



智能运输系统:智能化绿色结构设计(原书第2版)

Intelligent Transportation Systems:Smart and Green Infrastructure Design (Second Edition)

Sumit Ghosh (美) Tony S.Lee 著

胡郁葱 译









关于本书

"这本书的优点就在于它包含了对ITS及其在运输中的作用和地位的创新性认识和先进的理念。作者对其独特的见解进行了非常充分地论证,并在论证"怎样改变认识"和"从改变认识的过程中得到什么"这类问题时,提供了很多实际的案例。……运营者和计算机仿真专家们无论是在技术上,还是在理念上,都能从本书中获益。"

—Dr. Tim Lomax 德克萨斯运输研究所

"·····是对运输以及ITS有关文献的一个极好的补充。作者非常清楚地了解到运输以及ITS中存在的问题,并对这个问题进行了很好地归纳和总结。"

—Prof. Chelsea C. White, III 乔治亚理工学院

"……是一个在铁路和车辆路径仿 真和模拟方面非常了不起的原创工作, 对未来的研究非常有意义。提出的基本 软件架构在未来,可能会被作者以及其 他研究者进一步加强和完善,这是极有 教育意义的。特别要指出的是,这里提 出的DTMC的概念也是极好的……"

> —Raj Ghaman 美国交通部ITS联合项目办公室前任经理



智能运输系统:智能化 绿色结构设计 (原书第2版)

(美) Sumit Ghosh Tony S. Lee 著

胡郁葱 译



机械工业出版社

本书是作者在第1版的基础上,充分结合近10年来全世界在智能运输领域取得的最新研究及实践成果修订而成的。本书共分为10章,全面系统地总结了智能运输系统(ITS)的发展现状和存在的问题,阐述了近10年中发掘出的新的元层原理,对ITS的基本特征进行了归纳,提出铁路网络决策算法DARYN和RYNSORD,介绍了智能车路系统的一种新型体系架构DICAF,并对运输过程中出现干扰时上述算法的稳定性进行了研究。此外,本书还研究了ITS设计中的建模和仿真技术,介绍了所附带的CD-ROM中RYNSORD仿真器的简单使用方法。

本书可供运输工程和计算机科学的研究生,与运输相关的各学科研究 人员,铁路、高速公路和航空从业人员,运输系统政策制定者等使用。

Intelligent Transportation Systems; Smart and Green Infrastructure Design (Second Edition)/by Sumit Ghosh, Tony S. Lee /ISBN; 978-1-4398-3518-0

Authorized translation from English language edition published by CRC Press, part of Taylor & Francis Group LLC; All rights reserved.

China machine Press is authorized to publish and distribute exclusively the Chinese (Simplified Characters) language edition. This edition is authorized for sale throughout Mainland of China. No part of the publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

本书中文简体翻译版授权由机械工业出版社独家出版并在中国大陆地区销售,未经出版者书面许可,不得以任何方式复制或发行本书的任何部分。

本书版权登记号: 图字 01-2010-6643 号。

图书在版编目 (CIP) 数据

智能运输系统:智能化绿色结构设计(原书第2版)/(美) 戈什(Ghosh, S.),(美)托尼(Tony, S.L.)著;胡郁葱译.—北京:机械工业出版社,2012.4

(国际电气工程先进技术译丛)

书名原文: Intelligent Transportation Systems: Smart and Green Infrastructure Design, Second Edition

ISBN 978-7-111-37676-7

I.①智··· Ⅱ.①戈···②托···③胡··· Ⅲ.①公路运输 - 交通运输管理 - 自动化系统 - 设计 Ⅳ.①U491 - 39

中国版本图书馆 CIP 数据核字(2012) 第 040481 号

机械工业出版社(北京市百万庄大街22号 邮政编码100037)

策划编辑: 刘星宁 责任编辑: 刘星宁

版式设计: 霍永明 责任校对: 张 薇

封面设计:马精明 责任印制. 乔 宇

三河市国英印务有限公司印刷

2012 年 5 月第 1 版第 1 次印刷

169 mm × 239 mm · 11.75 印张 · 233 千字

0001 —3000册

标准书号: ISBN 978-7-111-37676-7

ISBN 978-7-89433-443-5 (光盘)

定价:58.00 元 (含1CD)

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

电话服务 网络服务

社服务中心: (010) 88361066

L 版 第 中 心: (010) 88301000 门户网: http://www.cmpbook.com

销售一部: (010) 68326294

销售二部: (010) 88379649 教材网: http://www.cmpedu.com

读者购书热线: (010) 88379203 封面无防伪标均为盗版

译者序

随着社会经济的不断发展,人们对交通运输的需求不断增加。由于道路资源有限,人们更希望能通过提高现有道路的利用率,来提高道路交通的安全性和道路使用的舒适性,这种情况下,智能运输系统(ITS)应运而生。ITS是一门同时涉及科学性和工程性的学科,其主要目标是在行驶速度不断提高,出行者数量不断增大,出行者迫切需要更精确、及时的信息的情况下,通过现有资源的公平分配,减少所有出行者和商品的总运输时间,同时确保出行的安全。

本书主要侧重于研究 ITS 如何才能为出行提供更及时的信息和更高质量的引导,也就是在现有基础设施范围内如何改善运输的质量。贯穿本书的一个中心思想是:随着运输模式日益多样化,运输规模的日益壮大,运输系统的体系结构必然要由集中式模式向异步分布式模式转化,因为这种异步分布式模式能在当今计算机技术高度发达的信息化社会,通过新型的计算机算法将快速计算机和高性能计算机网络有机地结合起来,极大地提高 ITS 的服务效率和稳定性。

本书作者 Sumit Ghosh 和 Tony S. Lee 多年来致力于电气工程和计算机等领域的研究工作,成果显著。Sumit Ghosh 曾先后在贝尔实验室、Fairchild 电气先进技术研究和发展实验室及 Silvar-Lisco 公司进行过研究工作,目前任教于德克萨斯大学泰勒分校。他于 2004 年获得 IEEE 计算机学会技术成就奖, 2008 年任美国设计与流程科学学会 (SDPS) 主席,已是 5 本书的第一作者。Tony S. Lee 曾是加利福尼亚州远创科技数据转换公司的资深高级工程师,还曾是美国国家航空航天局(NASA) 埃姆斯研究中心的一名研究员。Tony 目前是雅虎公司的科学带头人,其研究领域包括运输系统、银行系统和军事指挥与控制等。他也是高保真分布式网络模拟器 ATMSIM 软件包的发明者之一,该软件包能帮助网络研究人员为新的协议、算法以及识别复杂的交互行为而开发现实模型。

本书作者通过对ITS 的发展现状和存在问题的总结,归纳出一些新的适用面广泛的元层原理,并据此提出适用于各类运输系统发展的基本原则。本书以高速公路系统和铁路系统为例,与传统的集中式控制体系相对比,对分布式控制体系进行了研究,分别提出铁路网络的分布式决策算法 DARYN,以及更为精确的带有软预留的算法 RYNSORD,并且研究了 RYNSORD 在干扰下的稳定性。本书还针对运输问题的复杂性,提出一种智能车路系统的分布式、可扩展体系结构 DICAF。最后,本书专门针对 ITS 设计中重要的建模和仿真技术进行了阐述。本

书附带了 CD-ROM, 读者可以根据本书最后一章内容对 RYNSORD 仿真器进行简单的实验。

本书部分章节的初译得到研究生商慧丽、凌美宁、张琦的大力协助,在此表示感谢,陈海伟、向曾哲协助校对了部分初稿,全书由胡郁葱统稿审校。

由于译者水平有限, 书中难免有疏漏或错误, 敬请广大读者批评指正。

第2版前言

自1999年出版《智能运输系统:新的学科及其架构》以来,已经过去了10多年。在那以后,全世界交通运输领域中产生了许多重要的新成果,但同时,很多地方也出现了退步的情况。几乎所有人都承认,智能运输系统(ITS)将会有更好的发展前景,但还需要我们努力去实现。拥堵、不确定性以及未来交通发展方向的不明确等问题仍然存在。目前对ITS的发展形成阻碍的两个关键因素包括:缺乏大量有相应知识背景、受过相关培训的ITS工程技术人才;ITS研究和发展的推动力不足。本书希望能提醒美国政府,不仅要重视直接面向未来运输从业人员的ITS教育发展计划,还应注重提高当前运输从业人员的创新性和专业能力。这两方面都会给我们带来巨大的挑战,事实上,在21世纪,我们几乎在所有领域都已经开始遇到这种挑战,这也是学科间、跨学科或者交叉学科的研究和教育密切联系的必然结果。

在过去10多年中,我们通过在ITS 及其相关领域中的各种活动和努力,得到了许多经验,也获得了一些惊人的发现和独特的见解,还归纳总结出了某些元层原理,这些都包含在我们这个修订版中。我们曾经强调智能决策和效率的重要性,并将其作为ITS的标志性特征,但在这一版本中,我们将重点强调智能和创造力的分形特征,这一特征将成为我们巨大的灵感源泉,也许能帮助我们以前所未有的深度去分析最复杂的问题,为几乎不可能解决的问题找到最巧妙的答案。经过适当的培育,这种特性能为持续不断地实现量的增长提供实际可行的途径。我们还观察到ITS的效率属性和地球上到处可见的"绿色"生活是类似的,因为ITS和"绿色"生活的目标都是要利用最少的能源获得最大的有序度,即熵或无序的对立面。我们的研究还发现效率在本质上是等同于分形的。

我们要感谢 Frank Kreith 博士,他同时也是本书的编委和全国州立法委员会成员。他在 2009 年 4 月下旬打电话给我,并建议我们重新修订此书,当时我非常惊讶并且激动,这说明和 1999 年相比,如今人们对 ITS 的发展产生了更为浓厚的兴趣。有一个现象可以解释为什么会有这种不断增长的兴趣产生,那就是美国国会根据 "2009 美国回收再投资法案",在修路、修桥以及其他交通基础设施上已经投入了几十亿美元,而交通运输问题却没有得到本质的改善。除此之外,还有一个更深层次的不可见原因:美国是一个有着丰富的创新和改革经验,并充满商业化思想的国家。更重要的是,这个国家曾经以惊人的效率和改革精神迅速摆脱了经济萧条时期。从 1930 年到 1950 年,雷达、核能、州际高速公路、晶体管以及计算机的发展

都极大地改变了美国乃至于全世界的经济局面。到了2010年,学科范围几乎不受限制的ITS已经准备加入到这个行列中来。

在这一版中,我们增加了两个新的章节。第1章重点回顾了ITS 领域中重要的理论发展和工程实施情况,并指出ITS 将要面临的主要挑战。第2章分为三部分,第一部分描述了元层的原理和思想,第二部分解释了如何将元层原理和思想应用到实际的ITS 问题中,第三部分指出了ITS 必须要面对的一些挑战,当我们可以利用元层和ITS 原理解决这些问题时,就意味着我们在技术和社会利益上都将会发生一个量的突变。我们希望全美国的市、县、州,还有印度、中国以及其他国家都能参考本书中的观点,并能认真考虑将其应用于现有和规划的道路、高速公路、桥梁以及其他基础设施工程项目中,使这些工程的质量得以提高,运行寿命也能得以延长。ITS 的基本特征和原理都在这一版本的第3章中做了进一步的阐述。

