面以此为例，说明该神经元状态变化前后能量函数 E 值的计算。

设变化前($V_m = 0$) 的能量函数值为 E_1：

$$\begin{aligned} E_1 &= -\frac{1}{2}\sum_i \sum_{j\neq i} W_{ij}V_iV_j - \sum_i \theta_i V_i = \\ &-\frac{1}{2}\sum_{j\neq m} \sum_{j\neq i,m} W_{ij}V_iV_j - \sum_{i\neq m} \theta_i V_i - \sum_{j\neq m} W_{mj}V_mV_j - \theta_m V_m = \\ &-\frac{1}{2}\sum_{i\neq m} \sum_{j\neq i,m} W_{ij}V_iV_j - \sum_{i\neq m} \theta_i V_i - 0 - 0 \end{aligned} \tag{2-36}$$

变化后($V_m = 1$) 的能量函数值为 E_2：

$$\begin{aligned} E_2 &= -\frac{1}{2}\sum_i \sum_{j\neq i} W_{ij}V_iV_j - \sum_i \theta_i V_i = \\ &-\frac{1}{2}\sum_{i\neq m} \sum_{j\neq i,m} W_{ij}V_iV_j - \sum_{i\neq m} \theta_i V_i - \sum_{j\neq m} W_{mj}V_mV_j - \theta_m V_m = \\ &-\frac{1}{2}\sum_{i\neq m} \sum_{j\neq i,m} W_{ij}V_iV_j - \sum_{i\neq m} \theta_i V_i - \sum_{j\neq m} W_{mj}V_j - \theta_m \end{aligned} \tag{2-37}$$

由此，即可求得能量函数 E 值的变化量 ΔE：

$$\Delta E = E_2 - E_1 = -\left(\sum_{j\neq m} W_{mj}V_j + \theta_m\right) \tag{2-38}$$

根据式(2－35)，由于此时神经元的输出是由 0 变 1，括号中的值是正，则 $\Delta E < 0$。有趣的是，当按同样方法计算神经元的输出由 1 变 0 时，能量函数的变化值也是 $\Delta E < 0$。也就是说，任意一个神经元，当其输出发生变化时，能量函数值都将减小。或者说，在神经网络的状态变化过程中，能量函数 E 是单调下降的。

$$\Delta E = \Delta V_i \frac{\partial E}{\partial V_i} = \Delta V_i \left[\sum_{i\neq j}\left(-\frac{1}{2}W_{ij}V_j - \theta_j\right)\right] \leqslant 0 \tag{2-39}$$

由于 E 有界，系统必趋于稳定状态，并对应于 E 函数在 V 状态空间的局部最小值，如图 2.11 所示。

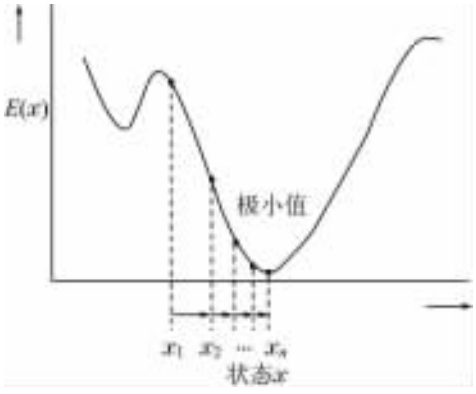


图 2.11　能量函数的极小值

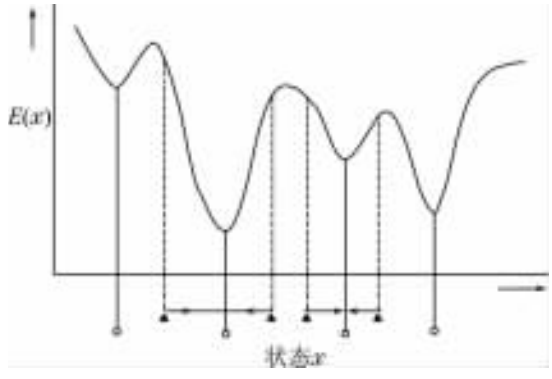


图 2.12　Hopfield 模型的联想

适当选取神经元兴奋模式的初始状态，则网络的状态就将按照上述的运算过程，到达初始状态附近的极小点。因此，如果贮存的样本是对应于网络的极小点(稳定点)，则当输入其附近的模式时，网络将“想起”极小点处的样本，如图 2.12 所示，亦即这种 Hopfield 神经网络模型是按照一种联想记忆装置进行工作的，具有联想记忆、模式识别、分类和误差自校正能力等智能功能。

2.3 基于 Agent 的建模方法

2.3.1 概述

随着科学技术的迅速发展和认识的深化，复杂系统建模问题的需求应运而生。研究复杂系统，就是研究它的演化规律。很自然的想法是建立系统的演化模型，以研究系统整体的涌现性。诚然，建立的演化模型可以为数学模型、知识模型或者是关系模型。但是，这种思路的基点在于研究系统的演化机制，事实告诉人们，这通常是非常困难的。那么，能否换一个角度来考虑问题呢？回答是肯定的。SFI 学者提出一种新的思路，其基点是：系统何时、何处出现涌现，出现什么样的涌现，是自下而上的，它们取决于构成系统的构件(Building Block)及其相互之间的非线性作用。SFI 学者称构件为主体(Agent)，并且用计算机程序来表述少数支配主体相互作用的规则，通过计算机仿真考察系统的涌现行为。在本书中，称 SFI 学者的这种方法为“基于 Agent 的建模方法”，它是生成论的研究方法。它建立在归纳逻辑之上，所用模型是由计算机程序表示的，根据模型让系统在计算机上产生、演化，让宏观整体行为由下而上、自然而然的涌现出来，使研究者能够直接观察系统的生成、演化过程，从观察现象中发现规律，提炼概念，形成洞见，建立理论。关于圣菲研究所的 CAS 理论，将在第 6 章第 7 节有进一步的介绍。

2.3.2 关于 Agent 的理念

1) 词汇 Agent

Agent 一词来源于拉丁语 agens，主要意思是“代理(人)”，即一个人代表另一个人或另一个组织去完成某种事情。国内文献中对 Agent 的翻译有多种，例如代理、智能体、主体、智能 Agent 等，这里建议不翻译而直接使用 Agent，这样表述更清楚。

2) Agent 的定义和属性

Agent 是一个运行于动态环境的具有较高自治能力的实体(它是一个自治体，可以是系统、机器等，软件 Agent 是一个计算机应用程序)。其根本目标是接收另外一个实体(可以是 Agent、用户、计算机程序、系统或机器等)的委托并为之提供帮助和服务，能够在该目标的驱动下主动采取包括社交、学习等手段在内的各种必要的行为，可感知、适应并对动态环境的变化进行适当的反应，它与其服务主体之间具有较为松散和相对独立的关系。

简单地说，Agent 是一种实体，而且是一种具有智能的实体。这种实体可以是智能软件、智能设备、智能机器人或智能计算机系统等等，甚至也可以是人。

目前，关于 Agent 还没有公认的定义，这里将之分为狭义 Agent 和广义 Agent。

狭义 Agent 的定义[17]是：一种具有自治性、社会能力、响应性、能动性的行为特征的智能实体。下面是对 Agent 的几个属性的说明：

(1) 自治性(Autonomy)

Agent 对自身状态和行为有一定程度的控制能力。在完成建模和仿真任务时,无需人类或其他 Agent 的直接干预。

(2) 社会能力(Social Ability)

当 Agent 认为合适时,能够与人类或者其他 Agent 进行交互以完成自身建模和仿真的任务,或者支持和帮助其他 Agent 完成它们要执行的任务。

(3) 响应性(Responsiveness)

Agent 能够理解自身所处的环境(包括物理世界、用户、协作的 Agent 和 Internet 等),可对环境变化作出及时和快速的响应。

(4) 能动性(Ability)

Agent 不仅能够对环境变化作出及时和快速的反应,而且显示出有意识的不失时机和目标导向的行为表现。

广义 Agent 的定义是:不仅是一种具有自治性、社会能力、响应性、能动性的行为特征的智能实体,而且还有知识、信念、责任、承诺等精神状态方面的特征。

2.3.3 Agent 的基本结构

Agent 是一个能够与外界自主交互,并拥有一定的知识和推理能力,能够独立完成一定任务的具有一定社会性的实体。从应用角度看,Agent 可以分为移动 Agent、界面 Agent、合作 Agent、信息 Agent、控制 Agent、路口 Agent、区域 Agent、协作 Agent、反应 Agent 等。不同类型的 Agent,基本结构相同,只是某些功能得到不同强化。在诸多属性中,自治性是 Agent 的关键属性。据此可以推断出 Agent 的一种基本结构,具体如图 2.13[18]所示。

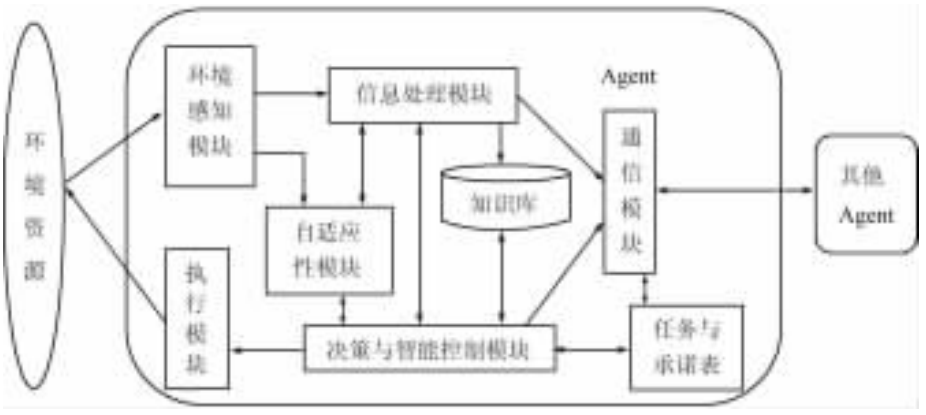


图 2.13 Agent 的一种基本结构

从图 2.13 可以看到,Agent 的基本结构包括:环境感知模块、执行模块、通信模块、信息处理模块、自适应模块、决策与智能控制模块、知识库、任务与承诺表等。通过提供一种形式化的定义,就能具体实现和用于系统中了。

图 2.13 可用于城市交通信号灯控制系统或电子交易业务(如拍卖等)的 Agent。如果用于研究系统的演化和涌现性的决策 Agent,也可以采用图 2.14 的结构[20]。

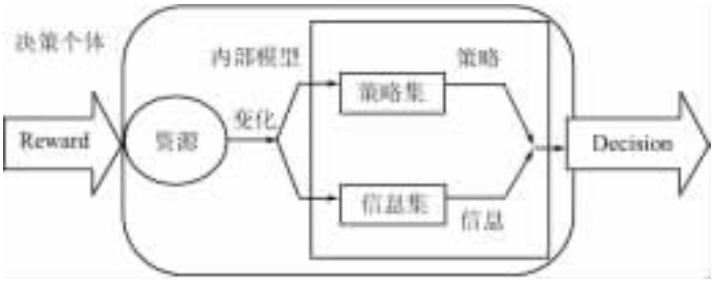


图 2.14　决策 Agent 的基本结构

复杂系统的研究专家 Goldenfeld 和 Kadanoff 曾指出：自然界即使在简单情况下也能产生复杂的结构，在复杂情况下也能遵循简单的规律。因而，无论哪种基本结构，只要适于人们的研究，满足人们的应用要求，即可采用。如果需要，还可另辟蹊径。

2.3.4　MAS 组织结构模式

1）多 Agent 系统定义

所谓多 Agent 系统，英语是 Multi-Agent Systems，缩写为 MAS，是指由多个 Agent 组成的一个较为松散的多 Agent 联邦，这些 Agent 成员之间相互协同，相互服务，共同完成一个任务。各 Agent 成员的活动是自治和独立的，其自身的目标和行为不受其他 Agent 成员的限制，通过竞争或磋商等手段协调和解决各成员 Agent 的目标和行为之间的矛盾和冲突。MAS 的数据和资源是分散的，每个 Agent 对于所要完成的任务拥有不全面的信息和能力，不存在全局的控制系统，任务的执行和计算是并行的。

在 MAS 中，各 Agent 是自治的，各 Agent 之间只能通过通信及对环境的改变来相互影响，各 Agent 自身的行为最终是由自己决定的。

MAS 是典型的分布式计算机系统，多个自主的 Agent 交互合作执行任务，满足一组目标，每个 Agent 有它自己的能力和与公共环境相联系的角色。

2）MAS 组织结构模式[18]

MAS 组织结构模式有 3 种视图：组织方式视图、系统开放性视图、系统结构关系视图。

（1）组织方式视图

MAS 是由多个 Agent 组采用集中或分布方式组成一个相互协作、相互作用、完成某些复杂的任务或目标的系统，如图 2.15 所示。最简单的 MAS 可以由一个集中式或分布式 Agent 组组成。小型的 MAS 可以采用集中式或分布式，再组织更大规模的 MAS，即允许 MAS 的嵌套。与传统模块化程序系统相似，MAS 系统结构可以通过由上到下和由下到上两种方式进行组织和实施。前一种方式是通过一个 MAS 中某些复杂的 Agent 成员进行扩展或者分解，由多个较为简单的 Agent 来实现该 Agent 成员的目标和任务，使该成员成为由多个较为简单的 Agent 组成的一个多 Agent 组，常用于新系统的设计和建立阶段。而后一种由下向上的方法是通过某种方式将现有的一些相对简单和小型的个体 Agent 或多 Agent 系统进行联合或集成，从而组成一个规模更大和功能更强的 MAS，常用于系统集成、升级和功能扩充等。根据 MAS 中 Agent 组的组成方式可以将 MAS 分为完全集中式 MAS、完全分布式 MAS 和混合式 MAS 3 类。

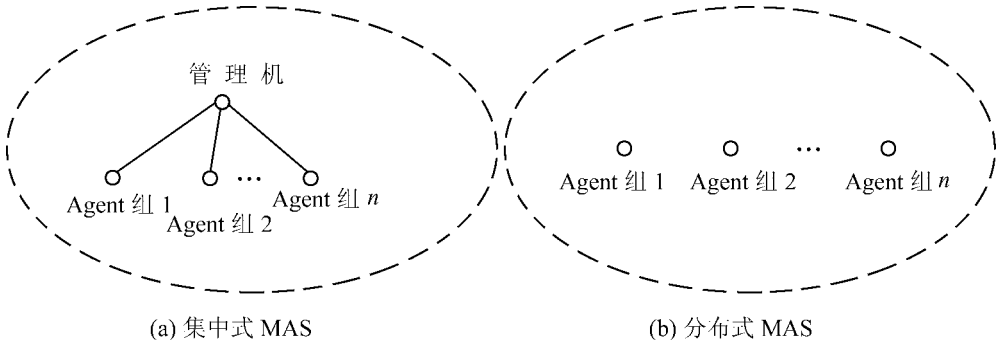


图 2.15　MAS 的基本结构

① 完全集中式 MAS

完全集中式 MAS 中的所有 Agent 成员组成一个集中式结构，且所有 Agent 成员组本身为集中式结构。多个集中式 MAS 再通过集中方式组合可以构成更大规模的集中式 MAS，如图 2.16 所示。在完全集中式 MAS 中，不同 Agent 组或不同管理服务机构的 Agent 成员之间的协作和对共享资源的访问需要，通过各自的上级并由其共同的管理服务机构进行控制和协调，因此 MAS 的嵌套不宜过深，以免影响系统成员之间的协作效率。

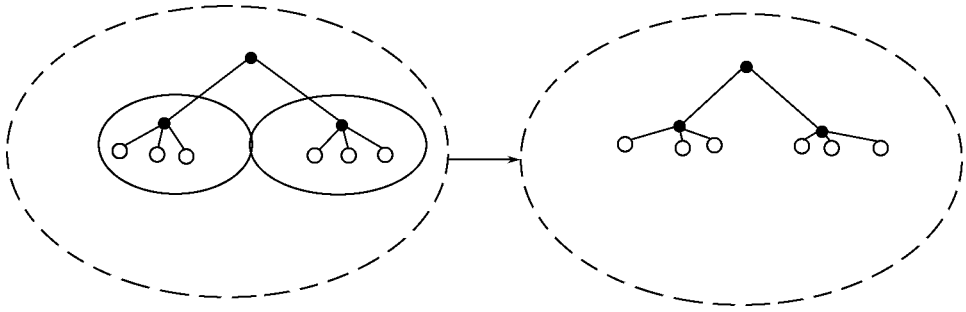


图 2.16　完全集中式 MAS 结构

② 完全分布式 MAS

完全分布式 MAS 中的所有 Agent 成员组构成一个分布式结构。多个分布式 MAS 再通过分布式组合可以构成更大规模的分布式 MAS，如图 2.17 所示。完全分布式 MAS 中可能存在多个中介服务机构，这些中介服务机构可以进行合作或合并，当某个 Agent 成员寻求协

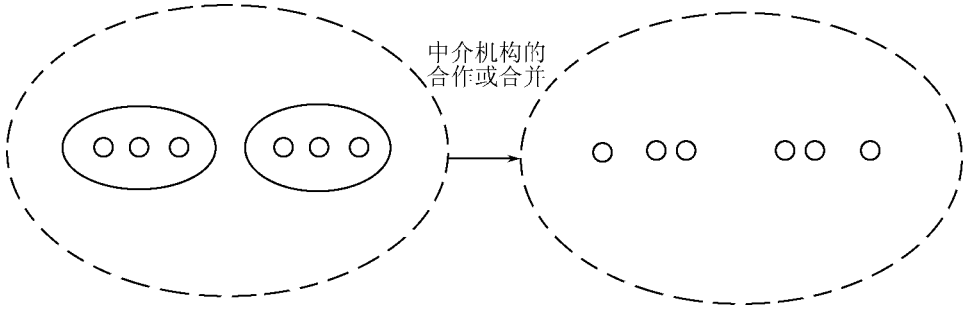


图 2.17　完全分布式 MAS 结构

作伙伴时，可以通过向多个中介服务机构查询或直接通过高层中介服务机构查询其他的中介服务机构来实现。

③ 混合式 MAS

由于分布式 Agent 组通常规模较大，且大多为开放的，其整体行为比较灵活且复杂，通常不与其他 Agent 组组成集中式结构，因此混合 MAS 一般由集中式或分布式 Agent 组构成一个两类结构共存的分布式结构。即两类不同类型的多个集中式 Agent 组采用分布形式构成一个混合式 MAS。

混合式 MAS 中拥有一个或多个管理服务机构，但这些管理服务机构只对部分 Agent 成员以某种方式进行统一管理，这些 Agent 之间的任务划分和分配、共享资源的分配和管理、冲突的协调、行为的一致性等均需要有管理服务机构的参与才能实现。而其 Agent 成员之间平等关系，它们的所有行为可以没有管理服务机构的参与，有其自身作出决策。

多个 Agent 组或小型的 MAS 组成一个大型的 MAS，其目标就是多个 Agent 组之间实现良好的协作关系，已完成更为复杂的任务和目标。因此，MAS 的研究重点是组与组的协作关系。

(2) 系统开放性视图

Agent 组根据是否允许 Agent 成员的自由进出，又可分为封闭式和开放式两类。封闭式 Agent 组是封闭的，不允许 Agent 成员的自由进出，其 Agent 成员是固定的。开放式 Agent 组是开放的，允许 Agent 成员的自由进出，其 Agent 成员不是固定不变的，而是动态变化的。对于开放的集中式 Agent 组，Agent 成员的退出或新的 Agent 的进入需要接受管理服务机构的管理。要进入该组的新的 Agent 首先向管理服务机构提出注册申请，管理服务机构受理并根据情况接受或拒绝其注册申请，同时对接受的新的 Agent 成员进行相应的授权并记录相关信息。该 Agent 成为该组的一个成员，与其他成员相互协作并接受管理服务机构的统一管理。对于开放式的分布式 Agent 组，Agent 成员的退出或新的 Agent 的进入相对比较自由，根据情况可以采取直接进入或经过认证机构的认证或向中介服务机构发布信息等方式加入一个分布式 Agent 组。

(3) 系统结构关系视图

Agent 组可以是静态的或动态的。所谓静态 Agent 组是指该 Agent 组的结构及其成员的作用和相互之间的关系是相对稳定和较为持久的，不因任务或环境的不同而发生改变；通常是针对一类目标和任务专门设计的结构。而动态 Agent 组的结构及其成员的作用和相互之间的关系是不稳定和临时的，随着任务或环境的不同而发生改变，动态结构的形成通常是各 Agent 成员相互竞争和系统演化的结果。动态 Agent 组通常是根据特殊的任务或目标临时组成的一个不稳定的结构，当该任务或目标完成后，该 Agent 组也可能随之解散，对于新的人物或目标将重新构造新的不同结构或不同成员的 Agent 组。

2.3.5 复杂系统实时仿真的多线程模型[22]

Agent 是具有特定完整功能的、独立的、高度智能化的个体，同时具有代理、智能、自主、交互、反应、主动、学习和自成长等特性。在一个复杂系统中，系统整体往往可由许多的Agent按一定规则结合而成，因而基于多 Agent 的复杂系统建模方法成为一种研究复杂系统的新方法。该方法利用 Agent 之间的局部连接准则和 Agent 的局部细节模型，建立复杂系统的整体模型，借助计算机来研究从小规模到大规模系统实现等方面的问题。在基于多 Agent 的复杂

系统的建模及其计算机实现中，智能体的数量和种类很多，每个 Agent 的功能可能会相当复杂，满足其相应功能的计算机程序的实现和运算难度较大，并且各 Agent 之间的相互通信比较频繁。由于对多 Agent 复杂系统的模拟对实时性的要求，各 Agent 必须具有相对独立性，对系统或其他 Agent 的同一变化，同时作出判断，并对自身作出调整，对外界作出响应。

要对多 Agent 复杂系统进行实时模拟，就必须解决如何有效处理众多 Agent 的并行计算问题。实现一个模拟系统的并行计算有多种手段。对于一个中等规模的基于多 Agent 的复杂系统的实时模拟，运用多线程技术实现计算的并行性，是一种经济、实用的手段，它比大型的并行计算系统实现起来要容易得多。

1）多线程技术

多 Agent 复杂系统实时模拟，首先要解决各 Agent 的并行计算和相互通信问题，这可以用多任务操作系统下的多线程技术来解决。线程是计算机应用程序实例中的一条执行路径，一个应用程序实例可以同时启动多个线程。从用户的角度看，程序中的线程是同时运行的。操作系统通过线程间反复地快速切换控制来达到并行计算这一效果的模拟（若通过多处理器或分布式并行计算方法，可实现真正的多任务并行计算）。若某个多 Agent 复杂系统在某个时间需同时完成多个任务，并且各任务之间需实时交换信息时，将每个任务放在不同的线程中，即可达到对该系统的实时模拟的目的。

在 Visual C++中，程序可以通过调用 AfxBeginThread()函数来创建一个线程：CWinThread * AfxBeginThread(AFX_THREADPROC, pfnThreadProc, LPVOID pParam, int nPriority=THREAD_PRIORITY_NORMAL, UINT nStackSize=0, DWORD dwCreatFlags=0, LPSECURITY_ATTRIBUTES lpSecurityAttrs=NULL)。该函数创建了一个新的 CWinThread 线程对象，并调用对象的 CreatThread 函数启动线程的执行。线程执行由 pfnThreadProc 指针指向线程控制函数，并通过参数 pParam 向线程控制函数传递其他线程的信息。多个线程可利用线程局部存储区访问同一线程函数或其他动态链接对象。同时可将多线程公用的代码以公共函数或动态链接库的形式予以单独封装。

2）基于多 Agent 的复杂系统中的 Agent 线程模型

复杂系统中的 Agent 线程模型是指将复杂系统中每个 Agent 的功能和行为放入一个相对独立的线程中运行，从而形成的 Agent 实时模拟模型。它是用来完成该 Agent 功能和行为的一条执行路径。Agent 线程由系统主线程创建后，其执行过程可以分为以下 3 步：

（1）系统主线程在创建 Agent 线程过程中，将系统环境变量传递给 Agent 线程控制函数 m_pAgentThread(UINT AgentThreadFunc, MULTIAGENTSYSTEMINFO * pParam)。其中，系统主线程将指向 MULTIAGENTSYSTEMINFO 系统环境数据结构的指针参数 pParam 传递给 Agent 控制函数 AgentThreadFunc()，并返回创建的 Agent 线程的指针。

（2）Agent 线程执行 Agent 控制函数 AgentThreadFunc(MULTIAGENTSYSTEMINFO * pParam)。该函数是 Agent 线程的功能和行为的核心，它主要实现以下功能：① 调用 Agent 系统探测函数和数据搜集函数；② 调用 Agent 数据分析和行为决策函数；③ 调用 Agent自学习和自适应函数；④ 调用 Agent 行为函数。

（3）通过同步机制，向系统主线程发送该 Agent 的公共数据。

3）基于多 Agent 线程模型的复杂系统实时模拟中的主线程

在基于多 Agent 线程模型的复杂系统实时模拟中，主线程起着整合各 Agent 线程的作用。主线程和各 Agent 线程的关系（以 4 个 Agent 为例）见图 2.18，它形成了基于多 Agent 线程模型的复杂系统模型。

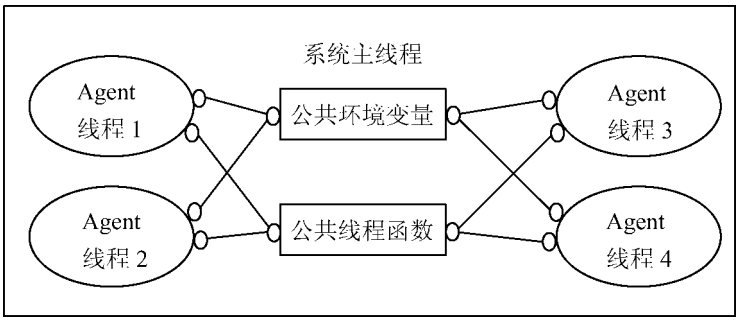


图 2.18　由主线程和 4 个 Agent 线程构成的复杂系统模型

主线程除了创建 Agent 线程外，还要完成以下功能：

（1）为 Agent 线程间通信提供环境。Agent 线程间通信可通过在主线程中定义全局变量获取指向同一公共对象的指针或利用主线程中的窗口消息等方法来进行。在过程中，主线程的全局变量和公共函数起着举足轻重的作用。全局变量是各 Agent 线程之间及其与主线程之间相互通信的重要手段，而将各 Agent 中能完成相似功能的代码块抽象成公共函数，放入主线程中，供所有 Agent 线程直接调用，这本身就大大降低了模拟过程的实现难度。

（2）提供多线程操作的同步机制。例如，在多个 Agent 线程同时读或写，其他 Agent 线程正在写入和更改变量时，必须根据线程的优先级，调用同步类对象的线程隔离函数，阻塞优先级低的线程操作，执行优先级高的线程操作，以实现各线程的同步机制。

4）实例分析

实时模拟具有“智能”的狗和兔子构成的追逐系统[22]。假定在系统中，狗追赶兔子，跑动速度已知，狗和兔子都不知道对方的跑动准则（决策方法）。其中，狗的跑动方法是随时沿着与兔子最近距离的直线向兔子跑，兔子的跑动方法是随时沿着与兔、狗最近距离连线垂直的方向跑。狗和兔子只有通过随时获取对方坐标点，然后根据各自的跑动准则来确定下一步的行动

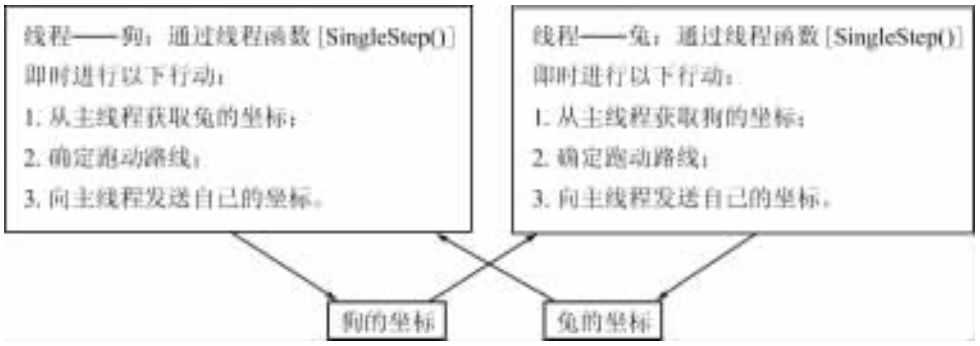


图 2.19　狗、兔追逐系统模拟

方向，兔和狗成为两个相互独立的 Agent。这里只给出实现模型（见图 2.19）和利用 Visual C++设计的整个动态模拟过程的最后结果截图（见图 2.20），图 2.20 中的坐标是狗和兔每决策 30 次后所显示的坐标。利用计算中生成的狗和兔的追逐曲线的坐标数组，并根据狗和兔的跑动准则，不难验证上述模拟结果的正确性。

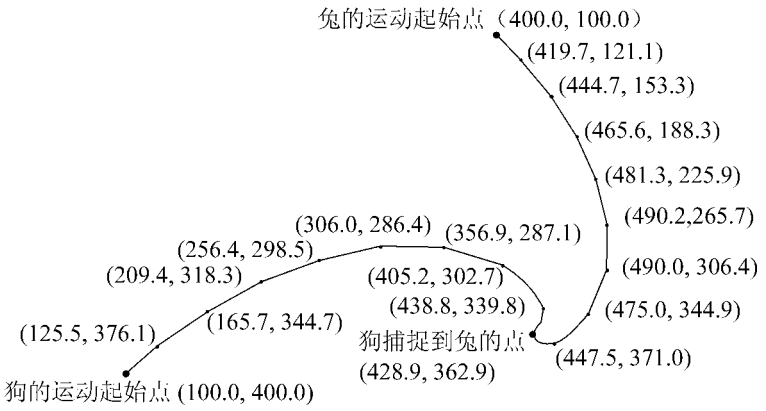


图 2.20　狗、兔追逐系统模拟结果

2.3.6　复杂系统多 Agent 分布仿真平台[23]

复杂系统与复杂性是 21 世纪的核心科学问题之一，涉及范围广泛，包括自然现象、工程、生物、经济、管理、军事、政治、社会等各个方面。例如，经济系统中的金融证券市场的运行涉及交易者的行为、交易者的交互、国家的经济走势、国家政策等诸多因素。正是由于诸多因素的随机性和交叉性，经典的金融市场理论对市场的行为难于进行解释，并对某项经济政策或者调整措施可能造成对金融市场的影响进行可信度的预测。据《经济观察报》报道，由于股评家的推荐而上套的股民占 29%，推荐股的有效性低于 50%。同样，战地对抗也是典型的复杂系统，敌我双方的作战单位、战略意图、战术思想、武器装备等相互制约，各种因素的影响程度难于正确描述和确定，其宏观行为也就难以进行推测。由此可见，研究复杂系统对国家的社会经济发展和军事技术都具有重要的意义。

对于复杂系统的复杂性研究存在着两类方法。第一类为朴素的还原论法，其基本观点为：复杂系统的运动形式和规律与物理系统运动形式和规律相似。可以按照物理系统建模来构造复杂系统的模型框架。例如，忽略很多因素的人口模型、经济发展预测模型、股评模型等。第二类方法为归纳推理方法，或称行为仿真方法。它把仿真目标定位在行为一级，根据复杂系统诸多因素的交互关系去建立系统同态模型，观测因素的个体行为来研究复杂系统的复杂行为趋向。由于复杂系统具有涌现性、非线性和复杂的关联性等特点，难于从上而下建立传统的数学分析模型。20 世纪 80 年代，在诺贝尔物理学奖获得者盖尔曼（Marray Gell-Mann）和安德逊（Philip Anderson）、经济学奖获得者阿罗（Kenneth Arrow）倡导下，人们企图通过科学融和的方法自下而上地从个别到整体、从微观到宏观来研究复杂系统的复杂性。具有代表性的有美国 SFI 的仿真平台 Swarm，Source Forge 的 Repost，芝加哥大学社会与经济动态性研究中心的 Ascape，麻省理工大学媒体实验室的 StarLogo，Sandia 国家实验室的 Aspen 等项目。

随着回路仿真技术的发展，形成了系统仿真、人工智能和分布计算相结合的分布智能仿真

(Distributed Intelligence Simulation)。相比之下,上述复杂系统平台结构比较简单,大多属于基于下一事件调度(Swarm Like)算法的单机系统,适用于规模不大的系统。为此,有必要对复杂系统计算机仿真进行基于分布智能仿真技术的新的研究与设计。

1) 复杂系统分布智能仿真平台

(1) 单机自适应 Agent 仿真平台

具有重大影响的美国 SFI 提出了用于复杂系统的自适应 Agent 仿真平台 Swarm,如图 2.21所示。平台由单机中若干进程或者线程组成的不同功能智能主体 Agent,包括控制时间推进和对象管理及通信的全局或局部 Agent、人机交互的显示/控制 Agent、完成网络交互的交互 Agent 以及具有自适应能力的反应 Agent。

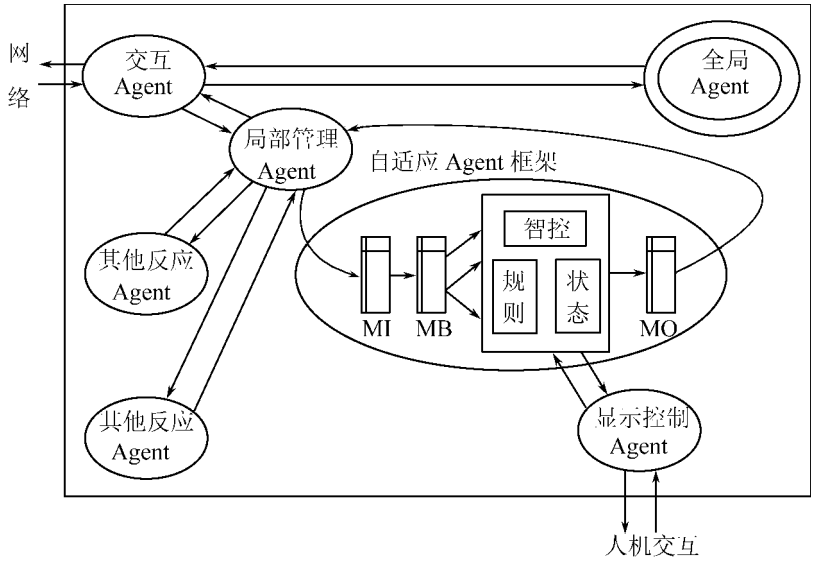


图 2.21　单机自适应 Agent 仿真平台框架结构

自适应反应 Agent 的结构框架同样在图中给出,它由信息接口 MI、信息缓冲 MB、反应体(规划、状态、智能控制)和行为输出 MO 组成。MI 和 MB 负责感知环境,反应体根据感知做出行动决策。例如,普通的证券交易人员(散户)的交易原则十分简单,仅有:

① 如果证券价格呈上升趋势,则出售呈上升趋势的若干证券。

② 如果证券价格呈下降趋势,则买进呈下降趋势的若干证券。

(2) 分布智能仿真(DIS)平台

随着分布计算机和人工智能技术的发展,SFI 的 Swarm 自适应仿真平台变得相对落后,特别受到仿真规模和慎思智能的限制。为此,设计了复杂系统的分布智能仿真(DIS),它的物理结构在图 2.22 中给出。