作为一个产业,ITS 为私人和公共部门的合作提供了巨大的潜在发展空间。然而同样因为作为产业,ITS 还需要进行更彻底的研究。ITS 涉及的范围是非常宽泛的,它涉及技术、运营、政策、个人隐私、标准、商业、财政、政治等各个层面,以及如何与其他产业合作,以促进其发展并创造财富等。本书主要讨论的是 ITS 技术和运营上的问题,其次也涉及一些 ITS 政策方面的问题。

我们非常感谢德州农工大学德克萨斯运输研究所的 Tim Lomax 博士和乔治亚理工学院的 Chelsea White 教授,他们对本书进行了公正的审阅,并提供了有深刻见解和有建设性的建议。

我们还要深深感谢美国运输部 (US. Dot) ITS 联合项目办公室前任经理 Raj Ghaman 先生。

最后,我们也要感谢 Joe Clements 及全体编委,感谢 CRC 出版社的全体职员为我们提供的一贯而热情的支持。

Sumit Ghosh Tony S. Lee

第1版前言

运输的概念是指人和货物经过一定地理距离的移动,运输拥有与人类文化同样久远的历史。运输的范围是巨大的,从人们在地球表面的步行到手推车的出现,再到畜力车、小汽车、火车、飞机和轮船。尽管运输模式是多样化的,但它们都有一个共同的特性——在其整个发展历程中,不断通过为其组成部分提供信息和指引来提高效率。过去,信息是由人和某种材料(纸等)以消息的形式携带并传递的,所以,信息的传送速度与运输的模式有很大关系。随着电磁通信技术的出现,在20世纪,运输的概念发生了巨大的转变。提供信息和指引的速度要比人和货物实际移动的速度快得多,这就意味着整个运输系统出现了质的提升。于是,在铁路系统中,站A的管理员在火车刚刚通过该站时,就立刻连线告知站B的管理员火车即将到达(站B)的信息,这已成为一个标准的操作过程。出于简化结构或保持控制的一致性需要等种种原因,人们通常会利用一个集中式单元来为运输系统提供信息和指引。在一个特定的运输系统中,由一个中央控制系统收集系统中所有实体的信息,在必要时,也向它们提供信息和指引。而对于孤立的,成员数量很少的小系统,通常也会采用非集中控制。可是,当系统由于联合等原因不断扩展时,还是会转而使用集中控制。

从20世纪末起,许多运输系统,特别是高速公路系统,包含的个体单元数量一直在飞速增长。仅仅在美国,车辆数已超过20亿,铺装道路里程超过4mile (1mile=1609.344m)。此外,在高峰时间段,许多运输系统的局部使用率已经超过了其容量限制,导致了交通阻塞以及其他相关问题的出现。此外,当今社会中的每个人都希望能有更多的自由选择和灵活性。但是,对许多运输系统来说,扩展现有基础设施的成本过于昂贵,这就导致了人们将主要发展战略从修建更多的基础设施转移到提供更及时的信息和更高质量的指引上来,也就是想在现有基础设施范围内改善运输的质量。当然,在经过深思熟虑,并利用模拟和仿真工具进行了科学论证之后,某些被认为是绝对必需的基础设施,还是应该继续建设的。基于上述原因,集中式控制模式将要面临几个挑战:实体绝对数量的增长、对更快获取准确信息的需求和对获取高质量指引的需求。

要满足未来这些需求,2000年后,运输系统必然要经历另一种根本性的转变,即集中式模式转向异步分布式模式,这种异步分布式模式将通过新型计算机算法将快速计算机和高性能计算机网络有机地结合起来。本书将重点研究未来运输系统的基本原则,提出新的系统架构。本书利用这些基本原则,经过科学的论证,阐述了

在运输系统中创新和有创造性的设计方法的重要性和必要性。本书还强调了对某些典型系统进行计算机模拟和仿真研究的必要性,这可以帮助我们完成一些对复杂的、大型系统的设计和论证工作,也可以帮助我们设计出一些新的性能指标,以评价这些系统的表现。尽管运输系统的范围非常广泛,但本书中的重点还是在两个应用最为普遍的系统上,即高速公路系统和铁路系统。书中介绍了在进行与出行相关的决策时要用到的基本原则,这些决策包括协调、控制和路径计算等。这些基本原则构成了运输系统的核心,也适用于整个运输系统,包括航空客运、航空货运、个性化快速公交系统等。这些原则是引领运输系统进入下个世纪的核心成果之一,因此本书中就不再讨论与此有关的其他问题,如驾驶员行为分析、人的因素、人的行为引起的拥挤和其他交通因素等。

本书的撰写受到两个动机的推动。第一个是我自己在使用当前的运输系统时经常会遇到问题,并且相信我们已经拥有了解决这些问题所需要的技术。和成千上万的人们一样,我们遇到的问题包括交通拥堵、在航空旅行过程中丢失行李、在火车站或汽车站得到错误的到达信息、夜间行驶在不熟悉的城市而迷路等。通用电气化机车建设公司根据 Carley 在华尔街日报 1998 年 6 月 29 日的报道,指出由于缺乏有效的信息、协调和控制,铁路走廊上的交通瓶颈导致的机车闲置率高达 40%。在对运输系统进行了全面分析和研究后,我们相信,新的分布式控制算法将是未来运输系统架构设计的最合理选择。我们可以预见,新型计算机控制算法和大型分布式软件在未来运输系统中将起到越来越重要的作用:它能为我们提供一种全新范围的个性化的出行服务;使全系统中每个实体的移动效率得到本质的提升;实现高速公路入口匝道的先进控制;在综合考虑高速公路和街道路面交通流的基础上,提供个性化的快速公交服务;促进环境朝更安全的方向发展等。

第二,我们相信,要实现智能运输系统(ITS)质的飞跃,未来的高速公路运营工程师和交通专家们必须接受在土木和电子工程、运输工程和规划、人文和最为重要的计算科学与工程(CSE)等学科交叉领域上的训练。CSE 学科中的分布式系统、算法设计、网络、计算机模拟和分布式仿真等基本知识尤为重要,因为它们是理解最复杂的、高水平系统结构体系的关键。1998 年 1 月,Massachusetts 大学的John Collura 教授在一门新课程中向本科生和硕士研究生详细介绍了 ITS 的最新进展,同一时间,Michigan 大学的 Chelsea White 教授在交通技术国际研讨会上指出:未来的运输问题本质上一定是各学科之间的问题。最近,Arizona 州立大学启动了一个关于运输系统的"跨学科认证等级计划",该计划受到来自规划和园林建筑学、土木与环境工程、地理和航空管理与技术等专业教师的支持。本书中的思想和理念也是"自治的分散式交通运营系统(ATOS)"的基础,该系统控制着世界上最大的运输系统——东日本铁路公司。由东芝公司建立的 ATOS 包括 5000 台自主计算机,每天控制着超过 6200 列火车[1]。

第3章检验了所有运输系统的基本特征,特别是从未来需求的角度,并根据分 析提出了其发展的基本原则。这一章还提出了未来系统的关键设计问题,包括控制 算法、系统中不同实体之间的交互特征、连接各实体的网络等。该章还强调了在未 来系统设计过程中模拟和仿真的重要作用。第4~6章介绍了几个案例研究情况, 对第3章中提出的观点进行了论证,并进行了概括和总结。第4章详细描述了在铁 路系统中计算火车路径的一种新型的分布式方法。该章先对火车控制中的传统方法 作了解释,并对相关文献进行了详细综述。然后,提出一种新型的分布式控制算法 DARYN。该算法先在一个小型铁路网中模拟,然后才在 ARMSTRONG——一个松 耦合并行处理器上进行模拟,获得了该算法的测评指标。第5章提出一种更为精密 的算法——RYNSORD,该算法主要用于在铁路网络中进行有效调度和传递拥堵信 息。在RYNSORD中,每列火车都使用前向值来动态规划其前方路径,即在每个阶 段、火车都会预定前方将要使用的一节或多节轨道。该预定过程具有"软"特性, 也就是说,当火车和车站之间进行协商时,少了一些硬性规定,多了些灵活性,这 和传统的死板的"硬"预留技术是相反的。RYNSORD 在美国东部铁路网的一个子 网上建立了模型,并在一个由65+SUN Sparc 10 工作站组成的计算机网络上进行了 仿真。在这一章中,对该算法性能表现的分析,主要是通过对火车出行时间的比较 和路径决策的质量比较来实现的,路径决策方案的质量是文献中对系统的唯一度量 标准。第6章介绍了 DICAF——一种智能车路系统 (IVHS) 的新型结构。该结构 中,全部动态路径引导任务和智能化的拥挤转移任务都被分配给系统中每个实体 (即 IVHS 系统中所有汽车和高速公路基础设施)来共同完成。IVHS 这个词仅限于 在与高速公路车辆有关的情况下使用,而ITS 这个词的应用面则更广,包括了所有 运输系统。DICAF使用了一个连续函数——拥堵程度、来影响路径引导过程、并 针对现实中的某高速公路系统、提出一种新的度量标准、即将 DICAF 的性能与绝 对最好值相比较。DICAF 和 RYNSORD 都有一个共同的目标,就是实现系统资源的 有效配置、它们只是根据两者之间在本质上的区别、采取了不同的策略而已:火车 受限于轨道,所以在使用轨道前必须要进行专门的预留,而汽车则有更大的灵活 性,可以与道路上的其他汽车一起分享道路资源。第7章对运输过程中出现干扰时 各种新型计算机算法的稳定性进行了系统的和详细的研究。这一分析对于理解复杂 的运输系统的鲁棒性和弹性是非常关键的。这章以 RYNSORD 为例阐述了分析的基 本原则。第8章介绍了一些在ITS设计中为模拟和仿真开发出的新技术。

综上所述,本书的组织思路如下:第3章提出一般性原则,第4~6章解决现实世界中的某些具体问题。作者的想法是,尽管看起来是基本类似的,但是每一个运输问题在它的本质、特征、要求等方面都是独特的,都需要建立一种服从于该系统基本原则的常规解决方案。我们希望读者能真正理解这些既一般化却又很特殊的概念,并将它们融入到未来问题的革新方案中去。

本书可供运输工程和计算机科学的研究生,不同运输学科的研究人员,铁路、高速公路和航空从业人员,运输体系的政策制定者等使用。它是为数不多的可作为研究生课程"先进的运输工程"教材的书籍之一。除了理论概念、计算机模拟和实验分析以外,本书还附带了CD-ROM,针对第5~7章中的每个实例,为读者提供了可执行的仿真器,使读者能在Linux工作站组成的网络上,针对不同参数选择、不同网络配置和不同输入数据的情况进行仿真,以通过动手实验来加深理解。导师们还可以将仿真器作为实验环境的一部分,供学生在实验室进行练习。第10章介绍了仿真器的使用。当然这些仿真器只是供学术研究而用,范围有一定限制,读者如果需要获得仿真器的商业版本,可以联系sumit.ghosh@ieee.org。

作者还要感谢许多将他们对运输系统的理解贡献给本书的人,包括第一作者以前的学生, Raj Iyer, M. D.、Noppanunt Utamapathei 和 Kwun Han, 他们在第一作者于布朗大学获得终身教职期间正随他攻读学位; 联邦高速公路管理局 (波士顿)的 Rick Backlund、罗德岛运输部的 Robert Shauver 等。也感谢运输方面各篇文献的作者, 他们帮助我们理解了这个学科, 我们对此深表感谢。我们还要感谢美国国防部的资助机构的一贯支持。最后, 我们要向 CRC 出版社工作人员的热情和一丝不苟的工作态度表示衷心的感谢。

关于作者

Sumit Ghosh 于1980 年在坎普尔的印度理工学院获得电子工程学士学位, 1984年,他在加利福尼亚的斯坦福大学计算机系统实验室(电子工程和计算机系 联合设立)获得硕士和博士学位。Sumit 曾经在贝尔实验室 (霍姆德尔,新泽西)、 Fairchild 电气先进技术研究和发展实验室(帕洛阿尔托、加利福尼亚)及 Silvar-Lisco 公司 (门洛公园, 加利福尼亚) 接受过工业研究训练。他曾经在布朗大学、 亚利桑那州立大学、史蒂文斯理工学院都持有学术职位、现在是在得克萨斯大学泰 勒分校任职。他是 IEEE 计算机学会 2004 技术成就奖的获得者, 是设计与流程科 学学会 (SDPS) 会员,并且在 2008 年被推选为 SDPS 主席,而 SDPS 是跨学科研究 和教育方面的领头羊。Sumit 是以下 5 本参考书的第一作者:《Hardware Description Languages: Concepts and Principles》 (IEEE 出版社, 2000); 《Modeling and Asynchronous Distributed Simulation of Complex Systems》(IEEE 出版社, 2000);《Intelligent Transportation Systems: New Principles and Architectures》(CRC 出版社, 2000); 《Principles of Secure Network Systems Design》 (Springer-Verlag, 2002); 《Algorithm Design for Networked Information Technology Systems: Principles and Applications (Springer-Verlag, 2003)。同时,他还是《Guarding Your Business: A Management Approach to Security》(Kluwer 学术出版, 2004)的主要合编者之一。

Tony S. Lee 在罗德岛州普罗维登斯的布朗大学计算机工程专业和政治学专业获得学士学位,同样在布朗大学获得电气工程硕士和博士学位。他目前是雅虎公司的科学带头人。Tony 已经在十几本重要刊物和会议上发表论文,涉及的领域包括运输系统、银行系统和军事指挥与控制。他还和 Sumit Ghosh 合作编写了《Modeling and Asynchronous Distributed Simulation of Complex Systems》 (IEEE 出版社,2000,6)。在获得当前职位以前,Tony 曾是加利福尼亚州桑尼维尔远创科技数据转换公司的资深高级工程师,工作范围是分布式软件开发和网络技术。在他获得远创科技终身职位之前,还曾是美国国家航空航天局 (NASA) 埃姆斯研究中心 (Moffett Field,加利福尼亚州)的一名研究员。在 NASA,他参加了几个高速网络的研究项目,研究内容包括网络服务质量和信息电网等。Tony 也是 ATMSIM 软件包的发明者,这是一个高保真的、分布式网络模拟器,它能帮助网络研究人员为新的协议、算法以及识别复杂的交互行为而开发现实模型。

目 录

译者序	
第2版	反前言 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)
第1版	反前言
关于作	E 者
第1章	■ ITS 技术的发展现状
1. 1	ITS 的广义范围 1
1. 2	ITS 的一般性定义 2
1.3	ITS 的发展现状 2
1.4	ITS 技术发展回顾 4
第2章	章 待发掘的 ITS 新的元层原理 ····································
2. 1	元层原理 7
2. 2	ITS 的新挖掘点和挖掘措施 8
2. 3	挑战和机遇的案例 11
第3章	章 交通运输系统存在的基本问题 ····································
3. 1	ITS 的主要特性 17
3. 2	通过建模和仿真科学地验证 ITS 的设计 22
第4章	DARYN :铁路网络的分布式决策算法 24
4. 1	概述
4. 2	DARYN 方法······ 26
4.	2.1 算法
	2.2 避免死锁的依据
	2.3 在松耦合并行处理器上模拟 DARYN
	在 ARMSTRONG 上执行 DARYN
	DARYN 的性能
4. 5	DARYN 算法的局限性

第5章	RYNSORD: 一种新型的基于软预留机制的铁路网络分散式	
	控制算法	46
5. 1	引言	46
5. 2	RYNSORD 算法 ·····	47
5.3	在精确、实时、并行处理的实验台上模拟 RYNSORD	55
5.4	算法的执行问题	60
5. 5	仿真数据和性能分析	63
5. 6	RYNSORD 的局限性	76
第6章	DICAF: 一种智能车路系统的分布式、可扩展结构	77
6. 1	引言	77
6. 2	DICAF: 一种新颖、分布式、可扩展的 IVHS 方法	83
6. 3	在一个精确的并行处理试验台上进行 DICAF 的现实模拟	
6. 4	实施操作和调试	96
6. 5	仿真结果和 DICAF 的性能分析	97
第7章	RYNSORD 在干扰下的稳定性······	118
7. 1	稳定性	118
7. 2	RYNSORD 稳定性的正式定义	121
7. 3	RYNSORD 稳定性分析模型 ······	122
7.4	RYNSORD 的稳定性分析	123
7. 4.	.1 稳定性分析的误差标准	123
7. 4.	. 2 稳态分析	124
7. 4.	.3 对输入率的干扰和稳定性分析	125
7. 4.	.4 对系统特征的干扰和稳定性分析	130
7.	. 4. 4. 1 对车站之间和火车与车站之间通信的干扰	130
7.	. 4. 4. 2 与轨道区间段有关的干扰	132
第8章	ITS 设计中的建模和仿真技术 ······	
8. 1	引言	141
8. 2	ITS 设计中移动策略的虚拟和物理的处理过程	141
8. 2.		
8. 2.	.2 物理进程移动策略	144
8.3	进程移动策略中的软件技术	
8. 3.	, , , , , , , , , , , , , , , , , , , ,	
8. 3.	. 2 PPM 中的软件技术 ······	150

8.4 实施情况 实施情况 8.5 仿真结果和性能分析	
第 9 章 ITS 未来的问题	
第 10 章 CD- ROM 中的 RYNSORD 仿真器及其实验范围说明	161
10.1 安装	161
10.2 概述	162
10.3 准备运行	163
10. 3. 1 NETWORK. OUT	163
10. 3. 2 帮助文件	164
10.3.3 输入生成	164
10.3.4 输出文件	164
10.3.5 故障处理	166
10.3.5.1 如何知道仿真系统正在工作	166
10.3.5.2 问题	166
10.3.6 轨道和通信故障	166
10.4 结论	167
参考文献	168

第1章 ITS 技术的发展现状

1.1 ITS 的广义范围

自 1999 年以来,人们越来越清晰地认识到,如何高效地实现旅客和货物运输这一问题远比想象的要复杂得多。该问题不仅存在于由车辆和道路组成的地面交通中,它还影响到火车、客机、航空货运、渡轮和船舶等所有可用的以及当前在用的运输方式。对这一问题的认识,使得人们放慢了研究和应用智能车路系统(IVHS)的脚步,而促进了更广泛意义上的智能运输系统(ITS)的发展。两名研究人员,Kan Chen 和 Bob Ervin,重新定义了 IVHS,扩大它涉及的范围,并将该系统重新命名为 ITS(Chelsa White 教授,2010)。显然,要形成一种全国甚至是全球通用的、真正可行的解决办法,必须从全局着眼,考虑多种运输方式之间复杂的相互独立、相互依存的关系,同时需要建立一个基本目标,即在公平分配可用资源的条件下,尽量减少所有旅客和货物的在途运输时间。在不久的将来,通过太空旅行到月球和太阳系中的其他人造卫星和行星去也将成为家常便饭,这些都可能需要在 ITS 中进行协调。因此,目前在对 ITS 体系结构基础进行规划时,就应该考虑到各种运输方式的特性,为未来实现各种运输方式之间的无缝一体化衔接做好准备。

根据《综合地面交通效率法(ISTEA)》(俗称冰茶法案)、《21世纪交通运输平等法案(TEA-21)》和《安全、可靠、便捷、高效的交通平等法案(SAFETEA-LU)》,美国 ITS 界普遍认为,解决这一复杂问题的关键是两个方面的科学研究和工程进展:首先是不断提升中的计算能力,它是以功能强大的台式工作站和移动笔记本电脑、掌上电脑、手持式个人数字处理器(PDA)等形式提供的。其次,是逐渐普及的通信和控制网络,包括有线和无线网络。然而,仅仅提高计算能力和普及网络本身并不会就自动形成一个解决方案。举一个例子:通用电气机车建设公司的 Carley^[2]指出,由于信息交流不畅、协调和控制不力等导致在铁路运输走廊产生的瓶颈,可能使机车闲置率高达 40%。由于柴油机车即使在闲置时其发动机也要运行,所以在 20世纪后期,许多铁路公司每年的燃料预算都超过 800 万美元,浪费非常惊人。无论以什么标准来看,这种低效率都是不可持续性的。在本书中提到,要成功解决这一复杂运输问题,关键就在于要全面了解控制和协调算法。抽象地说、这些算法能将计算能力和网络资源以一种协同的方式统一起来。

1.2 ITS 的一般性定义

ITS 是一个同时涉及科学性和工程性的学科,其主要目标是在行驶速度不断提高,旅客数量不断增大,旅客迫切需要更精确、及时的信息的情况下,通过现有资源的公平分配,减少所有旅客和商品的运输时间,同时确保安全。为了实现这一目标,在社会规范、政策和指引的约束下,ITS 必须通过异步分布式控制和协调算法使道路交通,铁路,航空客、货运,海上渡轮及其他不同运输方式有效地衔接起来。这种衔接的结果将表现为:①旅客可以从系统中的任意位置获得任何一种运输方式准确的运行信息;②旅客能利用个人出行辅助决策系统计算出最有效的路径,或通过处理所掌握的资料,在不同运输方式中重新设定出行路线;③将允许旅客动态地,甚至是在途中,在各种运输方式中执行有效的预订操作。实际上,ITS 涵盖了运输管理的各个领域的工作,包括州际公路管理和交通信号控制、包含多方式旅客出行信息在内的出行管理、公共交通和运输管理、涉及事故或铁路平交路口以及紧急救援服务等内容的安全管理、先进的车辆控制以及付费和收费管理等。