图 2.22 中 Agent 支撑库主要支撑 Agent 结构的创建和仿真结果统计分析,仿真支撑软件提供系统时钟同步和消息组播技术。整个分布智能仿真平台中,只有一个面向全局服务 Agent,其主要功能是提供各 Agent 全局唯一的 Agent 标识,创建和消除 Agent,提供 Agent 的物理位置和特征查询。每个结点机(Node)上有一个局部控制 Agent,它负责时间推进、对象管理、所有权变更和结点内部通信。

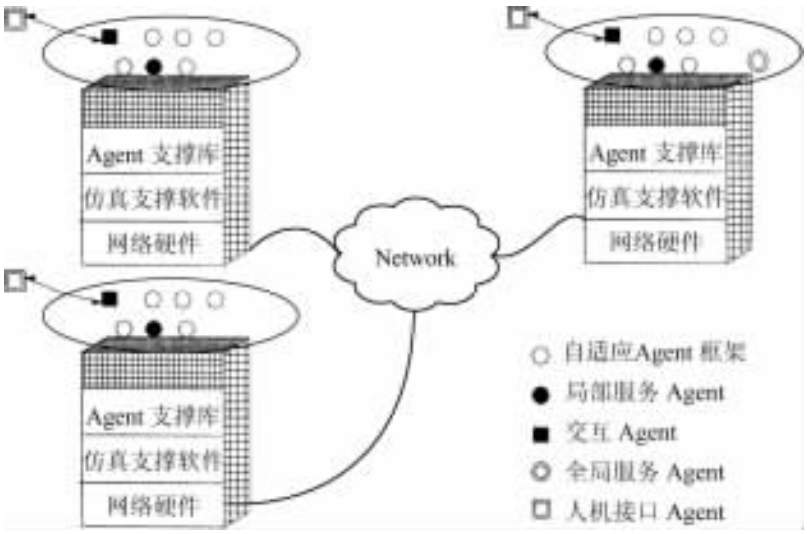


图 2.22　分布智能仿真平台框架结构

(3) 基于 HLA/RTI 的分布智能仿真平台

分布智能仿真平台实质上包括两大组成部分:一部分是具有智能的主体(自适应 Agent 以及显示/控制 Agent),它代表金融证券市场的散户、机构、公司、国家等复杂系统的基本元素。另一部分是完成基本元素关联的交互 Agent、局部 Agent 和全局 Agent 等,这实际上是分布交互仿真高层框架协议 HLA/RTI (IEEE1516)所规范的内容。分布智能仿真平台的结构如图 2.23 所示。

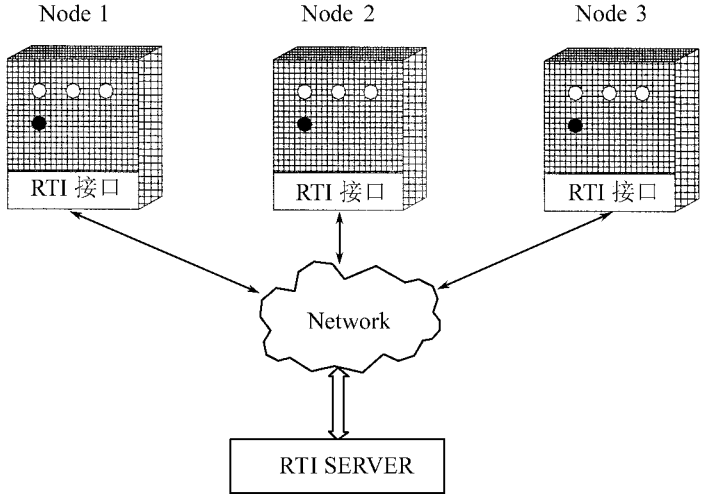


图 2.23　基于 HLA/RTI 的分布智能仿真平台框架结构

2) Agent 和 HLA/RTI 功能

(1) Agent 的理念和 Agent 的混合结构

分布式人工智能的研究目标是创建描述自然和社会系统的概念模型,并构造表现出一定的智能行为的 Agent,这正是复杂计算机仿真所需的基石。分布式人工智能一般分为分布问题求解(Distributed Problem Solving,DPS)和多主体系统(Multi-Agent Systems, MAS)。前者是纯粹

分布式问题求解，将问题分解为任务，并分布执行，是一种自上而下的设计系统，而后者符合自下而上的设计系统。在原理上，分散自主的 Agent 首先被定义，Agent 之间可能的交互是协作、竞争甚至是敌对。所以计算机仿真复杂系统理所当然地将 Agent 作为首选研究对象。

智能 Agent 可以分为慎思 Agent（Deliberative Agent）和反应 Agent（Reactive Agent）。德国学者 Fischer、Muller 和 Pischel 将反应、慎思和协作能力结合起来[25]，构成混合 Agent。最著名的混合 Agent 结构是 MAPE 混合结构，如图 2.24 所示。

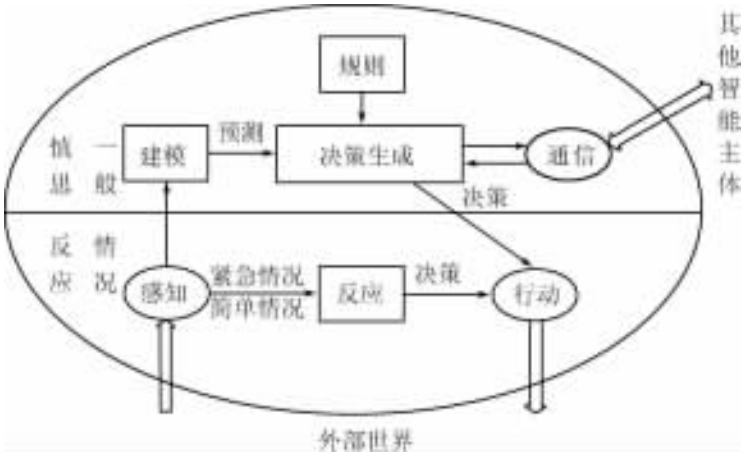


图 2.24　MAPE 混合结构

MAPE Agent 除了要保持对紧急情况的及时反应，还要使用一定的策略对其行为做出规划，进而通过对世界的其他 Agent 的建模分析预测未来的状态，以及通过通信实现和其他 Agent的协商，以达到做出慎思决策。以证券市场为例，在股市大起大落时，买卖股票要当机立断，及时作出反应。而在股市平稳时，有时间比较全面地了解形势，甚至与相关单位协商，评估预测，再作出行动决策。

（2）HLA/RTI 功能

RTI 是 HLA 框架的核心，其目的是将仿真应用与底层通信及基础功能相分离。如图 2.25所示，在复杂系统仿真过程中，所有的 Agent 按照 HLA 接口规范所要求的方式同 RTI 进行数据交换，实现 Agent 之间的相互操作。

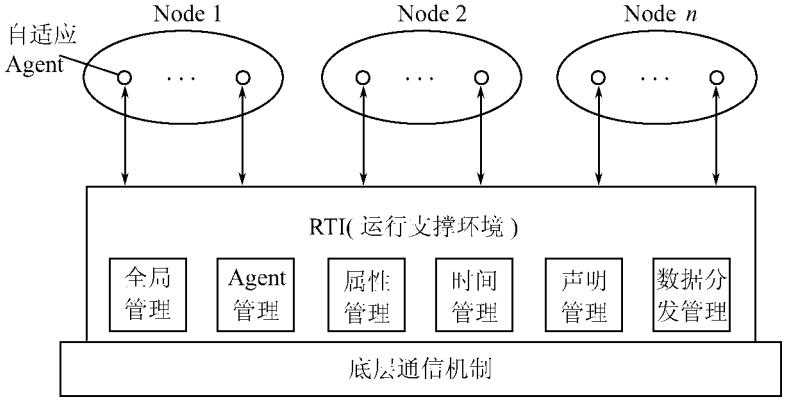


图 2.25　HLA/RTI 功能

RTI 的综合管理提供六大服务功能。

① 全局管理：主要提供全局服务，包括创建、动态控制、修改 Agent 执行等。

② 声明管理：提供 Agent 声明，希望提供和接收的交互信息。

③ 数据分发管理：目的是限制网络上交互的流量，通信时采用组播技术。

④ Agent 管理：提供 Agent 的控制服务。

⑤ 属性管理：对 Agent 对象属性所有权控制、迁移。

⑥ 时间管理：目的是保证以适当的方式和顺序进行时间推进。

(3) 基于 HLA/RTI 的多 Agent 分布仿真平台

多 Agent 仿真环境需要的大部分功能都可以由 HLA 框架中的 RTI 服务提供，包括了 Agent控制、信息分发、事件处理和时间推进。其中对实体状态更新和明确事件的处理有强大的支持，时间管理机制也相当灵活。采用 HLA/RTI 作为 MAS 支撑环境，可以充分利用 HLA 的建模范型以及 RTI 的强大功能，使 Agent 分布仿真环境更加通用和灵活。多 Agent 仿真环境的功能需求和 RTI 服务的功能对应如表 2.4 所示。

表 2.4　多 Agent 仿真环境的需求与 RTI 服务对照表

多 Agent 仿真环境的需求	RTI 服务
Agent 控制	联邦管理 FM/对象管理 OM
Agent 的加入、退出	注册(register)/发现(discover)类
信息分发	数据管理 DM/数据分发管理服务 DDM
Agent 的信息	发布(publish)/订购(subscriber)对象类
环境信息	更新(update)/映射(reflect)属性
事件处理	数据处理 DM/数据分发管理服务 DDM，发送(send)/接收(receive)交互
时间	时间管理服务 TM
通信语言(ACL)	

从 Agent 的角度看，完全采用 HLA/RTI 服务作为多 Agent 仿真平台还不完备，其中最重要的是缺少对 Agent 之间通信的支持。基于 Agent 的应用需要跟踪远程 Agent 的能力和状态，Agent 之间需要支持特定通信。在 Agent 的决策过程中，Agent 之间的通信服务十分重

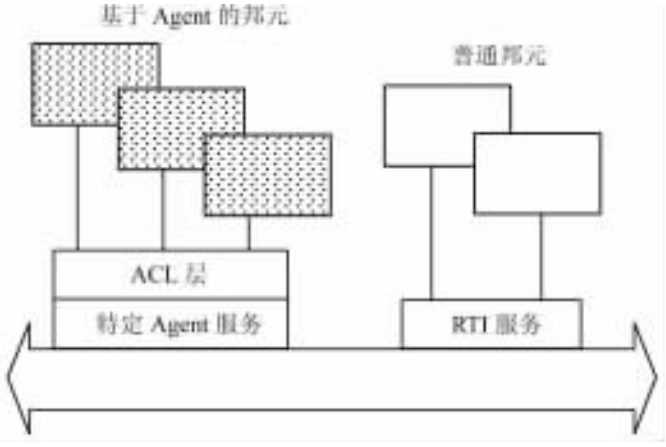


图 2.26　在 RTI 基础上增加 ACL 层

要，这时 Agent 不必关心这些事件具体是什么，而 Agent 之间的通信，需要 Agent 通信语言（ACL）来定义（见图 2.26），而在 HLA/RTI 框架中，无法进行具体描述。因此利用 HLA/RTI 中的数据订购不能满足 Agent 之间的通信需求。

要解决上述问题，可以采用在 Agent 和 RTI 软件之间增加中间层（ML）的办法（见图 2.27）。中间层 ML 的作用，是对所有作为 HLA 仿真邦元加入到仿真环境中的自适应 Agent 之间的通信进行封装和解释，从而在 HLA/RTI 仿真环境中增加了对 Agent 之间的通信以及对 ACL 的支持。

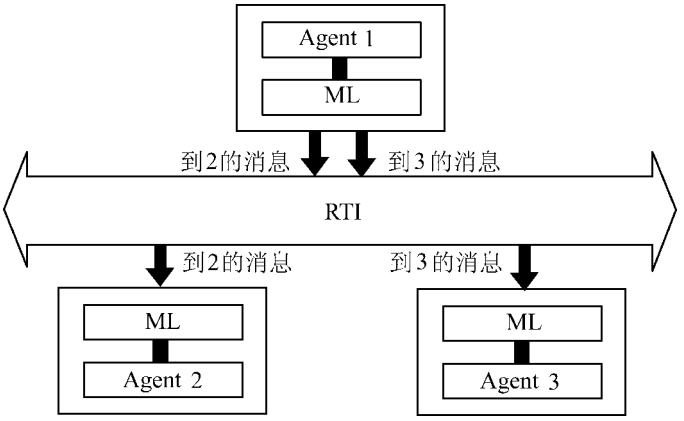


图 2.27　利用数据分发管理控制数据流量

由于经过中间层封装的信息，Agent 邦元之间的数据流通量可能会有所增加，这时可以采用 RTI 的路由空间限制联邦中数据分发的数量。在 RTI 中通过定义对发送或接收数据的约束，邦元能保证只得到需要的信息。例如，一个电台的邦元在不同的频段发出和接收信息，路由空间可以通过所使用的频段来约束发出的信息，只有在这个频段上接收的邦元能接收到相关信息。

为限制信息，联邦中每个 Agent 可以使用其邦元 ID 作为约束参数创建一个路由空间。若发送者想要所有 Agent 接收到信息，则不使用路由空间的概念。

综上分析，HLA/RTI 功能特征适用于多 Agent 分布仿真环境，如果是 HLA/RTI 进一步扩展，可以方便地构建多 Agent 分布仿真平台。HLA 的高级体系结构是美国国防部建模与仿真办公室（DMSO）于 1995 年 10 月提出的。

2.3.7　多 Agent 在区域交通信号灯协调优化中的应用[26]

近年来，随着经济的高速发展，城市交通拥挤现象日益严重，尤其在一些特大城市交通阻塞、交通事故更是频发。为了解决交通拥挤与阻塞问题，从 20 世纪 60 年代开始，发达国家就开始进行交通控制研究，这些研究的主要方法是通过建立交通流的数学模型，运用运筹学和最优控制论来控制和优化整个交通系统。尽管这些研究已经得到广泛应用，但由于交通系统的复杂性、随机性以及交通流模型的局限性等因素，控制优化的效果不尽如人意。具有自主性、移动性、协作性等特点的 Agent，为交通这个复杂巨系统的管理与控制，开辟了新的解决途径。

区域协调是多 Agent 城市交通控制系统的一个主要问题，它是指在交通中心的宏观调控作用下，根据不同的交通流量，最大限度地发挥路口之间互补的优势，均衡每个路口之间（包括高速公路与高速公路、高速公路与普通公路、普通公路与普通公路等不同层次规模的路口）的

良好协作，然而路口之间是互相影响、互相作用的，因此为实现区域协调必然会引起路口之间出现一定程度的冲突。如何解决这些冲突便是一个急需解决的重要问题。博弈论是研究理性的主体之间冲突及合作的理论，它研究主体的行为是如何相互影响的，主体是如何在相互作用中作出自己的行为选择和行为决策的。用博弈论的方法来分析问题，使问题的研究不仅局限与站在某个决策方的立场上找出针对他方的对策，更重要的是在分析这些决策过程时能够发现各方相互制约、相互作用的规律，从而导出合理的结果并用以解决相应的实际问题。

下面介绍的是中国科学院李振龙、陈德望两位学者成果[26]。他们以多 Agent 技术为基础，利用博弈论的相关知识提出了交通信号的区域优化的协调模型，利用此模型能够使各个路口充分合作，使一定区域内的交通信号得到协调优化，从而较好地解决交通拥挤问题。

1）系统结构

基于多 Agent 的城市交通控制系统通过 Agent 间的相互协调与合作来解决大规模的复杂的交通问题，使城市交通保持通畅的状态。路口是交通系统的基本控制单位，一个区域由多个路口及连接这些路口的道路组成，一个城市的交通系统由多个区域组成。多 Agent 交通控制系统主要包括两类 Agent：区域 Agent（Area-Agent）和路口 Agent（Cross-Agent）。每个区域有一个区域 Agent，每个路口有一个路口 Agent，每个区域内有若干路口 Agent，系统结构如图 2.28 所示。

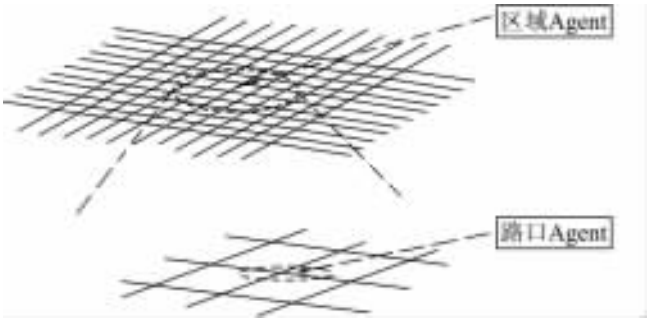


图 2.28　系统结构示意图

每个 Agent 主要包括 3 个层次：通信层、协作层和控制层。通信层由通信模块构成，主要完成与其他 Agent 的信息交互；协作层由学习机、推理机、规则库和知识库 4 部分组成，主要完成和其他 Agent 进行协调并生成最终决策的功能；控制层由控制模块构成，主要完成指导控制任务，并将控制任务的信息通过通信层传递给其他 Agent。

所有的路口 Agent 有着共同的全局目标——使得区域交通畅通，每个路口 Agent 有自己的局部目标——尽量使本路口交通畅通；所有的区域 Agent 有着共同的全局目标——使得整个交通系统畅通，每个区域 Agent 有自己的局部目标——尽量使本区域交通畅通。路口 Agent之间、路口 Agent 与区域 Agent 之间、区域 Agent 之间是相互影响、相互作用的，因此，每个 Agent 的决策必然要受到另一些 Agent 策略选择的影响，Agent 之间必然会发生一定程度的冲突。下面应用博弈论的相关知识，分析 Agent 之间的合作与冲突机理，建立区域协调模型来协调相互关系，实现博弈均衡，使交通顺畅。

2）协调模型

（1）基本符号和概念定义

为建立区域协调模型，引入以下符号和定义：

① $\boldsymbol{Q}_i(t)$ 表示 t 时刻等候在第 i 路口的车辆数量的向量。$\boldsymbol{Q}_i(t)=\{Q_{i,\mathrm{E}}(t),Q_{i,\mathrm{S}}(t),Q_{i,\mathrm{W}}(t),$

$Q_{i,N}(t)\}$，$Q_{i,E}(t)\mid Q_{i,S}(t)$、$Q_{i,W}(t)$、$Q_{i,N}(t)$ 分别表示 t 时刻等候在第 i 路口的东、南、西、北 4 个方向的车辆数。

② $\boldsymbol{Q}_i$ 表示第 i 路口的车辆数阈值的向量，$\boldsymbol{Q}_i=\{Q_{i,E},Q_{i,S},Q_{i,W},Q_{i,N}\}$，$Q_{i,E}$、$Q_{i,S}$、$Q_{i,W}$、$Q_{i,N}$ 分别表示不同方向等候车辆数的阈值，阈值可根据具体情况进行修改。

③ A 表示博弈协调中的行为和决策 Agent，它的目的是通过选择行动策略以最大化自己的支付(效用) 水平，是所有 Agent 的集合，$A=\{\text{Agent }1,\text{Agent }2,\cdots,\text{Agent }n\}$。

④ I 表示每个 Agent 拥有的信息，包括其他 Agent 的特征和行动策略的信息。

⑤ S 表示 Agent 的所有可能的策略或行动的集合，一个 Agent 的全部可行策略称为它的策略空间。每个 Agent 有一个纯策略的有限集，$S=\{$东西直行，南北直行，东西双左拐，南北双左拐$\}$。

⑥ U 表示 Agent 获得的利益，是指在既定策略组合条件下 Agent 的得失情况，即在一个特定的策略组合下得到的效用水平。

⑦ Nash 均衡。

设有 n 个 Agent 的博弈描述为 $G=\{A,S,U\}$，若此问题中战略组合 $S^*=\{s_1^*,\cdots,s_i^*,\cdots,s_n^*\}$ 是一个纳什均衡，则必须满足

$$U_i(s_i^*,s_{-i}^*)\geqslant U_i(s_i,s_{-i}^*),\quad \forall s_i\in S_i \tag{2-40}$$

式中：s_i^* 表示第 i 个 Agent 选择的战略；s_{-i}^* 表示除 i 之外的所有 Agent 的策略组成的向量；U_i 表示第 i 个 Agent 的效用水平；S_i 表示第 i 个 Agent 的策略空间。

(2) 协调模型

在整个交通中，在一定的时刻，车辆就是那么多，路口就是那么宽，由此就会引发路口相互之间的冲突，一个 Agent 的决策会影响其他 Agent 的决策，同时，也受其他 Agent 决策的影响，因此，一个 Agent 在做决策时，应考虑其他 Agent 可能采取的战略来决定自己的战略。通过 Agent 间的相互通信，每一个 Agent 对其他 Agent 的特征(策略空间、效用函数等) 有完全的了解，这决定了 Agent 间的协调过程是基于完全信息的博弈过程。

一次博弈协调，定义为

$$G=\{A,I,S,U\} \tag{2-41}$$

式中：$A=\{\text{Agent }1,\text{Agent }2,\cdots,\text{Agent }n\}$，$I$ 为每个 Agent 拥有的信息，$S=\{$东西直行，南北直行，东西双左拐，南北双左拐$\}$，U 为 $Q_i(t)$ 的收益函数。

每个 Agent 依据它所拥有的信息 I，在 S 中选择合适的策略，通过不断的协调，使它们的盈利达到纳什均衡，即

$$U_i(t\mid s_i^*,s_{-i}^*)\geqslant U_i(t\mid s_i,s_{-i}^*),\quad \forall s_i\in S_i \tag{2-42}$$

整个协调过程分为 3 个层次，下层是路口 Agent 与其相邻的路口 Agent 之间的协调；中间层是区域 Agent 与路口 Agent 之间的协调；上层是区域 Agent 与其相邻的区域 Agent 之间的协调，如图 2.29 所示。

图 2.29　协调层次图

(3) 协调算法

第一步，路口 Agent 的车辆排队数超过阈值，则向相邻的路口 Agent 发出请求。

第二步，相邻的路口 Agent 响应请求，并构建如图 2.30 所示的博弈树(博弈树分支上的

字母代表 Agent 的策略，框图内字母代表 Agent 的盈利，图中只给出 3 组盈利值，其他类推），通过搜索博弈树，根据式(2－42)寻找 Nash 均衡。

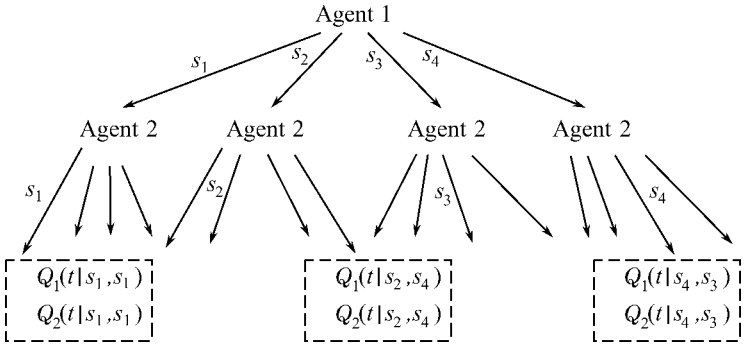


图 2.30　博弈树

第三步，如果 Nash 均衡存在，则 Agent 的行动策略就是达成 Nash 均衡时的策略，每个 Agent按照该策略控制路口器，本次协调结束。如果没有 Nash 均衡，则向该路口 Agent 所在的区域 Agent 发出请求。

第四步，区域 Agent 响应请求，对其所管辖的路口 Agent 进行博弈协调，寻求 Nash 均衡，如果 Nash 均衡不存在，则该区域 Agent 向相邻的区域 Agent 发出请求。

第五步，相邻的区域 Agent 响应请求，进行博弈协调，寻求 Nash 均衡，如果 Nash 均衡不存在，协调失败，则每个 Agent 保持原先的策略不变。

3）计算案例

用图 2.31 所示的一个简单交通区域来说明上面的协调算法。Agent 1、Agent 2、Agent 3 分别是 3 个路口 Agent，它们由区域 Agent 管辖。为分析方便，每个 Agent 的策略集 S 简化为{s_1 东西直行，s_2 南北直行，s_3 东西双左拐，s_4 南北双左拐}，路口间的距离为 200 m。图中的数字为

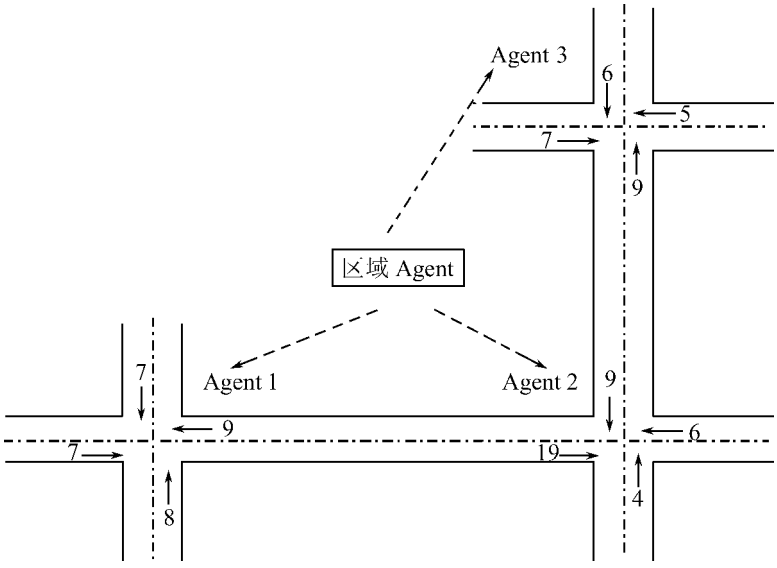


图 2.31　简单交通区域

t_0 时刻各个路口各个方向的排队车辆数。

Agent 2 在 t_0 时刻西口的排队车辆数 $Q_{2,W}(t_0)=19$，大于阈值 Q_W（设 $Q_W=15$），则 Agent 2 向 Agent 1 发出请求，Agent 1 响应请求并进行博弈协调，博弈协调的收益即是排队车辆数，它是 Agent 1、Agent 2 博弈协调的目标和得失情况的体现，收益的多少取决于它们的策略组合。假设每个周期，到达路口每个方向的车辆数为 λ，每个绿灯时间通过路口的车辆数为 μ，并假设 $\mu=1.5\lambda$，那么路口 1 和 2 在 t_1 时刻的排队车辆数是由 Agent 1、Agent 2 在 t_0 时刻采取的策略决定的，通过构建博弈树可以知道有以下 4 种情况：

① Agent 1 和 Agent 2 都选择 s_1，两个路口的停车数分别为 $15+2\lambda$，$13+3.5\lambda$。

② Agent 1 选择 s_1，Agent 2 选择 s_2，两个路口的停车数分别为 $15+2\lambda$，$25+2.5\lambda$。

③ Agent 1 选择 s_2，Agent 2 选择 s_1，两个路口的停车数分别为 $16+2.5\lambda$，$13+2\lambda$。

④ Agent 1 和 Agent 2 都选择 s_2，两个路口的停车数分别为 $16+\lambda$，$25+\lambda$。

Agent 1、Agent 2 在 t_0 时刻所采取的策略导致在 t_1 时刻两个路口的排队车辆数用下面的收益矩阵表示

$$
\begin{array}{cc}
 & \text{Agent 2} \\
 & \begin{array}{cc} s_1 & s_2 \end{array} \\
\text{Agent 1}\begin{array}{c} s_1 \\ s_2 \end{array} & \begin{pmatrix} (15+2\lambda,13+3.5\lambda) & (15+2\lambda,25+2.5\lambda) \\ (16+2.5\lambda,13+2\lambda) & (16+2.5\lambda,25+\lambda) \end{pmatrix}
\end{array}
$$

每个 Agent 根据其拥有的信息进行决策，决策问题由收益矩阵决定，从收益矩阵可以知道：当 $\lambda<12$ 时，Agent 1 选择策略 s_1，Agent 2 选择策略 s_1 达到纳什均衡；当 $\lambda>12$，Agent 1 选择策略 s_2，Agent 2 选择策略 s_2 达到纳什均衡。所以，当 $\lambda<12$ 时，Agent 1 的行动策略是 s_1（东西直行），Agent 2 的行动策略是 s_1（东西直行）；当 $\lambda>12$ 时，Agent 1 的行动策略是 s_2（南北直行），Agent 2 的行动策略是 s_2（南北直行）；Agent 1、Agent 2 根据其相应的策略控制路口。

以上介绍的是下层协调，如果路口 3 南口的车辆排队数 $Q_{3,S}(t_0)$ 等于 18 而不是图 2.31 所示的 9，那么 $Q_{3,S}(t_0)$ 大于阈值 Q_S（设 $Q_S=15$），则 Agent 3 向 Agent 2 发出请求，而此时 Agent 2 也正向 Agent 1 发出请求，这种情况下 Agent 3 和 Agent 2、Agent 2 和 Agent 1 之间的纳什均衡很难得到。于是它们向区域 Agent 发出请求，区域 Agent 响应请求并进行博弈协调，开始中间层协调。路口 Agent 的收益用图 2.32 的博弈树表示。

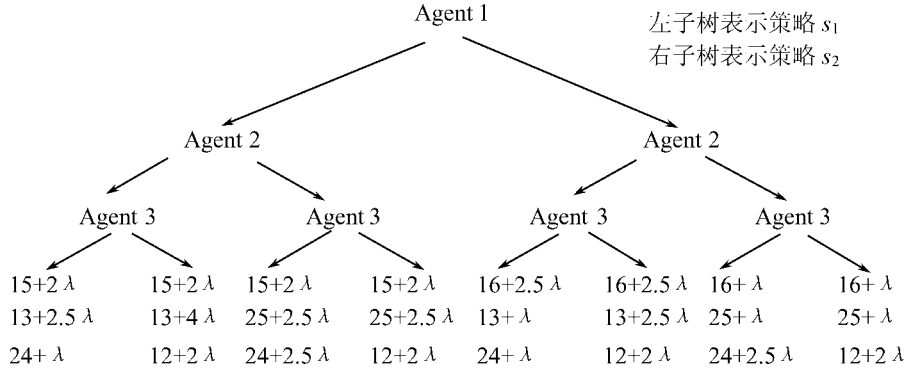


图 2.32　路口 Agent 的收益

通过搜索博弈树可知，当Agent 1 选择策略 s_2，Agent 2 选择策略 s_2，Agent 3 选择策略 s_2 时

达到纳什均衡，Agent 1、Agent 2、Agent 3 的策略都是 s_2 南北直行，虽然路口 2 西口的车辆排队数有所增大，但从整个交通区域的角度看，Agent 1、Agent 2、Agent 3 选择策略 s_2 是优于其他策略的。

上层协调与下层协调类似，将区域 Agent 当作路口 Agent 来对待，在此不再赘述。

2.4 基于 CGP 的建模方法[27]

2.4.1 概述

确定性的系统有其内在的随机性——混沌，而随机性的系统又有其内在的确定性——涌现。混沌和涌现是复杂系统的两个显著特征，是自然界普遍存在的现象。人们对涌现的研究还很不够，有关涌现的理论还不完善，还没有给出精确的定义。但是，许多学者都从他们各自的角度对涌现进行研究。美国学者 Holland、Casti 等人对涌现进行了较系统的研究。Casti 把涌现定义为“是从许多参与者的交互行为中产生的整个系统的行为，它不能从系统元素的独立行为知识中预测到”。Holland 通过对大量的复杂适应系统的研究发现，在复杂适应系统中，即使系统元素的结构、性质、行为规则等都很简单，也可能产生极为复杂的整体结构、性质和行为。因此，他把系统的这种从简单的局部产生复杂的整体的现象称为涌现，即“涌现是从小原因中产生的大结果”。

涌现的表现形式具有明显的上下文相关性，但是表现形式不同的涌现具有一些共同的特征，使得它们能够被识别出来。一般来说，涌现具有不可预测性、相关性、动态性、不变性、宏观约束性。涌现存在于不同领域的各种系统中，这些领域都用它们各自不同的专业术语对涌现进行了描述。但是，从根本上讲，产生涌现的系统应该具备非线性、自组织、非平衡和多个吸引子这几个条件。当系统含有具有适应性或学习能力的元素时，涌现的产生就更为复杂了。

对上述的产生涌现的系统所应具备的条件，构成复杂性理论基本框架的几个学派已经分别对它们进行了较为深入的研究。SFI 在复杂适应系统理论研究中，用涌现一词表示从系统中的多个 Agent 的相互作用中产生的宏观层次的模式。对涌现的研究表明，当复杂系统的元素结构能够比基于各个部分的解释对系统的动力学特性作出更多、更深层的解释的时候，应该考虑应用有关涌现的理论。但是在讨论涌现时，应注意区分真正的涌现和 Holland 所说的“偶然出现的新奇事物”。两者的区别在于“偶然出现的新奇事物”不一定具有真正的涌现所具有的适应能力。

涌现是复杂系统中普遍存在的现象，现实世界中，很多的现象都可以归结为涌现。但是由于产生涌现的系统的组成元素、系统结构以及动力学特性等千差万别，使涌现的表现形式具有明显的上下文相关性，这给人们从不同领域中的涌现现象总结共同的规律带来了很大的困难。人们一直在试图寻找一种能够描述和研究不同系统中的涌现的通用方法，以便从中找到它们的共同规律。下面介绍的有约束的产生过程（Constrained Generating Procedure，简记为 CGP）模型就是这样一种研究不同系统中涌现的共同或相似规律的方法。CGP 模型具备描述产生涌现的不同系统的组成、结构和演变过程的能力，通过比较不同系统的 CGP 模型及相应的模型实验结果，有可能从中发现它们的共性，从而得到关于涌现的一般性结论。在参考迟妍、谭跃进、邓宏钟 3 位学者发表的成果[27]的基础上，介绍 CGP 模型及其应用实例。

2.4.2 CGP 模型

1) CGP 模型概述

CGP 模型是由 Holland 提出的用于涌现研究的一种模型,它指的是有约束的产生过程。所谓的产生过程指的是作为组成模型的基本单位的机构(mechanism)产生了模型的动态行为,同时,因为模型描述的是系统动态变化的性质,所以称之为"过程";所谓有约束指的是机构本身的状态变化规则和对机构之间相互作用关系的约束将系统可能的动态行为限制在特定的集合之内。

机构是 CGP 模型的基本元素。机构对行为或信息作出反应,并将此作为输入,产生新的结果即输出包括行为或信息。CGP 模型是以机构为结点,以机构间的相互作用关系为有向边的网络。机构有状态,状态按照一定的规则不断地变化,而整个网络的状态与各个机构的状态直接相关,因而也是不断变化的。网络的状态在变化的过程中表现出来的,不能从机构的状态变化规则中推导出的规律就是通常所讲的涌现现象。因此,CGP 模型是一种研究各种涌现的有力工具。

2) CGP 模型的建立过程

CGP 模型的基本思想是:系统由一些基本的元素组成,它们相互联系、相互作用、相互制约的关系缩小了系统可能的状态集,而它们状态的演变过程就是系统的发展过程。该模型的建模过程如下:

(1) 确定组成系统的基本元素——机构。一个系统中可能含有多个机构,它们可能属于同一种类,也可能属于不同的种类。对机构应先定义它的状态,然后在此基础上定义它的转换函数。一个(组)转换函数定义了一(多)种机构。