1.3 ITS 的发展现状

尽管早在1991年就颁布了"冰茶法案"(ISTEA),但目前州际和国家公路系统上的拥堵问题仍很严重。这不是一个新问题,因为拥堵已经出现很长一段时间了。2003年伦敦的平均车速为17mile/h(1mile/h=0.44704m/s),相当于100年前马车时代的速度。如今在曼哈顿,行人的平均速度是3.3mile/h,具有讽刺性的是这竟然要高于机动车3.1mile/h的平均速度。根据对5个最拥堵的地区(洛杉矶、华盛顿、旧金山-奥兰多、迈阿密和芝加哥)10年数据的研究,Lomax^[3]在2002年指出,每年每个司机由于停车等待造成的平均出行延误相当于一周工作时间,而每年仅在洛杉矶由于拥堵导致停车而造成的燃料浪费就高达86亿美元。最严重的出行延误发生在圣贝纳迪诺,为每年每车75h,而最高的人均燃料浪费为华盛顿特区的860美元。拥挤增加导致的另一个后果是美国州际高速公路系统和其他主要道路严重受损^[4]。路面的碎裂部分和坑洞不仅给高速行驶的车辆带来危险,而且碎片很可能变成飞行的导弹,引发交通意外和事故。为了避免拥挤,重量超过10万1b(11b≈0.4536kg)的重型拖车越来越多地选择使用农村道路,这反而又加剧了农村路面碎裂的程度,因为农村道路是为荷载小于10万1b的轻型车辆服务的。

但是,我们仍然可以从两个不同的角度看到 ITS 带来的进步。第一个角度是从科学文献综述中总结出的下列领域的研究成果:系统规划、交通流建模、系统评价、车辆跟踪、基于 GPS 导航的自动驾驶、信号控制、车辆制动、车道检测和转

向控制、智能驾驶控制、向司机传达路况信息、噪声污染控制、队列控制、ITS 系统的评价与仿真以及 ITS 工作人员培训等。通信系统中越来越多地采用无线和光纤取代铜缆方式,提高了交通运输系统中的快速通信能力,因此可以做到不断地重新计算和调整交通灯信号中的红灯、黄灯和绿灯时间,使道路交通流和车辆更有序地行驶。在对车道进行追踪和对车道变换的自操纵研究过程中,1000Hz 线性摄像机在距离为 10m、速度为 60km/h 和加速度为 - 2m/s²的情况下能高效地提供精确到 1mm 的相对距离精度。到目前为止,ITS 最重要的成就就是能利用信息技术为用户提供有关信息,这些信息包括从家里、公司,以及全球的移动电话,或通过互联网获得航班和火车到达和离开的信息;自动通知延误和取消信息;通过移动电话和短信方便地预订座位和出租车;以及从因特网上获得从起点到终点的街道方向和地图等。获得这些信息的能力极大地体现了 ITS 的进步。

第二个角度就是,自 1991 年以来,ITS 产业不断发展,带来了许多新的市场 产品和服务,通过这些产品和服务也可以检验 ITS 的进步。例如,美国许多城市都 提供免费电话号码(如波士顿)和公共网站(如西雅图),使人们可以从中获取城 市道路拥挤的最新状态报告 (Tim Lomax 教授, 2010) 等信息。这种服务是非常有 价值的,但有三个关键问题需要解决:第一,最新的路况信息往往仅限于为数不多 的主要道路,而对于更多的次要道路,要么没有这些信息,要么只能提供过时信 息;第二,在全部公路上传播完整的信息所需的时间通常很长,因此,对某些驾驶 员来说,这些信息的用处是有限的;第三,系统的信息服务不是实时动态的,在驾 驶途中, 信息可能会由于意外和事故而发生重大变化。如今, 许多载重汽车、机场 接送服务车,以及豪华汽车都配备了 GPS 导航装置,获取信息应该不成问题,因 而 ITS 面临的主要问题还是在干缺少一种实时的方式来传递关于拥挤和道路封闭情 况的准确且实时动态的信息。许多公路部门已经转向使用调频广播,播报由于施工 导致的匝道关闭、车道压缩以及其他相关公路状况等信息。这种做法有一定意义、 但是还是可能存在由于电磁干扰导致驾驶员不能及时收到广播的问题。在某些交通 运输网络中,人们喜欢沿着高速公路和快速路安装大量摄像头,将视频信号反馈回 集中式交通控制中心,监测拥挤情况。虽然摄像头的安装和维护成本较高,但是它 们确实能有效地引导途中的警察和救护人员到达事故现场,还可以记录可疑车辆的 车牌。然而,在对拥挤进行监测与控制这一问题上,还有一些其他设备可以使用, 如铺设在人行道下的光纤等,它们都比摄像头更为便宜,而且非常可靠。上述这些 措施都促进了非官方的旅行速度数据市场的逐步形成和发展。电子收费系统(如 美国东部和沿95 号州际公路走廊的 E- ZPass 系统) 已被看做是 ITS 领域的一项重 大成就。这套系统中,车流可以以 15mile/h 的速度通过收费站,避免了车辆在收 费站停车交费,看似可以明显减少延误。然而,实际上 E- ZPass 电子收费系统的优 点只有在道路不拥堵的时候才能发挥出来。在实际道路条件下,它的表现只能说是

差强人意。在这个国家的某些地区,任意某个工作日,尤其是在交通高峰期,成千上万的汽车开往收费站,当车辆从当前车道转向对应的收费车道时会十分危险,这必然会导致一些车辆减速,另一些则完全停止行驶,从而使车辆处于一个高度紧张和不安全的环境中。当现金收费站的排队很长时,一般通过 E-ZPass 收费通道的车辆平均速度几乎都不会超过 2 或 3 mile/h。在收费站出口,由于 10 个收费车道突然减少为 3 或 4 个车道,加上司机为弥补交费所耽误的时间而加速出站,使收费站出口变得更加危险。按照目前的情况,无论是电子或者是非电子的收费模式都是低效的,因为它都不必要地减缓了车流速度;同时也是不安全的,因为它造成了一个危险的驾驶环境;上述这些问题都从根本上背离了 ISTEA 的基本目标与 ITS 的宗旨。

在对铁路算法的研究中,我们早在 1992 年就揭示了集中调度下的火车空闲等待时间比分布式调度时要高 5~40 倍。1998 年 Carley 指出高达 40% 的机车由于得不到调度指令而空置在侧线停车站上。然而,直到 2010 年 1 月,我们从一个美国交通部官员那里得知,铁路系统仍然依赖于集中控制。以整个东部沿海地区为例,美国铁路运输的火车仍由佛罗里达州的杰克逊维尔集中控制,并且在任何一个时间,在卡罗来纳州的农村地区,长达 20mile 的轨道上只有一列火车在运行。这种低效率必将不可避免地导致拥挤和延误。

1.4 ITS 技术发展回顾

我们先后与美国的主要交通部门[包括交通部(DOT)、美国联邦公路管理局和各州运输部等]进行了广泛、深入和坦诚的交流。仔细分析讨论后发现,到目前为止ITS仍没有取得长足进步的原因,也许是因为我们面临的一些强大且持久的挑战。对政策制定者和交通管理者来说,不断深入研究、讨论和思考这些问题是非常重要的。否则,ITS可能会继续停滞不前,这最终可能会导致公众对其失望而过早地放弃它。

第一个挑战是,ITS 内部存在一个强大的障碍,阻碍我们去想象ITS 将带来怎样的革命性变化,而这些变化将大大改善我们的生活质量,并且给当今乃至未来的社会创造丰厚的利益。历史表明,阻碍我们进步的一个最大的障碍在于我们缺乏对未来的想象力,并且我们还拒绝承认这一事实。如果缺乏信念,我们就会失去为获取成功而努力的动力,那么,即使是最伟大的想法也将会以失败告终。当我们在这个障碍中挣扎时,一个事实可能会对我们产生极大帮助:伟大的教育家和思想家,无论是什么民族、文化,在什么时间,他们一直都在强调想象力的重要性。今天,大部分科学家和工程师们把想象力看作是一种软科学,与之相对应的是硬科学,即物理、数学等。目前,在世界上任何地方都没有将想象力放入正式的高等教育课程。然而,具有讽刺意味的是,在所有领域,包括科学、数学和工程中的每一个伟

大的概念和原理无一例外地都开始于直觉和想象的世界,然后,再通过巨大的努力,这些概念才最终成为社会的基础和支柱。战车的车轮、可远距离投掷的武器、整个数字系统、自然语言中的语法、音乐的节奏、傅里叶分析、相量分析(电气工程中最难的概念之一)、晶体管、离散数字计算机、杜比噪声消除系统、炸药,甚至是打败黄金成为世界第一法定储备货币的强大的美元等,所有这些都是起源于想象。这种现象已经成为"humble beginnings"一词的来源。我们难以想象如果没有这些宝贵的概念和根据这些概念生产的产品,世界会是什么样吗?在第2章中,我们将重点阐述这个现象,并解释我们要怎样去接受它。正如拉丁语 FIAT 的意思一样:让它去做吧。FIAT 纸币是法定货币,它的价值仅仅来自于政府的有关法令:这些钱是法定货币。而它本身不具有任何内在价值,也不依赖于任何有形的商品,包括黄金、白银和贵重金属,而对它的信心却来自于对国家经济和稳定的信任。因此,要相信我们的想象力,必将能引领 ITS 的进步。

第二个挑战或许可以称为是自我设限。大部分的运输从业人员似乎已经渐渐相 信现实世界的问题非常复杂,永远也不可能通过 ITS 得到解决。许多交通官员与我 们分享了他们在不同的背景和环境下碰到的挫折。几年前交通部的一位高官在作观 察报告时(这里改述其意思),得出了一个有代表性的经典评论:任何 ITS 项目都 不可能满足实际的交通需求、因此很难说在设计决策和选择方案过程中到底需要哪 些知识,所以一味探究 ITS 项目的效果是没有意义的。看看下面这个真实的故事: 在 1995~1996 年间, 10 号州际公路 (I-10) 上, 从凤凰机场到美国的亚利桑那玉 女溪那段路只有单向两车道,在高峰期常常堵塞严重。后来,亚利桑那州的人口经 历了奇迹般的增长,新小区沿着 I-10 公路四处涌现。人们的出行逐渐感到不便, 并且出行费用很高。为此亚利桑那州交通部开始了一项雄心勃勃的计划:拓宽 I-10 公路为单向 3 车道。将近 1 年的工作之后,这条路重新开放,成为双向 6 车 道。起初,这极大地改善了当地交通状况,拥挤几乎消失了,沿 I-10 公路乘车上 下班变得轻而易举。但是,这种现象出现后不到6个月的时间内,交通拥堵又重新 出现、情况又回到了一年半前的状态。也就是说、不到一年半的时间、由于人口的 膨胀, 使 6 车道的 I-10 公路变得毫无作用。因此, 交通官员的沮丧是很容易理解 的。然而,幸运的是,我们已经引进了一个新的原理[5]:间接测定输入刺激;或 以一个稳定的判断标准,去弱化问题的严格性。它是基于音响工程和逻辑原则,体 现了一个全新的且具有实用性的思想,本书将在后面的章节中解释这个原理。

第三个挑战是教育。2000 年末,一位交通部的高级官员告诉我们,交通部迫切需要先进的模拟器以研究和评估 ITS 项目的效果。他们已委托了一家研究所,对顶级供应商和一间著名的学术机构提供的 51 套最先进的 ITS 模拟器进行评估。经过综合测试,研究所和交通部的结论都是:这些模拟器全部不可接受。这件事证明了我们长期坚持的专业意见是正确的:即任何一个项目的完整模拟研究,都需要具

有真才实学和专业知识的团队根据这个项目的需要从零开始进行开发。"罐装式"的模拟器是不可能满足 ITS 项目对灵活性的要求的。我们知道,一个复杂的问题永远不可能会被全部预知,这意味着使用有限选项集的"预处理式"模拟器来预测一个既定问题的所有复杂需求,即使可能,也是很困难的。在美国国防部,尤其是美国陆军,在过去 30 年中已经花费了没有数十亿,也有数亿美元,去完善他们自己的罐装式软件模拟器,以用于战区的规划和实际操作。通过与军方高层官员坦诚讨论后得知,它的研究进展可以描述为突飞猛进。我们相信,如果在各军部阶层普及建模和仿真的专业知识,就可以在和敌人实际战斗前对作战计划进行更深入的研究,就可以避免许多任务的失败和人员的伤亡。美国交通部关于"预处理式"模拟器的调查结果证实了我们的想法:即为了保持在 ITS 研究和高科技产业中的领先,美国需要一系列卓越的创新,以实现①将行为模型和异步分布式仿真引入主流的研究和教育领域;②以学士到博士甚至更高的教育水平训练学生,在这个领域中培养出真正优秀的人才。

第四个挑战表现在缺乏对 ITS 研究和发展的正确引导,不仅如此,更糟糕的是,还可能无意间强化了不当的动机,在优先发展哪项 ITS 技术这一问题上对运输从业人员进行了误导。自 1997 年以来,为了监测拥挤和检测事故,政府投入了大量资金和精力,在红绿灯信号控制路口安装了许多摄像头。而实际上,检测拥挤的自动化设备和装置早已研发成功,相对于高空摄像头的开销,它只需要 1% 的安装成本,而在系统整个生命周期中的运营成本甚至不到高空摄像头的 0.01%。交通部官员不是不知道这些事实,但是他们依然选择花费高额费用安装高空摄像头,主要原因是因为这些摄像头能让人们产生一种成就感和进步的错觉。其实这些摄像头本身也并不能代表 ITS 的真正进步,在全国范围内强制安装摄像头,只是使我们的注意力从改善路况的需求上转移开而已。在这个高科技产业占主导地位且充满竞争的时代,我们必须为 ITS 的工程研究人员提供真正自由的空间,使他们能认真而耐心地面对 ITS 中真正艰巨的挑战。

第2章 待发掘的 ITS 新的元层原理

2.1 元层原理

在过去 10 年里,出于某些独特的考虑,我们开发出一系列新的元层原理,我们会在这里把它们列出来。洞察力来源于观察,自然界中几乎所有(虽然偶有例外)伟大的发明和发现都是偶然出现的,发明者似乎都是通过纯粹的巧合偶然发现了它们。例如青霉素的发现、晶体管的发明和炸药的发明等。我们还注意到,伟大的发明和发现寥寥可数,因为我们大多数人在自然进化过程中只是担当一个被动者的角色。今天,我们站在人类文明史中一个特殊的十字路口上:这是历史上的第一次,我们拥有充足的资金,只要有足够的意愿,我们就可以从容地进行发现和发明。这种能促使我们成功的东西就是计算引擎,目前一个众所周知的表现形式就是数字离散计算机。

要理解计算机如何帮助我们实现想法,就需要研究计算引擎的基本性质,它是 根据以下这组元层原理而得出的:第一,计算引擎归根结底是人类智慧的放大器, 它的放大系数是10亿(还在不断增长中)。人类的智慧具有两种最基本但却是高 水平的功能,就是做决策和与人进行交流。计算引擎不仅同时具备了这两个功能, 还能以极快的速度执行这两个功能。第二,我们所能想到的任何东西,即便是完全 随意的想法,现在也都可以使用计算引擎在抽象的系统里实现,并且通过行为模型 和异步分布式仿真进行更深入的研究。至于说这个想法是否违背已知的物理世界和 社会的规律,已无关紧要,它仍可以在一个被我们称为网络空间的全新维度的计算 机抽象世界里实现。由于建模与仿真速度极快,这就使我们能够在有限的时间内检 测、探索和研究最复杂的想法,并且努力在现实中去实现。第三,与其被动地等待 某些问题的自然降临,如SARS、艾滋病或某个超级火山喷发,然后被动地努力去 寻找解决的办法,不如主动去想象一下未来可能面临的各种挑战,并且积极地开始 寻找正确的解决方案。我们会有足够的时间来评估各种解决方案的优劣,并不断完 善它们,说不定还可以得到意外的让人惊喜的结果。美国加州大学伯克利分校名誉 教授 C. V. Ramamoorthy 认为,应该在我们的科学和工程学科系列中创立一门全新 的分支学科, 名为"需求工程"。他和我们都认为, 站在今天的角度, 对各种方便 性需求进行适当的思考和想象,就可能真的演化为未来的社会需求。第四个元层原 理是,我们必须认真而充满热情地致力于跨学科的研究和教育,才可能获得成功。

2.2 ITS 的新挖掘点和挖掘措施

虽然 2.1 节介绍的元层原理适用于任何领域,并且都可能会带来巨大的进步, 但是在本书中我们将重点研究 ITS 这一领域。从本质上来说、ITS 就是一座矿藏、 它不产生任何具体的物质产品,诸如煤矿、黄金或者钻石等,但同时它也不会枯 竭,它是能让世界变得更美好的智慧思想的源泉。鉴于我们对文明的渴望正迅速地 从物质转变为思想,对ITS 这个矿藏的认识来得非常及时。建造摩天大楼和大桥已 不再是空想,我们已经成功建成了很多,这已经没什么稀奇的了。我们正在寻找能 帮助我们进一步成长的新挑战,这些挑战将主要存在于想象的世界里。有这样一个 例子,现在听起来可能比较荒谬,但是将来肯定会实现,那就是希望能与 Panini 进 行一次有意义的交流,问问他为什么以及怎样创造了梵文语法,这样说不定我们就 能够为某种新计算机设计出一种全新的计算机语言。另外一个现在看来同样荒谬的 例子,就是去询问阿基米德他为什么要发明里程计,而且在罗马公路沿线作英里标 记,除了为罗马帝国制定作战计划、保障后勤以镇压叛乱这个众所周知的目的之 外,还有什么其他原因?第三个例子是我们非常希望能与托马斯·杰斐逊进行真诚 的交流,以得到他对于那部给现代人带来困扰的《人权法案》中各种问题的见解。 在创作《人权法案》时,他和他的同事关心的是什么、恐惧、幸福还是对美国未 来的希望?这些见解对未来任何一个可能产生的新政权来说都会是无价之宝。当 然,我们也明白,无论多么复杂的推理技术、多么先进的计算机记忆模型、左右大 脑的非线性相互作用、建模与仿真技术,都无法让时间倒流。

我们的发掘工具将包括我们所熟悉的原理、概念、理论和科学、数学、计算机科学、电子工程、生物制药、力学、土木和航空工程、经济、商务、会计、人类学、艺术和哲学等各学科各自掌握的规律,在此基础上,还会用一种前所未有的方式将它们融合起来。也就是说,我们的发掘技术将建立在智能和发散型创造性思维的基础上:即我们挖掘得越深,就会有越来越多的可能性出现,我们可能从中获得更好的知识,但也可能面临越来越复杂的子原则;我们爬得越高,就可能会得到越多更具体、更先进、更完整的理念,也可能使我们的思想不断变得更开阔。我们把这种新的方法称为跨学科的研究和教育理念。然而,关于跨学科研究和教育的详细范例讲解超出了本书的范围,在这里我们只介绍几个关于如何使用这种方法的例子,6.7节和6.8节中将为读者提供更多相关的讨论。

第一个事例是这样: J. C. Bose 之所以出名是因为他在 1895 年首次发现了无线电^[9],但是很少有人知道大约 60 年后,他为贝尔实验室发明晶体管所做的工作。Bose 曾经痴迷于麦克斯韦的高频电磁波存在假设和赫兹关于在 10~60GHz 微波段范围内能产生高频电磁波的证明,但是不知道如何检测它们的存在。今天,每一个

移动电话都有一个微波检测器,但是在 19 世纪 90 年代,微波绝对是一个谜。Bose 提出了一种新颖的方法来探测 60 GHz 的微波。他用不同的金属和材料做成一些不同的接头,用电线把他们连接成了一个电路,并和一个非常灵敏的电流检测器相连。当他把接头暴露在微波下时,电路就会显示出了瞬间电流迹象。他这种测试仪器敏感性的设计方法完全是由他的科学好奇心所激发的:植物是否会和人类有类似的反应?它们是否也是"活着的"?尽管科学史并没有告诉我们他选择这个方法的原因,但还是为我们提供了一个符合逻辑的解释。我们都知道,双金属片的使用可追溯到远古时代,它们通常被广泛用于建造坚固的战车车轮。这两种金属片在加热和冷却时的膨胀性和收缩性不同。天才 J. C. Bose 一定思考过:任何两种材料都不具有绝对相同的性质。因为金属可以导电,并且电磁场和电流彼此相关,当不同材料的接头组成部分暴露在微波下时,很有可能表现出不同的电性。很显然,Bose就是一个跨学科的科学家,或许是现代最早的跨学科科学家之一。他的思想里融合了电磁学、放射学、电力学、化学、冶金、植物生理学等各个学科的概念和原理。