(2) 定义机构的连接方式,构成网络,这个网络就是 CGP 模型。一种比较简单的方法就是将一个机构的输出作为另一个机构的一个输入即可将它们连接。对系统中的所有机构重复这一过程即可构造整个网络。

(3) 用各个机构的状态定义整个 CGP 模型的状态,比如说用各个机构的状态组成的向量作为整个模型的状态。

(4) 从基本的机构开始,逐层定义更复杂的机构,再用这些更高层的机构去定义 CGP 模型。通过这种方式可以描述产生涌现的系统的一个主要特征:层次性。

3) 各种模型与 CGP 的对应关系

涌现普遍存在于各种复杂系统中,而对各种复杂系统也有各种各样的模型,这些模型与 CGP 模型的关系如表 2.5 所示。从表 2.5 可以看出,各种看似不同的模型都可以统一于 CGP 模型的概念之下。

表 2.5 各种模型与 CGP 模型的关系

模 型	机 构	约 束
博弈模型	博弈者	博弈规则
网络模型	结点	结点连接和转换函数选取规则
元胞自动机	元胞	元胞的连接和状态变化规则
神经网络	神经元	神经元的连接和转换函数
Multi-Agent Systems	主体	主体的行为规则

2.4.3 CGP 模型的应用实例

下面以一个关于采集机器人的仿真实验来说明如何用 CGP 模型研究涌现。

1）基本思想

一个机器人是一个复杂的系统，要设计制造一个具有一定功能的机器人是一项复杂的系统工程。而从已有的关于涌现的研究结果中可以知道，即使系统组成元素的局部规则很简单，也有可能产生复杂的全局行为。根据这个结论，考虑是否可以让一个由多个结构简单、功能简单的机器人组成的机器人组来完成由一个复杂机器人完成的任务。在以上的思路和 CGP 建模思想的指导下，同时在 SFI 开发的 Swarm 仿真平台上建立了一个具有采集功能的机器人组的 CGP 模型，并且进行了仿真实验。

2）模型描述

该模型中共有 5 个结构和功能完全相同的机器人，它们的任务是将随机分布在一个二维的 30×30 的正方形网格内的 64 个相同的物体堆积成堆。建模过程如下：

（1）确定机构

模型中的机构共有两类：机器人和物体。

机器人的属性包括：

① 当前位置 $p(x,y)$，x 与 y 分别为机器人在网络中的横坐标与纵坐标。

② 运动方向 d 机器人可以在上、下、左、右 4 个方向上作直线运动。

③ 最大推力 f 以能推动的最多的物体数表示，模型中假设每个机器人一次最多能推动 3 个物体。

这 3 个属性构成的向量 $(p(x,y),d,f)$ 是机器人的状态向量。

物体的属性只有它的当前位置 (x,y)，因此它的状态就是它的当前位置。

机器人的状态转换函数定义如下：

① 如果机器人正前方的 3 个网格内的物体数少于等于 3，则机器人将它们向前推进一格，机器人在原运动方向上前进一格。

② 如果机器人或它所推动的物体已经到达网格的边界，则机器人放下物体，随机选择一个新的运动方向。

③ 如果一个机器人遇到其他的机器人，它随机选择一个新的运动方向。

物体的状态转换函数定义为：如果物体被机器人推动，则它的当前位置为原位置在机器人运动方向上的下一网格，否则保持原值不变。

（2）定义机构的连接方式

设机器人的当前位置为 (x,y)，它的运动方向为向右，则网格 $(x+1,y-1)$、$(x+1,y)$、$(x+1,y+1)$ 内的物体数和网格 (i,j) 内的机器人是否存在的状态为机器人状态转换函数的输入，其中 $i \in \{x+1,x+2\}$，$j \in \{y-3$ 到 $y+3$ 的整数$\}$。当机器人的运动方向为左、上、下时，相应的状态转换函数的输入与此类似地定义。

物体如果在某个机器人的探测范围之内，则它与该机器人建立连接关系，否则无连接关系。而物体之间无连接关系。

(3) 定义 CGP 模型的状态

整个 CGP 模型的状态为所有的机器人和物体的位置构成的向量。

模型中只有一个层次的机构:机器人和物体。

3) 仿真实验

仿真实验开始之初,5 个机器人和 64 个物体随机分布在 30×30 的二维网格内。将这个模型共运行了 50 次,每次改变机器人和物体的初始分布,保持其他的参数不变。图 2.33、图 2.34、图 2.35 分别给出了第 1 次、第 25 次和第 50 次仿真进行到第 5 000 个仿真周期时的模型状态。从这 3 幅图中可以看出,从不同的初始条件开始,在经过大约 5 000 个仿真周期之后,模型都表现出了相似的动态稳定状态,即物体基本上都分布在网格区域的 4 个角上。之所以称这种状态为动态稳定的是因为模型中物体聚集在区域的 4 个角上的状态基本保持不变,但是各个物体和机器人的状态不断地变化,机器人仍在不断地运动,物体从一堆被移动到另外一堆。可以认为,在这种存在随机因素(机器人和物体的初始位置的分布的随机性)的条件下,多次仿真中表现出来的物体堆积在网格的 4 个角上的相似的规律是一种涌现行为。

图 2.33　第 1 次仿真在第 5 000 个仿真周期的模型状态

图 2.34　第 25 次仿真在第 5 000 个仿真周期的模型状态

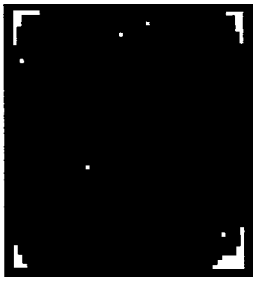


图 2.35　第 50 次仿真在第 5 000 个仿真周期的模型状态

2.5　遗传算法[28~31]

2.5.1　概述

遗传算法(Genetic Algorithms)是在 20 世纪 70 年代初期由美国密歇根(Michigan)大学的 Holland 教授发展起来的。1975 年,Holland 发表了第一本比较系统论述遗传算法的专著 *Adaptation in Natural and Artificial System*。

遗传算法是基于"适者生存"规律的一种高度并行、随机和自适应的优化算法,它将问题的求解表示成"染色体"的适者生存过程,把搜索空间(欲求解问题的解空间)映射为遗传空间,即把每一个可能的解编码为一个向量,称为一个染色体(chromosome)或个体。向量的每个元素称为基因(genes)。所有染色体组成群体(population)或集团。并按预定的目标函数(或某种评价指标),计算其适应值。根据适应值对诸染色体进行选择、交叉、变异等遗传操作,剔除适应值低的染色体,留下适应值高的染色体,从而得到新的群体。由于新群体的成员是上一代群体的优秀者,继承了上一代的优良性态,因而明显优于上一代。遗传算法就这样反复迭代,向着最优解的方向演化,直至满足某种预定的优化指标。所以遗传算法是一种随机优化算法,但

它不是简单的随机比较搜索，而是通过对染色体的评价和对染色体中基因的作用，有效地利用已有信息来指导搜索。简单遗传算法的主要步骤可描述如下：

① 随机产生一组初始个体构成初始种群，并评价每一个体的适应值(Fitness Value)。

② 判断算法收敛准则是否满足。若满足则输出搜索结果，否则执行以下步骤。

③ 根据适应值大小以一定方式执行复制操作。

④ 按交叉概率 p_c 执行交叉操作。

⑤ 按变异概率 p_m 执行变异操作。

⑥ 返回步骤②。

简单遗传算法的 3 个基本运算：选择运算、交叉运算和变异运算。

选择运算又称繁殖、再生或者复制运算，用于模拟生物界去劣存优的自然选择现象。它从旧种群中选择出适应性强的某些染色体，放入匹配集，为染色体交叉和变异运算产生新种群做准备。适应值越高的染色体被选择的可能性越大，其遗传基因在下一代群体中的分布就越广，其子孙在下一代出现的数量就越多。

选择运算虽然能够从旧种群中选择出优秀者，但不能创造新的染色体。因此，遗传算法的开创者提出了交叉运算。交叉操作通过交换两父代个体的部分信息构成后代个体，使得后代继承父代的有效模式，从而有助于产生优良个体。交叉操作的方法比较多，有部分匹配交叉、顺序交叉、循环交叉等。

变异运算用来模拟生物在自然遗传环境中由于各种偶然因素引起的基因突变，以很小的概率随机的改变遗传基因(表示染色体的数字串的某一位)的值。在染色体以二进制编码的系统中，它随机的将染色体的某一个基因由 1 变成 0，或由 0 变成 1。若只有选择和交叉，而没有变异操作，则无法在初始基因组合以外的空间进行搜索，使演化过程的早期就陷入局部解而终止演化过程，从而使解的质量受到很大限制。通过变异操作，可确保群体中遗传基因类型的多样性，以使搜索能在尽可能大的空间中进行，避免在搜索中丢失有用的遗传信息而陷入局部解，获得质量较高的优化解答。

遗传算法不同于传统的搜索和优化方法，它的主要特点在于[32]：

① 自组织、自适应和自学习性(智能性)。应用遗传算法求解问题时，在编码方案、适应度函数及遗传算子确定后，算法将利用演化过程中获得的信息自行组织搜索。由于基于自然的选择策略为“适应者生存，不适应者被淘汰”，因而适应度大的个体具有较高的生存概率。通常，适应度大的个体具有更适应环境的基因结构，再通过基因重组和基因变异等遗传操作，就可能产生更适应环境的后代。遗传算法的这种自组织、自适应特征，使它同时具有根据环境变化来自动发现环境的特性和规律的能力。自然选择消除了算法设计过程中的一个最大障碍，即需要事先描述问题的全部特点，并要说明针对问题的不同特点算法应采取的措施。因此，利用遗传算法的方法，可以解决那些复杂的非结构化问题。

② 遗传算法的本质并行性。遗传算法按并行方式搜索一个种群数目的点，而不是单点。遗传算法是内在并行的(Inherent Parallelism)，即遗传算法本身非常适合大规模并行运算。最简单的并行方式是让几百甚至数千台计算机各自进行独立的演化计算，等到运算结束进行比较，选取最佳个体。这种并行处理方式对并行系统结构没有什么限制和要求，可以说，遗传算法适合在目前所有的并行机或分布式系统上进行并行处理，而且对并行效率没有太大影响。

③ 遗传算法不需要求导或其他辅助知识，而只需要影响搜索方向的目标函数和相应的适

应度函数。

④ 遗传算法强调概率转换规则，而不是确定的转换规则。

⑤ 遗传算法可以更加直接的应用。

⑥ 遗传算法对给定问题，可以产生许多的潜在解，最终选择可以由使用者确定。

2.5.2 遗传算法的改进及其应用

1）遗传算法的改进

传统的简单遗传算法存在许多缺陷，在处理一些复杂系统优化问题时存在整体收敛性差，种群容易早熟，从而使种群陷入局部最优点，不能收敛到全局最优。

由于它存在着这些局限性，研究者提出了改进方法来提高遗传算法的性能，如杂和算法[33]。每种改进算法都在某一方面改进了遗传算法的性能，因此，本书在这里考虑将这些方法加以综合，进而从整体上改进遗传算法性能的作用，仍然称之为杂和遗传算法。具体改进有如下几点：

（1）保护优秀个体

将每一代种群中的最佳个体（适应值最大的个体）保留下来，不参加杂交和变异过程，使之直接进入下一代，这样可以防止优秀的个体由于选择、交叉或变异中的偶然因素而被破坏。

（2）标记淘汰算子取代原有变异算子

在进行演化计算时，都要将每一代种群竞争下来的最优个体与停机准则匹配，看是否已经达到全局最优解。因此，每进行一轮遗传算子操作后，真正有用的只是保留下来的当前最优个体，而应对所有其他适应值较小的个体设置淘汰标记，禁止其重复参与竞争，但仍允许其参加本轮的交叉和变异操作，以用来提供下一代新的个体和指导下一代的搜索。对于包含约束条件的寻优问题，还应在计算适应值之前将不满足约束条件的个体也设置上淘汰标记。这样，每一次新的竞争前，在全局范围内随机产生未参与竞争的新的个体以替代已经淘汰的个体进入下一代竞争，可达到提高收敛速度的目的，同时也增加了整个种群的多样性。

（3）模糊大变异操作

大变异操作是当某代种群中所有个体集中在一起时，以一个远大于通常的变异概率的概率执行一次变异操作，这种大变异操作能够随机、独立地产生许多新的个体，从而使整个种群脱离“早熟”。在大变异操作中要判断当前代中所有个体的集中程度并保证当前代中具有最大适应度值的个体不被大变异操作破坏掉。

个体的集中程度是采用式(2－43)进行判断的。

$$a * F_{\max} < F_{\text{avg}} \tag{2-43}$$

式中：a 为密集因子，在[0.5,1.0]内取值；$F_{\max}$ 为当代种群的最大适应度函数值；F_{avg} 为当代种群的平均适应度函数值。

显然，密集因子的选择会影响大变异操作的调用。如果密集因子 a 值选得过小，大变异调用过多，就会影响算法的稳定性，甚至退化为随机搜索；如果密集因子 a 值过大，大变异操作调用较少，就不能起到维持群体多样性，提高算法性能的作用。因此，本书采用模糊大变异操作，对于不同的集中程度进行不同的大变异操作。

通过判定式(2－44)中 m 值的大小来确定演化中种群的集中程度，不同的集中程度采用不同变异概率 p_{big} 的大变异操作，如式(2－45)所示。

$$m = (f_{\max} - f_{\text{avg}})/f_{\max} \tag{2-44}$$

$$p_{\text{big}} = \begin{cases} 0.015, & 0.4 \leqslant m < 0.5 \\ 0.020, & 0.3 \leqslant m < 0.4 \\ 0.030, & 0.2 \leqslant m < 0.3 \\ 0.035, & 0.1 \leqslant m < 0.2 \\ 0.040, & m < 0.1 \end{cases} \tag{2-45}$$

(4) 动态自适应交叉、变异

通常交叉概率 p_c 和变异概率 p_m 取(0,1)之间的常数,而且在整个算法迭代过程中保持不变。然而,在生物演化过程中,生存环境处在不断变化之中,生物适应环境的能力也不同,如果在整个算法过程中将 p_c 和 p_m 取为一个固定的常数,就不免带有一定的局限性,因此,p_c 和 p_m 应该与生物演化代数和个体本身的适应值有关。算法在开始迭代时,交叉算子起主要破坏作用,为了保持较高的破坏概率,产生更多的新模式,应增加 p_c。当算法接近于收敛时,变异算子起主要的破坏作用,为使算法收敛,被破坏的模式数量必须逐步减少,此时,由于 p_c 已基本不起作用,p_m 必须随演化代数的增加而减少。

综上所述,在动态自适应遗传算法中,交叉和变异概率分别定义为式(2-46)和(2-47)。

$$p_c = k_1 \frac{t(f_{\max} - f_c)}{(f_{\max} - f_{\min})} \tag{2-46}$$

$$p_m = k_2 \frac{(f_{\max} - f_m)}{(f_{\max} - f_{\min})t} \tag{2-47}$$

式中:k_1、k_2 为常数;t 为演化代数;f_c、f_m 分别为交叉和变异个体的适应值。

2) 杂和遗传算法的计算步骤

杂和遗传算法的基本计算步骤如下:

第一步,产生初始化种群 $p(0)$,$t = 0$。

第二步,计算种群中每个个体的适应度函数值,检查是否满足收敛条件,如果满足,输出结果,不满足则进行第三步。

第三步,执行选择操作。

第四步,判断是否模糊大变异操作条件,如果满足则进行模糊大变异操作,否则进行动态自适应变异操作。

第五步,进行动态自适应交叉操作。

第六步,生成新的种群 $p(t+1)$,$t = t+1$,转第二步。

3) 杂和遗传算法在暴雨强度公式参数辨识中的应用

暴雨强度公式是确定雨水设计流量的基本依据之一。暴雨强度公式[34] 形式为

$$i = \frac{A_1(1 + C\lg P)}{(t+b)^n} \tag{2-48}$$

式中:i 为降雨强度;P 为重现期;t 为降雨历时。

用遗传算法来求解暴雨强度公式参数 A_1、C、b、n,适应度函数为

$$F = \frac{1}{Q} \tag{2-49}$$

$$Q = \sum_{k=1}^{m} \sum_{j=1}^{n} \left[\left(\frac{A_1(1 + C\lg P_j)}{(t_k + b)^n} \right) - i_{kj} \right]^2 \tag{2-50}$$

根据文献[34]中表1.69提供的数据，将参数A_1、C、b、n表示成一维数组，每个参数都用十进制数表示，该一维数组就是一条染色体，采用简单遗传算法和杂和遗传算法，进行暴雨强度公式参数的辨识，所得结果如表2.6所示。

表2.6 暴雨强度参数计算结果

计算方法	A_1	b	C	n	绝对均方差和
简单遗传算法	11.700	17.840	0.751 0	0.700	0.521 0
杂和遗传算法	10.820	17.620	0.752 8	0.684	0.476 1

由此可以看出，杂和遗传算法比简单遗传算法效果更好。

2.6 粒子群优化算法[35]

2.6.1 概述

当前，通过模拟生物群体的行为来解决计算问题已经成为新的研究热点，形成了以群体智能(Swarm Intelligence)为核心的理论体系，并已在一些实际应用领域取得突破性进展。通过对生物群体的观察和研究发现，生物群体内个体间的合作与竞争等复杂性行为产生的群体智能往往能对某些特定的问题提供高效的解决方法。例如，动物行为学家曾仔细观察过蚂蚁的觅食行为，发现不管初始时同一蚁巢的蚂蚁从蚁巢到食物的觅食路径是如何的随机，随着觅食的蚂蚁往返次数的增加，蚁群总能找到最短的觅食路径。著名的蚁群算法正是受蚁群觅食行为的启发而产生的。实践证明，蚁群算法在组合优化、车间作业调度、网络路由选择等领域已经取得成功的应用，对此，将在下一节介绍。美国的Kennedy和Eberhar受鸟群觅食行为的启发，于1995年提出了粒子群优化算法(Particle Swarm Optimization，简记为PSO)。最初的设想是仿真简单的社会系统，研究并解释复杂的社会行为，后来发现粒子群优化算法可以用于复杂优化问题的求解。

2.6.2 粒子群优化算法

如上所述，粒子群优化算法的提出，是受鸟群觅食行为的启发，并用于解决复杂优化问题的。算法采用速度-位置搜索模型。每个粒子代表解空间的一个候选解，解的优劣程度由适应函数决定。速度$\boldsymbol{v}_i=[v_{i1},v_{i2},\cdots,v_{id}]$决定粒子在搜索空间单位迭代次数的位移。其中，适应函数根据优化目标定义。PSO随机初始化为一群粒子，其中第i个粒子在d维解空间的位置表示为$\boldsymbol{x}_i=[x_{i1},x_{i2},\cdots,x_{id}]$。每一次迭代，粒子通过动态跟踪两个极值来更新其速度和位置。第一个是粒子从初始到当前迭代次数搜索产生的最优解：个体极值$\boldsymbol{p}_i=[p_{i1},p_{i2},\cdots,p_{id}]$。第二个是粒子种群目前的最优解：全局极值$\boldsymbol{g}=[g_1,g_2,\cdots,g_d]$。粒子根据以下公式来更新其速度和位置：

$$\boldsymbol{v}_i=\boldsymbol{v}_i+c_1\cdot\text{rand}()\cdot(\boldsymbol{p}_i-\boldsymbol{x}_i)+c_2\cdot\text{rand}()\cdot(\boldsymbol{g}-\boldsymbol{x}_i) \tag{2-51}$$

$$\boldsymbol{x}_i=\boldsymbol{x}_i+\boldsymbol{v}_i \tag{2-52}$$

式中：rand()是均匀分布在(0,1)区间的随机数。一般取学习因子$c_1=c_2=2$。粒子在解空间内不断跟踪个体极值与全局极值进行搜索，直到达到规定的迭代次数或满足规定的误差标准为止。粒子在每一维飞行的速度不能超过算法设定的最大速度$v_{\max}$。设置$v_{\max}$较大可以保证粒子

种群的全局搜索能力，v_{max} 较小则粒子种群的局部搜索能力加强。粒子群优化算法在解空间搜索的示意图如图 2.36 所示，粒子群优化算法的流程如图 2.37 所示。

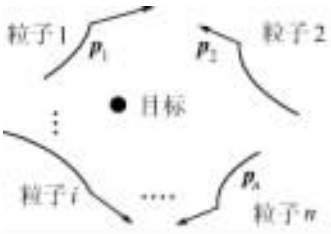


图 2.36　PSO 优化搜索示意图

粒子群优化算法是基于群体智能理论的优化算法，通过群体中粒子间的合作与竞争产生的群体智能指导优化搜索。与演化算法比较，PSO 保留了基于种群的全局搜索策略，但是其采用的速度-位移模型操作简单，避免了复杂的遗传操作。它特有的记忆使其可以动态跟踪当前的搜索情况调整其搜索策略。与演化算法比较，粒子群优化算法是一种更高效的并行搜索算法。

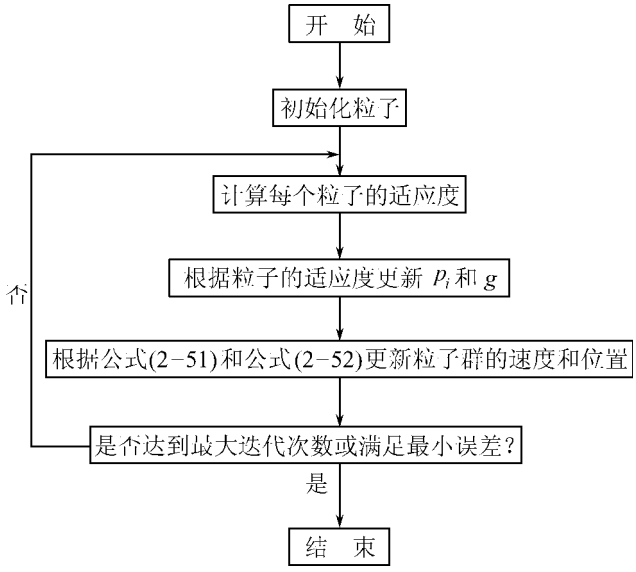


图 2.37　粒子群优化算法流程图

2.6.3　粒子群优化算法在车辆路径优化中的应用[30, 35~37]

粒子群优化算法结构相对简单，运行速度很快，但是算法运行过程中，如果某粒子发现一个当前最优位置，其他粒子将迅速向其靠拢，如果该最优位置为一局部最优点，粒子群就无法在解空间内重新搜索，因此，算法陷入局部最优，出现了所谓的早熟收敛现象。另外，基本的 PSO 算法大多用于连续优化问题中，为了使其适合于离散域的优化，需要对其作一些修改。基于以上的原因，这里对粒子群优化算法进行改进，使其便于进行配送路径优化问题的解决，并能得到较好的结果。

1）群体适应度方差与粒子收敛

试验证明，粒子群优化算法无论是早熟收敛还是全局收敛，粒子群中的粒子都会出现“聚集”现象。要么所有粒子聚集在某一特定位置，要么聚集在某几个特定位置，这主要取决于问题本身的特性以及适应度函数的选择。研究粒子群中所有粒子适应度的整体变化就可以跟踪粒子群的状态。为了定量描述粒子群的状态，下面先给出群体适应度方差的定义，同时也给出了粒子收敛的定义。

定义 2.1 设粒子群的粒子数目为 n，f_i 为第 i 个粒子的适应度，f_{avg} 为粒子群目前的平均适应度，σ^2 为粒子群的群体适应度方差，则 σ^2 可以定义为

$$\sigma^2 = \sum_{i=1}^{n} \left(\frac{f_i - f_{\text{avg}}}{f} \right)^2 \tag{2-53}$$

式中：f 是归一化定标因子，其作用是限制 σ^2 的大小。f 可以取任意值，但需注意以下两个条件：

① 归一化后，整个粒子群 $| f_i - f_{\text{avg}} |$ 的最大值不大于 1。

② f 随算法的演化而变化。在本书算法中，f 的取值采用如下公式：

$$f = \begin{cases} \max\{| f_i - f_{\text{avg}} |\}, & \max\{| f_i - f_{\text{avg}} |\} > 1 \\ 1, & \text{其他} \end{cases} \tag{2-54}$$

定义 2.1 表明：群体适应度方差 σ^2 反映的是粒子群中所有粒子的“收敛”程度。σ^2 越小，则粒子群趋于收敛；反之，粒子群则处于随机搜索阶段。

定义 2.2 设粒子群中某个粒子在 t 时刻的位置为 $x(t)$，p 为搜索空间内的任意位置，则粒子收敛定义如下：

$$\lim_{t \to +\infty} x(t) = p \tag{2-55}$$

该定义表明，粒子的收敛是指粒子最终停留在搜索空间内某一固定位置 p。

如果粒子群优化算法陷入早熟收敛或者达到全局收敛，粒子群中的粒子将聚集在搜索空间的一个或几个特定位置，群体适应度方差 σ^2 等于零[36]。所以仅凭群体适应度方差等于零不能区别早熟收敛与全局收敛，还需进一步判断算法此时得到的最优解是否为理论全局最优或者期望最优解。如果此时已经得到全局最优，则可认为算法达到全局收敛；反之，则表明算法陷入局部最优。

2）交换子和交换序

定义 2.3 设路径优化问题的解序列为 $X_i = (x_{id})$，$d = 1,2,\cdots,D$。定义交换子 $SO(d_1, d_2)$ 为交换解 X_i 中的点 x_{id_1} 和 x_{id_2}，则 $X_i' = X_i + SO(d_1, d_2)$ 为解 X_i 经算子 $SO(d_1, d_2)$ 操作后的新解，这里为符号“+”赋予了新的含义。

例如，有一个含有 5 节点（包括虚拟节点）的路径优化问题，其解为 $X_i = (1\ 3\ 5\ 2\ 4)$，交换子为 $SO\ (1,2)$，则

$$X_i' = X_i + SO\ (1,2) = (1\ 3\ 5\ 2\ 4) + SO\ (1,2) = (3\ 1\ 5\ 2\ 4)$$

定义 2.4 一个或多个交换子的有序队列就是交换序，记作 SS，即

$$SS = (SO_1, SO_2, \cdots, SO_n) \tag{2-56}$$

式中：$SO_1, SO_2, \cdots, SO_n$ 是交换子，它们之间的顺序是有意义的。

交换序作用于一个解上，意味着这个交换序中的所有交换子依次作用于该解上，即

$$X'_i = X_i + SS = X_i + (SO_1, SO_2, \cdots, SO_n) = [(X_i + SO_1) + SO_2] + \cdots + SO_n \tag{2-57}$$

定义 2.5 不同的交换序作用于同一解上可能产生相同的新解，所有具有相同效果的交换序的集合称为交换序的等价集。

定义 2.6 若干个交换序可以合并成一个新的交换序，定义 $\oplus$ 为两个交换序的合并算子。

例如，设两个交换序 SS_1 和 SS_2，按先后顺序作用于解 S 上，得到新解 S'。假设另外有一个交换序 SS' 作用于同一解 S 上，能够得到相同的解 S'，可定义

$$SS' = SS_1 \oplus SS_2 \tag{2-58}$$

SS' 和 $SS_1 \oplus SS_2$ 属于同一等价集。一般来说，SS' 不唯一。

定义 2.7 在交换序等价集中，拥有最少交换子的交换序称为该等价集的基本交换序。

可按如下的方法构造一个基本交换序。设给定两个解路径 A 和 B，需要构造一个基本交换序 SS，使得 $B+SS=A$。A:(1 2 3 4 5)，B:(2 3 1 5 4)。

可以看出，$A(1)=B(3)=1$，所以第一个交换子是 $SO(1,3)$，$B_1=B+SO(1,3)$，得到 B_1:(1 3 2 5 4)。

$A(2)=B_1(3)=2$，所以第二个交换子是 $SO(2,3)$，$B_2=B_1+SO(2,3)$，得到 B_2:(1 2 3 5 4)。

同理，第三个交换子是 $SO(4,5)$，$B_3=B_2+SO(4,5)=A$。这样，就得到一个基本交换序：

$$SS = A - B = (SO(1,3), SO(2,3), SO(4,5))$$

3）改进的粒子群优化算法

由于车辆路径问题是离散域上的优化问题，基本 PSO 算法中的速度算式已不适合此类优化问题，所以需要重新构造速度算式。新的速度迭代公式如下：

$$v_{id} = v_{id} \oplus \alpha(p_{id} - x_{id}) \oplus \beta(p_{gd} - x_{id}) \tag{2-59}$$

式中：α、β 为随机数，α、$\beta \in [0,1]$；$\alpha(p_{id}-x_{id})$ 表示基本交换序 $(p_{id}-x_{id})$ 中的所有交换子以概率 α 保留；同理，$\beta(p_{gd}-x_{id})$ 表示基本交换序 $(p_{gd}-x_{id})$ 中的所有交换子以概率 β 保留。由此可以看出，α 的值越大，$(p_{id}-x_{id})$ 保留的交换子越多，p_{id} 的影响越大；同理，β 的值越大，$(p_{gd}-x_{id})$ 保留的交换子就越多，p_{gd} 的影响越大。

另外，从前面的分析知，在粒子群优化算法运行过程中，如果群体适应度方差等于零，且此时得到的最优解不是理论最优解或者期望最优解，则粒子群陷入局部最优，算法将出现早熟收敛。因此，如果要克服早熟收敛问题，就必须提供一种机制，让算法在发生早熟收敛时，能够跳出局部最优，进入解空间的其他区域继续进行搜索，直到最后找到全局最优解。

根据式(2-59)和式(2-52)，粒子下一时刻的位置由当前位置与当前速度共同决定，速度大小决定移动距离，速度方向决定粒子前进方向。根据式(2-59)，粒子当前速度由 3 个因素决定：原来的速度、个体极值 p_{id} 与全局极值 p_{gd}。全局极值 p_{gd} 是算法目前找到的最优解。如果算法出现早熟收敛，全局极值 p_{gd} 一定是局部最优解。结合式(2-59)，如果此时改变全局极值 p_{gd}(变异操作)，就可以改变粒子的前进方向，从而让粒子进入其他区域进行搜索，在其后的搜索过程中，算法就可能发现新的个体极值 p_{id} 以及全局极值 p_{gd}。如此循环，算法就可以找到全局最优解。

考虑到粒子在当前 p_{gd} 的作用下可能发现更好的位置，因此新算法将变异操作设计成一个随机算子，即对满足变异条件的 p_{gd} 按一定的概率 p_m 变异

$$p_m = \begin{cases} k, & \sigma^2 < \sigma_d^2 \\ 0, & \text{其他} \end{cases} \tag{2-60}$$

式中：k 可以取 $[0.1,0.3]$ 之间的任意数值；σ_d^2 的取值与实际问题有关，一般远小于 σ^2 的最

大值。

综上所述，对粒子优化算法进行了改进，改进后的粒子优化算法具体步骤如下：

(1) 随机初始化粒子群众粒子的位置与速度，即给每个粒子赋一个随机的初始解和一个随机的交换序。

(2) 如果满足结束条件，转步骤(10)，否则执行(3)。

(3) 根据粒子当前位置计算其下一个位置，即新解：

① 计算 p_{id} 和 x_{id} 之间的差 A，$A = p_{id} - x_{id}$，其中 A 是一个基本交换序，表示 A 作用于 x_{id} 得到 p_{id}。

② 计算 $B = p_{gd} - x_{id}$，其中 B 也是一个基本交换序。

③ 根据式(2-59) 更新速度 v_{id}。

(4) 计算搜索到的新解 $x_{id} = x_{id} + v_{id}$。

(5) 如果粒子适应度优于 p_{id} 的适应度，p_{id} 设置为新位置。

(6) 如果粒子适应度优于 p_{gd} 的适应度，p_{gd} 设置为新位置。

(7) 根据式(2-53)、(2-54) 计算群体适应度方差 σ^2。

(8) 根据式(2-60) 计算变异概率 p_m。

(9) 产生随机数 $r \in [0,1]$，如果 $r < p_m$，执行变异操作；否则，转步骤(2)。

(10) 输出结果，算法运行结束。

从上述流程可以看出，改进的粒子群优化算法实际上是在粒子群优化算法的基本框架中引入交换序的概念，使之适应离散域的优化。同时增加了随机变异算子，通过对 p_{gd} 的随机变异来提高粒子群优化算法跳出局部最优解的能力。

4) 改进粒子群优化算法在车辆配送路径优化中的应用

(1) 粒子群优化算法在车辆配送路径问题中应用的具体步骤

将改进的粒子群优化算法稍加调整，可得到如下步骤：

① 编码[38]

用简单直观的自然数编码，用 0 表示配送中心，用 1，2，…，L 表示各需求点。因为共有 C 辆汽车，最多存在 C 条配送路径，每条配送路径都始于配送中心，也终于配送中心。为了在编码中反映车辆配送的路径，采用了增加 $C-1$ 个虚拟配送中心的方法，分别用 $L+1$，$L+2$，…，$L+C-1$ 表示。这样 1，2，…，$L+C-1$ 中 $L+C-1$ 个互不重复的自然数的随机排列就构成了一个个体，并对应一种配送路径方案。

② 产生初始群体

随机产生 M 个粒子的初始位置和速度，构成初始群体。

③ 粒子的适应度计算

对于某个粒子所对应的配送路径方案，要判定其优劣，一是要看其是否满足配送的约束条件；二是要计算其目标函数值(即各条配送路径的长度之和)。根据配送路径优化问题的特点所确定的编码方法，隐含能够满足每个需求点都得到配送服务及每个需求点仅由一辆汽车配送的约束条件，但不能保证满足每条路径上各需求点需求量之和不超过汽车载重量及每条配送路线的长度不超过汽车一次配送的最大行驶距离的约束条件。为此，对每个粒子所对应的配送路径方案，要对各条路径逐一进行判断，看其是否满足上述两个约束条件，若不满足，则将该条路径定为

不可行路径。对于某个粒子 j，若其对应的配送路径为可行路径，则个体的适应度值为

$$F_j = \frac{1}{Z_j} \tag{2-61}$$

Z_j 为其目标函数值(即各条配送路径的长度之和)，若对应的配送路径为不可行路径，则令 $Z_j = G$，G 为一个很大的整数。

④ 求新解

a. 计算 p_{id} 和 x_{id} 之间的差 A，$A = p_{id} - x_{id}$，其中 A 是一个基本交换序，表示 A 作用于 x_{id} 得到 p_{id}。

b. 计算 $B = p_{gd} - x_{id}$，其中 B 也是一个基本交换序。

c. 根据式(2-59)更新速度 v_{id}。

d. 计算搜索到的新解 $x_{id} = x_{id} + v_{id}$。

e. 如果粒子适应度优于 p_{id} 的适应度，p_{id} 设置为新位置。

f. 如果粒子适应度优于 p_{gd} 的适应度，p_{gd} 设置为新位置。

g. 根据式(2-53)、(2-54)计算群体适应度方差 σ^2。

h. 根据式(2-60)计算变异概率 p_m。

i. 产生随机数 $r \in [0,1]$，如果 $r < p_m$，执行变异操作；否则，转步骤 b。

j. 如果满足结束条件，转步骤 k；否则，执行 c。

k. 输出结果，算法运行结束。

(2) 应用实例

由一个配送中心向 8 个需求点送货，各需求点对配送中心的需求为 $q_i(i = 1,2,\cdots,8)$(单位：t)，配送中心有 2 辆车用于配送，每辆车的容量均为 8 t，配送中心与各需求点的距离如表 2.7 所示(其中 0 表示配送中心[39])。