第二个事例是一个不太常见的问题[10,11]。在 20 世纪 90 年代中期,加利福尼 亚州的71号公路需要维修。在维修期间,71号公路上的交通流将被迫分流到另一 条公路即83号公路上去,维修期预计是6~12个月。每天交通量的最大增量估计 会发生在下午,每小时约增加580~1460辆车。这一增量会使噪声级别提高3~ 4dB。美国鱼类和野生动物保护组织担心这会影响到一些濒临灭绝的鸣鸟类物种的 生存环境, 其中一种叫贝尔氏绿鹃的鸟, 就生活在 83 号公路沿线约 200ft (1ft = 0.3048m) 长的范围内,周围环境噪声级的增加会给这种鸟类的生存环境带来不利 影响。成年的鸟可能因惊吓,会丢下自己的蛋和脆弱的幼鸟而独自离开,最终可能 导致整个种群内鸟的数量越来越少。另外,也有人认为道路噪声的增加妨碍了鸟类 互相传递听觉信号,这些信号往往是鸟类互相传达有危险的信息、发现食肉动物以 及照看幼鸟等信息的主要手段。美国鱼类和野牛动物保护组织与加利福尼亚州的运 输部门 (DOT) 的谈判陷入了僵局。这时, 一个在信号 (S)、噪声 (N) 和噪声 比率 (S/N) 等方面开展过深入研究的专家提出了一个独特的方案。他认为, 尽管 周围存在噪声(N),这种鸟仍能活下来,这意味着 S/N 比率(其中 S 代表鸟类的 听觉信号强度)还在鸟类的容忍极限范围内。显然,如果信号噪声比率 S/N 能维 持在当前水平,公路建设就可以继续进行。为此,接下来做了这样几件事:首先, 测量和记录当前83号公路沿线的噪声水平,记为N1。显然,鸟类已学会了忍受 N1 的环境。早在 1987~1988 年间, 生物学家 John Rieger 就已经计算出了鸟类对噪 声的容忍极限在 60dB。其次、公路部门将把 83 号公路上的限速从 55mile/h 降到 35mile/h. 并沿 83 号公路建造了隔音屏障, 以控制道路噪声增长水平: 此外, 当 71 号公路上的交通开始向 83 号公路分流时,连续监测隔音屏障外侧的噪声级,以 确保噪声级别绝对不会超过 N1。这个计划后来执行得非常顺利。

第三个事例是关于在亚利桑那的一间学校里做过的一个实验。首先,给一组学 生分配一些阅读任务,其中一个任务是跟第二个事例有关的。在最后的测试中,要 求学生回家完成,并回答一个问题。这个问题的背景是:日本的关西机场[12]是一 座建造在大阪湾的人造岛,修建时,特意选择远离日本大陆的地方,以便使它能一 年 365 天每天 24h 不停地运作,而不会导致大阪居民失聪或失眠。在离大陆这么远 的地方修建机场,意味着水下建筑结构将不得不面对冲积土和洪积层的问题。人们 对冲积土的特性, 特别是在承受沉重负荷时的特性已经基本了解, 然而洪积层的特 性却还相对未知。这直接导致了在施工过程中遇到严重的问题: 整个岛沉没得比预 期更快更深、更严重的是整个岛并不是均匀下沉的。最终的结果是建设成本超支了 几十亿美元, 延迟一年后该机场才得以开放, 并且直至今天, 还不得不面对岛屿继 续下沉的险峻形势。对于这个问题,要求学生们查阅每一个关于这个主题的文件, 这本身就很具有挑战性,因为日本政府已经抹去了一切可得到的记录以避免公众恐 慌。老师要求学生一方面研究下沉这个难题,另一方面深刻反思问题的根本所在, 提出纠正的措施,并为未来岛内建设机场的设计指南提出建议。学生们给出了非常 简单但很出色的答案:正如加利福尼亚州 83 号公路沿线保护濒危鸟类的例子一样, 关西项目团队本应该首先根据机场的不同设计交通量规模计算可能产生的不同噪声 级别,了解在径向方向随着远离机场距离增加,噪声逐步减弱的情况,然后综合实 际测量、调查和计算机模拟的结果来确定白天和晚上人们可容忍的平均噪声水平 (例如, 纽约州的法律明确将所有噪声水平限制在 90dB)。有了这两组数据, 项目 团队就能计算出机场应该在大海中离大陆多远的地方。我们相信,这个机场的位置 本应更接近大陆,那就可能完全避免洪积层的难题。这个建议虽然看似简单,有马 后炮之嫌, 但是并非微不足道。相反, 日本团队如果早些采用这种方式, 就可以避 免后来遇到的那些巨大的困难。

为了更清楚地看到发掘 ITS 这一矿藏的各种可能途径,再来看下面的例子,这同样是一个真实的故事。如果有人驾车到达拉斯-沃思堡国际机场接一个来自国外的游客,他一般会把车停在国际抵达航站楼(D)并且步行到休息室,因为进站旅客在成功地完成了海关和入境手续后也会被引导到这里。海关和出入境大厅通往休息室的门外是一个显示国际航班和它们的预定到达时间的电子信息板。休息室里都是来接朋友和家人的人。过了一会儿,很多人会变得很不耐烦,因为航班到达时间过了好久,却还没有看见他们的朋友或亲人的身影。电子显示板从来不显示某航班是否延迟或降落等信息,因为这些信息在机场和航空公司各自的网站上一应俱全,但休息室中的等待者却看不到这些信息。再看另外一个例子,也是一个真实故事。琼斯先生乘坐从伦敦到纽约的航班,经过7h飞行到达肯尼迪机场,这时他非常想赶快回到市内的家中,但他不知道外面正在下大雪。他快速通过海关和出入境大厅,跑到航站楼外面叫了一辆出租车,结果又在路上耗费了8h,因为此时整个城

市正处于交通堵塞之中。因为刚刚抵达纽约,琼斯先生是不可能预先知道这个城市的驾驶条件的,但是当地的主要交通官员却是知道这些信息的。只要告诉琼斯先生这种情况,他完全有可能会选择步行到附近的旅馆先登记入住,或在机场贵宾室过夜,甚至只是在机场坐下来吃个漫长的晚餐,这样就不会遇到交通堵塞了。这两个简单的例子揭示了潜在问题的严重性:如今,信息技术(IT)正在迅速退化成盲目提供信息服务的手段,却没有考虑如何才能使这些信息被有需要的人更好地利用,反而可能误导人们,或者在错误的地点错误地使用了错误的信息。这反过来又为我们提供了一个巨大的机遇,即在那些熟悉交通信息及其含义的交通部官员和那些能获取最新信息并能确定及时呈现信息的最佳形式的IT专家之间建立宝贵的合作关系,给最需要的人提供尽可能多的信息,这样才能给社会带来最大效益,使信息真正地服务于社会。

2.3 挑战和机遇的案例

本节介绍了 ITS 的一些问题,在其范围不断扩大后,这些问题也许看起来非常有挑战性。由于人类具有创新精神,所以这些问题在未来一定会被解决。而一旦这些问题被成功地解决了,前景将是非常美好的。本节的目的就是要激发和引导我们找到正确的方向。

- 1) 今天我们制造和使用的所有汽油和柴油发动机,无论是在汽车、内燃机车、飞机还是发电机中使用的,均是起源于100多年前就发现的控制突释原理。除使用了合金等新材料、多了几个电子芯片和计算机芯片等几个微小改动以外,今天的技术跟100年前相比几乎没有变化。目前的技术反映的只是一个对原型逐渐改进后的结果。基本上来说,过去100年以内的新发现和发明还没有被人们真正地吸收消化,并融合到新的发动机设计中去。这意味着我们将有数不清的机会,可能设计出前所未有的具有特殊性能的新型发动机,这些发动机将具有全新的属性,并在效率上有飞跃性的提高。
- 2)人们从远古时代起就开始建设桥梁来横跨溪流、江河甚至是面积较小的海洋。虽然桥的设计原理多年来都没有什么变化,但如今我们的工程技术水平已经逐步提高,可以由计算机进行辅助分析(即有限元分析)。最新的研究成果包括利用桥梁上的应变计和相应的数据采集和传输技术实现动态称重功能(Chelsea White教授,2010)等。然而,到目前为止,所有的桥梁,无论大小、木质还是钢制、是否悬索桥等,都有一个明显的共同点:它们都是固定的和静止的,给人的感觉是被动的和无生命的,它们不能动态地适应环境的变化。虽然这可能听起来有点耸人听闻,但是我们没有理由不把智能化应用到桥梁上,智能化应能使桥梁感知和适应快速变化的情况,并能在一定程度上保护桥梁自身和使用者。正如现在很多计算机

处理芯片中,传感器不断地检测中央处理器(CPU)的热度,如果温度超过预定阈值,CPU 就会自动关闭一样,各种智能技术的组合运用就能帮助计算机芯片避免遭到毁灭性的损坏。

- 3) 电动汽车,尤其是同时可以有汽油驱动和电力驱动两种功能的混合动力电动汽车,由于具有实用性,正逐渐普及。在拥挤的大城市中使用这种汽车,一个主要优势就是它们基本是无污染的。在城市里,驾驶中的停停开开很常见,而使用电力驱动能消除传统汽油发动机由于长时间怠速而产生的废气。科罗拉多州的某区域评估测试(Jim Himelick, 2009)表明,插电式混合动力电动汽车(PHEV)在通宵充电的状态下,还能大大减少温室气体(GHG)的排放^[13,14]。当前未解决的问题是用电功耗的变异性,以及直接在燃油和电力消耗之间进行转换的难题。非插电式混合动力电动汽车基本上仍主要使用汽油,而那些完全由电池驱动的电动汽车主要从电网充电,它的主要能量来源可能是矿物燃料。从根本上来说,这种电动汽车可以帮助我们使污染远离城市而分散到乡村地区,因为那里建有发电厂,取电更为方便,但实际上整个环境中的污染总量仍然没有改变。当然,如果全世界能完全实现核能和水能发电,那么就会减少甚至消除二氧化碳、一氧化碳、二氧化硫以及燃烧矿物燃料所产生的其他有毒气体。因此,设计可以降低总排放量的车辆将是未来交通可持续发展的重点。
- 4) 在心理学领域有确切的证据证明人类缺乏对深度的精确感知能力,而这正是造成许多事故的一个主要因素,特别是涉及汽车和火车的事故。大部分在晚上发生而且涉及老人的车祸,很可能就是因为人们无法精确地判断一个在我们正前方的物体和自己的距离,不能确定它是在远离还是接近我们而造成的。在世界各地人口密度极高的城市,在繁忙的道路上,尤其是在晚上或能见度较低的时候,造成车辆与穿越街道的行人之间的交通事故的主要原因,也许就可以归咎于人类视觉系统的缺陷。随着世界人口的不断老化,这个问题只会变得更糟,这意味着在研究和缓解这一问题上我们将面临巨大的挑战。
- 5) 自从有了人类文明就开始有道路。罗马帝国的道路长达数百英里,德国的高速公路也有数千英里,而美国的州际公路已达数十万英里。修建道路的材料和方式已经有过千年的发展历史。铺面道路的定义就是在全部长度范围内进行了路面铺装的道路。世界范围内数百万英里的铁轨也都是按相同的规则铺设的。当然,公路的基础设施建设是按每单位长度计算成本的,可如今的公路建设成本相差极大。在美国,修建一条设计年限同样为 20 年的 4 车道州际公路,成本预算最低可以是100 万美元/mile,最高却可达 1 亿美元/mile。更重要的是,道路的维修费用不断攀升,甚至到了连美国政府都难以承受的程度。其结果是,在公路建设项目启动50 年后,现在美国的州际公路系统已经开始崩溃了[4]。现实情况是,随着使用年限的增加,混凝土路面出现了微小裂缝,而融化的积雪和雨水渗入裂缝,导致裂缝