表 2.7　需求点间距离及各需求点的需求量表

d_{ij}	0	1	2	3	4	5	6	7	8
0	0.0	4.0	6.0	7.5	9.0	20.0	10.0	16.0	8.0
1	4.0	0.0	6.5	4.0	10.0	5.0	7.5	11.0	10.0
2	6.0	6.5	0.0	7.5	10.0	10.0	7.5	7.5	7.5
3	7.5	4.0	7.5	0.0	10.0	5.0	9.0	9.0	15.0
4	9.0	10.0	10.0	10.0	0.0	10.0	7.5	7.5	10.0
5	20.0	5.0	10.0	5.0	10.0	0.0	7.0	9.0	7.5
6	10.0	7.5	7.5	9.0	7.5	7.0	0.0	7.0	10.0
7	16.0	11.0	7.5	9.0	7.5	9.0	7.0	0.0	10.0
8	8.0	10.0	7.5	15.0	10.0	7.5	10.0	10.0	0.0
需求量		1	2	1	2	1	4	2	2

采用 C 语言编程，在 PⅢ850 的计算机上对改进粒子群优化算法的性能进行了验证：选取群体规模为 80，$k = 0.3$，$\sigma_d^2 = 0.01$，运行 10 次，每次运行 50 代，得到配送路径为：0－1－3－5－8－2－0、0－6－7－4－0，收敛到满意解的概率为 90％。

为作比较，该例也用隔离小生境遗传算法进行计算：选取群体规模为 100，分为 5 个子群

体，最大允许规模为 40，最小保护规模为 10，取各子群体的交叉概率 $p_{ic}=0.75$，变异概率 $p_{im}=0.02$，$T_0=100$，温度冷却系数 $C_i=0.9$。计算最优结果为 67.5，配送路径为：0－1－3－5－8－2－0、0－6－7－4－0，收敛到满意解的概率为 90％。

可见，粒子群优化算法具有较好的收敛性和稳定性的，对于解决物流配送中的车辆路径问题是可行和有效的。

2.6.4 粒子群优化算法和遗传算法的对比[40]

自然界经过数万年的演化，存在的方式、规律都是最优的。所以，人们总是在找寻自然界的优化规律，发展形成所谓的演化算法，并应用到各类优化问题，这里介绍的遗传算法以及近年兴起的粒子群优化算法就是借鉴自然界优化规律而产生的演化算法。

遗传算法是 20 世纪 70 年代由美国密歇根大学 J. H. Holland 教授提出的，它是一类基于自然选择和遗传学原理的有效寻优方法，目前在很多方面都有成功应用。粒子群优化算法最初由 Jim Kennedy 于 1995 年提出并成功地用于连续非线性函数的优化，它是对鸟群觅食过程中的迁徙和聚集的模拟，也可以说是对社会心理学的一种模拟。粒子群优化算法对优化问题求解来说，最大特点就是收敛速度较快。

工程优化应用的大多数问题，都需要先对其进行数学建模，即向数学问题转化，然后对函数进行优化。工程优化所建立的数学函数，往往是带有多种约束条件的复杂函数，而且大多是既不连续、也不可微的隐函数。对于这些复杂函数来说，用常规的数学方法很难或根本无法处理。这里讨论的这两种演化算法都能处理这类复杂函数的优化问题，但由于算法的原理不同，在优化过程中所表现出来的特性和优点也会不同。两种算法原理的基本特点，前面已作了介绍，下面通过原理简单对比和一个较为典型的数学问题的优化，对这两种演化算法的性态进行了分析，以此来说明它们在工程优化中的有效性及其差异。

1）遗传算法基本思想

遗传算法通过对参数的编码进行操作，而不是对参数本身进行操作，这使得它处理优化问题时，既不要求函数连续，更不要求可微。因此，要优化的函数既可以是数学解析式所表达的显函数，也可以是映射矩阵甚至是神经网络等隐函数，因而其应用范围较广。进一步，该方法通过目标函数来计算适应度函数值，不需要其他的推导和辅助信息，从而对问题的依赖较小。遗传算法是从许多初始点开始并行操作，通过交叉、变异、选择，以选取最优解，而不是从一个点开始，因而可以有效的防止搜索过程收敛于局部最优，而且有较大的可能求得全部的最优解。遗传算法具有并行计算的特点，因而可通过大规模并行计算来提高计算速度。上述这些特点使得遗传算法较适合复杂问题的优化。

2）粒子群优化算法基本思想

粒子群优化算法是对鸟群觅食过程中的迁徙和群集的模拟。鸟群在觅食时，在它们找到有食物的地方之前的迁徙过程中，既有分散又有群集的特点。所谓鸟语花香，鸟也有鸟的语言，它们总是通过自己一套独有的方式相互传递信息，其实，这种个体之间信息的交换在群居的生物群体中都存在，如蜜蜂、蚂蚁，也包括人类。对于鸟群来说，在它们找到食物源之前的从一地到另一地的迁徙过程中，总是有那么一只鸟对食物的嗅觉较好，即对食物源的大致方向具有较好的洞察力，因而这只鸟就拥有食物源的较好信息。由于在找寻食物源的途中，它们又在

相互传递信息，特别是这种较好的信息。所以，在这种“好消息”的指引下，最终导致了鸟群“一窝蜂”的飞向食物源，达到了在食物源的群集。粒子群优化算法中，解群相当于一个鸟群，一地到另一地的迁徙相当于解群的演化，“好消息”相当于解群每代演化中的最优解，食物源相当于全局最优解。Jim Kennedy 正是从鸟群觅食的自然现象中得到启发，提出了粒子群优化算法。

3）两种算法对比的案例

改进遗传算法和粒子群优化算法的对比分析，用下面数学问题来说明。

$$\min f(x,y) = (x-10)^3 + (y-20)^3$$

$$\text{s.t. } (x-5)^2 + (y-5)^2 - 100 \geqslant 0$$

$$(x-6)^2 + (y-5)^2 - 82.81 \geqslant 0$$

$$13 \leqslant x \leqslant 100, 0 \leqslant y \leqslant 100$$

该问题为一带边界约束和非线性约束的二元函数的优化问题。由此可知，其全局最优解为 $(x,y) = (14.095, 0.842\,96)$，$f(x,y) = -6\,961.813\,381$。

对于改进遗传算法而言，控制参数设定：群体规模大小为 20；个体串长度为 2；交叉概率为 0.98，变异概率为 0.01；最大遗传代数为 500；初始温度（T_0）为 100[40]（这里遗传算法是改进的，引入了模拟退火思想，T_0 为模拟退火的初始温度）。

对于粒子群优化算法而言，参数设定：粒子群的最大演化代数为 20；粒子群的种群大小为 20，维数为 2；粒子群算法速度矢量中的参数为 C_1：3.0，C_2：3.0，ω：3.0[40]（这里速度公式是改进的，和式(2-51) 比较，等号右边 v_i 前面有常数 ω）。

对上述数学问题的优化，采用的计算机配置为 CPU：赛扬 1 G，内存：128 MB，在这种配置下，两种算法的对比如表 2.8 所示。

表 2.8　两种演化算法的性能对比

项目	演化代数	耗费 CPU 时间	最优解	最优解出现代数
改进遗传算法	500	48 h	$(x,y)=(14.095\,010, 0.842\,980\,2)$ $f(x,y)=-6\,961.792\,000$	488
粒子群优化算法	20	10 s	$(x,y)=(14.095\,000, 0.842\,965\,4)$ $f(x,y)=-6\,961.809\,000$	17

由算例可以看出，改进遗传算法和粒子群优化算法都能在有限的演化代数内找到其最优解。从这两种演化算法的解群随演化代数的变化情况可以看出，粒子群优化算法在较少的演化代数内，其解群就向最优解的方向收敛，可见粒子群优化算法的优化效率较高。改进遗传算法花费了较长的计算时间，进行了较多的演化迭代，最终找到了最优解，其解群是逐渐收敛到最优解的，收敛速度比粒子群优化算法慢一些。从算例看出，粒子群优化算法和改进遗传算法都表现出较好的健壮性。

2.7　蚁群优化算法[43～47]

组合优化问题很早就引起了人们的兴趣，但是由于许多组合优化问题都是 NP 难题，对于这类问题，至今尚无很好的解决办法，而一般采用启发式算法来解决。近年来，随着计算技术

的发展，一些新的智能算法如遗传算法[GA]、模拟退火算法[SA]等得到了迅速发展和广泛应用。一些新的模拟演化算法，如粒子群优化算法、蚁群算法也逐渐发展起来。本节介绍的人工蚁群算法就是一种新型的模拟演化算法。该算法是由意大利学者 M. Dorigo、V. Maniez-Zo、A. Colorini 等人首先提出的，称之为蚁群系统（Ant Colony System），并应用该算法求解 TSP 问题、分配问题、Job-Shop 调度问题，取得了较好的结果。受其影响，蚁群系统模型逐渐引起了其他研究者的注意，D. Costa 和 A. Hertz 在 M. Dorigo 等人研究成果的基础上，提出了一种求解分配类型问题（Assignment Type Problem）的一般模型，并用来研究图着色的问题。G. Bilchev、I. C. Parmee 研究了求解连续空间优化问题的蚁群系统模型。虽然这些成果仅是初步的，但是这些研究已显示出蚁群算法的一些优点。

2.7.1　基本原理

人工蚁群算法是受到对真实的蚁群行为的研究的启发而提出的。为了说明人工蚁群系统的原理，先从蚁群搜索食物的过程谈起。例如，蚂蚁、蜜蜂、飞蛾等群居昆虫，虽然单个昆虫的行为极其简单，但由简单的单个个体所组成的群体却表现出极其复杂的行为，原因是什么呢？仿生学家经过大量细致观察研究发现，蚂蚁个体之间是通过一种称之为外激素（pheromone）的物质进行信息传递的。蚂蚁在运动过程中，能够在它所经过的路径上留下该种物质，而且蚂蚁在运动过程中能够感知这种物质，并以此指导自己的运动方向，因此，由大量蚂蚁组成的蚁群的集体行为便表现出一种信息正反馈现象：某一路径上走过的蚂蚁越多，则后来者选择该路径的概率就越大。蚂蚁个体之间就是通过这种信息的交流达到搜索食物的目的。下面引用 M. Dorigo 所举的例子来具体说明蚁群系统的原理。如图 2.38 所示，设 A 是巢穴，E 是食物源，H、C 为一障碍物。由于障碍物的存在，蚂蚁只能经 H 或 C 由 A 到达 E，或由 E 到达 A，各点之间的距离如图 2.38 (a) 所示。设每个单位时间内有 30 只蚂蚁由 A 到达 B，有 30 只蚂蚁由 E 到达 D 点，蚂蚁经过后留下的激素物质量（以下称之为信息）为 1。为方便，设该物质停留时间为 1。在初始时刻（$t=0$，如图 2.38 (b)），由于路径 BH、BC、DH、DC 上均无信息存在，位于 B 和 D 的蚂蚁可以随机选择路径，从统计的角度可以认为它们以相同的概率选择 BH、BC、DH、DC。经过一个单位时间后，在路径 BCD 上的信息量是路径 BHD 上信息量的 2 倍。$t=1$ 时刻（如图 2.38 (c)），将分别有 20 只蚂蚁由 B 和 D 到达 C，分别有 10 只蚂蚁由 B 和 D 到达 H。随着时间的推移，蚂蚁将会以越来越大的概率选择路径 BCD，最终完全选择路径 BCD。从而找到由蚁巢到食物源的最短路径。由此可见，蚂蚁个体之间的信息交换是一个正反馈过程。

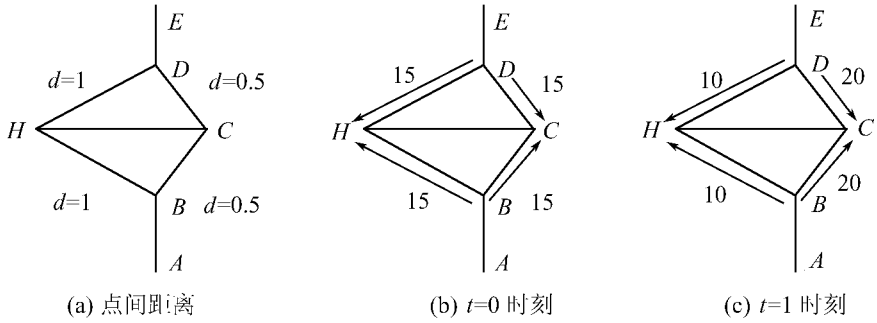


图 2.38　蚁群系统示意图

2.7.2 系统模型及其实现

以求解 n 个城市的 TSP 问题为例说明蚁群系统模型。对于其他问题，对此模型稍作修改便可应用。为模拟实际蚂蚁的行为，首先引进如下记号：设 m 是蚁群中蚂蚁的数量，$d_{ij}(i,j=1,2,\cdots,n)$ 表示城市 i 和城市 j 之间的距离，$b_i(t)$ 表示 t 时刻位于城市 i 的蚂蚁的个数，$m=\sum_{i=1}^{n} b_i(t)$，$\tau_{ij}(t)$ 表示 t 时刻在 ij 连线上残留的信息量。初始时刻，各条路径上信息量相等，设 $\tau_{ij}(0)=c(c$ 为常数)。蚂蚁 $k(k=1,2,\cdots,m)$ 在运动过程中，根据各条路径上的信息量决定转移方向，$p_{ij}^{k}(t)$ 表示在 t 时刻蚂蚁 k 由位置 i 转移到位置 j 的概率

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\,\eta_{ij}^{\beta}(t)}{\sum\limits_{s\in \text{allowed}_k}\tau_{is}^{\alpha}\,\eta_{is}^{\beta}(t)}, & j\in \text{allowed}_k\\ 0, & \text{otherwise}\end{cases} \tag{2-62}$$

式中：$\text{allowed}_k=\{0,1,\cdots,n-1\}-\text{tabu}_k$，表示蚂蚁 k 下一步允许选择的城市。与真实蚁群系统不同，人工蚁群系统具有一定的记忆功能，这里用 $\text{tabu}_k(k=1,2,\cdots,m)$ 记录蚂蚁 k 目前已经走过的城市。随着时间的推移，以前留下的信息逐渐消逝，用参数 $1-\rho$ 表示信息消逝程度，经过 n 个时刻，蚂蚁完成一次循环，各路径上信息量要根据下式作调整

$$\tau_{ij}(t+n)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij} \tag{2-63}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{2-64}$$

式中：$\Delta\tau_{ij}^{k}$ 表示第 k 只蚂蚁在本次循环中留在路径 ij 上的信息量；$\Delta\tau_{ij}$ 表示本次循环中留在路径 ij 上的信息量。

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过 } ij\\ 0, & \text{否则}\end{cases} \tag{2-65}$$

式中：Q 是常数；L_k 表示第 k 只蚂蚁在本次循环中所走路径的长度。在初始时刻，$\tau_{ij}(0)=C$，C 为常数，$\Delta\tau_{ij}=0$，其中，$i,j=0,1,\cdots,n-1$。α、β 分别表示蚂蚁在运动过程中所积累的信息及启发式因子在蚂蚁选择路径中所起的不同作用。η_{ij} 表示由城市 i 转移到城市 j 的期望程度，可根据某种启发式算法具体确定。根据具体算法的不同，$\tau_{ij}(t)$、$\Delta\tau_{ij}(t)$ 及 $p_{ij}^{k}(t)$ 的表达形式可以不同，要根据具体问题而定。M. Dorigo 曾给出 3 种不同模型，分别称之为 Ant-Cycle System，Ant-Quantity System，Ant-Density System。停止条件可以用固定循环次数或者当演化趋势不明显时便停止计算。以上是蚁群系统模型，是一个递推过程，很容易在计算机上实现。

上述 3 个模型的差别在于式(2-65)表述不同，式(2-65)为 Ant-Cycle System 模型。在 Ant-Quantity System 模型中，有

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{d_{ij}}, & \text{若第 } k \text{ 只蚂蚁在时刻 } t \text{ 和 } t+1 \text{ 之间经过 } ij\\ 0, & \text{否则}\end{cases} \tag{2-66}$$

在 Anti-Density System 模型中，有

$$\Delta\tau_{ij}^{k}=\begin{cases}Q, & \text{若第 } k \text{ 只蚂蚁在时刻 } t \text{ 和 } t+1 \text{ 之间经过 } ij\\ 0, & \text{否则}\end{cases} \tag{2-67}$$

这 3 个模型的区别在于，在后两种模型中，利用的是局部信息，而前者利用的是整体信息，因此性能较好，通常采用它作为基本模型。

由于蚂蚁在每个阶段所能选择的候选路径是有限、固定的，传统蚁群算法要求离散的解空间，因而不能直接用于求解线性和非线性规划等连续空间的优化问题。近年来，很多学者进行了将蚁群算法用于连续空间求解的研究，取得了大量的成果。下面介绍其中一些研究成果的应用。

2.7.3 蚁群算法的实际应用

1）蚁群算法在发电企业报价策略中的应用

（1）发电企业的报价模型

发电企业的竞价决策模型[48,49]，目标函数是单个时段单个机组利润最大化，即

$$\max J = \max \sum_{i=1}^{n} (\lambda_{ki} q_{ki} - C_{ki}(q_{ki})), n \in \{\text{报价点数}\}, k \in \{1,2,\cdots,\text{种群数}\} \tag{2-68}$$

机组最大最小出力约束为

$$q_{ki\min} \leqslant q_{ki} \leqslant q_{ki\max} \tag{2-69}$$

机组单时段报价限制为

$$\lambda_{ki} < \lambda_{kj}, i < j \tag{2-70}$$

式中：J 为利润函数；λ 为预测边际电价或接受市场的发布信息，单位为元 /MW·h^{-1}；q 为机组出力，单位为 MW；k 为报价方案编号；i 为报家点数编号。$C_{ki}(q_{ki})$ 是机组的生产成本，包括固定成本和运行成本。

（2）发电企业报价的蚁群算法[47]

利用蚁群算法求解连续函数的成果，解决发电企业的报价问题[50]。设系统中的蚂蚁数为 m，在选取 m 个初始解以后，它们的第 i 个分量的 m 个取值，构成该分量的候选组。将 n 个分量看成是 n 个顶点，在第 i 个顶点到第 $i+1$ 个顶点之间有 m 个连线，代表第 i 个分量候选组中的 m 个不同的候选值。记其中第 j 条连线为(i,j)，在 t 时刻它上面的信息量记为 $\tau_{ij}(t)$。每只蚂蚁都从第一个顶点出发，按照一定的策略依次选择 $n-1$ 条连线。到达第 n 个顶点后，再在 m 条连线中选择某一条到达终点。每个蚂蚁所走过的路径代表一个解，其 n 条连线表示它的 n 个分量。为了使解的分布具有多样性，在各个分量选取 m 个值后对其实行分叉、变异操作，所得到的值作为新一代解的相应分量。在得到 m 个解后，要根据它们的适应度值，更新各条边上的信息量。在每一次迭代后，对各个分量的候选组中的值，要动态的更新，要在新产生的值和候选组的值中选取信息量较高的 m 个值作为新的候选组。重复迭代，直至满足停止条件。具体做法如下：

① 定义解的适应度函数

定义解的适应度函数如下：

$$F = A + J - \Phi_k \sum_{k=1}^{3} V_k \tag{2-71}$$

$$V_1 = \max\{0, R - J(\lambda)\} \tag{2-72}$$

$$V_2 = \max\{(\underline{q}_t - q_t), \max(0, q_t - \bar{q}_t)\} \tag{2-73}$$

$$V_3 = \max\{(\underline{\lambda}_t - \lambda_t), \max(0, \lambda_t - \bar{\lambda}_t)\} \tag{2-74}$$

式中：A 是一个正数，可保证个体适应值为正；J 是目标函数；R 是利润的下限值；Φ_k 是约束条件 k

的惩罚系数，初值为1，在算法演化中，若约束条件被违反，则逐步增大惩罚系数，以保证约束条件得到满足；q 是机组出力，λ 是电价，$\overline{\lambda_t}$ 和 $\underline{\lambda_t}$ 分别表示 λ 的上下限值；$\overline{q_t}$ 和 $\underline{q_t}$ 分别表示 q 的上下限值。

② 编码及初始化

本书采用十进制编码，每个报价时段对应一个解，每个报价点的报价为解的分量。随机产生 m 个初始解，计算这 m 个初始解的适应度，由这 m 个初始解得到各个分量值的候选组，并根据候选组中的值按它们所在解的适应度计算它们的信息量。

③ 迭代过程

while not 结束条件 do

a. for $i = 1$ to n do(对 n 个分量循环)

for $k = 1$ to m do(对 m 个蚂蚁循环)

根据概率 $p_{ij}^k(t)$，在第 i 个分量候选组里选择第 k 个蚂蚁在该分量的初值 end for k

对所选择的第 i 个分量的 m 个初值，实行交叉、变异操作，生成第 i 个分量的新一代的 m 个值并加入候选组 end for i

根据所选择的分量构成 m 个新一代解，并计算新一代解的适应度值。

b. 修改各分量的候选组中各候选值的信息量。

c. 取各分量的候选组中信息量较高的 m 个值作为新的候选组 end while

上述算法的 b 中，选取第 i 个分量的候选组中第 j 个值的概率为

$$p_{ij}^k(t) = \frac{\tau_{ij}(t)}{\sum_{r=1}^{m}\tau_{ir}(t)} \tag{2-75}$$

式中：$\tau_{ij}(t)$ 表示此时第 i 个分量的候选组中在做动态变化的第 j 个值的信息量；在得到新一代的解之后，本算法的 b 按照式(2-63) 和式(2-64) 相应地更新各个分量的候选组中的信息量 $\tau_{ij}(t)$，但对 $\Delta\tau_{ij}^k(t)$ 的更新按照下式进行：

$$\Delta\tau_{ij}^k = \begin{cases} W \cdot F_k, & \text{若蚂蚁 } k \text{ 的解的第 } i \text{ 个分量选中第 } j \text{ 个候选值} \\ 0, & \text{否则} \end{cases} \tag{2-76}$$

式中：W 为一常数，F_k 为第 k 只蚂蚁所对应的解的适应度。

(3) 计算结果

结合江苏某电厂2002年3月某日运行的实际数据，采用蚁群算法进行仿真。5个时段对应4条报价曲线，出力组合有135 MW，200 MW，300 MW，600 MW共4个，结果如图2.39所示。它对电厂的报价和运行有较好的参考价值。

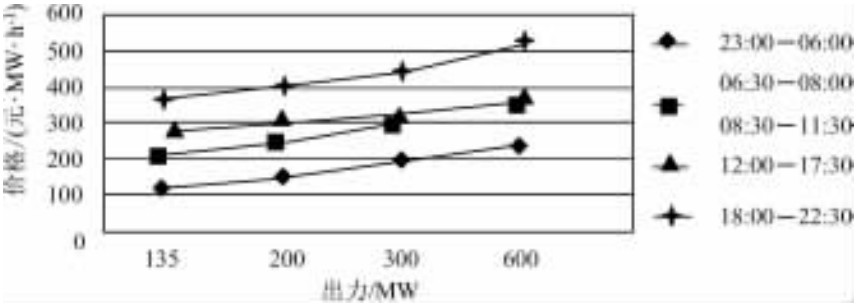


图 2.39　价格-出力报价曲线

2）蚁群算法在中央空调故障诊断中的应用[45]

下面介绍是根据华东理工大学杨欣斌等3位学者研究成果整理出来的。

随着我国经济的迅速发展和人民生活水平的提高，办公楼和居民别墅使用中央空调已经相当普遍，因此根据系统运行数据对中央空调系统进行诊断有重要的实际应用价值。中央空调系统的运行数据有：送风温度、送风压力、送回风流量、送风机控制偏差、回风机控制偏差、阀门控制偏差。系统故障一般有：正常状态（相当于没有故障）、回风机故障、送风机故障、阀门故障、温度传感器故障、压力传感器故障、送风风速仪故障、回风风速仪故障等。每一组测量数据可以看作为多维空间的一个点，于是故障诊断问题可归结为数学上的聚类分析问题来解决。

“物以类聚，人以群分”是自然界的普遍现象，聚类分析是根据这一现象经过抽象发展起来的一种数学分析方法。从数学角度来说，聚类分析是统计学的一个分支，也是无导师监督的机器学习方法，目前在模式识别、知识发现及数据挖掘等方面得到了广泛的应用。通过对人进行聚类判断所遵循的原则分析发现，人们在实施聚类行为时，有一种类似于“能量场”的特点，即首先人们会确定一个“核心”（即聚类中心），然后将周围的对象“吸引”（归并）到该“核心”周围，从而完成聚类过程。进一步分析比较发现，蚂蚁在寻找食物过程中也遵循相同的原则，即发现食物源（可类比为聚类中心）后，蚂蚁就会被“吸引”（类比于归并）到食物源周围，因此，下面将蚁群算法运用到聚类分析中。

（1）基于蚁群算法的聚类学习方法的基本原理

在基于蚁群算法的聚类分析中，将数据视为具有不同属性的蚂蚁，聚类中心看作是蚂蚁所要寻找的“食物源”。所以，数据聚类过程就看作是蚂蚁寻找食物源的过程。设 $\boldsymbol{X}=\{\boldsymbol{X}_i \mid \boldsymbol{X}_i=\{x_{i1},x_{i2},\cdots,x_{im}\},i=1,2,\cdots,N\}$ 是待进行聚类分析的数据集合。令 $d_{ij}=\|P(\boldsymbol{X}_i-\boldsymbol{X}_j)\|_2=\sqrt{\sum_{k=1}^{m}p_k(x_{ik}-x_{jk})^2}$，$d_{ij}$ 表示 $\boldsymbol{X}_i$ 到 $\boldsymbol{X}_j$ 之间的加权欧氏距离，其中 P 为权因子，可以根据各分量在聚类中的贡献不同而设定。设 r 为聚类半径，ε 为统计误差，$\tau_{ij}(t)$ 是 t 时刻数据 $\boldsymbol{X}_i$ 到数据 $\boldsymbol{X}_j$ 路径上残留的信息量，设 $\tau_{ij}(0)=0$，即在初始时刻各条路径上的信息量相等且为0。路径 ij 上信息量由下式给出

$$\tau_{ij}(t)=\begin{cases}1, & d_{ij}\leqslant r\\ 0, & d_{ij}>r\end{cases} \tag{2-77}$$

$\boldsymbol{X}_i$ 是否归并到 $\boldsymbol{X}_j$ 由下式给出

$$p_{ij}(t)=\frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{s\in S}\tau_{sj}^{\alpha}(t)\eta_{sj}^{\beta}(t)} \tag{2-78}$$

其中

$$\boldsymbol{S}=\{\boldsymbol{X}_s \mid d_{sj}\leqslant r,s=1,2,\cdots,j,j+1,\cdots,N\} \tag{2-79}$$

若 $p_{ij}(t)\geqslant p_0$，则 $\boldsymbol{X}_i$ 归并到 $\boldsymbol{X}_j$ 领域。令 $\boldsymbol{C}_j=\{\boldsymbol{X}_k \mid d_{kj}\leqslant r,k=1,2,\cdots,J\}$，$\boldsymbol{C}_j$ 表示所有归并到 $\boldsymbol{X}_j$ 领域的数据集合。根据下式求出理想的聚类中心：

$$\overline{\boldsymbol{C}}_j=\frac{1}{J}\sum_{k=1}^{J}\boldsymbol{X}_k \tag{2-80}$$

式中：$\boldsymbol{X}_k\in\boldsymbol{C}_j$。

（2）基于蚁群的聚类学习方法的算法步骤

① 初始化：设定 $N,m,r,\varepsilon_0,\alpha,\beta,\tau_{ij}(0)=0,p_0$。

② 计算：

$$d_{ij}=\|P(\boldsymbol{X}_i-\boldsymbol{X}_j)\|_2=\sqrt{\sum_{k=1}^{m}p_k(x_{ik}-x_{jk})^2}$$

③ 计算各路径上的信息量：

$$\tau_{ij}(t)=\begin{cases}1, & d_{ij}\leqslant r\\ 0, & d_{ij}>r\end{cases}$$

④ 计算 $\boldsymbol{X}_i$ 归并到 $\boldsymbol{X}_j$ 的概率：

$$p_{ij}(t)=\frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{s\in S}\tau_{sj}^{\alpha}(t)\eta_{sj}^{\beta}(t)}$$

⑤ 判断 $p_{ij}(t)>p_0$ 是否成立。

⑥ 按 $\bar{\boldsymbol{C}}_j=\frac{1}{J}\sum_{k=1}^{J}\boldsymbol{X}_k$ 式计算该类的聚类中心。

⑦ 按 $\boldsymbol{D}_j=\sum_{k=1}^{J}\sqrt{\sum_{i=1}^{m}(x_{ki}-\boldsymbol{C}_{ji})^2}$ 计算第 J 个聚类的偏离误差，其中 $\boldsymbol{C}_{ji}$ 表示第 J 个聚类中心的第 i 个分量。

⑧ 计算总体误差 $\varepsilon=\sum_{J=1}^{k}\boldsymbol{D}_J$。

⑨ 判断 $\varepsilon\leqslant\varepsilon_0$，若成立，则停止，并输出聚类个数 k 和聚类中心 $\bar{\boldsymbol{C}}_j$；若不成立，转②继续迭代。

(3) 计算结果

对某大楼的中央空调故障诊断的40个6维样本数据进行仿真测试，取 $\eta=1,\alpha=1,\beta=1,p_0=0.60,r=0.1,\varepsilon=0.8$。数据见表2.9。

用前面介绍的步骤计算得结果如下：

$$\boldsymbol{C}_1=\{X_1,X_9,X_{37},X_{39},X_{40}\}$$
$$\boldsymbol{C}_2=\{X_2,X_{10},X_{23},X_{29},X_{32}\}$$
$$\boldsymbol{C}_3=\{X_3,X_{11},X_{17},X_{24},X_{30}\}$$
$$\boldsymbol{C}_4=\{X_4,X_{12},X_{19},X_{25},X_{31}\}$$
$$\boldsymbol{C}_5=\{X_5,X_{13},X_{18},X_{33},X_{36}\}$$
$$\boldsymbol{C}_6=\{X_6,X_{14},X_{20},X_{26},X_{34}\}$$
$$\boldsymbol{C}_7=\{X_7,X_{15},X_{22},X_{27},X_{35}\}$$
$$\boldsymbol{C}_8=\{X_8,X_{16},X_{21},X_{28},X_{38}\}$$

符号的意义是：$\boldsymbol{C}_1$ 类是正常状态；$\boldsymbol{C}_2$ 类表示回风机故障；$\boldsymbol{C}_3$ 类表示送风机故障；$\boldsymbol{C}_4$ 类表示阀门故障；$\boldsymbol{C}_5$ 类表示温度传感器故障；$\boldsymbol{C}_6$ 类表示压力传感器故障；$\boldsymbol{C}_7$ 类表示送风风速仪故障；$\boldsymbol{C}_8$ 类表示回风风速仪故障。$X_1\sim X_{40}$ 表示相应的测量数据，即系统的状态。

表 2.9　仿真数据

序号	送风温度	送风压力	送回风流量	送风机控制偏差	回风机控制偏差	阀门控制偏差
X_1	0.003 8	0.027 7	−0.040 5	−0.010 7	0.013 8	−0.161 8
X_2	−0.030 9	0.050 6	2.160 4	−0.021 7	−0.807 3	0.320 6
X_3	0.320 1	−0.827 4	−1.553 6	−0.948 3	−0.016 6	−0.408 6
X_4	−0.062 3	0.097 9	0.085 5	−0.007 2	−0.007 1	2.179 7
X_5	−1.494 5	0.000 4	0.025 7	0.000 1	−0.011 5	3.378 1
X_6	−0.086 7	−0.951 2	−0.088 6	0.028 1	0.013 5	−0.565 9
X_7	0.121 2	−0.062 6	−1.192 4	0.034 5	−0.029 4	0.625 0
X_8	−0.182 1	−0.008 6	2.096 8	−0.019 3	−0.040 5	0.841 2
X_9	0.004 0	0.028 3	−0.041 2	−0.011 0	0.014 2	−0.170 1
X_{10}	−0.031 4	0.051 7	2.160 1	0.022 8	−0.810 1	0.319 8
X_{11}	0.319 8	−0.830 1	−1.560 1	−0.950 0	−0.017 2	−0.410 1
X_{12}	−0.061 7	0.098 1	0.086 1	−0.008 1	0.007 8	2.200 0
X_{13}	−1.520 0	0.000 5	0.030 1	0.001	−0.011 7	3.400 1
X_{14}	−0.090 1	−1.12	−0.090 1	0.029 0	0.013 8	−0.570 1
X_{15}	0.110 0	−0.063 7	−1.200 1	0.034 0	−0.030 1	−0.570 1
X_{16}	−0.080 1	−0.009 1	2.098 7	−0.020 0	−0.040 7	−0.570 1
X_{17}	0.321 0	−0.825 0	−1.554 7	−1.000 1	−0.015 9	−0.408 9
X_{18}	−1.489 7	0.000 3	0.026 9	0.000 0	−0.012 0	3.379 2
X_{19}	−0.062 9	0.096 3	0.085 9	−0.006 9	0.006 9	2.180 1
X_{20}	−0.085 3	−0.962 1	−0.089 2	0.030 1	0.014 1	−0.567 2
X_{21}	−0.183 7	−0.008 1	2.110 1	−0.018 7	−0.040 9	0.843 1
X_{22}	0.128 9	−0.061 8	−1.193 7	0.033 9	−0.030 2	0.626 1
X_{23}	−0.030 6	0.052 0	2.200 1	−0.023 1	−0.809 2	0.319 9
X_{24}	0.319 6	−0.828 7	−1.549 8	−0.945 0	−0.016 0	−0.409 2
X_{25}	−0.063 1	0.097 0	0.084 9	−0.007 4	0.007 3	2.178 9
X_{26}	−0.087 8	−0.953 8	−0.087 9	0.027 9	0.013 2	−0.564 7
X_{27}	0.109 8	−0.063 0	−1.194 8	0.034 7	−0.029 1	0.625 7
X_{28}	−0.179 3	−0.008 9	2.098 0	−0.019 5	−0.041 0	0.842 7
X_{29}	−0.031 2	0.049 8	2.170 2	−0.019 7	−0.808 5	0.320 0
X_{30}	0.321 4	−0.826 9	−1.553 9	−0.949 2	−0.016 9	−0.409 7

续表 2.9

序号	送风温度	送风压力	送回风流量	送风机控制偏差	回风机控制偏差	阀门控制偏差
X_{31}	−0.061 2	0.098 2	0.086 2	−0.007 0	0.006 8	2.179 3
X_{32}	−0.031 0	0.049 6	2.160 0	−0.021 0	−0.810 0	0.320 5
X_{33}	−1.510 0	0.000 4	0.027 1	0.000 2	−0.011 9	3.377 9
X_{34}	−0.090 0	−0.952 7	−0.090 1	0.030 2	0.014 3	−0.564 9
X_{35}	0.123 1	−0.062 1	−1.190 1	0.035 1	−0.029 7	0.626 3
X_{36}	−1.497 8	0.000 5	0.026 3	0.000 1	−0.012 1	3.378 4
X_{37}	0.003 7	0.025 6	−0.039 8	−0.009 8	0.012 7	−0.162 1
X_{38}	−0.183 0	−0.009 2	2.097 8	−0.020 1	−0.041 3	0.840 9
X_{39}	0.004 7	0.030 1	0.042 1	−0.009 9	0.013 3	−0.162 3
X_{40}	0.003 9	0.029 0	−0.039 9	−0.010 9	0.014 1	−0.161 3

3　离散事件动态系统建模

3.1　概述

随着计算机和信息技术的迅猛发展，在机械制造、交通管理、信息处理、军事指挥等领域，出现了一批人造的系统，例如，柔性加工系统、大规模计算机和通信网络、机场交通管理系统等。这类系统的特点是：对系统进程起决定作用的是一些离散事件，所遵循的是一些复杂的人为规则，这种系统称为离散事件动态系统，英文是 Discrete Event Dynamic Systems，简记为 DEDS。DEDS 的名词是美国哈佛大学何毓琦教授引入的。

清华大学郑大钟、赵千川两位教授对离散事件动态系统的概念、分析方法已经有比较系统的描述[51]。由于本书是从复杂系统建模角度对其进行讨论和研究，因而在模型的分类、问题的提法、研究的方法或思路，将会有差别。

由第 1 章内容可知，对模型的理解是广义的，一般包括知识模型(KM)、数学模型(MM)和关系模型(RM)，以此为基点，本章归纳的建模方法有：极大代数建模方法、Petri 网建模法、任务/资源图建模法、基于知识的建模法、基于系统理论形式化的建模法。

3.2　极大代数建模方法及其应用

3.2.1　极大代数法[51]

极大代数(Maximum Algebra)形成于 20 世纪 60 年代初，也称为路径代数(Path Algebra)。早在 1962 年，英国学者 Cuninghame-Green 就指出了极大代数的一些应用前景，而把极大代数成功用于 DEDS 的建模与分析，并且为这种方法做了开创性工作的则是法国学者 Cohen 及其合作者。在极大代数中，只定义了两种基本运算，即圈加和圈乘运算，分别记为$\oplus$和$\otimes$，其中以“圈加运算$\oplus$”取代“运算 max”，以“圈乘运算$\otimes$”取代“运算＋”。下面对极大代数的基本概念和运算规则给以简要的介绍。

(1) 极大代数的定义域 $\bar{R}$：

$$\bar{R} = \mathbf{R} \cup \{-\infty\} \tag{3-1}$$

式中：$\mathbf{R}$ 为实数域；$-\infty$ 为负无穷大。

(2) 圈加运算 $\oplus$：对于标量 a 和 b，$a,b \in \bar{R}$，有

$$a \oplus b = \max(a,b) \tag{3-2}$$

圈加运算 $\oplus$ 满足交换律、结合律和对圈乘运算 $\otimes$ 的分配律，即有

$$a \oplus b = b \oplus a \tag{3-3}$$

$$(a \oplus b) \oplus c = a \oplus (b \oplus c), c \in \bar{R} \tag{3-4}$$

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c), c \in \bar{R} \tag{3-5}$$

对于矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，且 $\boldsymbol{A},\boldsymbol{B} \in \bar{R}^{m\times n}$，有

$$(\boldsymbol{A}\oplus\boldsymbol{B})_{ij}=(\boldsymbol{A})_{ij}\oplus(\boldsymbol{B})_{ij}=\max\{(\boldsymbol{A})_{ij}\oplus(\boldsymbol{B})_{ij}\} \tag{3-6}$$

(3) 圈乘运算 $\otimes$：对于标量 a 和 b，$a,b\in\bar{R}$，有

$$a\otimes b=a+b$$

圈乘运算 $\otimes$ 满足交换律、结合律和对圈加运算 $\oplus$ 的分配律，即有

$$a\otimes b=b\otimes a \tag{3-7}$$

$$(a\otimes b)\otimes c=a\otimes(b\otimes c),c\in\bar{R} \tag{3-8}$$

$$a\otimes(b\oplus c)=(a\otimes b)\oplus(a\otimes c),c\in\bar{R} \tag{3-9}$$

(4) 零元 ε 和单位元 e：

$$\varepsilon=-\infty \tag{3-10}$$

$$e=0 \tag{3-11}$$

零元 ε 和单位元 e 满足式(3-12)、(3-13)、(3-14)的运算律，

$$\varepsilon\oplus a=a \tag{3-12}$$

$$\varepsilon\otimes a=\varepsilon \tag{3-13}$$

$$e\otimes a=a \tag{3-14}$$

(5) 幂指数运算：设 k 为正整数，任意标量 $a,a\in\bar{R}$，则极大代数幂指数关系式定义为

$$a^{k}=a\otimes a\otimes\cdots\otimes a=ka \tag{3-15}$$

相应的零幂指数、负幂指数和分数幂指数分别为式(3-16)、(3-17)、(3-18)，

$$a^{0}=e \tag{3-16}$$

$$a^{-k}=-ka \tag{3-17}$$

$$a^{\frac{1}{k}}=\frac{a}{k} \tag{3-18}$$

3.2.2 极大代数法在串行生产加工系统建模中的应用[51,52]

离散事件动态系统是一种特殊的、复杂的系统。主要特点表现在：离散性、异步并发性、随机性、动态性。它的进程是由发生在离散时刻的事件所驱动的，并且事件之间存在着复杂的相互作用关系，因而这种系统运行规律，很难用通常的微分方程和差分方程来准确描述。法国学者 Cohen 将极大代数方法应用于对离散事件系统的研究，这对于对系统特性的研究和对系统运行规律的准确描述，是十分有意义的，将为该类系统的分析、设计优化提供重要的依据和基础。下面介绍极大代数方法在串行生产加工系统建模中的应用。

1) 串行生产加工系统的数学模型

设状态 $x_{ij}(k)$ 表示机床 M_j 对工件 P_i 的加工条件，若机床 M_j 处于可利用状态和工件 P_i 处于到位状态，则可建立串行生产加工系统的数学模型，具体如下：

状态方程为

$$\left.\begin{aligned}
&x_{11}(k)=\max\{u_1(k),u_{n+1}(k)\}\\
&x_{1j}(k)=\max\{u_j(k),x_{1,j-1}(k)+t_{1,j-1}\},\quad j=2,3,\cdots,n\\
&x_{21}(k)=\max\{x_{11}(k)+t_{11},u_{n+2}(k)\},\\
&x_{2j}(k)=\max\{x_{1j}(k)+t_{1j},x_{2,j-1}(k)+t_{2,j-1}\},\quad j=2,3,\cdots,n\\
&\vdots\\
&x_{m1}(k)=\max\{x_{m-1,1}(k)+t_{m-1,1},u_{n+m}(k)\},\\
&x_{mj}(k)=\max\{x_{m-1,j}(k)+t_{m-1,j},x_{m,j-1}(k)+t_{m,j-1}\},\quad j=2,3,\cdots,n
\end{aligned}\right\} \tag{3-19}$$

输出方程为

$$\left.\begin{aligned} y_j(k) &= x_{mj}(k) + t_{mj}, \quad j = 1,2,\cdots,n \\ y_{n+i}(k) &= x_{in}(k) + t_{in}, \quad i = 1,2,\cdots,m \end{aligned}\right\} \tag{3-20}$$

式中：$k = 1,2,\cdots$ 表示工作批量。

由状态方程(3-19)和输出方程(3-20)可以看出，串行生产线的代数模型有着如下的一些基本特点：

(1) 状态方程严重非线性。状态方程(3-19)是一组逻辑型非线性方程，直接由此来分析系统的运行规律和行为性能，难于得到简明的解析表达式。

(2) 模型维数大。系统模型的维数为 $m \times n$，随着机床台数 n 和工件类数 m 的增加，模型维数将急剧增大。这是 DEDS 状态空间模型的一个主要缺点。

(3) 模型为确定性。对实际串行生产线，如果考虑不可避免的随机因素，那么系统参量 t_{ij} 应为随机量，系统状态变量相应地也应为随机变量。所以，严格地说，系统模型应当为一个随机性模型，需要采用概率统计方法来加以处理。但由于引入系统参量 t_{ij} 为确定性的基本假设，使在理想化处理后得到的系统模型为确定性模型。研究表明，按此分析的结果在很多情况下是有效的，而分析过程则可大为简化。

(4) 模型具有线性化的可能性。从模型(3-19)的逻辑型非线性可看出，如果引入一种特殊的代数运算体系，使把“逻辑取大运算 max”转化为“求和运算 $\oplus$”，把“常规加运算”转化为“求积运算 $\otimes$”，那么就可把逻辑非线性方程(3-19)线性化。本节中引入的极大代数就是具有这种运算特性的一种代数运算体系。

2) 串行生产加工系统在极大代数下的开环线性模型

在上述极大代数运算规则的基础上，可以导出串行生产线的开环线性模型。为此，在串行生产系统代数模型(3-19)和(3-20)中，以“圈加运算 $\oplus$”取代“运算 max”，以“圈乘运算 $\otimes$”取代“运算 +”。则可得到

$$\left.\begin{aligned} x_{11}(k) &= 0 \otimes u_1(k) \oplus 0 \otimes u_{n+1}(k), \\ x_{1j}(k) &= 0 \otimes u_j(k) \oplus t_{1,j-1} \otimes x_{1,j-1}(k), \quad j = 2,3,\cdots,n \\ x_{21}(k) &= t_{11} \otimes x_{11}(k) \oplus 0 \otimes u_{n+2}(k), \\ x_{2j}(k) &= t_{1j} \otimes x_{1j}(k) \oplus t_{2,j-1} \otimes x_{2,j-1}(k), \quad j = 2,3,\cdots,n \\ &\vdots \\ x_{m1}(k) &= t_{m-1,1} \otimes x_{m-1,1}(k) \oplus 0 \otimes u_{n+m}(k), \\ x_{mj}(k) &= t_{m-1,j} \otimes x_{m-1,j}(k) \oplus t_{m,j-1} \otimes x_{m,j-1}(k), \quad j = 2,3,\cdots,n \end{aligned}\right\} \tag{3-21}$$

和

$$\left.\begin{aligned} y_j(k) &= t_{mj} \otimes x_{mj}(k), \quad j = 1,2,\cdots,n \\ y_{n+i}(k) &= t_{in} \otimes x_{in}(k), \quad i = 1,2,\cdots,m \end{aligned}\right\} \tag{3-22}$$

其中，运算的优先序规定为“先圈乘运算 $\otimes$，后圈加运算 $\oplus$”，工作批量 $k = 1,2,\cdots$

为表达上的简洁，需要进一步将标量方程组形式的状态方程(3-21)和输出方程(3-22)化为向量方程的形式。为此，对系统变量引入向量表示：

$$\text{状态向量 } \boldsymbol{x} = [x_{11},\cdots,x_{1n};x_{21},\cdots,x_{2n};\cdots;x_{m1},\cdots,x_{mn}]^{\mathrm{T}} \tag{3-23}$$

$$\text{输入向量 } \boldsymbol{u} = [u_1,\cdots,u_n;u_{n+1},\cdots,u_{n+m}]^{\mathrm{T}} \tag{3-24}$$

输出向量 $\boldsymbol{y}=[y_1,\cdots,y_n;y_{n+1},\cdots,y_{n+m}]^{\mathrm{T}}$ (3-25)

相应地再引入具有适当维数的系统参数矩阵 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，那么串行生产加工系统在极大代数下的开环线性模型可以简记为

$$\boldsymbol{x}(k)=\boldsymbol{A}\boldsymbol{x}(k)\oplus\boldsymbol{B}\boldsymbol{u}(k) \tag{3-26}$$

$$\boldsymbol{y}(k)=\boldsymbol{C}\boldsymbol{x}(k) \tag{3-27}$$

式中，为了表达简洁起见，圈乘 $\otimes$ 符号被省略掉，各向量和矩阵的维数分别为

$$\dim(\boldsymbol{x})=mn,\dim(\boldsymbol{u})=\dim(\boldsymbol{y})=n+m$$

$$\boldsymbol{A}\in\overline{R}^{mn\times mn},\boldsymbol{B}\in\overline{R}^{mn\times(n+m)},\boldsymbol{C}\in\overline{R}^{(n+m)\times mn}$$

3）串行生产加工系统在极大代数下的开环线性模型的解

对串行生产加工系统的开环线性模型，即式(3-26)和式(3-27)，令 $p=mn$ 和 $q=n+m$，n 和 m 为系统的批工件数和机床台数。进一步定义 p 维参数矩阵 $\boldsymbol{A}$ 的“星运算”$\boldsymbol{A}^*$，即

$$\boldsymbol{A}^*=\boldsymbol{E}\oplus\boldsymbol{A}\oplus\boldsymbol{A}^2\oplus\cdots\oplus\boldsymbol{A}^{p-1},p=\dim(\boldsymbol{A}) \tag{3-28}$$

式中：$\boldsymbol{E}$ 为极大代数下的单位矩阵，那么，串行生产加工系统开环线性模型的状态解和输出解分别为

$$\boldsymbol{x}(k)=\boldsymbol{A}^*\boldsymbol{B}\boldsymbol{u}(k),\quad k=1,2,\cdots \tag{3-29}$$

和

$$\boldsymbol{y}(k)=\boldsymbol{C}\boldsymbol{A}^*\boldsymbol{B}\boldsymbol{u}(k),\quad k=1,2,\cdots \tag{3-30}$$

证明　对状态方程(3-26)，逐次左圈乘矩阵 $\boldsymbol{E},\boldsymbol{A},\boldsymbol{A}^2,\cdots,\boldsymbol{A}^{p-2},\boldsymbol{A}^{p-1}$，可得

$$\left.\begin{aligned}
&\boldsymbol{x}(k)=\boldsymbol{A}\boldsymbol{x}(k)\oplus\boldsymbol{B}\boldsymbol{u}(k)\\
&\boldsymbol{A}\boldsymbol{x}(k)=\boldsymbol{A}^2\boldsymbol{x}(k)\oplus\boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(k)\\
&\boldsymbol{A}^2\boldsymbol{x}(k)=\boldsymbol{A}^3\boldsymbol{x}(k)\oplus\boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}(k)\\
&\vdots\\
&\boldsymbol{A}^{p-2}\boldsymbol{x}(k)=\boldsymbol{A}^{p-1}\boldsymbol{x}(k)\oplus\boldsymbol{A}^{p-2}\boldsymbol{B}\boldsymbol{u}(k)\\
&\boldsymbol{A}^{p-1}\boldsymbol{x}(k)=\boldsymbol{A}^{p}\boldsymbol{x}(k)\oplus\boldsymbol{A}^{p-1}\boldsymbol{B}\boldsymbol{u}(k)
\end{aligned}\right\} \tag{3-31}$$

进而，将式(3-31)的各个方程由下而上依次迭代，可导出

$$\begin{aligned}\boldsymbol{x}(k)&=\boldsymbol{A}^p\boldsymbol{x}(k)\oplus[\boldsymbol{E}\oplus\boldsymbol{A}\oplus\boldsymbol{A}^2\oplus\cdots\oplus\boldsymbol{A}^{p-1}]\boldsymbol{B}\boldsymbol{u}(k)=\\&\quad\boldsymbol{A}^p\boldsymbol{x}(k)\oplus\boldsymbol{A}^*\boldsymbol{B}\boldsymbol{u}(k),\quad k=1,2,\cdots\end{aligned} \tag{3-32}$$

容易证明 $\boldsymbol{A}^p=\boldsymbol{\phi}$(零矩阵)，由式(3-32)即可得到状态解表达式(3-29)。再将式(3-29)代入式(3-27)，可导出输出解表达式(3-30)。证毕。

4）极大代数方法在轧钢厂 DEDS 中的应用[52]

下面的实例，是中国科学院系统研究所陈文德等学者成果。

太钢五轧厂热轧线上有3台轧机 m_1,m_2,m_3，主要生产卷板与中板，相应轧件记为 a，b。这里 a 顺序经 m_1,m_2,m_3 轧制，b 经 m_1,m_2 轧制。用 $\bar{t}_i^a,\bar{t}_i^b$ 分别表示 m_i 轧制 a，b 的平均所需时间，见表3.1。再用 C_i^a,C_i^b 表示 a，b 在 m_i 轧机前的辊道上的传送时间。现设每批轧件为 $p_1=a$，$p_2=\cdots=p_{M+1}=b$。问：如何选择 M 的值(即所谓轧制节奏)及每个 p_j 的出炉进入 m_1 辊道的时间，使生产线的班(8 h)总产量最高，而轧件无阻塞(即无碰撞)？

表 3.1　每台轧机轧制 a, b 的平均所需时间　　(单位:s)

时　间	对　象		
	m_1	m_2	m_3
$\bar{t}_i^a$	43.730 00	41.088 10	251.992 00
$\bar{t}_i^b$	55.375 50	63.888 54	—

因为只考虑确定性模型,故服务时间用均值 $\bar{t}_i^a$,$\bar{t}_i^b$。下面建立一个简单模型[53]:把轧机与它前面的辊道合在一起,形式上看作一个机器 m_i,对轧件 a 服务时间记为 $T_i^a = C_i^a + \bar{t}_i^a$;但易知在 m_i 接收 a 后经 $t_i^{ab} = T_i^a - C_i^b$ 时间即可在辊道上接收 b 而不至于阻塞,类似地记 $t_i^{ba} = T_i^b - C_i^a$,$t_i^{bb} = \bar{t}_i^b$,$t_i^{aa} = \bar{t}_i^a$,$T_i^b = C_i^b + \bar{t}_i^b$;用 (i,j) 表示 m_i 轧制 p_j。一批轧件的生产过程的有向图如图 3.1 所示。

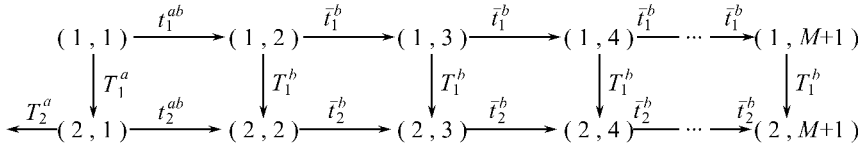


图 3.1　轧件生产过程的有向图

从图 3.1 得矩阵 $\boldsymbol{F},\boldsymbol{G},\boldsymbol{H}$:$\boldsymbol{F} \in \bar{R}^{(2M+3)\times(2M+3)}$,$\boldsymbol{G} \in \bar{R}^{(M+4)\times(2M+3)}$,$\boldsymbol{H} \in \bar{R}^{(2M+3)\times(M+4)}$,$\bar{R}$ 为极大代数,其定义如前所述。记 $\boldsymbol{E}$ 为 $\bar{R}$ 上单位阵,$\boldsymbol{F}^* = \bigoplus_{i=0}^{\infty} \boldsymbol{F}^i$。计算 $\boldsymbol{GF} * \boldsymbol{H}$,再取 $k_m = \boldsymbol{E}_i$,从 $\boldsymbol{GF} * \boldsymbol{H}$ 可得[53,54]

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k)\boldsymbol{A} \oplus \boldsymbol{u}(k+1)\boldsymbol{B} \tag{3-33}$$

式中:$\boldsymbol{x}(k) = [x_1(k), x_2(k), x_3(k)]$,$x_i(k)$ 表示第 $k+1$ 批中活动 $(i,\ 1)$ 的最早开始时刻,即 $x_i(k)$ 表示 m_i 辊道接收第 $k+1$ 批的 p_1 的最早时刻;$\boldsymbol{u}(k) = [u_1(k), \cdots, u_{M+1}(k)]$,$u_i(k)$ 表示第 k 批 p_i 的出炉时刻;

$$\boldsymbol{A} = \begin{bmatrix} t_1^{ab}(\bar{t}_1^b)^{M-1}t_1^{ba} & a_{12} & T_1^a\ T_2^a\ \bar{t}_3^a \\ \varepsilon & t_2^{ab}(\bar{t}_2^b)^{M-1}t_2^{ba} & T_2^a\ \bar{t}_3^a \\ \varepsilon & \varepsilon & \bar{t}_3^a \end{bmatrix} \tag{3-34}$$

$$\boldsymbol{B} = \begin{bmatrix} t_1^{ab}(\bar{t}_1^b)^{M-1}t_1^{ba} & b_{12} & T_1^a\ T_2^a\ \bar{t}_3^a \\ (\bar{t}_1^b)^{M-1}t_1^{ba} & T_1^b(\bar{t}_1^b \oplus \bar{t}_2^b)^{M-1}t_2^{ba} & \varepsilon \\ \vdots & \vdots & \vdots \\ \bar{t}_1^b\ t_1^{ba} & T_1^b(\bar{t}_1^b \oplus \bar{t}_2^b)t_2^{ba} & \varepsilon \\ t_1^{ba} & T_1^b\ t_2^{ba} & \varepsilon \end{bmatrix} \tag{3-35}$$

这里

$$a_{12} = b_{12} = (T_1^a\ t_2^{ab}(\bar{t}_2^b)^{M-1} \oplus t_1^{ab}\ T_1^b(\bar{t}_1^b \oplus \bar{t}_2^b)^{M-1})t_2^{ba} \tag{3-36}$$

$$\varepsilon = -\infty \tag{3-37}$$

计算 $\boldsymbol{A},\boldsymbol{B}$ 的过程中用了 $\bar{R}$ 上易验的公式

$$\sum_{i+j=s} (\bar{t}_1^b)^i (\bar{t}_2^b)^j = (\bar{t}_1^b \oplus \bar{t}_2^b)^s \tag{3-38}$$

若供料及时,即 m_1 允许接收 p_j 时,就出炉,这时 $u_j(k+1) = x_1(k)t_1^{ab}(\bar{t}_1^b)^{j-2}$,$j > 1$,

$u_1(k+1)=x_1(k)$，易知有 $\boldsymbol{x}(k)\boldsymbol{B}$ 各分量大于等于 $\boldsymbol{u}(k+1)\boldsymbol{B}$ 各分量，系统(3-33)变成

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)\boldsymbol{A} \tag{3-39}$$

$\boldsymbol{A}$ 为 d 阶 $\boldsymbol{\Lambda}$ 分场周期阵[55]，

$$\boldsymbol{\Lambda}=\begin{bmatrix}\lambda_1 & \sum\limits_{j=1}^{2}\lambda_j & \sum\limits_{j=1}^{3}\lambda_j\\ \varepsilon & \lambda_2 & \sum\limits_{j=2}^{3}\lambda_j\\ \varepsilon & \varepsilon & \lambda_3\end{bmatrix} \tag{3-40}$$

式中：$\lambda_1=t_1^{ab}(\bar{t}_1^{b})^{M-1}t_1^{ba}$；$\lambda_2=t_2^{ab}(\bar{t}_2^{b})^{M-1}t_2^{ba}$；$\lambda_3=\bar{t}_3^{a}$。式(3-39)为多周期系统，$x_i(k)$ 的周期为 $\sum\limits_{j=1}^{i}\lambda_j$[55]。

现按控制原则来取状态反馈[56]：当 m_1 允许接收 a,b 时，再经 Δ_a,Δ_b 延时后就令 a 或 b 出炉，这时 $\boldsymbol{u}(k+1)=\boldsymbol{x}(k)\boldsymbol{K}$，

$$\boldsymbol{K}=\begin{bmatrix}\Delta_a & \Delta_a t_1^{ab}\Delta_b & \Delta_a t_1^{ab}\Delta_b(\bar{t}_1^{b}\Delta_b) & \cdots & \Delta_a t_1^{ab}\Delta_b(\bar{t}_1^{b}\Delta_b)^{M-1}\\ \varepsilon & \varepsilon & \varepsilon & \cdots & \varepsilon\\ \varepsilon & \varepsilon & \varepsilon & \cdots & \varepsilon\end{bmatrix} \tag{3-41}$$

于是系统(3-33)变成

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)(\boldsymbol{A}\oplus\boldsymbol{K}\boldsymbol{B})=\boldsymbol{x}(k)\tilde{\boldsymbol{A}} \tag{3-42}$$

由式(3-38)、式(3-41)，经计算可得

$$\tilde{\boldsymbol{A}}=\begin{bmatrix}\tilde{a}_{11} & \tilde{a}_{12} & \Delta_a T_1^{a}\ T_2^{a}\ \bar{t}_3^{a}\\ \varepsilon & (\bar{t}_2^{b})^{M}\bar{t}_2^{a} & T^{a}\bar{t}_3^{a}\\ \varepsilon & \varepsilon & \bar{t}_3^{a}\end{bmatrix} \tag{3-43}$$

式中：$\tilde{a}_{11}=(\bar{t}_1^{b})^{M}\bar{t}_1^{a}\Delta_a\Delta_b^{M}$；$\tilde{a}_{12}=a_{12}\Delta_a\oplus\sum\limits_{i=1}^{M}t_1^{ab}(\bar{t}_1^{b})^{i-1}(\bar{t}_1^{b}\oplus\bar{t}_2^{b})^{M-i}T_1^{b}\ t_2^{ba}\Delta_a\Delta_b^{i}$。于是 $x_i(k)$ 的周期为 $\sum\limits_{j=1}^{i}\tilde{\lambda}_j$，这里 $\tilde{\lambda}_1=\tilde{a}_{11}$，$\tilde{\lambda}_j=\lambda_j$，$j=2,3$。由部分周期配置理论[54]及式(3-38)易知，仅有 $\tilde{\lambda}_1$ 可在$[\lambda_1,+\infty]$中配置，而其他 $\tilde{\lambda}_j$ 不变，式(3-41)正是实现这种配置的一类反馈。显然，系统(3-33)能达，因而也可用状态反馈合并配置周期，这提供了其他可能有用的更多的反馈方式，但本例仅限于研究式(3-41)这种简单方式。

取稳态后的班产量为优化的目标函数：

$$f=\frac{(W_a+MW_b)\cdot 8\cdot 3\,600}{\sum\limits_{j=1}^{3}\tilde{\lambda}_j} \tag{3-44}$$

式中：W_a,W_b 分别为 a,b 的平均质量 4.5 t 与 2 t，由式(3-43)与式(3-44)式易知

$$f=\frac{(W_a+MW_b)\cdot 28\,800}{\tilde{\lambda}_1},\text{当}\ \tilde{\lambda}_1\geqslant\tilde{\lambda}_2\oplus\tilde{\lambda}_3 \tag{3-45}$$

用普通运算写式(3-45)

$$f=\frac{28\,800(4.5+2M)}{[\bar{t}_1^{a}+\Delta_a+(\bar{t}_1^{b}+\Delta_b)M]},\text{当}\ \tilde{\lambda}_1\geqslant\tilde{\lambda}_2\oplus\tilde{\lambda}_3 \tag{3-46}$$

显然

$$\partial f/\partial \Delta_a < 0, \partial f/\partial \Delta_b < 0$$

所以，只需在 Δ_a，Δ_b 的最小值上寻找使 f 最大的 M，由计算知

$$\partial f/\partial M < 0 \Leftrightarrow 2(\bar{t}_1^a + \Delta_a) - 4.5(\bar{t}_1^b + \Delta_b) < 0 \tag{3-47}$$

现在先研究无阻塞条件，m_i 上无阻塞的充要条件是：第 k 批的 p_i 实际到达 m_i 的时刻大于等于 m_i 允许接收第 k 批的 p_j 的时刻。不失一般性，可设 $\boldsymbol{x}(0) = [0, T_1^a, T_2^a\ T_1^a]$。考察图 3.2 所示的有向图。对图中每个结点，其充要条件为

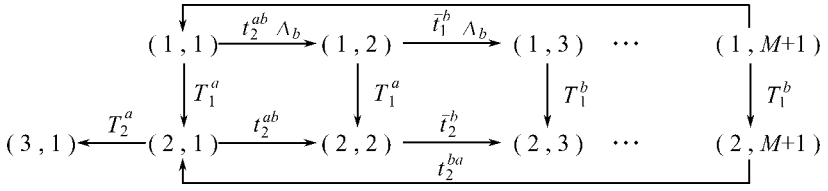


图 3.2　示例的有向图

$$t_1^{ab}\Delta_b T_1^b \geqslant T_1^a\ t_2^{ab}, \bar{t}_1^b \Delta_b T_1^b \geqslant T_1^b\ \bar{t}_2^b, t_1^{ba}\Delta_a T_1^a \geqslant T_1^b\ t_2^{ba} \tag{3-48}$$

易知：考虑多批轧件时还需加上条件

$$\bar{\lambda}_1 \geqslant \bar{\lambda}_2 \oplus \bar{\lambda}_3 \tag{3-49}$$

式(3-48)、(3-49) 经计算可表达成普通运算下的不等式组：

$$\left.\begin{aligned} &\Delta_a \geqslant \bar{t}_2^b - \bar{t}_1^a + C_2^b - C_2^a, \quad \Delta_b \geqslant \max\{\bar{t}_2^b - \bar{t}_1^b, \bar{t}_2^a - \bar{t}_1^b + C_2^a - C_2^b\} \\ &\Delta_a + M\Delta_b \geqslant \max\{\bar{t}_2^a - \bar{t}_1^a + M(\bar{t}_2^b - \bar{t}_1^b), \bar{t}_3^a - \bar{t}_1^a - M\bar{t}_1^b\} \end{aligned}\right\} \tag{3-50}$$

式(3-50) 就是无阻塞的充要条件。设 $C_i^a = C_i^b$，由表 3.1，式(3-50) 可得

$$\left.\begin{aligned} &\Delta_a \geqslant 20.158\,54, \quad \Delta_b \geqslant 8.513\,04 \\ &M \geqslant (208.262\,2 - \Delta_a)/(55.375\,5 + \Delta_b) \end{aligned}\right\} \tag{3-51}$$

f 的最优值必在

$$\Delta_a = 20.158\,54, \Delta_b = 8.513\,04 \tag{3-52}$$

上达到。这时，式(3-51) 是 $M \geqslant 2.944\,24$，再由式(3-47) 知：$\partial f/\partial M < 0$，所以 f 在 $M = 3$，及式(3-52) 上达到最高班产量：$f = 1\,183.310\,81$ t。

上述结果与张迪生等学者研究成果基本一致[56]，Δ_b 得到了改进。

3.3　基于 Petri 网建模方法

3.3.1　概述[51,57]

Petri 网(Petri Nets)，简称 PN，中文可翻译为“佩特里网”，是由德国科学家 C. A. Petri 于 1962 年在他的博士论文中首先提出的一种建模工具。此后，随着其理论上的不断完善和应用上的不断扩大，这种方法已经成为对离散事件动态系统(DEDS) 进行建模和分析的主要方法之一；Petri 网基本构造十分简单，但它能对复杂的大型系统进行精确的描述。本质上，Petri 网是由库所(place)、变迁(transition) 和连接库所与变迁的有向弧组成的图形研究工具，是一种研究系统组织结构和动态特性的理论，尤其适合于对异步并发系统的建模和分析。Petri 网在许多领域的建模中取得令人瞩目的成果，将其用于模型管理和建模支持一定会有广阔前景。

1) Petri 网

从抽象和一般的角度，一个 Petri 网表示为一个六元组，即

$$PN = (P, T, F, \boldsymbol{W}, M, M_0) \tag{3-53}$$

式中：$P = \{p_1, p_2, \cdots, p_m\}$ 为有限位置(place)集，称为库所，也可用 S 表示；$T = \{t_1, t_2, \cdots, t_n\}$ 为有限变迁(transition)集，称为变迁；$F \subseteq (p \times T) \cup (T \times p)$ 为节点流关系集，即为有向弧集；$\boldsymbol{W}: F \to \{1, 2, \cdots\}$ 为有向弧的权函数；$M: p \to \{0, 1, 2, \cdots\}$ 为状态标识含有托肯(token)的数量；$M_0: p \to \{0, 1, 2, \cdots\}$ 为初始标识含有托肯(token)的数量。

说明：① 在上述给出的 Petri 网表示中，网 $N = (P, T, F, \boldsymbol{W})$ 构成 Petri 网的结构。② 网 (N, M) 成为标识 Petri 网，其特征是在网的各个位置中引入了标识(token)或直接称其为“托肯”，有的学者称 token 为令牌。

2) Petri 网图

Petri 网图是对 Petri 网的六元组的一种图形化表示，在 Petri 网图中，P 代表“位置”节点集，T 代表“变迁”节点集，F 代表节点间的有向弧集，$\boldsymbol{W}$ 代表有向弧的“权”为元所构成的向量。从结构上看，标识 Petri 网图就是由 $(P, T, F, \boldsymbol{W})$ 所决定的一类有向二元图。

(1) 位置节点和变迁节点：对给定的一个标识 Petri 网，P 中的所有元表示为图的“位置”节点，采用符号“○”(圆圈)表示；T 中的所有元表示为图的“变迁”节点，采用符号“▬”(粗线段)或小方块表示。

(2) 有向弧：F 可用来决定在图的各个位置节点和各个变迁节点之间所存在的所有的有向弧。

(3) 有向弧的权 $\boldsymbol{W}$：画出一条有向弧并在其旁标注“权”，以表示输入或输出重数。当权数为 1 时，标注常常可以省略。

(4) 位置中“托肯”数的表示，在位置节点中标注相应的黑点数，以表示此位置所包含的“托肯”(token)数。

3) 变迁的发射规则

Petri 网本身一般仅能描述 DEDS 的静态结构。系统的动态过程是由状态标识的变化迁移过程来表征的。进一步，状态标识能否变迁和变迁结果，则是由 Petri 网中变迁的发射规则所决定的。

对标识 Petri 网，变迁发射有两个规则：

(1) 对变迁 t 的每一个输入位置 p_i 中包含的“托肯”数 $M(p_i)$ 不少于对有向弧 (p_i, t) 的权重 $W(p_i, t)$，即，$M(p_i) \geqslant W(p_i, t)$；对变迁 t 的每一个输出位置 p_i 的容量 $K(p_j)$，$K(p_j) \geqslant M(p_j) + W(t, p_j)$；对变迁 t 的每一个既为输入又为输出的位置 $p_r \in I(t) \cap O(t)$，位置 p_r 同时满足上述两个关系式，即位置 p_r 的容量 $K(p_r) \geqslant M(p_r) + W(t, p_r)$，$M(p_r) \geqslant W(p_r, t)$。

(2) 从变迁节点 t 的各个输入位置中减去“托肯”数即黑点数，减去“托肯”数即黑点数等于各输入位置之变迁节点 t 的输入有向弧的权；在变迁节点 t 的各个输出位置中加上“托肯”数即黑点数，增加“托肯”数即黑点数等于变迁节点 t 的各个输出位置的输出有向弧的权。

3.3.2 Petri 网在建模中的应用[57]

1) 基于 Petri 网的建模

建模支持的研究对象并不是子模型本身，而是子模型之间的连接关系。与子模型之间的连

接关系有关的因素有输入、输出和约束条件(通过算子的运算来体现)等,这些因素决定了模型之间的固有连接关系,而模型的状态则决定模型的先后执行次序。另外,对于由规则知识构成的模型中,规则决定了问题求解路径。因此,在 Petri 网建模中,主要考虑这些如何用 Petri 网表示。

(1) 数学模型的 Petri 网表示

① 库所。库所一般被用来描述模型的输入和输出变量,或者模型的状态。

② 变迁。变迁一般用来描述模型的算子,在有条件输出时,变迁还要用来描述约束条件。

③ 标识。标识是用来说明变量的值是否存在的符号,如库所上的标识为 1,则它所表示的变量值存在。当变迁的所有输入库所的标识均存在时,该变迁表示的算子就可以运算;若变迁的输入库所集中的一个库所的标识不存在,则该变迁表示的算子就不能运算。

④ 弧线。弧线表示模型库中的各个算子与变量之间相互关系,用双向弧线"↔"表示算子与算子的输入变量之间关系。若库所存在托肯,该库所的托肯不会再改变,它可作为不止一个表示算子变迁的输入。单向弧线"→"表示算子与算子的输出之间关系,还可表示状态与算子之间关系。