不断扩大。随着时间的推移,裂缝越来越大,越来越宽,最后将路面分解成一小块一小块有害的碎片和碎石。由于对成千上万英里的路面进行彻底检查需要巨大的资金,交通官员不得不"一边打补丁一边祈祷",选择性对部分路面进行浅表层的修补。这种情况非常糟糕,因为它有可能会逐渐影响到这个国家在世界上的竞争力。我们面临的挑战是,能否从基础物理和数学研究出发,利用所有可能的先进科学手段,将道路基础设施建设和维护成本降低一个数量级,或者大大延长道路的使用寿命。目前印度和中国正在走美国的老路,投入巨资,努力进行道路基础设施的建设。那么他们在未来50年内就很可能面临同样的问题。如果在道路建设的早期阶段,他们能意识到这一点,并以一种全新的方式去思考这个问题,未来他们将会从中大大受益。

- 6) 一般情况下,汽车设计的经济时速为 100~120mile/h,但由于各种因素的影响,目前美国高速公路上的汽车速度普遍低于 60mile/h。美国交通部使用一种经验概测法测定车速,即流量为 10000 辆/h 的 4 车道公路上,假定前后两车车头间距为 3 个车身长,意味着此时的平均时速是 30mile/h。一方面,更高的速度意味着更短的旅行时间,这也意味着在商业上具有竞争优势,也能较大地改善生活质量;另一方面,美国高速公路沿线张贴的"超速危险"标志,提醒我们记住一个简单的物理定律,即速度越快,重伤甚至死亡的概率越高。在美国,超速经常被起诉和罚款,有时开太快甚至会被逮捕。在德国,高速公路没有车速限制,但政府总在要求不断改善路面条件,严格培训驾驶员以及优化汽车的安全设计参数。很明显速度和安全问题是相互对立的。我们面临的挑战是要尽力找到新的设计思想,提出非比寻常的设计原理,使汽车在不亚于当今安全水平的前提下能以 250mile/h 甚至更快的速度行驶。如果 1000mile/h 或者更远的远距离旅程能在 4h 内完成,那就意味着陆地上每个最偏僻的角落都会有不错的可达性,汽车出行,将能在除了需要跨越大洋的航线以外的地方,真正代替航空旅行。
- 7) 改善制动性能,尤其是改善在光滑路面或结冰路面上的制动性能,可以显著减少小汽车、卡车、火车、甚至飞机的事故发生率,这已是一个共识。在正常的干燥路面条件下,一些高性能赛车强大的制动系统可以使速度为 120km/h 的车辆在大约 60m 或 10s 内完全停下来。这里面临的一个严峻的挑战就在于如何基于对减速原理更深入地理解,发掘出一个颠覆性的制动原理,它能在确保驾驶员同样安全的情况下,将制动距离减少到 5m 或把制动停车时间缩短到 1s。这一发明将给交通工具带来本质上的改进,甚至可能避免很多交通事故。
- 8)全球范围内车辆交通事故导致的结果,包括了从轻微交通事故到轻伤、重伤,有时甚至会导致死亡。事故不但影响个人和家庭生活,还成为阻碍现代化进步的一大祸根。更重要的是,即使最先进的医疗诊断器械也不能百分百检测出那些看不见的和隐蔽的伤口。即使引入了安全笼、可拆卸的发动机组件、安全带、安全气

囊和其他先进器件,极大地提高了乘坐者的安全度,但据美国一名高级官员说,美国交通事故的死亡人数一直保持在每年4万人左右。其中一个主要的事故形式就是街道交叉口处的碰撞。另外,在行人安全方面,相应的改善工作一直进展缓慢。我们面临的挑战是需要更深层次地去理解造成事故的原因和事故的性质,并不断改进技术,以减轻或控制事故的发生。作为这一新思想的开创者,我们可以从这样的一个想法开始,即交通事故的发生是因为违背了基本的物理定律:两个事件不可能在同一时间占据同一个空间位置。但同时我们还需要仔细思考一个事实,即这个定律只适用于物质,而不适用于能量。

- 9)由于地球表面超过 2/3 的面积是被水覆盖的,所以永远都会有船舶等水上载运工具的存在。古时候,它们的行驶速度一般介于 10~15kn 之间,现代远洋货轮和核动力战舰以及潜艇的行驶速度一般是 20~40kn,特制的小型高速船只行驶速度可以达到 70~80kn 以上。这里提出一个似乎是不可能的挑战:设想出一种全新的设计思想,使远洋货船时速达到 400kn 甚至更高。如果我们真的能实现这一设想,我们的思维方式就可能发生根本性的转变。在合理的时间内到达地球上每一个角落将成为一天内就能实现的现实。由于地球的大部分地区是被海洋覆盖的,大部分的人口聚集中心也在沿海区域,而且世界上大多数水域是相互连通的,原有的运输路线,甚至是我们的生活环境都很可能发生质的变化。
- 10) 虽然个性化快速公交(PRT)的基本理念在 20 世纪 70~90 年代期间就被提出[17,18,19],瑞典和美国也对其进行了探索研究[20],但它尚未得到世界各地的政策制定者和运输官员的认可。在 PRT 中,旅行者一旦到达一个城市,就会在机场的指定车站乘上 2 座或 3 座的私人自动汽车,然后从城市中一系列地点中选择他(她)期望到达的目的地,包括酒店、会议中心、购物商场、餐馆、博物馆、政府部门和市中心主要办公楼等,这辆车沿着地面轨道滑行,并且会自动且安全地在环形导轨中进行切换,最终抵达目标车站。旅客下车后,这辆车可以马上被另一位已经等候在车站的旅客使用,也可以由算法引导空车回到最近的停车场。PRT 可以大大减少人口高度密集的城市中心区的交通拥挤。而我们面临的挑战是缺乏实用的PRT 架构,更重要的是缺乏科学的方法以协助决策者系统地分析对 PRT 网络的需求,决定是否在某一特定环境中建立 PRT 网络。但可以肯定的是,一定要制订详细的工程参数来协调和控制综合的异步式分散算法。
- 11)在商用飞机的发展初期,在确定事件和事故原因过程中,黑匣子的作用就已经是无可取代的了。它对于综合分析飞机的安全性能,重新设计飞行的控制和通信协议起到了很大作用。然而,在最近的空难中,黑匣子似乎越来越没用。原因也许来自两方面:第一个,如果空难发生在世界各大海洋上空,黑匣子往往沉入海底,很难找到。鉴于地球表面的 2/3 都被海水覆盖着,而许多国际航线都穿越海洋,所以我们需要调整黑匣子的设计思想,重新设计适于越洋航线的黑匣子。第二

- 个,也是更重要的一个原因,是因为今天的许多灾难事故发生得非常突然且迅速,以至于黑匣子的数据要么太少,信息价值不高,要么是驾驶舱声音记录的数据与事故调查完全无关,就仿佛全体驾驶人员都来不及应对这一威胁一样。和以前不同的是,今天的航空控制器由于使用了快速计算机和复杂的软件而变得更加复杂。当出现问题而导致飞机坠毁时,计算机本应能做出快速反应并加以控制,但却没有做到,所以有一点是明确的,就是这个问题是极其复杂的,超出了目前技术认知所能控制的范围。还有一个很大的可能性,就是目前的黑匣子设计无法识别并记录在正确的方案下飞机正确运行时的参数设置情况。对我们来说,开发一种全新的方法来确保飞行安全是一个挑战。这种新方法要更加注重信息的快速检测和威胁的自动消除,这样就不用总是强调事后事故分析过程中掌握飞行数据和驾驶员声音信息的重要性了。
- 12) 在错误的时间、错误的地点,过多的水会对生命和财产造成严重损害。由于上游降水过量,河流水位持续上升,可能会威胁到堤坝。当由于洪水、暴雨、风暴等导致积水或堤坝崩溃时,水灾会摧毁家园、农作物、社区和我们的生活。过去几千年间,每次遇到这些情况时,我们只能依赖万有引力的怜悯和其他自然力量逃过这些劫难。如何构造一种全新的思维方式,并利用各种科学原理整合成一个特别的方法,实现以惊人的速度排水,这对我们来说是一项艰巨的挑战。它的好处当然是不言而喻的,如果能做到,就能避免 2005 年的卡特里娜飓风造成的那种无法挽回的损失。

第3章 交通运输系统存在的基本问题

从根本上来说,交通运输系统就是实现物理位移的系统,即经过一个或多个中转站(如客货运枢纽或沿线各种站点等),沿着既定线路,把人和货物从一个地点运送到另一个地点。建设枢纽站不仅可以促进资源的共享,还可以为运输提供如加油等其他便利服务。因此,运输系统应该包括分散在各地的需要被运送的人和货物、客货运中转站等各组成部分在内。在电信业出现之前,人和货物的运输信息是随着要被运输的人或货物本身进行传播,或者是通过其他物理方法传播的,所以信息的传播速度就近似等于人和货物的运输速度。因此,从本质上说,人和货物的运输速度和相关的运输信息的传播速度是一样的。

交通运输的第一次革命与电信业几乎同时出现,它使人和货物运输信息的传播速度明显快于实物运输的速度。因此,一个现代化的运输网络从理论上可以看做是由两个主要部分组成:基于电磁能量传播的信息传播网络以及运送人和货物的实体运输网络。实体运输网络是实现运输必须具备的基本条件,而信息网络虽然不是必需的,但是它具有很强的实用性。

计算机的应用促进了交通运输系统的第二次重大革命,人们开始使用快速和精确的计算机,对整个系统的人和货物运输进行有效的控制和协调。在目前的运输系统中有效应用的例子包括铁路集中控制^[21]、空中交通集中管制、道路交通管理中心^[22]等。为了使这项工作简化并便于理解,协调和控制功能从一开始就被集成到集中控制单元中,至今为止这种做法仍然占据主流地位。早期由于计算机的昂贵成本及其庞大的体积,使人们不得不采用集中式控制。在集中控制模式下,首先要从被集中控制的单元系统的各个组成部分中获得相关信息,这些信息可能包括被运输单元的起点和终点,还可能包括有些单元在运输上的优先权、各中转点之间的运输路径及其被占用的状态,以及信息网络的状态等。然后,集中式控制单元会执行一个复杂的决策程序,其目的是使整个系统在符合预先设立的标准的前提下,实现高效运行。该程序为每个需要运输的单元逐个计算与运输有关的决策,最后,这些决策被传播到各单元中并依次得以实现。