(2) 产生式规则的 Petri 网表示

在产生式系统中,产生式规则是由规则头和规则尾两部分组成,当规则头为真时,规则就激活,从而可推出规则尾为真。若把产生式规则的激活当作 Petri 网的变迁发生,把产生式规则的规则头(即条件)当作 Petri 网的输入库所,把它的规则尾(即结构)当作网的输出库所。这样产生式规则库就可连接成一个复杂的 Petri 网。

在 Petri 网中,库所的标志为 1 则表示该库所表示的谓项为真,根据产生式系统的事实库,就可确认某些库所的标志为 1,其他的则为 0。这样,Petri 网就可运行,其运行过程就是产生式系统的正向推理过程,在推理过程中出现的冲突问题,也就是 Petri 网的冲突问题。

(3) 建模过程

一个大中型的模型库所生成的 Petri 网是极其巨大的,它的连线非常复杂,一般很难用肉眼区别。在利用 Petri 网进行建模时,因要同时支持不同层次的用户,所以必须充分利用各类用户的信息,逐渐使模型完善,下面讨论建模的各个过程。

① 系统结构的 Petri 网表示

这一过程是支持信息用户,用 Petri 网表示模型的功能流程,描述各个模型执行的先后次序,而不研究各个子模型的具体内容,在 Petri 网中,库所表示各个子模型状态,状态的变化过程就是模型执行的次序,变迁还是用来表示子模型的算子,算子仅仅是一个约定的符号。这样,可把子模型之间连接关系通过 Petri 网表示出来。实际上,这一过程是对系统结构的描述,它体现了整个系统的结构模型。

② Petri 网的进一步完善

根据上面的 Petri 网,再进一步明确各个模型,包括输入、输出库所的连接和算子的刻画。这样,只要确定 Petri 网的起始变迁和终止变迁,就可运行和分析。在运行过程中,若发现它同实际情况不相符,可对 Petri 网进行修改。

③ Petri 网的简化

在模型库中,如果算子被刻画得非常基本,也就是模型被划分得非常细,那么它所形成的 Petri 网将十分巨大。为了简化 Petri 网,把可组合的算子重新组成一个新算子。这样,表示新算子的变迁可代替原有的所有表示被组合算子的变迁,而且表示中间变量的库所可省略。

2）模型的 Petri 网表示的示例[57]

（1）数学模型的 Petri 网表示（见图 3.3）

在图 3.3 中，Petri 网表示了瓦斯涌出量模型，它是由下列公式组成：

$$q = \mu K_{\rho} W_{含} \tag{3-54}$$

$$W_{含} = \frac{65.5(100 - A^5 - W^5)}{(a/P + b)(V^5)^{0.146} e^{n}(1 + 0.31V^5) \cdot 100} + \frac{K_{孔} P}{100 K_{压} V_{容}} \tag{3-55}$$

$$a = 2.4 + 0.21V^5 \tag{3-56}$$

$$b = 1 - 0.004V^5 \tag{3-57}$$

$$n = 0.026/(0.993 + 0.007P) \tag{3-58}$$

$$K_{压} = 1 - 0.002\,3P + 0.004t \tag{3-59}$$

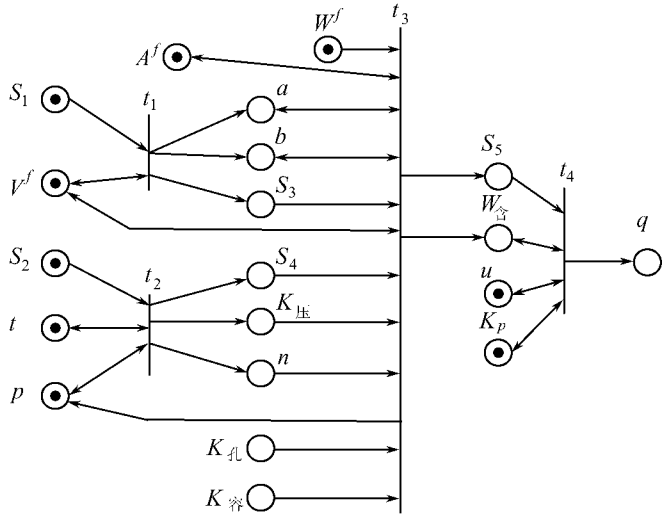
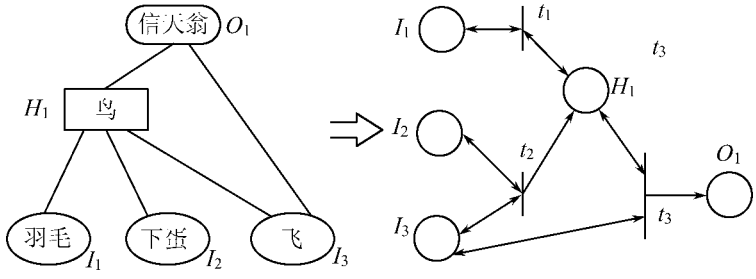


图 3.3　单一煤层瓦斯涌入量模型的 Petri 网表示

库所 S_1、S_2、S_3、S_4、S_5 表示各个子模型的状态，从图 3.3 中可知道，S_1、S_2 的标志为 1。变迁 t_1 和 t_2 就有发生权，当它们发生后，S_3、S_4 的标志为 1，从而使变迁 t_3 有发生权，最后变迁 t_4 才有发生权。变迁 t_1 表示的算子为式（3-56）和式（3-57），变迁 t_2 表示的算子是式（3-58）和式（3-59），变迁 t_3 表示的算子是式（3-55），变迁 t_4 表示的算子是式（3-54）。

（2）产生式规则的 Petri 网表示（见图 3.4）



(a) 动物识别推理网络　　(b) 对应的 Petri 网表示

图 3.4　动物识别推理网络及其 Petri 网表示

对于图 3.4(a) 的动物识别推理网络，可用图 3.4(b) 的 Petri 网表示。在图 3.4(b) 中，库所 I_1 表示是否有羽毛，I_2 表示是否下蛋，I_3 表示是否会飞，H_1 表示是不是鸟，O_1 表示是不是信天翁。若在事实库中，某一动物会飞且下蛋，那么库所 I_2、I_3 的标志为 1，I_1 的标志为 0。经过 Petri 网的运行，最终库所 O_1 的标志变为 1，得出结论，该动物是信天翁。

3.3.3　Petri 网的分层递归建模方法[58]

1) 基本理念

Petri 网的分层模型是用 Petri 网对复杂系统建模的一种手段和方法。原型 Petri 网用于复杂系统建模，模型规模相当庞大。在 Petri 网的分层模型中，一个 Petri 网中的某些基本元素(库所或变迁) 连同其外延可以加细成一个 Petri 网。把 Petri 网中某些基本元素加细的过程称为求精。分层模型的提出使复杂系统的 Petri 网模型的规模得到缩减，并可以清晰地反映出模型的层次，也便于用逐步求精的方法对模型及被模拟系统进行性能分析。

传统的 Petri 网分层模型的理论方法中，是不考虑递归的。这样的分层模型虽然给系统的建模和分析带来了方便，但并没有增强模拟能力，也就是说，不考虑递归的 Petri 网分层模型的模拟能力等价于原型 Petri 网。

进一步允许递归，从模拟能力上扩展了原型 Petri 网的概念。可以证明任意上下文无关语言都可以被一个分层递归 Petri 网识别(而原型 Petri 网是不具备识别任意上下文无关语言的能力的)。

2) 分层模型的定义和定理

Petri 网的分层模型在不少文献中已经出现过，但一般都只是对其用自然语言描述，尚未对其给出严格的形式定义。下面，对一般的分层模型给出一个形式定义。

定义 3.1　设 $N=(S,T,F)$ 为一个网，$x \in S \cup T$(S 为库所，T 为变迁)，

(1) 称 ${}^{\cdot}x^{\cdot}={}^{\cdot}x \cup x^{\cdot}$ 为 x 的外延。

(2) 称 $F_x=\{(x,y) \in F\}$ 为 x 的关联弧集。

定义 3.2　设 $N=(S,T,F)$ 为一个网，

(1) 若 f_T 是这样一种操作：对 $\forall t \in T$，用一个三元组 (S_1,T_1,F_1) 替换 t 和 F_t，即 $f_T(t,F_t)=(S_1,T_1,F_1)$，使得 $(S_1 \cup {}^{\cdot}t^{\cdot},T_1,F_1)$ 构成一个网，则称 f_T 为对网 N 的一个变迁替换。

(2) 若 f_S 是这样一种操作：对 $\forall s \in S$，用一个三元组 (S_1,T_1,F_1) 替换 s 和 F_S，即 $f_S(s,F_s)=(S_1,T_1,F_1)$，使得 $(S_1,T_1 \cup {}^{\cdot}s^{\cdot},F_1)$ 构成一个网，则称 f_S 为对网 N 的一个库所替换。

定义 3.3　设 $N=(S,T,F)$ 为一个网，f_T 是对网 N 的一个变迁替换。若 $t \in T$，使得 $f_T(t,F_t)=(\phi,\{t\},F_t)$，则称 f_T 对于变迁 t 施加的是为一个不变替换，简记为 $f_T(t)=I_t$；若对 $t \in T$，f_T 对 t 施加的都是不变替换，则称 f_T 为一个变迁恒等替换，记为 $f_T=I_T$。

同理，可以定义对库所 s 施加的不变替换 $f_S(s)=I_s$ 和库所恒等替换 $f_S=I_S$。

显然，对于任意一个网 $N=(S,T,F)$，如果对 N 施加一个变迁恒等替换 $f_T=I_T$ 和一个库所恒等替换 $f_S=I_S$，则 N 没有发生任何变化。

定义 3.4　设 $N=(S,T,F)$ 为一个网，若对网 N 施加一个变迁替换 f_T 和一个库所替换 f_S，且 f_T 和 f_S 中至少有一个不是恒等替换，则称 (N,f_T,f_S) 为一个以 N 为基础的二层网模型。

为了表述简略，在描述一个分层模型时，如果施加在某变迁(库所) 上的替换是一个不变

替换，则不必写出。同样地，当描述一个变迁（库所）替换 $f_T(f_S)$ 时，若 $f_T(t)(f_S(s))$ 的值未给出，则表示 $f_T(t)=I_t(f_S(s)=I_s)$。

下面给出示例，具体见图 3.5(a)，该图为一个网 $N=(S,T,F)$，其中 $S=\{s_1,s_2,s_3,s_4,s_5\}$，$T=\{t_1,t_2,t_3,t_4,t_5\}$，$F$ 如图所示。若对网 N 施加一个变迁替换 f_T 和一个库所替换 f_S，使得 $f_T(t_2,F_{t_2})$ 和 $f_T(t_3,F_{t_3})$ 如图 3.5(b) 所示，$f_S(s_2,F_{s_2})$ 和 $f_S(s_5,F_{s_5})$ 如图 3.5(c) 所示，则 (N,f_T,f_S) 是一个二层网模型。

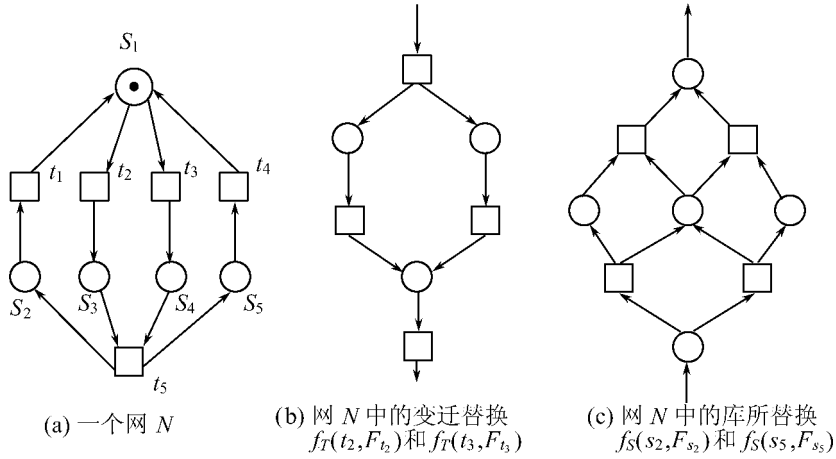


图 3.5　一个二层网模型 (N,f_T,f_S)

如果把二层网模型 (N,f_T,f_S) 中的替换 f_T 和 f_S 嵌入到网 N 的各个变迁和库所中，就得到一个复杂的、规模更大的网 N_2，记为 $N_2=(N,f_T,f_S)$，如图 3.6 所示。从二层网模型 (N,f_T,f_S) 得到 N_2 的过程称为（分层模型的）细化或求精。

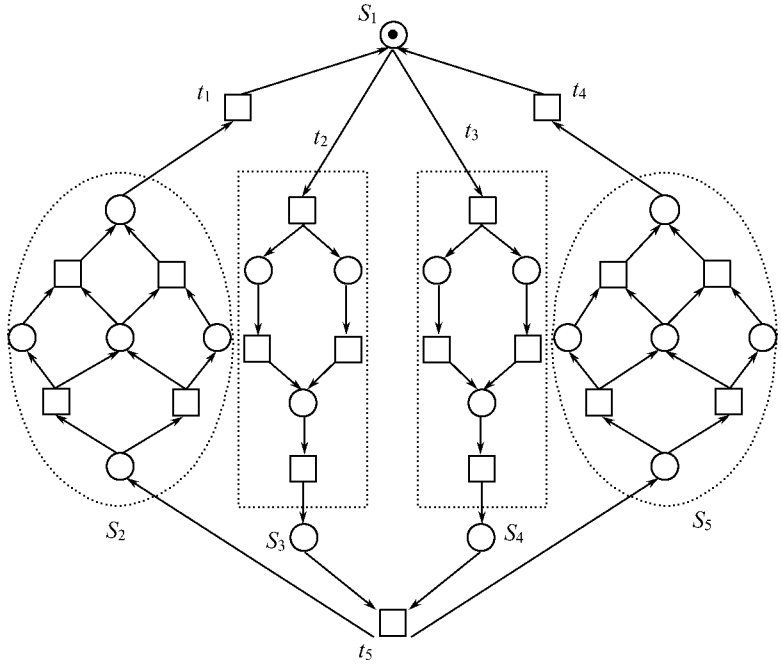


图 3.6　二层网模型 (N,f_T,f_S) 的求精

通过这个例子可以看出，若 $N_1=(S_1,T_1,F_1)$ 是一个网，f_{T_1} 和 f_{S_1} 分别为对 N_1 的变迁替换和库所替换。那么二层网模型 (N_1,f_{T_1},f_{S_1}) 等价于另一个网 N_2。如果 f_{T_1} 和 f_{S_1} 中至少有一个不是恒等替换，那么 N_2 比 N_1 有更大的规模。

定义 3.5　设 $N_{k-1}=(S_{k-1},T_{k-1},F_{k-1})$ 是一个以 $N_1=(S_1,T_1,F_1)$ 为基础的 $k-1$ 层网模型($k\geqslant 2$)，$f_{T_{k-1}}$ 和 $f_{S_{k-1}}$ 分别为 N_{k-1} 上的变迁替换和库所替换。如果 $f_{T_{k-1}}$ 和 $f_{S_{k-1}}$ 中至少有一个不是恒等替换，则称 $(N_{k-1},f_{T_{k-1}},f_{S_{k-1}})$ 为一个以 N_1 为基础的 k 层网模型，记为

$$N_k=((((N_1,f_{T_1},f_{S_1}),f_{T_2},f_{S_2}),\cdots),f_{T_{k-1}},f_{S_{k-1}}) \tag{3-60}$$

显然，对于任意一个确定的正整数 k，k 层网模型都等价于一个网。

定义 3.6　设 $\Sigma_1=(S_1,T_1,F_1,M_{01})$ 为一个 Petri 网，$N_k=(((N_1,f_{T_1},f_{S_1}),\cdots),f_{T_{k-1}},f_{S_{k-1}})$ 是以 $N_1=(S_1,T_1,F_1)$ 为基础的 k 层网模型，则称 $\Sigma_k=(N_k,M_{0k})$ 为一个以 Σ_1 为基础的 k 层 Petri 网模型，其中

$$M_{0k}(s)=\begin{cases}M_{01}(s), & \text{当 } s\in S_1\\ 0, & \text{其他}\end{cases} \tag{3-61}$$

由于分层网模型实质上是一个网的分层表示，所以可以直接得到下面结论。

定理 3.1　对任意确定的正整数 k，k 层 Petri 网模型的模拟能力等价于一个原型 Petri 网。

可以只用图形来简便、直观地表示一个分层网模型。其要领为：在被施加替换的网中，对那些要施加替换的库所和变迁(如果对一个库所或变迁施加的是不变替换，则认为对其不施变换)，分别用  和  表示，并在这些元素旁标上替换的结果(确切地说，标上替换的三元组连同该元素的外延构成的网的名称)。另用附图把这些替换结果表示出来，其中被替换元素的外延用虚线表示。例如，图 3.5 的二层网型可以用图 3.7 表示。

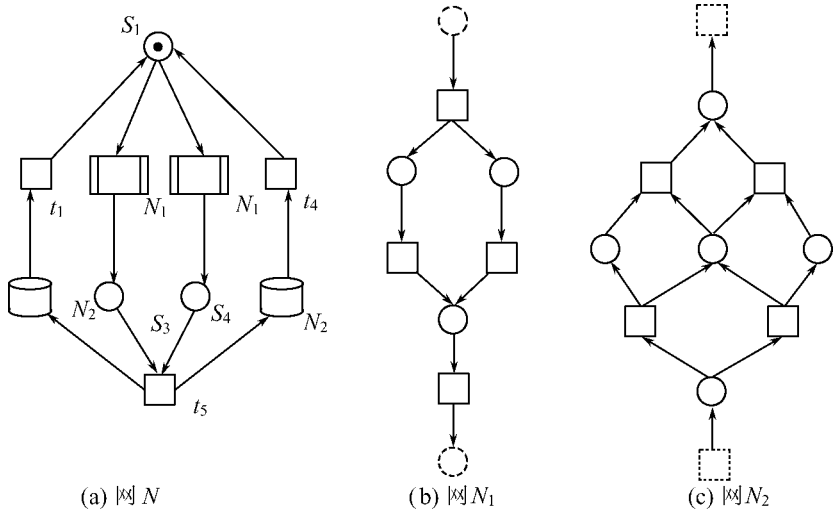


图 3.7　图 3.5 的二层网模型的另一种表示方法

3) Petri 网的分层递归模型

定义 3.7　设 $N_k=((((N_1,f_{T_1},f_{S_1}),f_{T_2},f_{S_2}),\cdots),f_{T_{k-1}},f_{S_{k-1}})$ 为一个 k 层网模型，如果下面两点：

(1) 存在 $t \in T_{k-1}, f_{T_{k-1}}(t, F_{(k-1)t}) = (S', T', F')$ 使得 $(S', T' \bigcup \cdot t^{\cdot}, F') = N_1$。

(2) 存在 $s \in S_{k-1}, f_{S_{k-1}}(s, F_{(k-1)s}) = (S', T', F')$ 使得 $(S', T' \bigcup \cdot t^{\cdot}, F') = N_1$ 中。

至少有一点成立，则称 N_k 为一个 k 层循环网模型。

定义 3.8 设 $N_2 = (N_1, f_{T_1}, f_{S_1})$ 为一个二层模型，如果下面两点：

(1) 存在 $t \in T_1, f_{T_1}(t, F_{1t}) = (S', T', F')$ 使得 $(S' \bigcup \cdot t^{\cdot}, T', F') = N_1$。

(2) 存在 $s \in S_1, f_{S_1}(s, F_{1s}) = (S', T', F')$ 使得 $(S', T' \bigcup \cdot s^{\cdot}, F') = N_1$。

则称 N_2 为一个自循环模型。

定义 3.9 设 $N_k = (((N_1, f_{T_1}, f_{S_1}), \cdots), f_{T_{k-1}}, f_{S_{k-1}})$ 为一个 k 层循环模型，定义

$$N_k^2 = (((((N_1, f_{T_1}, f_{S_1}), \cdots), f_{T_{k-1}}, f_{S_{k-1}}), f_{T_1}, f_{S_1}) \cdots f_{T_{k-1}}, f_{S_{k-1}}) \tag{3-62}$$

$$N_k^i = N_k^{i-1} \cdot N_k, \quad i = 3, 4, \cdots \tag{3-63}$$

特别地，当 $N_2 = (N_1, f_{T_1}, f_{S_1})$ 是一个自循环模型时

$$N_2^2 = N_2 \cdot N_2 = ((N_1, f_{T_1}, f_{S_1}), f_{T_1}, f_{S_1}) \tag{3-64}$$

$$N_2^i = N_2^{i-1} \cdot N_2, \quad i = 3, 4, \cdots \tag{3-65}$$

定义 3.10 设 N_k 为一个 k 层循环模型，则 $N_k^* = \bigcup_{i=1}^{\infty} N_k^i$ 称为一个分层递归网模型，若 N_2 是一个自循环模型，则

$$N_2^* = \bigcup_{i=1}^{\infty} N_2^i \tag{3-66}$$

称为一个分层自递归网模型。

定义 3.11 设 $\Sigma_1 = (N_1, M_{01})$ 为一个 Petri 网，Σ_k 是以 Σ_1 为基础的 k 层 Petri 网模型，则

$$\Sigma^* = \bigcup_{i=1}^{\infty} \Sigma_k^i = \bigcup_{i=1}^{\infty} (N_k^i, M_{0k}^i) \tag{3-67}$$

称为 Petri 网的分层递归模型，其中

$$M_{0k}^i(s) = \begin{cases} M_0(s), & \text{当 } s \in S_1 \\ 0, & \text{其他} \end{cases} \tag{3-68}$$

同理，可以定义 Petri 网的分层自递归模型。

从定义 3.11 看出，Petri 网的分层递归模型（分层自递归模型）是 Petri 网的分层循环（自循环）模型的 Kleene 闭包。这种模型比原型 Petri 网有更强的模拟能力。譬如，$L = \{ww^R(w \in \{a, b\}^+)\}$ 是一个上下文无关语言，这种语言不能被原型 Petri 网识别[59]。然而可以构造出一个 Petri 网的分层自递归模型，使得它所识别的语言恰好为 L。下面给出示例：识别 $L = \{ww^R(w \in \{a, b\}^+)\}$ 的分层递归 Petri 网模型。

图 3.8 是构造出的识别语言 $L = \{ww^R(w \in \{a, b\}^+)\}$ 的分层自递归 Petri 网模型 $\Sigma = (S, T, F, M_s)$，其中：$T = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6\}$，$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$，$M_s = (s_0)$，$T = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6\}$，影射 φ：使得 $\varphi(t_1) = \varphi(t_2) = a, \varphi(t_3) = \varphi(t_4) = b, \varphi(t_5) = \varphi(t_6) = \varphi(t_0) = \varepsilon$。

那么，Petri 网的分层递归模型的模拟能力有多强呢？下面的定理表明，它至少包括全部上下文无关语言。

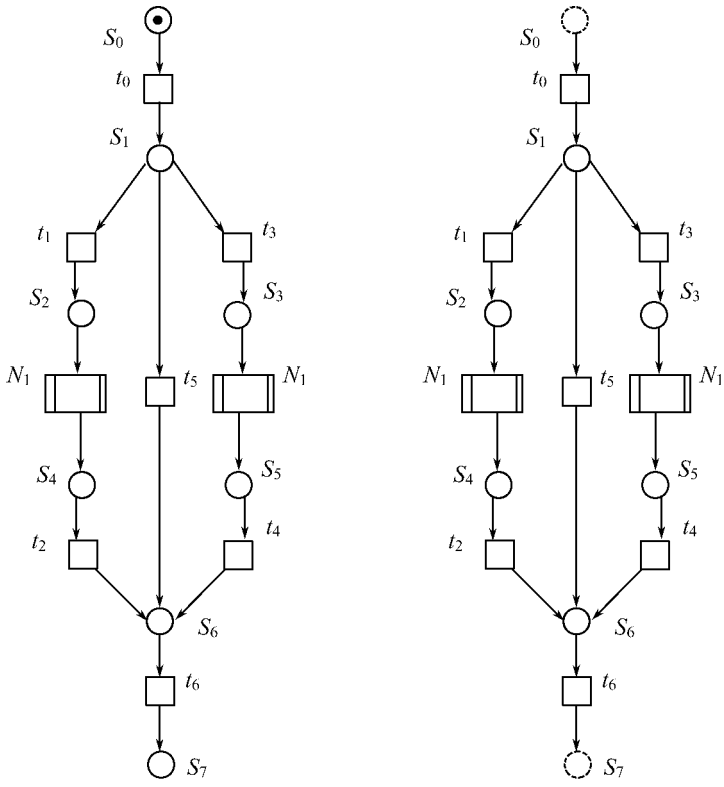


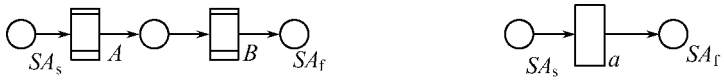
图 3.8　语言 $L=\{ww^R(w\in\{a,b\}^+)\}$ 的分层自递归 Petri 网模型

定理 3.2　如果 L 是一个上下文无关语言，则 L 可被一个 Petri 网的分层递归模型识别。

证明　由于 L 是上下文无关语言，所以 L 可以由上下文无关文法 Chomshy 范式 $G=(V,T,P,S)$ 产生，即 P 中的产生式只有两种形式：$A\to BC$，$A\to a$，其中：A、B、$C\in V$，$a\in T$。对于每种类型的产生式，都可以用一个分层(Petri) 网模型来描述。即具有下列的转换形式：

(1) $A\to BC$，对应的分层网模型见图 3.9(a)。

(2) $A\to a$，对应的分层网模型见图 3.9(b)。



(a) $A\to BC$ 的网模型　　(b) $A\to a$ 的网模型

图 3.9　产生式的分层网模型

图中的 SA_s、SA_f 分别表示非终结符 A 对应的网模型的起始位置和终止位置。

利用产生式对应的分层网模型，可以构造出一个标准 Petri 网 Σ 使得：$L(G)=L(\Sigma)$，其构造过程为：

(1) 对集合 P 中的每个产生式，根据产生式形式，构造出其相应的分层网模型。

(2) 对于任意 $A\in V$，将 A 对应的所有的分层网模型进行“共享合成运算”，构成一个新的分层模型 N_A。

(3) 对开始变量 $S \in V$,构成的对应的分层网模型 N_S,置 $M_S = (SS_s)$,形成分层 Petri 网模型 $\Sigma = (S, T, F, M_s)$。

进一步,利用归纳法,可以证明:$L(G) = L(\Sigma)$。

3.3.4 面向对象的 Petri 网建模[60]

面向对象技术是一种非常有效的程序设计范型,在制造、控制、计算机科学领域中越来越受到人们的重视和欢迎,这是因为用它开发出的应用系统能适应不断变化的客观环境的需要,并可在构件级上实现软件重用。但在构造复杂大系统时,需要构造大量的对象实体,对象间的关系也变得非常复杂,对象间隐含有并发性,相互通信与制约关系的表达不直观,容易出错,因而需要一种更直观、形式化的方法。Petri 网是一种图形化建模工具,与其他建模方法相比,Petri 网理论具有更严格的数学基础和直观易懂的图形表示,能充分描述系统的并发性、异步性、非确定性和并行性等特点,适于描述离散事件系统。但用它分析复杂系统时,模型非常复杂,这将增加计算的复杂程度,而且占用大量的存储空间。面向对象与 Petri 网技术相互结合,可使传统的 Petri 网技术模型得以简化,也使表达更为直观。由上述分析可知,这样可以综合两种方法的优势,弥补各自的缺点,为系统模型的设计、建立与分析提供更为有效的工具。这里研究一种面向对象的 Petri 网建模方法——OOPN(Object-Oriented Petri Net),它在将两种技术的优点结合的基础上,又考虑了机器人系统的特点,并在 AUV(Autonomous Underwater Vehicle,智能水下机器人) 系统建模上进行了实践。

1) 面向对象的 Perti 网

在 OOPN 中,系统可描述为相互通信的物理对象和它们之间的联系,为了便于描述,下面给出一些基本定义。

(1) 对象

用 OOPN 对系统进行建模时,一般可将对象分为两类,即模块对象和门对象。模块对象是根据实际的系统实体抽象得来的,和实际实体有着对应关系,其对外接口用消息库所来表示。门对象是为了对象间传递消息而增加的一种对象,它用一种特殊类型的变迁表示。若一个模块对象 A 由 A_1 和 A_2 组成,可用框子框住 A_1 和 A_2,如图 3.10 所示,表示封装与抽象,外部接口由消息库所(特殊类型的库所)、门(特殊类型的变迁) 和它们之间的流关系给出,各对象内部可分别有若干实体(用黑点表示 token),表示系统当前的状态。

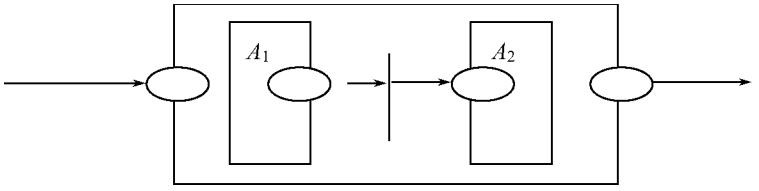


图 3.10 对象的示例

(2) 形式化描述

定义 3.12 $O_i = \{P_i, T_i, IP_i, OP_i, F_i\}$,其中 P_i 为 O_i 中的状态库所集,T_i 为活动变迁集,IP_i 为输入消息库所集,OP_i 为输出库所集,F_i 为活动变迁与状态/消息库所间的输入/输出关系。

定义 3.13　$R_{ij}=\{(OP_i,G_{ij},IP_j)\},(i\neq j)$，其中，$R_{ij}$ 表示消息发送对象 OP_i 与消息接受对象 IP_j 之间的通信互连关系，G_{ij} 表示一种特殊类型的变迁，称为门。

定义 3.14　$S=(O,R)$，其中 O 为对象集合，可记为 $O=\{O_i \mid i=1,2,\cdots,n\}$；$R$ 为关系集合，可记为 $R=\{R_{ij} \mid i=1,2,\cdots,m;j=1,2,\cdots,m;i\neq j\}$；$S$ 表示系统模型。

(3) 建立系统模型

对于一个应用系统，建立其 OOPN 模型的算法步骤如下：

步骤 1　确定系统中的各个对象。

① 根据系统构成的特点，生成系统的对象。

② 明确各对象内部的行为及其与其他对象间的联系。

③ 用消息库所集来表示对象的外部接口。

步骤 2　进行对象间的连接。

① 确定对象间的通信关系。

② 通过门连接对象间的外部接口。

步骤 3　确定系统的初始状态，即给定系统的初始状态，将 token 放入相应的库所中。

经过以上 3 步，就可以得到系统的模型了。该算法充分体现了面向对象的思想和 Petri 网图形化的优点。

下面对算法的空间复杂性进行讨论。假设每个库所在寄存器中所占空间为 1，系统模型中共有 m 个模块，每个模块中有 n 个库所，则利用传统 Petri 网建模方法进行建模时的空间耗费为 mn，而利用面向对象建模时的空间耗费为 Cm，其中 C 为常数。两种建模方法的空间复杂性分别为 $O(mn)$ 和 $O(m)$。由此可见，加入面向对象技术后，其空间复杂性得以明显降低，大大地节约了存储空间。

2) AUV 系统的描述

AUV 系统中包括 4 个模块：广播模块(BRO)、感知模块(DET)、规划模块(PLAN) 和控制模块(CON)。每个模块的动态行为用其相应的 OOPN 模型来表示，图 3.11 ～图 3.14 分别依次给出了 4 个模块的模型。这里将 OP_i，IP_j 缩写为 MP_i。对于系统中对象的描述，仅以广播模块为例，其他模块的只给出变量的说明。

(1) 广播模块

AUV 系统中，在每个时间节拍开始时，广播模块通过广播向整个系统发送节拍信号和系统状态，并起到同步的作用。可把广播模块看作一个对象，表示为 $O_1=\{P_1,T_1,IP_1,OP_1,F_1\}$，其中：

$$
\begin{aligned}
&P_1=\{P_{11}\}\\
&T_1=\{T_{11},T_{12}\}\\
&MP_1=\{MP_{11},MP_{12}\}\\
&F_1=\{(P_{11},T_{11}),(P_{11},T_{12}),\quad (T_{11},MP_{11}),\quad (MP_{12},T_{11}),(T_{12},P_{11})\}
\end{aligned}
$$

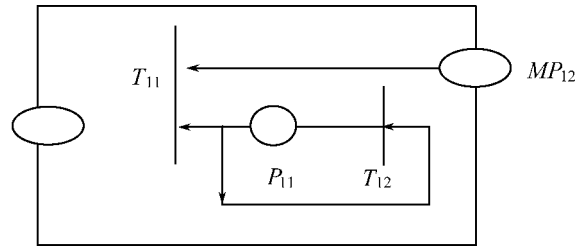


图 3.11　广播模块

图 3.11 中，MP_{11}：系统发出的节拍信号；MP_{12}：新的状态文件；P_{11}：

时钟计时结束产生的时钟信号；T_{11}：产生节拍信号；T_{12}：时钟计时。

（2）感知模块

由图 3.12 可以看出，感知模块包括 4 个子模块，分别为定位与避碰声纳模块、前视觉模块、下视觉模块和声视觉模块。它们之间为并行的，又通过节拍信号进行并发，每个子模块内部是通过顺序执行信号处理和写网，从而完成从感知到形成数据文件的过程。该图中，$P_{21} \sim P_{25}$ 分别为各模块的获得的信息，$P_{26} \sim P_{29}$ 分别为各模块信号处理后得到的数据，$P_{210} \sim P_{213}$ 分别为各模块更新后的数据文件，$P_{214} \sim P_{217}$ 分别为各模块得到的有效数据文件，$T_{21} \sim T_{24}$ 表示各模块进行的信号处理，$T_{25} \sim T_{28}$ 表示各模块进行写网，$T_{29} \sim T_{212}$ 表示各模块得到有效文件，T_{213} 表示各数据文件对系统数据库进行更新，MP_{21} 表示上一次 AUV 行动后得到的反馈信息，MP_{22} 表示系统的节拍信号，MP_{23} 表示有效的数据库文件。

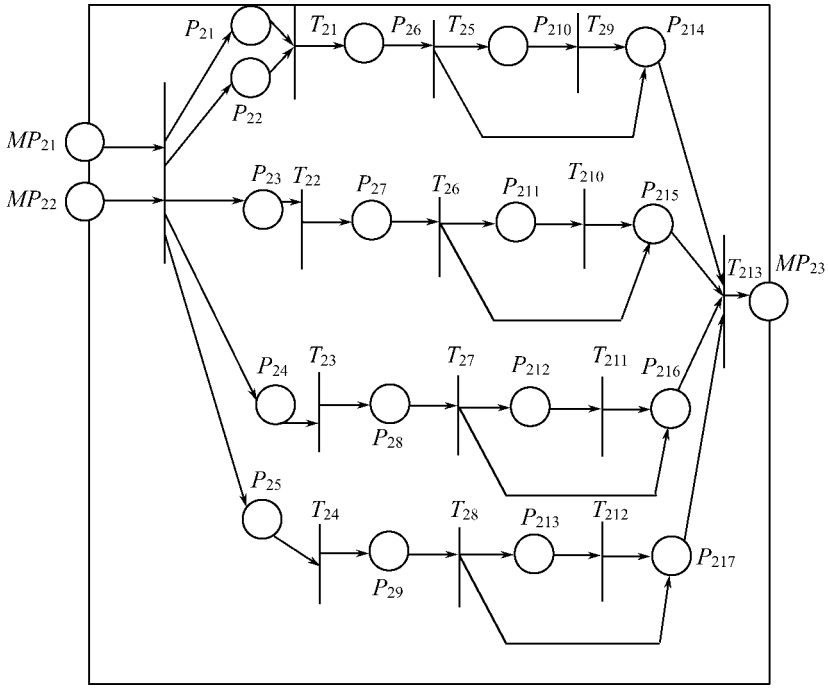


图 3.12　感知模块

（3）规划模块

规划模块主要是利用探测所得的数据文件，进行路径和作业的规划，并将规划的结果存贮到数据文件中。图 3.13 为规划模块的模型，其中，P_{31} 为路径规划所需数据，P_{32} 为作业规划所需数据，P_{33} 和 P_{35} 为路径规划和作业规划所得到的数据，P_{34} 为规划后改变系统状态数据；T_{31} 表示系统读取数据文件，T_{32} 和 T_{33} 分别表示路径规划和作业规划，T_{34} 和 T_{36} 分别表示将路径规划和作业规划数据写入文件，T_{35} 表示更新系统状态文件；MP_{31} 表示数据库文件，MP_{32} 表示系统的节拍信号，MP_{33} 和 MP_{35} 分别表示得到的路径规划文件和作业规划文件，MP_{34} 表示更新后的系统状态文件。

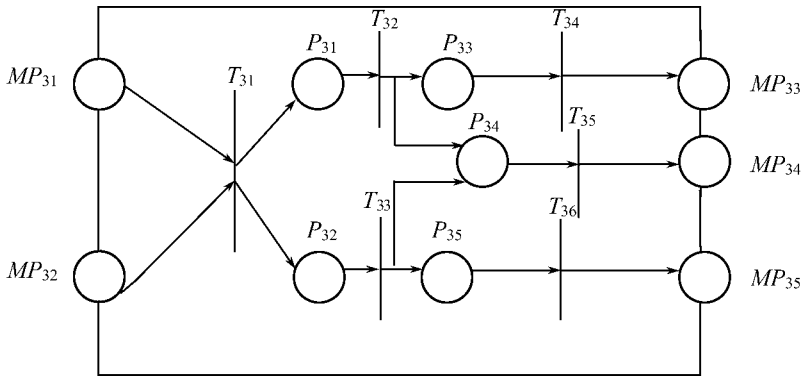


图 3.13　规划模块

(4) 控制模块

控制模块利用读到的路径规划和作业规划文件,并发地控制 AUV 的运动和作业,并得到 AUV 行动后的反馈信息,为下一次感知模块进行数据测量做好准备。图 3.14 为控制模块的模型,其中,P_{41} 和 P_{42} 分别表示系统读取的路径规划和作业规划数据,T_{41} 和 T_{42} 表示系统读取规划文件,T_{43} 表示系统对运动和作业动作的控制,MP_{41} 和 MP_{43} 分别表示路径规划文件和作业规划文件,MP_{42} 表示系统的节拍信号,MP_{44} 表示系统控制 AUV 行动后得到的反馈信息。

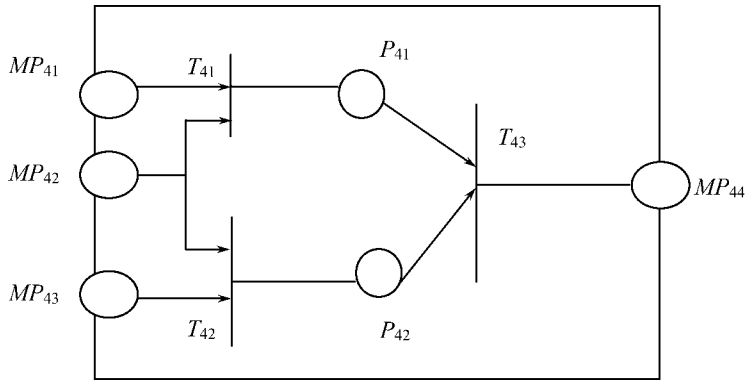


图 3.14　控制模块

(5) AUV 系统模型

在得到了各对象的模型后,就可以构造系统的模型了。AUV 系统模型如图 3.15 所示。$S=(OR)$,$O=\{O_1,O_2,O_3,O_4\}$,$R=\{(MP_{44},G_1,MP_{21}),(MP_{23},G_2,MP_{31}),(MP_{11},G_3,MP_{22}),(MP_{11},G_3,MP_{32}),(MP_{11},G_3,MP_{42}),(MP_{33},G_4,MP_{41}),(MP_{34},G_5,MP_{43}),(MP_{35},G_6,MP_{12})\}$。

图 3.15 中,G_1 表示将上次行动信息反馈给下次行动,G_2 表示读取数据库文件,G_3 表示发出系统节拍信号,G_4 和 G_5 分别表示读取路径规划和作业规划文件,G_6 表示读取新的状态文件。基本 OOPN 模型是由各对输入/输出消息库所及它们之间的连接和相关 OOPN 中相应的门组成的。如图 3.15 所示,反馈信息库所 MP_{44} 通过 G_1 将反馈信息传送给 MP_{21};MP_{11} 将系统的节拍信号分别传送给 3 个模块的 MP_{22},MP_{32} 和 MP_{42};MP_{23} 通过 G_2 将网上数据文件传给 MP_{31};MP_{41} 和 MP_{42} 通过 G_4 和 G_5 读取 MP_{33} 和 MP_{34} 中的路径规划文件和作业规划文件;

MP_{35} 通过 G_6 与 MP_{12} 相连，将新的状态文件传送到广播模块，同时钟一起产生节拍信号。

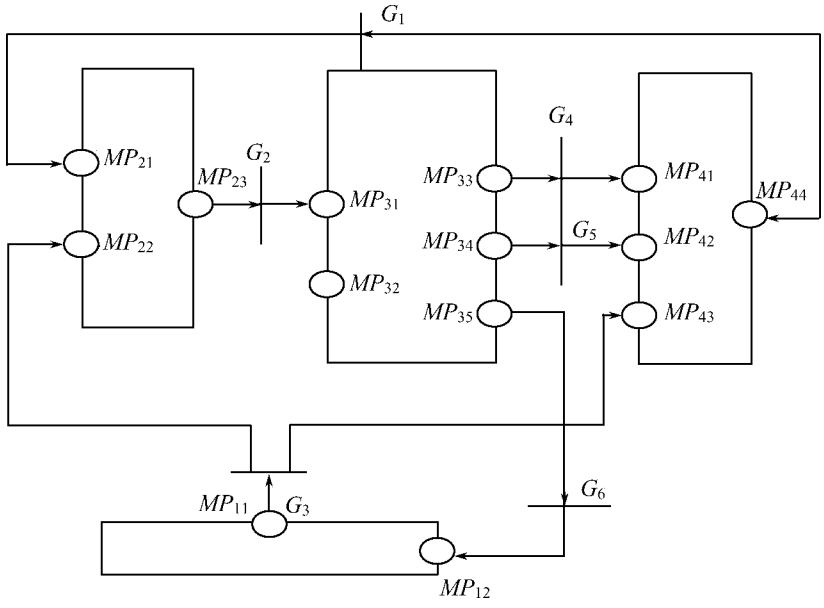


图 3.15　AUV 系统模型

3）小结

从 OOPN 建模思想及其在 AUV 系统中的应用可以看出，它具有以下几个特点：

（1）系统模型可以方便的进行细化或更高层次的抽象化；可以分层，具有很好的模块性。

（2）对象内部结构同外部结构相脱离，提高了可维护性。

（3）Petri 网模型中的库所可以和对象的状态、条件或某些数据相对应，而变迁则可和对象的行为进行对应，可进行相互映射，便于理解系统模型。

（4）兼顾形式化系统建模能力和分析能力，自底向上层次化、模块化建模。

（5）加入面向对象技术后，空间复杂性得以降低，大大地节约了存储空间，减小了“组合爆炸”出现的可能性；将面向对象技术与 Petri 网相结合，很好地满足了系统的要求，使系统具有较好的模块性，便于功能的添加和修改，加快了系统的开发周期。

3.3.5　模糊 H 网[61]

1）问题的提出

在 Petri 网中，转移结点可认为是完成一种简单处理的处理性结点，而位置结点可认为是存放标记（信息）用的存储性结点。为了分析系统的行为，先把处理结点和存储结点分开描述是有好处的，这样可以将系统的行为描述得更加清楚，也便于分析。可是，这种把处理和被处理的对象分开的模式有时对描述某些系统并不十分合适，例如，工厂生产，在那里并不是所有车间都把生产和仓储分得很清楚。此外，由于在 Petri 网中转移结点的输出量在输出时就马上被分送到各后续的位置结点去了，至于送去之后能否被利用或者够不够用就不再管而且也管不了，从而很可能在下一环节造成“积压”或“待料”现象，这与主观按计划分配资源时经常造成的不合理现象类似。要改善这种状况，客观上有必要引入一种把处理和

被处理对象合在一起描述的表示模式，使处理后所得的“成品”暂时存放在原地，待后续处理需要时再传输给后续部门。这种“按需供给”的模式可以缓解上述矛盾，此外，一旦分析清楚以后，为了实现或者模拟一个实际系统，分开处理就没有必要了。因为若用一些微处理器来实现网中的结点，微处理器本身就既具有处理能力也具有存储能力，如果非要把处理和存储分开反而违背常理。因此，从这个角度看，寻求一种能把两种功能结合在一起的表示模式既有理论意义，也有实用价值。

2）H 网

北京系统工程研究所何新贵研究员曾提出过一种模糊 Petri 网[62]，它使对现实系统的描述更加贴切，但并没有解决上述问题。于是，他引进一种对普通 Petri 网的改进，使之能部分满足上述要求，并取名为 H 网。它的功能与 Petri 网类似，但其中只有一种结点，称为 H 结点，这种结点既有处理功能又有存储功能，从而使得用它构成的网结构比 Petri 网更简单，而其表达能力却比 Petri 网强。

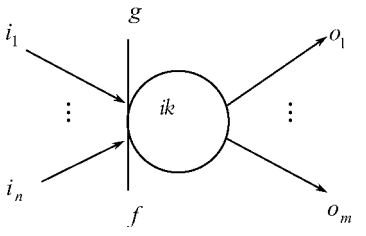


图 3.16　H 网的结点

每个 H 结点都有一个状态转移控制函数 $f(i_1, i_2, \cdots, i_n)$，它是定义在 n 个输入上的一个非负整数值函数，其输入 $i_1, i_2, \cdots, i_n$ 或者取 1 值，或者取 0 值。一个具有 n 个输入 $i_1, i_2, \cdots, i_n$ 和 m 个输出 $o_1, o_2, \cdots, o_m$ 的 H 结点用图 3.16 表示。

一个 H 网就是用这种 H 结点通过有向边连接而成的一个图，例如图 3.17 就是一个 H 网。处于一个有向边末端的结点称为处于该有向边始端的结点的输出结点，反之，始端结点称为末端结点的输入结点。

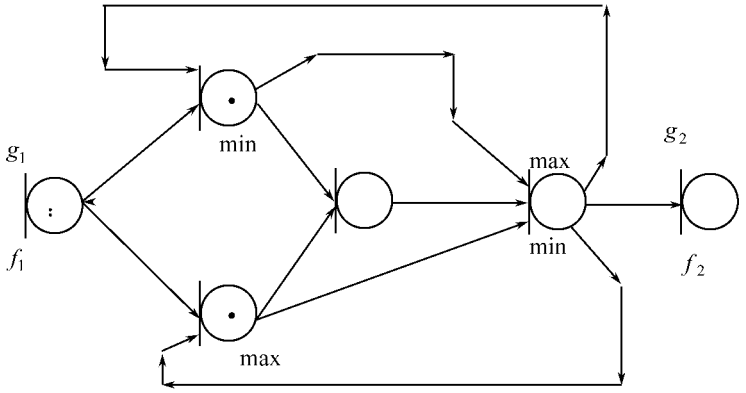


图 3.17　H 网示意图

H 网是一种可运行的图，H 网运行前，与 Petri 网一样需要给结点预先放置标记，如图 3.17 中圆点所示。运行时，计算 H 结点的状态转移控制函数 f 的值，其自变量值是当前该 H 结点的各输入量。如果一个 H 结点的某输入结点中放有标记（可以有多个），则对应输入线上的输入量为 1；如果其输入结点中未放标记，则对应输入线上的输入值为 0。若如此算出某 H 结点的 f 值大于等于 1，则该 H 结点就被点火，点火的结果是从其各输入结点中减去一个标记（当对应的输入值为 1 时），或者不减（当对应的输入值为 0 时）。并给本结点增加 g 个标记，其中 $g = g(i_1, i_2, \cdots, i_n)$，这里，函数 g 是定义在各输入量上的整数值函数，称为 H 结点的标记转移

函数。若如上算出的 f 值为 0，则该节点仍处休眠状态，不做任何动作。运行过程就是从一种初始标记状态开始，如此一步一步地点火并转移标记状态的过程。

定义 3.15　一个 H 网可形式地表示为一个五元组：

$$H=\{N,E,F(n),G(n),S_0(n)\} \tag{3-69}$$

式中：N 是一个 H 结点的有限集合；E 是 $N\times N$ 上的一个关系，表示 H 结点间的有向连接边的连接情况；$F(n)$ 是定义在 N 上的一个映射，它把 N 中的 H 结点 n 映射为一个定义在其输入结点的标记状态上的非负整数值函数，称为结点 n 的状态转移控制函数；$G(n)$ 是定义在 N 上的一个映射，它把 N 中的 H 结点 n 映射为一个定义在其输入结点的标记状态上的整数值函数，称为结点 n 的标记转移函数；$S_0(n)$ 是定义在 N 上的一个取值大于等于零的整数值函数，表示 H 网开始运行时各 H 结点的初始标记状态。

定理 3.3　任一 Petri 网都可用 H 网来等效地表示。

证明　从下面两对图（图 3.18 和图 3.19）的等效性显见，在图 3.18 中 $\min\times m$ 表示 $f(i_1,i_2,\cdots,i_n)=\min(i_1+i_2+\cdots+i_n)\times m$。函数 g 与 f 相同。在图 3.19 中，Σ 表示 $f(i_1,i_2,\cdots,i_n)=i_1+i_2+\cdots+i_n$。函数 g 与 f 相同。

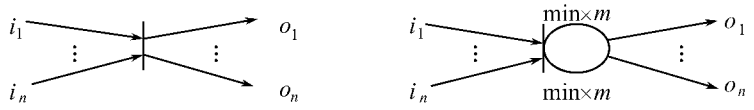


图 3.18　转移结点的等效性

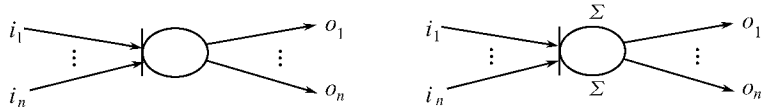


图 3.19　位置结点的等效性

该定理的逆定理是不成立的，因为附带在 H 结点上的状态转移控制函数和标记转移函数可以是取值于 $[0,\infty)$ 中整数的任意递增函数。可见 H 网比 Petri 网表达能力强得多。

3）模糊 H 网

为了描述和分析一些模糊系统的动态行为，并方便地用微处理器来物理地实现它们，将 H 网进一步模糊化为另一种模糊网是必要的。何新贵研究员引入模糊 H 网，这种网中也只有一种结点，称为模糊 H 结点。它兼有模糊 Petri 网的转移结点和位置结点的功能，用图 3.20 表示。其中，$0\leqslant(\alpha_i,\beta_j)\leqslant 1$ 表示有向边的连接强度，i_i 表示相应有向边上容许的最大输入量，o_j 表示相应有向边上容许的最大输出量，$i=1,2,\cdots,n$；$j=1,2,\cdots,m$。$0<\tau<\infty$ 称为结点的点火阈限。f 是定义在模糊 H 结点的输入量上的一个状态转移控制函数。g 是定义在模糊 H 结点的输入量上的一个标记转移控制函数。

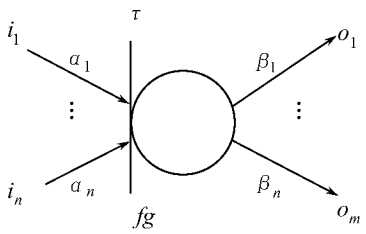


图 3.20　模糊 *H* 结点

模糊 H 网就是由这种结点通过有向边连接而成的图，如图 3.21 所示。由竖线一端进入的连接线称为输入连线，由圆圈一端引出的连接线称为输出连线。没有输入连线的结点称为该模

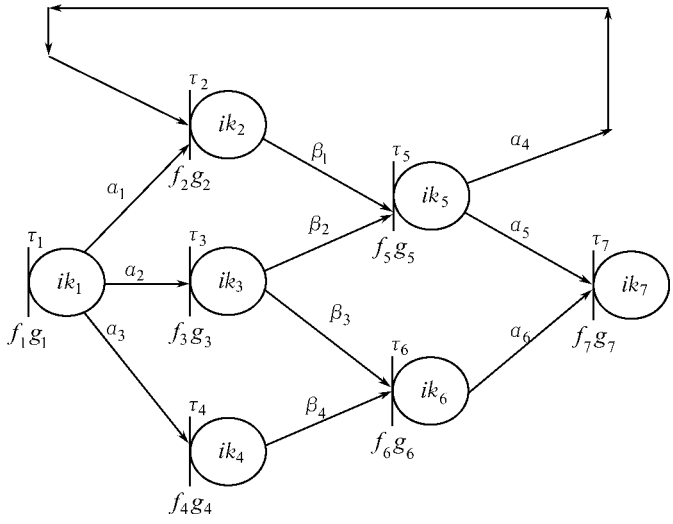


图 3.21　模糊 H 网

糊 H 网的输入结点，没有输出连线的结点称为该模糊 H 网的输出结点。模糊 H 网是可运行的，在运行之前，需先进行初始化，即给每个结点赋一个大于等于 0 的正实数(0 可以不标出)，称为结点的标记数。运行时，不断对各 H 结点计算从它的各输入结点传来的输入强度，输入强度是相应输入连线上的输入量 i_k(必须小于等于相应的最大输入量) 和连接强度 α_k 的一个二元实数值函数 $S(i_k,\alpha_k)$。一旦结点的各输入强度都计算出之后，接着就计算该结点的状态转移控制函数的值 f，f 是输入强度的一个非负函数。当 $f \geqslant \tau$ 时，该结点就具有点火能力，点火的结果是将其各输入结点的标记数减相应输入连线上的输入量 i_k，并将本结点的标记数加 $g = g(i_1, i_2, \cdots, i_n)$，其中标记转移函数 g 是一个实数值函数，特殊情况下可与 f 相同。模糊 H 网就是如此一步一步地不断运行。其中 $S(i_k,\alpha_k)$ 可根据具体情况采用不同的定义，例如令 $S(i_k,\alpha_k) = \min\{i_k,\alpha_k\}$ 或 $i_k \times \alpha_k$ 等等。

下面说明模糊 Petri 网与模糊 H 网的关系。

定理 3.4　任意一个模糊 Petri 网都存在一个与之等效的模糊 H 网，或者说任意一个模糊 Petri 网都可以表示成一个等效的模糊 H 网。反之不成立。

证明　因为模糊 Petri 网中一个模糊转移节点等效于一个模糊 H 节点，见图 3.22。其中 $f = \min \times m$ 表示 $f(s_1, s_2, \cdots, s_n) = \min(s_1, s_2, \cdots, s_n) \times m$，这里 $s_1, s_2, \cdots, s_n$ 为各输入连线上的输入强度。

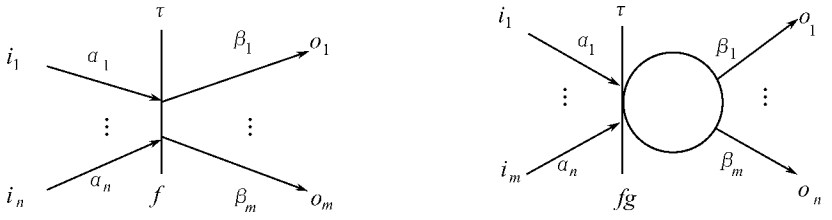


图 3.22　模糊转移结点的等效性

而模糊 Petri 网中的一个模糊位置结点等效于一个模糊 H 结点，见图 3.23。其中 $f = \Sigma$，表

示 $f(s_1,s_2,\cdots,s_n)=s_1+s_2+\cdots+s_n$，这里 $s_1,s_2,\cdots,s_n$ 为输入连线上的输入强度。

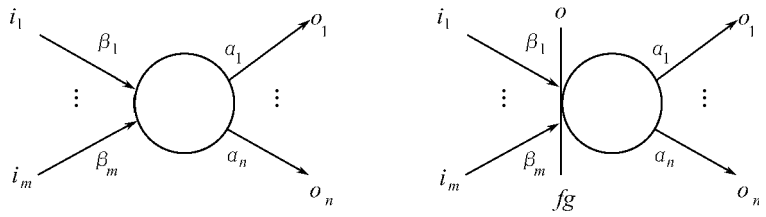


图 3.23　模糊位置结点的等效性

反之，显然逆定理是不成立的，因为模糊 H 网上的各种参数的选择范围很大。模糊 H 网在复杂系统模拟仿真、资源调度、并发网络和模糊电路设计，以及模糊推理等方面将有很广的应用前景。

Petri 网及其改进网，在复杂系统建模的应用范围是相当广阔的[51,57～59,63,64]。

3.4　任务/资源图建模法[4]

抛开严格的定义，可以说，复杂系统是指由多种类型的子系统组合而成的混合系统，包括连续系统和离散系统、非实时系统和实时系统等，如图 3.24 所示。尽管复杂系统由多个连续和离散子系统组成，然而当从整体角度去分析时，则可以将各个子系统之间的交互作用视为数字离散地进行。

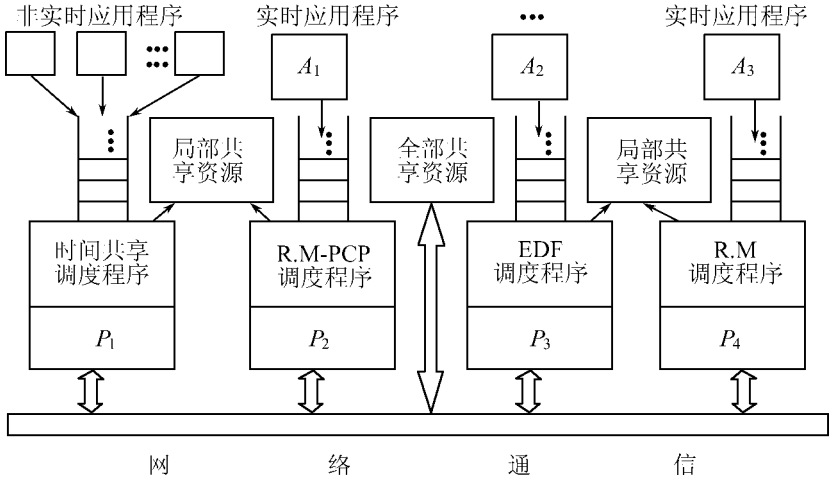


图 3.24　复杂系统模型

典型的离散事件动态系统建模方法有排队论方法、实体流图法、Petri 网方法等，虽然它们都能从不同角度描述目标系统，但由于这些方法都缺乏充分针对实时系统时间特性的机制，因而不太适合复杂离散实时系统建模、调度分析和仿真测试。2000 年，瞿继权、戴金海两位学者提出任务/资源图建模方法[4]，该方法是一种建立在定期任务模型基础上，结合了最新实时系统时间限制验证理论和非定期任务处理、资源访问控制等实时计算、调度理论的建模方法，它通过任务、边、资源等丰富的时间属性来分析实时系统的静态特性，通过大量的任务事件来分

析实时系统的动态特性。表 3.2 列出了任务/资源图建模方法与其他典型建模方法在系统特性描述方面的比较。

表 3.2　任务/资源图模型与其他典型仿真模型的比较

比较类别	建模方法			
	任务/资源图模型	排队论模型	实体流图法	Petri 网模型
实体	任务、资源	服务机构、客户	临时实体、永久实体	库所集、变迁集、令牌
属性	任务的时间参数、功能参数、互联参数，资源需求参数、资源参数	客户到达时间、等待时间、服务时间、队列长度，服务机构平均处理率等	临时实体到达时间、等待时间、服务时间、队列长度，永久实体平均处理率等	容量、权、标识等
事件	多达 29 种任务实例事件，用户还可以创立自己的事件	客户到达，服务机构开始服务、完成服务等几种事件	临时实体到达，永久实体开始服务、完成服务等几种事件	变迁发生，获取令牌，消耗令牌，变迁冲突等
进程	作业集按照优先级调度法则一次被调度执行的过程，其间作业可以被抢占	客户按照服务机构分配的处理器时间执行，其他客户不能抢占它	临时实体按照永久实体分配的处理器时间执行，其他临时实体不能抢占它	变迁任一输入库所中令牌数大于或等于输入弧的权值，输出库所中已有令牌数与输出弧权值之和小于输出库所容量，则变迁被授权发生
数据流	边及其参数	各实体服从 IID	各实体服从 IID	序偶
调度算法	优先级驱动可抢占调度算法，如 RM 、DM、EDF 等	FIFO、LIFO 根据设定客户优先级进行调度	FIFO、LIFO 根据设定客户优先级进行调度	一般根据实际研究的对象系统来确定调度算法

3.4.1　任务、边、资源及其参数

任务的参数主要由 4 部分组成，即时间参数、功能参数、互连参数及资源需求参数，其中时间参数和功能参数称为普通参数。时间参数包括就绪时间、执行时间、时限、周期、相位和结束时间。功能参数包括任务名称、权值、可抢占性、松弛类型、松弛函数、可选间隔和优先级。任务图中的边表示定期任务之间的数据、时间、控制等属性相互之间的依赖关系。边参数和互连参数都是表示任务之间相互依赖关系的属性，包括数据量参数和最小、最大时间间隔。资源是处理器、存储设备 I/O 设备、信号灯等系统资源的抽象，资源的参数有资源类型、资源数量、可消耗性等。

任务对资源的需求和使用是通过资源需求参数来实现的。资源需求参数包括资源名称、资源需求开始时间和结束时间、资源需求数量。例如，某项任务的资源需求参数为

Memory:[15→30]:400.0.0

Sensor:[18→50]:1.0.0

该参数表明执行任务时，在 15→30 时需要 400 单位数量的 Memory，其获取时间和释放时间都是 0；而在 18→50 时需要 1 个数量的 Sensor，Sensor 的获取时间和释放时间也都为 0。

3.4.2 任务图模型

1）任务图的基本概念

定义 3.16 任务图

任务图是以任务集 T 为顶点，由边集 E 连接而成的图 $G=(T,E)$，其中任意两项顶点任务之间的关系集 $R(T_i,T_j)$ 满足：

$$R(T_i,T_j)=\begin{cases}E, & T_i \text{ 与 } T_j \text{ 相关}\\ \Phi, & T_i \text{ 与 } T_j \text{ 相互独立}\end{cases} \tag{3-70}$$

利用任务图模型和任务、边丰富的时间属性可以对复杂实时系统的时间特性进行抽象建模，为系统的调度分析和仿真测试做好准备。

任务图有如下两个性质：

(1) 任意两个任务之间不能有多于一条边连接。

(2) 任务图是非环式的。唯一的例外就是当定期任务或不规则任务(由一系列不规则任务实例组成)是由一个任务实例链组成时，后面任务实例执行依赖于前面任务实例执行的结果，此时边可以形成自环。

如图 3.25 所示为一简单作战飞机航电设备的子任务图. 包括 Radar 任务和 Display 任务，Display 是 Radar 的当前后续任务。Radar 任务表示雷达的信号处理，而 Display 任务则把信号处理的结果显示给飞行员。根据实际系统可以确定各任务的参数，例如，如果 Display 任务的频率为 60 Hz，则每一帧信息显示的周期 Period 就可以设为 50 ms。假设每一帧信息包括飞机自己的位置、速度、加速度，目标的位置、速度和加速度，以及目标识别结果，是否可以开火判断，信号传递的附加信息等等，信息容量约为 200 byte，则可假设边 DI 的 Data Volume - 200 byte。类似地可以确定出其他参数值。

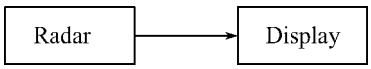


图 3.25 飞机航电设备子任务图

定义 3.17 可约子任务图

设任务图 G 是任务图 T 的子任务图，其中 G 的任务结点集为 $\{g_i\}$，T 的任务结点集 $\{t_i\}$，则若：

(1) $\{g_i\}$ 中存在唯一的结点 s，使得所有从 $\{t_i\}-\{g_i\}$ 到 $\{g_i\}$ 的边都形如 $\text{edge}(t_j,s)$，$t_j \in \{t_i\}-\{g_i\}$；

(2) $\{g_i\}$ 中存在唯一的结点 v，使得所有从 $\{g_i\}$ 到 $\{t_i\}-\{g_i\}$ 的边都形如 $\text{edge}(v,t_j)$，$t_j \in \{t_i\}-\{g_i\}$；

就称子任务图 G 是可约的。

可约子任务图由于与外部环境交互单一，因而很适合模块化结构和具有封装性，利于编程实现。如图 3.26 中的 A、B、C、D 4 项任务组成的子任务图就是可约的。

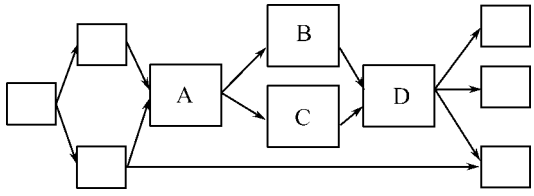


图 3.26 可约子任务图

2）任务图的多模式切换

实时系统可以有多种模式的操作，例如，一个航电系统，有针对飞行器起飞前的模式，针对起飞的模式，针对降落的模式，针对正常飞行操作模式和针对战斗操作模式，各种模式都具有

一些共同的任务。任务图存在有多种模式，就必然有多模式切换。模式切换总是由某些任务的执行或外部事件引起的。

系统每种模式里所做的工作是通过一个单独的任务图(或子任务图)来描述的。不同模式中相似的任务可以用相同的任务名称来描述，此时同名的任务描述了尽管相似但却不同的系统行为，任务名称可以在前面加上模式限制以示区别：modetaskname。不同模式的同名任务的参数值不必相同，例如航电系统中的巡航飞行模式 routine_flight 和敌对环境 hostile_environment 模式，两者都有监视和显示飞行器燃油状况的任务 fuel_level；当巡航飞行时，fuel_level 的 Period 为 1 s，而在敌对环境中 fuel_level 的 Period 为 3 s，以使系统资源更有效地对付敌对目标，此时

routine_flight. fuel_level. period＝1 s

hostile_environment. fuel_level. period＝3 s

有时候表示不同模式的任务图可能是相交的，即一项任务同时属于两个任务图(子任务图)，此时该任务在任一模式中都将执行。当某一任务的执行或外部事件(同样被视为一外部任务)触发一个模式切换时，就用一条从该任务指向新模式中某一任务的边来表示切换过程。如图 3.27 所示，控制核反应堆系统的 Start_Up 模式和 Normal_Operations 模式，任务 Check_Status 既属于 Start_Up 模式，又属于 Normal_Operations 模式，当 Check_Status 执行后，如果核反应堆已经在线运作了，则系统并行模式切换，进入 Normal_Operations 式，并执行 Start_Operations，否则系统继续在 Start_Up 模式中执行 Diagnosis 任务。

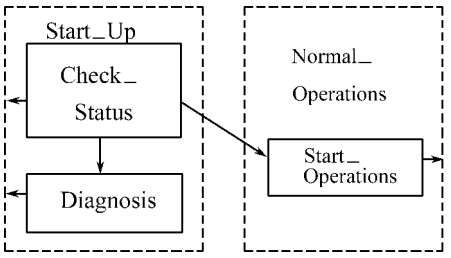


图 3.27　任务图的多模式切换

3.4.3　资源图模型

定义 3.18　资源图

下面给出资源图和资源图中边的定义：

若集合 $N=\{N_1,N_2,\cdots,N_n\}$，集合 $R=\bigcup_{i=1}^{n}R_i$ 和集合 F 构成的三元组 $G=\{N,R,F\}$ 满足：

(1) $N\cup R\neq\varnothing$　　(3-71)

(2) $N\cap R=\varnothing$　　(3-72)

(3) 对任意二元关系集 $F_1=N_i\times R_i$ 有

$$F_1\subseteq F\subseteq N\times R \tag{3-73}$$

(4) $\mathrm{dom}\{F\}\cup\mathrm{cod}\{F\}=N\cup R$　　(3-74)

则称 G 为资源图。

上述定义中，条件(1)、(2)表明资源图由 N 和 R 两类实体元素组成，N 表示结点集，R 表示资源集；条件(3)则表明 F 是由一个 N 元素和一个 R 元素组成的有序偶集合，一个结点总是对应于一个资源子集，F 称为边集；条件(4)表明 G 不能有孤立元素，从而 N、R、F 均不能为空集。

定义 3.19　包含边和访问边

若资源图的结点集 $N=\{N_1,N_2,\cdots,N_n\}$，资源集 $R=\bigcup_{i=1}^{n}R_i$，边集 $F=F_1\cup F_2$，若

(1) N_i 与 N_j 相异，$i\neq j$：

$R_i\cap R_j=\varnothing, i=1,2,\cdots,n; j=1,2,\cdots,n; i\neq j$

$F_1\cap F_2=\varnothing$

(2) $F_1=N_i\times R_i; F_2=N_i\times R_j,\ i\neq j$；

则称 F_1 中的元素为包含边，F_2 中的元素为访问边。

一般来说，资源都属于某一节点，从该节点到其所属资源的边就是包含边，而从一个节点到另一节点的资源的边就是访问边。如图 3.28 就是一个典型的资源图。

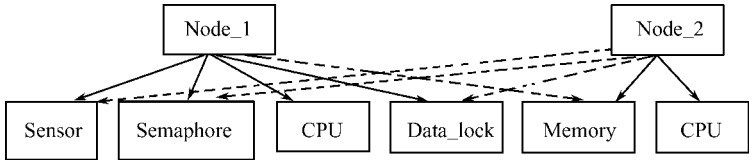


图 3.28　典型资源图模型

图中结点集 $N=\{\text{Node\_1},\text{Node\_2}\}$；$R_1=\{\text{Sensor},\text{Semaphore},\text{CPU},\text{Data\_lock}\}$，$R_2=\{\text{Memory},\ \text{CPU}\}$，$R=R_1\cup R_2$ 构成了资源集；Node_1 到 R_1，Node_2 到 R_2 的边组成了包含边集；Node_1 到 R_2，Node_2 到 R_1 的边组成了访问边集。

如图 3.29 为对应于图 3.25 飞机航电设备子任务图的资源图。在 Node_Radar 中，可以把信号处理设备全体抽象为 CPU 处理器，信号经 CPU 处理以后，其结果通过 Transmission_Line 传送给 Node_Display. Node_Display 除了有 CPU、Monitor 资源外，还与 Node_Radar 共享 Transmission_Line 资源。

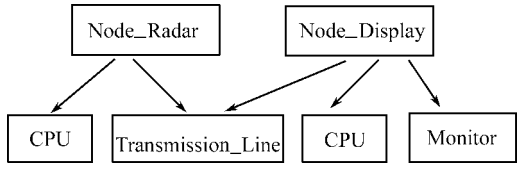


图 3.29　飞机航电设备资源图

任务/资源图模型由任务图模型和资源图模型结合而成，资源图中的处理器(CPU)通过调度算法来处理就绪的任务，同时根据任务执行对资源的需求通过资源访问控制协议，利用资源以满足任务执行。

任务/资源图建模方法由于引入了大量时间属性参数，因而比传统的排队论模型、实体流图、Petri 网等建模方法更适合于描述复杂离散实时系统。从 20 世纪 80 年代后期开始，任务/资源图建模方法在国外有较多的研究，并获得了迅速发展，然而迄今为止，似乎还没有研究者对其做出严格的定义。因此，对任务/资源图建模方法进行概念性研究具有重要意义。

3.5　基于知识的建模方法[65]

计算机的仿真语言基本上可以分成两大类：一般的计算机程序语言和专门的仿真语言。一般的计算机程序语言在构造仿真模型时具有很大柔性，可以应用于任何特殊领域的仿真，但要求建模者具有很高的编程技巧，而且相当费时，难于维护。专门的仿真语言被构造成面向模块的结构化语言，允许建模者使用预先定义好的几种模块构造仿真模型，因而对建模者的编程

技巧要求不高，但预定义的建模模块限制了建模柔性，使得某些特殊情况下很难实现系统仿真，其应用受到限制。此外，这两大类仿真语言的代码缺少重用性，没有充分利用建模者的知识，不能对仿真模型进行“What－If”或定性分析。同时，许多实际系统都是复杂系统，具有高度复杂性和病态结构，因此这些系统的模型不可能用单一的数学解析方法求解，而将数学解析法和符号推理技术结合起来的系统知识模型则能有效处理系统的复杂性和病态结构。这种知识模型相应地要求其仿真模型也必须能表达系统中实体之间或有关目标之间的数学、逻辑以及符号间的关系。

针对上述问题，1995 年，天津大学董明等学者提出采用面向对象的专家系统开发工具 DEST 14 对离散事件系统进行仿真建模的方法。该方法的思路是：在基于知识的仿真模型的表达、处理和使用方面，人工智能特别是专家系统技术能够提供有效帮助。通过使用 DEST 中的面向对象框架语言，不仅可以对基于知识的仿真模型进行表达、组织和处理，而且可以通过知识库用快速原型法构造仿真模型，这种仿真专家系统还具有良好的人—机界面。下面对该方法进行介绍。

3.5.1　基于知识的离散事件仿真模型

1）面向对象框架语言简介

有效的知识表达是 AI 程序成功的关键，DEST 系统知识表达语言可分为 4 个大部分，分别表达基于框架、规则、过程和面向对象的仿真建模知识。这 4 大部分分别是：框架表达语言、方法表达语言、规则表达语言、事实库非结构化数据变量定义语言。面向对象框架语言将多种单一的表达方法（规则、框架和过程）按照面向对象的程序设计原则组成一种混合知识表达形式，即以对象为中心，将对象的静态属性、动态行为特征和使用处理知识等有关知识（即元知识）“封装”在表达对象的结构中。在对象的知识表达中，可以采用面向对象的框架语言表达对象类和对象，与普通框架不同的是，这种框架一般具有动态属性（如规则槽和方法槽）。面向对象框架语言的巴科斯诺尔范式如下：

Unit：〈框架名〉in〈知识库〉；
{superclasses：〈超类框架名〉{，〈超类框架名〉；}
{subclasses：〈子类框架名〉{，〈子类框架名〉；}　　分类关系定义段
{member：〈成员框架名〉{，〈成员框架名〉；}
{memberof：〈类框架名〉{，〈类框架名〉；}

memberslot|ownslot：〈槽名〉from〈框架名〉；
valueclass：〈槽值类型〉；
inheritance：〈继承属性〉；
cardinality. Min：〈槽值的最少个数〉；
cardinality. Max：〈槽值的最多个数〉　　框架属性槽定义段
{，〈自定义侧面〉：〈侧面值〉；}
Values：〈槽值〉；
END Slot；
END Unit；

2）离散事件仿真模型的面向对象知识表达

离散事件仿真模型通常包含逻辑、数学、符号的关系式，这些关系式用系统的状态、实体及其属性、集合、事件、活动和延迟来描述系统。因此，离散事件仿真模型的结构通常包含以下元素：① 系统状态，它是变量的集合，包含所有在任何时间描述系统的必要信息；② 实体；③ 属性；④ 集合；⑤ 事件；⑥ 活动。

用面向对象框架语言表达仿真模型时，要以对象为中心，将其静态、动态特性以数值、符号等表达方式封装在框架（即对象）数据结构中，并以类层次的形式建造仿真模型知识库。类与GPSS等中的模块不同，具有很大柔性，易修改、使用，通过类的继承性可以方便地应用于不同系统，而模块定义好之后则不能改动。下面简述这种离散事件仿真模型类库的基本组成部分：

（1）事件类：离散事件仿真模型有两种基本事件类型，即到达事件和离开事件，这两种事件类型的算法可用框架中的方法槽来表达。

（2）实体类：包括机器类、运送器类、传送器类等。为了回答“如果……怎样……”（What - If）问题，知识库中应该包括关于设备的各种各样的事实、设备间的相互关系以及行为作用的过程知识。

（3）集合的实现：离散事件仿真中集合如队列适于用特殊的数据结构（如链表、数组等）来实现，而面向对象框架语言允许用户编写的子程序作为内部函数出现在方法体或规则槽的结论中，这是由 DEST 系统中的 def_function()函数实现的，使得面向对象框架语言具有开放性，易于扩展，从而满足用户的需要。离散事件仿真中的一个重要概念是将来事件表（FEL），FEL 是推进仿真时间和保证所有事件准确按序发生的关键。该表是一个特定的集合，它包括了所有安排在将来某些时刻发生的事件。FEL 可用 C 语言编写的双向链表来实现，并将其定义为 DEST 系统的内部函数。

3）仿真模型中各类框架间的通信

框架类间的通信是以消息传递的形式完成的，面向对象框架语言中的消息传递语句可完成此功能，其语法如下：

〈消息传递语句〉::=send(〈消息因子〉{〈消息因子〉}) to 〈目标框架名〉

〈消息因子〉::=〈串〉|〈实数〉|〈整数〉|〈变量标识符〉

〈目标框架名〉::=〈串〉|〈变量标识符〉

例如，当表示机器人将工件从机器人装卸站（Robot Station）搬运到钻床（LD）这一活动时，可用消息语句 send(“transport Robot Station LD”)to“ROBOT”，其中 transport 为关键词，Robot Station 和 LD 为消息匹配模式表中的参数，ROBOT 为接收消息的框架名。以这种方式进行通信有利于实现信息的封装性。

4）仿真策略和推理机

董明博士的方法采用事件调度法，即通过定义事件及每个事件发生对系统状态的改变，按时间顺序确定并执行每个事件发生时有关的逻辑关系。其算法如下：

执行初始化操作

置初始时间和结束时间

事件表初始化，置系统初始时间

实体状态初始化

TNOW＝事件时间

```
WHILE(TNOW<=结束时间)
  Case 事件类型 i
    1：执行第 1 类事件处理程序
    ……
    n ：执行第 n 类事件处理程序
  End Case
取出最早时间的事件记录
置仿真时间 TNOW=事件发生时间
End WHILE
```

此仿真策略隐含地采用正向推理(Forward Chaining)工作方式，即从已知证据出发，利用与事实相匹配规则的执行扩展新事实，这个过程反复进行直至达到给定目标或没有规则匹配为止，因而能够很好地被面向对象框架语言中的规则推理机支持。解释模块与推理机密切相关，它主要是通过推理过程中所记录的导出事实相关性论据来完成的。

3.5.2　知识基离散事件仿真模型的实施

1) 用活动循环图(Activity Cycle Diagrams，简记为 ACD)建立概念模型

从仿真模型的类库到其实施需要有一个形式化过程。一般地，直接由实际系统构造仿真的计算机模型有一定难度，而且容易丢失一些信息。因此，应首先构造系统的概念模型，再由概念模型导出仿真的计算机模型。该方法中采用活动循环图作为形式化手段，由仿真模型类库构造仿真的网络逻辑模型。活动循环图能描述出系统的基本逻辑，特别适合于具有队列结构的系统建模，而且十分容易理解和使用。

活动循环图又叫实体生命周期图(Entity Life Cycle Diagram)，其中实体被分成类，每类实体不必完全相同，但必须有相似的行为模式。换句话说，活动循环图是按照实体类的行为模式建立起来的，它与实体类中的实体数量无关，因而仿真模型类库用 ACD 图进行形式化十分合适。ACD 图对系统描述的原理是认为系统中的每一种实体都按各自的方式循环发生变化，而在这一循环中又只有两种状态——静止状态和活动状态，这两种状态在循环中交替出现。静止状态(或称队列)用圆形来代表，而活动状态用矩形来代表，它们之间的转换用有向弧(箭头)表示。一个活动发生的条件是所有前置队列中都具有按照排队规则挑选的、足够数量的令牌(token)。在 ACD 图中，总是规定实体的行为模式遵循“……→活动→队列→活动→……”的交替变化规则，即活动循环。另外，ACD 图可以根据问题需要，对所分析系统建立不同层次模型，高层次模型可进一步分解为低层次模型。

2) 知识基仿真模型的建立过程

知识基仿真模型的建立过程可用图 3.30 表示。

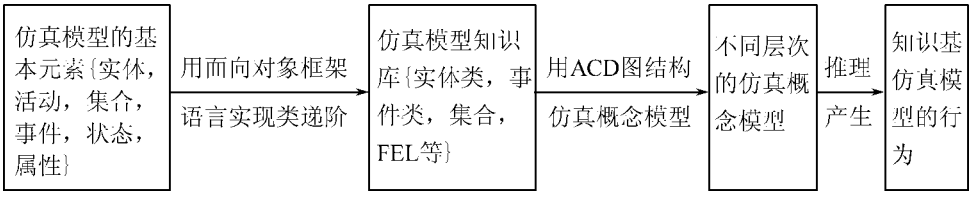


图 3.30　知识基仿真建模过程

3.5.3 应用举例

下面用一个 FMS 系统的仿真来说明上述知识基仿真模型的创建过程。该 FMS 的布局如图 3.31 所示，零件首先由 U/L 装卸站进入系统；然后在车床(LHL)上完成第一道工序；接着进入机器人装卸站；由铣床(SHM)完成第二道工序；再由机器人将其由铣床搬运到钻床(LD)上完成第三道工序；最后从 U/L 装卸站离开系统。

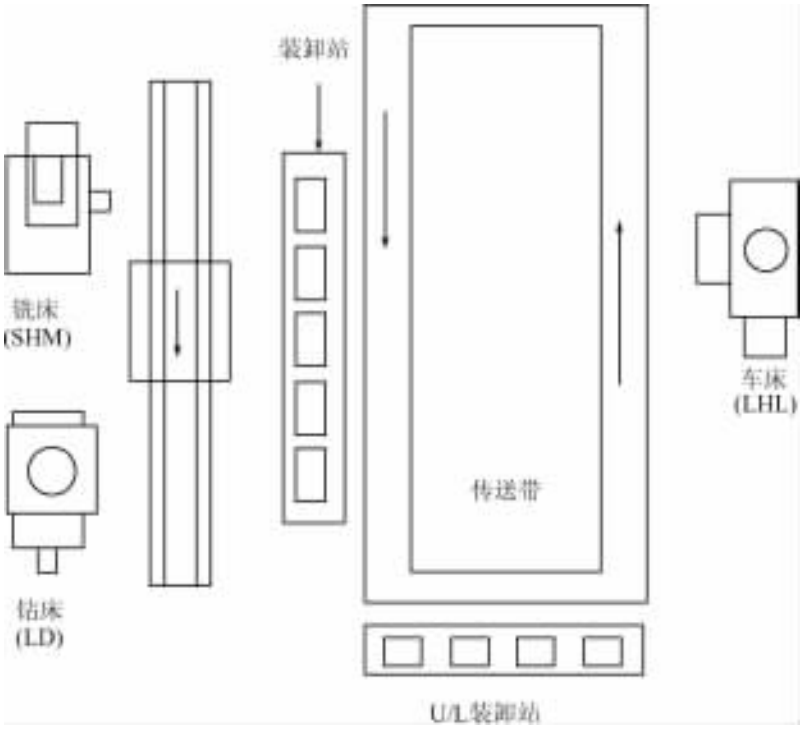


图 3.31 FMS 的布局

根据上述知识基仿真建模过程，用面向对象的框架语言构造该 FMS 系统的知识库(包括实体类、事件类、集合等)。下面为一个机器实体类的框架实现。

```
unit: machine in SIMULATE。KBS;/ * 机器类框架名 * /
memberslot: name from machine;/ * 机器名 * /
……
memberslot: status from machine;/ * 机器状态 * /
……
memberslot: INPUT_length_of_queue from machine;/ * 机器输入队长 * /
……
memberslot: load_rule from machine;/ * 机器装载或零件选取规则 * /
inheritance: override;
valueclass: rules;
values:
{rule1
```

```
fact_frame. name="LHL";/ * LHL 为机器名 * /
and_frame. status="FREE";
and_frame. INPUT_length_of_queue>0
and_frame. part_select_rule="SPT";
/ * SPT 表示处理时间最短调度规则,即 Shortest Processing Time * /
then _frame. status="BUSY";
      _frame. part_type_machined"part_type_4";
      _frame. INPUT_length_of_queue=_frame. INPUT_length_of_queue-1;
      _frame. TotalServiceTime=_frame. TotalServiceTime+part_type
      _4. ServiceTime;
      ……}
end slot;
end unit
```

根据该 FMS 的逻辑结构可构造其活动循环图,如图 3.32 所示,图中不同线型表示不同实

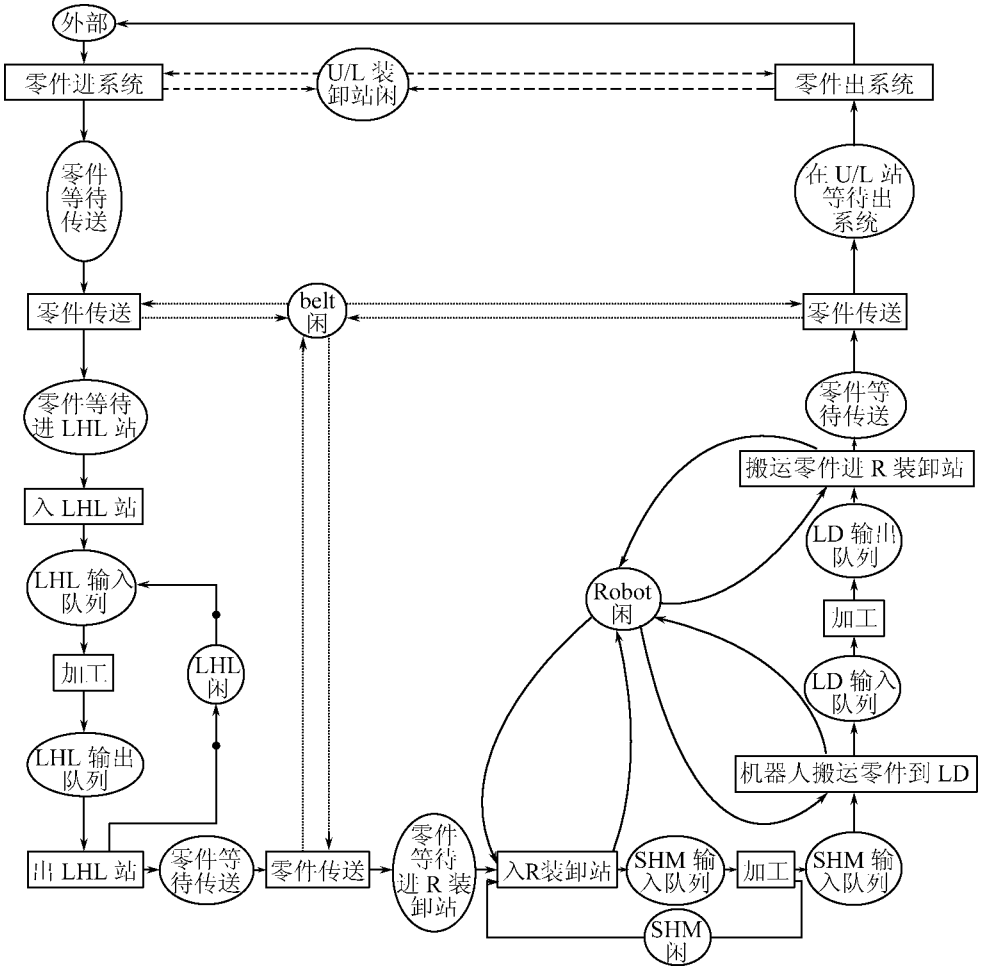


图 3.32　FMS 的活动循环图

体。其中实线表示系统的总循环活动流程，虚线表示零件进出系统与装卸站循环活动过程，点线表示传送带与零件传送间的循环活动。进一步可由该活动循环图建立其仿真计算机模型，通过 DEST 推理机能够产生仿真模型的行为，从而对该 FMS 的定性和定量指标进行分析，如找出该系统中的瓶颈机床，分析该系统的阻塞情况等。

3.6 基于系统理论形式化的建模方法[66]

传统的仿真观点把重点放在实验方面，强调如何获得系统中有关变量对时间的响应轨迹。因而，建模时强调如何编写产生模型行为的计算机程序，即用程序表示仿真模型。当然，在计算机仿真领域，不容易把程序和模型分开。但是，如果不作这种区分的话，在进行仿真研究时，就不能确定仿真结果和实际行为之间的差异是由于程序的错误引起的，还是由于错误地理解了实际系统的工作机理造成的。特别是在大型复杂系统的仿真研究中，如果实际系统的内部结构和工作机理仅隐含在程序中，则很难进行模型的确认和验证。

过去二十多年，人们在仿真建模的形式化方面进行了一些研究工作。目的是寻找独立于程序语言的模型描述方法。有些学者建议对离散事件模型要有一个标准化的模型表达机制，这种机制必须便于人们理解在各个抽象层次上的模型表达，从适合于决策应用的最高层到适合于程序实现的最底层，用这种方法描述的模型应适合于分析确定该模型采用哪一种仿真策略(下次事件、活动描述、过程交互)能达到最有效的仿真。近十年来，围绕仿真建模，利用系统理论、数学理论和控制论的观点和方法展开了全面深入的研究，提出了一些很有前途的仿真建模和形式化方法。这些方法对认识实际系统的定性/定量行为，模型的有效性检验、开发新的仿真语言、提高仿真语言效率，以及比较现有仿真语言的特点都是非常重要的。

下面，首先回顾仿真建模的研究现状和建模形式化方面所进行的一些研究工作；然后介绍把系统描述为集合结构的形式化方法，讨论系统描述的层次结构和离散时间仿真的数学形式体系。

3.6.1 建模形式化研究回顾

在过去几十年中，对仿真和建模方法学的研究极大地改变和影响着今天进行仿真研究的方式，传统仿真技术的能力在许多方面都得到了发展，由于功能强大的程序设计语言和高性能工作站的出现，特别是个人计算机的普及以及它的图形/图像显示能力、方便的人机交互能力，使仿真技术得到了更为普遍的应用。

然而，与连续系统相比，对离散系统本质的认识还处于发展和完善阶段。用微分方程描述连续系统已有相当长的历史，在计算机出现以前，描述连续系统的数学形式化体系就已得到相当充分的研究，形成了比较完备的理论体系。而离散事件仿真则是随着计算机的发展而发展起来的，描述离散时间系统的数学形式体系则是后来才提出来的，这与描述连续系统的微分方程理论是不同的。

现在人们越来越认识到基于系统和数学理论的模型描述形式化可以极大地增强对复杂系统的认识，从而构造出有效的系统模型。目前，两个研究发展趋势会对仿真建模领域的发展起到重要作用。一个趋势是对仿真建模概念理论的研究，以及支持这种理论的软件工具的实际应用；另一个趋势是出现了基于理论框架的仿真语言，虽然这些语言尚未达到商业化程度，但

也能足以说明基于理论的方法的功能和表达能力。

过去十多年，许多专家学者在这一领域进行了广泛深入的研究，提出了许多方法学和形式体系。下面简要回顾把系统理论、系统设计和仿真建模联系起来的比较典型的，而且很有发展希望的几个方面的工作。

最早把这些学科统一起来进行研究的是 20 世纪 70 年代初期 Ören 的工作[67]。他设计了一种通用系统理论实现语言(GEST)，这是一种用于构造模块化层次模型的语言，可以表示连续和离散仿真形式的描述。改进后的版本 GEST81 和它的一个高层开发外壳 MAGEST 提供了描述层次组合模型的方法。

Klir 提出了一个用于归纳建模的通用系统问题求解器(GSPS)。GSPS 概念框架的核心是一个称为认识论层次的系统描述层次结构，这个层次结构形成了对系统和系统问题进行最基本的分类基础。系统的描述从单纯的观察层上升到行为层，然后再到结构层。Klir 的认识论层次和 Zeigler 的系统描述层次有些相似，不同的是 Klir 的方法用于归纳建模，而 Zeigler 则主要用于仿真建模。关于 GSPS 的理论，将在第 4 章进一步介绍。

Fishwick 对仿真建模的过程抽象进行了分类，提出了各种有效的抽象方法的形式化定义，并阐述了抽象层次之间进行转换的基本概念。他还开发出一种计算机仿真语言，为研究过程抽象提供了一个实验基础。

Pichler 进行了一项新的实现系统概念的研究。这项研究称为计算机辅助系统理论(CAST)，其目的是为计算机辅助问题求解提供方法库。CAST 来源于 Pichler 的早期工作，即基于系统理论的问题求解(STIPS)。STIPS 已在一种面向对象的环境中实现了有限状态机。

Zeigler 提出了 DEVS，他用系统理论的方法来描述离散事件系统的数学结构。DEVS 可用于描述和开发离散事件模型和仿真语言，并在几种面向对象的环境中得到了实现。DEVS 具有严密的理论基础，Zeigler 曾把 DEVS 称为“离散事件系统的微分方程”。

以上讨论的许多工作都从不同的角度对仿真建模进行了系统的研究，概括起来主要包括两个方面：一方面是利用系统理论的概念和原理统一各种建模；另一方面是为仿真提供概念理论框架。在研究过程中 Zeigler 在这两个方面的工作最为出色，下面围绕 Zeigler 提出的概念框架来讨论仿真建模的形式化问题。

3.6.2　系统建模的形式化描述[68, 69]

1) 系统的分解与聚合

在仿真领域，把系统理解为实现世界分割出的一部分。因此，任何一个系统都有一定的边界，通过边界，系统与外部环境发生一定的联系和相互作用。系统是一个复杂的有机整体，它往往有许多组成部分，这些组成部分称为子系统。这些子系统虽然具有一定的相对独立性，但是，它们可以根据逻辑统一的要求相互连接从而形成一个有机的整体。因此，研究复杂系统的时候，一般采用两种手段：一是分解；二是聚合。分解是把一个系统整体分解成许多子系统，每个子系统又可以进一步分解成更小的子系统，这个过程隐含着递归分解的概念，分解的结果可以得到一个系统的多层次结构。聚合则是分解的逆过程，也就是把某个层次上的若干子系统相互连接形成一个较大的系统，这个过程隐含着重构的概念。因此，任何一个系统(子系统)都是一个大系统有机组成部分，每个系统都提供了一个界面，通过界面和其他系统相互作用。要使系统能进行有效的分解与聚合就必须精确地描述系统的内部结构，正是系统的内部结构决

定了系统边界处的相互作用。

模型和系统之间最重要的关系就是抽象，控制抽象是描述和解决实际问题最有效的方法。模型是对系统的抽象描述，它集中反映了我们所关心的系统某些本质方面的特征。因此，模型不是系统的复现，而是根据所研究的问题和目的，便于对实际系统进行研究的一个“替身”，通过对模型的研究来推断实际系统。

为了理解模型如何表现系统的内部结构，递归分解性和界面处的相互作用，就需要更为精确的数学形式来描述模型的一般概念。

2）系统的形式化描述

一个系统可以描述为一个集合结构：

$$S = \langle T, X, \Omega, Q, Y, \delta, \lambda \rangle \tag{3-75}$$

式中：T 为时基；X 为输入值集合；Ω 为输入段集合；Q 为内部状态集合；Y 为输出值集合；δ 为状态转移函数；λ 为输出函数。

时基 T，是表示时钟时间和为事件排序的集合，通常取 T 为整数 Z 或实数 R，S 相应地称之为离散或连续时间系统。

输入值 X，表示界面的一部分，外部环境通过它作用于系统。通常取 X 为 $\mathbf{R}^n$，$n \in \boldsymbol{I}^+$，即表示 n 个实值的输入变量；或选取 X 为 $X^{\varnothing}$，$X^{\varnothing} = X \cup \{\varnothing\}$，其中，$X$ 表示外部事件结合，$\varnothing$ 表示空事件。

输入段集合 Ω，输入段描述了在某时间区间内对系统的输入模式，一个输入段是一个映射 $\omega: \langle t_i, t_f \rangle \to X$，其中 $\langle t_i, t_f \rangle$ 表示从初始时刻 t_i 到终止时刻 t_f 的时间区间，$\omega \in \Omega$。所有这些的输入段构成的集合记为 (X, T)，输入段集合 Ω 是 (X, T) 的一个子集，通常取 Ω 为分段连续集，这时 $T = \mathbf{R}, X = \mathbf{R}$。或取 Ω 为离散事件段集，这时 $T = \mathbf{R}$，而且映射 $\omega: \langle t_i, t_f \rangle \to X^{\varnothing}$ 使得除有限的事件集合 $\{t_1, \cdots, t_n\} \subseteq \langle t_i, t_f \rangle$ 以外，均使 $\omega(t) = \varnothing$，$t \notin \{t_1, \cdots, t_n\}$。当 $T = \boldsymbol{I}$ 时，Ω 实际上是一个有限序列集。

内部状态集合 Q，Q 表示系统的记忆，它影响系统现在和将来的响应。内部状态集是前面提到的内部结构建模的核心。

状态转移函数 δ，状态转移函数是一个映射 $\delta: Q \times \Omega \to Q$。它的含义是，若系统 t_i 在时刻的状态是 q，这时施加一个输入段 $\omega: \langle t_i, t_f \rangle \to X$，那么 $\delta(q, \omega)$ 表示系统在 t_f 时刻的状态。为使状态结合概括以前所有有关的信息，就必须要求转移函数满足半群关系，即对每个 $q \in Q, \omega \in \Omega$ 和 $t \in \langle t_i, t_f \rangle$ 都有，即

$$\delta(q, \omega) = \delta(\delta(q, \omega_{t>}), \omega_{t<}) \tag{3-76}$$

式中：$\omega_{t>} = \omega \mid \langle t_i, t \rangle$，从 t_i 到 t 的部分；$\omega_{t<} = \omega \mid \langle t, t_f \rangle$，从 t 到 t_f 的部分。

半群关系可用图 3.33 表示。

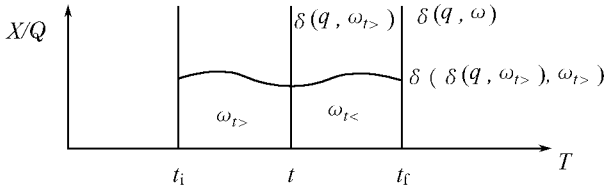


图 3.33　半群关系

输出集合 Y，Y 也表示界面的一部分，系统通过它影响环境。除方向不同外，输出集合的含

义与输入集合完全相同。如果该系统是一个大系统的子系统，则该系统的输入/输出是另外系统的输出/输入。

输出函数λ，输出函数是一个映射$\lambda: Q \to Y$，即当系统的状态是q时，输出值是$\lambda(q)$。因此，把系统内的内部状态与其环境的影响联系起来。换句话说，当系统的状态是q时，$\lambda(q)$可以由观察者测定。

3）系统行为

系统行为是其内部结构的外部表现形式，即系统在叉积$(X,T)\times(Y,T)$上的关系。这个关系的含义是，如果系统在某时间间隔的输入是$\omega \in (X,T)$，则在相同的区间内系统的输出是$\rho \in (Y,T)$，这样就可以得到一个输入/输出对(ω,ρ)，所有这些输入/输出对组成的集合就称为一个输入/输出关系。

这个关系可表述如下：

对每个状态$(q \in Q)$，和Ω中的输入段$\omega: \langle t_i, t_f \rangle \to X$，都存在一个唯一的状态轨迹$\mathrm{STRAJ}_{q,\omega}$，它是一个映射：

$$\mathrm{STRAJ}_{q,\omega}: \langle t_i, t_f \rangle \to Q \tag{3-77}$$

并且

$$\mathrm{STRAJ}_{q,\omega}(t_i) = q$$

$$\mathrm{STRAJ}_{q,\omega}(t) = \delta(q, \omega_{t>}),\ t \in \langle t_i, t_f \rangle \tag{3-78}$$

这个状态轨迹可在仿真运行过程中计算出来。这个轨迹的可观测结果是与$q \in Q$和$\omega \in Q$有关的输出轨迹$\mathrm{OTRAJ}_{q,\omega}$，它也是一个映射：

$$\mathrm{OTRAJ}_{q,\omega}: \langle t_i, t_f \rangle \to Y \tag{3-79}$$

并且

$$\mathrm{OTRAJ}_{q,\omega}(t) = \lambda(\mathrm{STRAJ}_{q,\omega}(t)) \tag{3-80}$$

于是，系统的行为就可以通过它的输入输出的关系来表示：

$$R_\varphi = \{(\omega,\rho) \mid \omega \in \Omega, \rho = \mathrm{OTRAJ}_{q,\omega}, \text{对某个 } q \in Q\} \tag{3-81}$$

以上叙述了系统模型形式化定义的含义。模型的构造过程包括系统静态结构和动态结构的描述。输入集合X、状态集合Q、输出集合Y和输出函数λ定义了系统静态结构；时基T、输入段集Ω和状态转移函数δ构成系统的动态结构。

上面描述的是模型的一般形式化，在实际仿真建模时，模型根据所研究的问题用某种特殊的建模形式化来描述。典型的形式化包括微分方程、有限状态机和离散事件模型。每一种形式化的模型描述都确定了一个系统，对该系统可能包括的静态和动态结构施加某种约束，就能确定该系统的分解。

3.6.3　系统描述的层次

上一节描述了系统界面、内部结构和递归分解性的概念，从而构成了一个系统的层次结构。根据实际研究的目的，还可把某一个层次上的系统再进行分解，同时也可能需要某一层次上的系统聚合在一起形成一个更抽象的系统，当然，分解和聚合的过程不是随心所欲的，必须满足一定的约束，正是系统的界面和内部结构约束了系统的分解和聚合，因此，就必须详细地描述每一层次上系统的界面和内部结构。

系统描述是从抽象的黑箱概念开始，也就是把系统看成一个黑箱，对它施加输入，然后对

它的输出进行测量和记录，这样就可以产生系统的输入输出行为，但它不能反映出系统内部的信息。因此，随着系统描述层次的增加，逐步给系统的内部结构增加细节，这样就可以了解更为详细的系统内部信息。

Zeigler 提出了系统描述的 5 个层次[70]：

(1) 输入输出关系观测。这一层次上的描述是系统输入输出行为可观测的记录，它给出了系统的 I/O 关系。由于对实际系统的实验是处于行为层次上，因此，这一层次上的描述是非常重要的。这实际上就是传统黑箱的例子。

(2) 输入输出函数观测。描述把系统的输入输出关系进行分类的输入输出函数集合。对每个给定的输入输出关系的输入函数恰好存在一个输出函数。

(3) 输入输出系统。描述一个输入输出系统，除了输出集合，还要定义状态集合、状态转移函数和输出函数。状态转移函数描述由输入段引起的状态到状态的转移，输出函数描述从状态到可观测输出的映射，这一层次上的描述是上一节讨论的系统描述的基础。

(4) 结构化系统描述。这一层次上的描述和第 3 层次上的描述相同，但集合和函数都是结构化的，也就是说把系统表示为更具体的基本集合和函数的叉积。

(5) 系统描述网络(耦合系统)。把系统描述为由许多分系统相互连接在一起构成的。为此，要确定一组成分系统和一种耦合机制，这一描述是以层次形式构造模型的基础。

上述系统描述的层次与任何特定的建模形式化无关，也就是说在任何一个层次上，可以利用任一种形式体系(如微分方程、离散时间系统或离散事件)来描述系统。Zeigler 讨论了几种主要建模形式体系，以及向一般系统描述的转换过程。

3.6.4 离散事件仿真的形式理论

DEVS 是第一个描述离散事件仿真的形式理论，它刻画了离散事件仿真语言如何描述离散事件系统的本质特征。DEVS 是 Zeigler 在 20 世纪 70 年代根据系统理论概念提出的离散事件系统的形式理论。这个形式理论已作为模型和仿真程序开发、高级仿真语言设计和仿真程序验证的形式基础。

DEVS 用一个集合结构描述离散事件系统

$$M = \langle X, Y, S, \delta, \lambda, t_a \rangle \tag{3-82}$$

式中：X 为外部事件集合；Y 为外部输出集合；S 为序贯状态集合；δ 为状态转移描述函数；λ 为输出函数；t_a 为时间推进函数。

这个形式结构是通过对前面讨论的一般建模形式施加某些约束而得到，约束为：

(1) M 描述的系统的总状态集合为

$$Q = \{(s, e) \mid s \in S, 0 \leqslant e \leqslant t_a(s)\} \tag{3-83}$$

(s, e) 是总状态，s 是序贯状态，e 是在状态 s 停留的时间。

(2) 状态转移描述函数

① 内部转移函数 δ_ϕ，

$$\delta_\phi : s \to S \tag{3-84}$$

如果没有外部事件到达，系统经过 $t_a(s)$ 时间单位后，从序贯状态 s 转移到 $\delta_\phi(s)$，同时停留时间 e 置为 0。

② 外部转移函数 δ_{ex}，

$$\delta_{ex}: Q \times X \rightarrow S \tag{3-85}$$

如果有一个外部事件 $x \in X$ 到达，系统在状态 s 已停留的时间为 e，则它立即转移到 $\delta_{ex}(s, e, x)$，同时停留时间 e 置为 0。

③ t_a 是从 S 到非负实数的映射，

$$t_a: S \rightarrow \mathbf{R}_{0,\infty}^{+} \tag{3-86}$$

t_a 可解释为如果没有外部事件到达，系统状态 $s \in S$ 停留的时间。

④ λ 是从 Q 到 Y 的映射，

$$\lambda: Q \rightarrow Y \tag{3-87}$$

基于 DEVS 形式理论，Zeigler 进一步提出了层次模块化 DEVS 模型和抽象仿真器的概念，并实现了一个基于 DEVS 形式理论、具有充分理论根据的支持层次、模块化离散事件模型构造和仿真的环境 DEVS-Scheme[69]。

模型形式化描述的层次结构为开发有效地表达模型的仿真系统提供了充分的理论基础。这个领域还有很多学者在默默的耕耘，不久的将来基于理论的仿真方法学将出现更先进的仿真系统。