理论上,集中式控制计算机作出决策的速度可能会很慢,尤其是在系统中单元数量很庞大的情况下。而且,逐个阅读所有远程单元的资料,再将相应的决策传达给每个远程单元,这些都可能会大大降低决策产生和实现的速度。较慢的决策往往意味着不精确和不严密。此外,集中式控制单元在遇到自然和人为灾害时更容易失效。在现实中,目前许多运输网络的组成单元数量都在急剧增加,人和货物的流通

速度不断加快,越来越迫切地需要更快、更精确的协调和控制。据估计,对于交通运输系统来说,即使使用当今最快的超级计算机进行集中式控制,可能也无法满足这种要求。最近,美国联邦航空管理局(FAA)提出一个"自由飞行"的概念^[23],力图改善航空安全,提高国家空域系统(NAS)的效率。这个概念就是要将 NAS 中飞行器和空中交通管制者之间的集中指挥控制系统转变为分布式系统,允许飞行员在实际操作中可以根据效率和经济的情况动态地选择适合自己的路线。为了实现安全的"自由飞行",美国联邦航空局必须建立引导系统来控制飞行路线。从根本上讲,目前的集中式控制方法要求针对组成单元 X,Y,…的决策是连续的,但实际上这不一定是必要的。如果与 X 运输有关的资源和与 Y 有关的资源相互独立,那么对 X 和 Y 的决策从理论上来说是相互独立并可以同步进行的。

今天,交通运输已成为一个正经历着第三次重大革命的交叉学科。这次革命可能是最复杂的——要从集中控制模式转化到异步、分布式控制模式,该模式能通过新计算机算法使快速计算和高性能计算机网络实现一体化。智能运输系统(ITS)^[24]学科中归纳了所有交通运输上所取得的进展,并向人们提出利用其基本原理设计新颖的有创造性的交通运输工具的要求,这将在未来得到全面的科学验证。

3.1 ITS 的主要特性

在未来, ITS 的重要性将与日俱增, 其主要特性将包括以下几点:

- 1)利用信息帮助人们消除和摆脱信息盲区:信息盲区是导致司机鲁莽驾驶、交通事故、路怒症(驾驶人因不耐烦前车或不满抢道而愤怒)以及车祸的最主要原因。当旅客在旅行途中遇到阻碍时,由于不知道将会发生什么,无论他们是否理性,都会担心不能按时到达目的地,这就有可能导致许多灾难性的后果。ITS 体系结构的最大贡献可能就在于能向处于信息盲区的旅行者及时提供相关的准确信息,驾驶员可以根据这些信息做出抉择,重新拟定出行计划。例如,某城市有一个公务繁忙的行政人员,名叫乔伊,他计划到附近的机场乘坐航班 XYZ。但由于太晚离开办公室了,乔伊在驾驶时非常慌忙急躁,差点撞倒一个行人。而当他刚好赶到机场时才发现由于飞机 XYZ 出现意外的机械故障,航班推迟了 2h。如果 ITS 可以及时获得航班延误的信息,并且自动把相关信息发送给每个 XYZ 航班的乘客,如果这种信息能传到乔伊的手机上,那么乔伊就基本上不会存在驾驶行为问题了。
- 2) 实现自动计算:现在,世界各地许多运输系统中的决策制定以及到达时间的计算仍然采用手工估算的方法。与之不同的是,未来的ITS 架构一定会采用自动的决策计算机系统来产生准确的信息,并实现精确地控制和协调。
- 3) 信息的准确性至关重要:信息必须是准确、及时、相关和一致的,否则, 驾驶员将会质疑 ITS 技术的价值,甚至可能会过早地放弃使用它。任何过时的数据

都可能导致错误信息的出现,从而产生无法预料的严重后果。在 ITS 中,像以下事例中出现的事件是绝对不允许发生的:在 2002 年 6 月,10 号州际公路上一个可编程的高架电子信息板提醒司机:菲尼克斯和图森之间的空气质量很差。阅读信息板的司机会发现电子信息板上这类信息内容相当丰富,但却没一条对驾驶员有用,因为他(她)正要离开凤凰城,向西开往洛杉矶。也就是说,这些信息不应该显示在这个方向,而是应该显示在对面的州际 10 号公路,即菲尼克斯和图森之间。

- 4) 尊重驾驶员的自由非常重要:除非在十分紧急的状况下,否则 ITS 体系结构不应该试图控制或命令司机的行为,原因有两个:首先,中央控制系统^[25]不可能确知每个司机的出行目标、目的或想法,因此,任何试图控制驾驶员的行为可能总是基于错误的假设。其次,从长远来看,驾驶员会讨厌自己做决定的自由被侵犯,这可能会最终导致他们放弃使用 ITS。实际上,ITS 的重要原则之一就是要向司机提供尽可能多的相关信息,给他们配备适当的网络和计算资源,使他们在某种情况下能够自己做出决定,并自由选择正确的行动方案。
- 5)需要更灵活和自由的选择方式:越来越多的出行者拒绝接受缺乏灵活性和个性化的服务,拒绝接受系统不准确的估计。他们非常反感以计算能力和网络带宽有限为借口。出行者对灵活、自由、个性化选择服务的需求正在不断增长,并且这一趋势很可能会延续到未来。
- 6) 需要精确、严密、实时的信息:传统的方法中,数据先被集中到一个中心单元并进行加工,然后,由此产生的信息再被发送到各地的出行者手中。考虑到地理距离以及电磁波有限的传播速度,当出行者得到关于运输过程中某单元的信息时,原始信息由于传播时间过长已经失效了。对于动态系统来说,这种延迟就意味着出行者收到的信息已经滞后,这种信息是不精确、不严密的。由信息的滞后性导致的信息错误程度与延误时间长短有关,而延误时间的长短又是与系统的动态特性以及精确度相关的。因此,每个运输系统都会有信息传输的滞后性,所以未来的系统架构必须将注意力放在分布式计划上,以消除一切不必要的可能导致滞后的因素,尽可能地实现高效、准确和及时的决策。
- 7) 个体出行者将成为关注的焦点: ITS 体系结构必须从根本上围绕每一个司机或出行者制订方案,当然必须首先服从所有可用资源的安全性和公平性要求。这也就是说,尽管策划者或运营者更关心一些总体性指标,如路面利用率、平均出行时间、网络可使用宽带和计算能力等,但还是需要把对个体出行者运行参数的关注放在首位。
- 8) 交通网络的一个基本特征:货物和人在实体运输网络中运输,和信息在信息网络中传输明显具有不同的含义。如今,某些实物在运输过程中可能携带了自己的计算引擎,这在当今社会中是非常有必要的。与之对应的,在通信网络中,信息单元是由纯电磁能量构成的,信息单元在传送过程中不可能携带自己的计算引擎。

而那些被货物或者出行者携带着的计算引擎,能随时动态地执行与出行相关的计算和决策。此外,由于采用了无线或红外线技术,运输系统的各组成单元在运输过程中可以互相交流,这就有望降低对集中式的信息采集和决策功能的要求。因此,人们正不断追求尺寸更小、成本更低、性能更高的计算机设计,以促使计算机在交通运输网络中得到更广泛的应用。

9)设计异步、分布式算法以控制、协调和管理资源:由于许多运输系统的组成单元和资源在地理上是分散的,所以未来的系统架构必然要采用分布式算法。在交通网络运输的各单元间,其运输服务请求很可能是相互独立的,而时间间隔也可能是没有规律的。因此,系统中的互动本质上应该是异步的,为了控制、协调和配置资源,必须设计异步、分布式算法。

设计出的运输系统的异步、分布式算法必须能确切反映系统的最高级元目标或 意图。该算法将在每个组成单元的行为上有所体现,它要使原有的开发系统发挥出 最大的潜力。此外,必须尽可能最大化地实现本地化计算,同时尽量减少实体之间 的通信,从而达到计算的高效性、鲁棒性和可扩展性。这种算法的主要特点如下:

- ① 实体的定义和识别:从控制、协调和配置资源的算法角度来看,实体构成了基本的决策单元,并决定了运输系统中的决策行为。从物理系统的角度来看,实体和它的自然组成部分有关,包括可用于运输的资源和待运的单元。每个实体必须是独立的,即在一切可能发生的情况下,它的行为是完全由自己决定的。每个实体与其他所有实体之间是相互独立的,因此只有它知道自己要做什么。除非一个实体告诉另一个实体自己要干什么,否则没有其他实体会知道它的行为。可以想象得到,一个实体在与其他实体互动的情况下,其行为的考虑因素中必然要包括自己和其他实体之间的相互关系及其影响范围。
- ② 实体的异步性质: 一般来说,系统单元的运输请求发出的时间间隔是不规则的。一个单元可以是单个出行者,也可以是为了某一特殊事件而一起出行的一群人。由于单元是在系统中不同地理位置零散出现的,它们互相并不知道对方的存在,运输的速度也可能不同,它们的目的地也可能不相关或完全不同,因此它们在系统中的交流也就是异步的。时间是实体自身行为以及实体间互相作用的一个重要组成部分。虽然每一个实体因为有各自的独立性,其运输时间可能不同,但是当一系列实体 E_1 , E_2 , …之间互相作用时,它们就一定会有共用的时间,即"通用时间",这正是它们之间相互作用的意义所在。通用时间来源于不同时间的最小公分母,它反映了所有互相作用的实体中最优的时间分配方案。然而,各实体的异步性还表现在这样几个地方:在实体 A 和 B 互相作用的地方,在它们每次互相作用的时间之间,A 和 B 都是独立和异步的。也就是说,对于 A,其相对于 B 的进展速度是不规则和不协调的,并且反映出对 B 的进展速度缺乏精确的了解,反过来也是一样的。但是,在同步点,A 和 B 的时间价值必定是一致的。

- ③ 实体的并行性质:由于每个实体都是独立存在的,那么除了与其他实体进行必要的互动以外,每个实体必然都是并行的。也就是说,它的执行进度和速度是独立于其他实体的,在运行过程中任意某个时刻,各实体所处的位置也是唯一的。在算法的实现过程中,每个实体都可以被映射到主机系统的某个并发进程中去。认识到实体的并行性是很重要的,原因有两个:首先,它反映了对一个实际运输系统的正确看法。正是以此为基础,我们才能说控制和协调算法更贴切地反映实际。其次,当算法在拥有足够计算资源的主机系统上运行时,并发实体也可在专门的计算引擎上同步运行,以实现更快的全局运行。更快的运行将节省大量时间,同时可针对不同的参数来思考系统设计的问题。
- ④ 实体之间的通信:显然,由于地理距离和有限的信息传播速度,任何一个实体都不可能得到它在每个连续运作的瞬间所需要的一切完整而准确的信息。因此,算法的一个重要组成部分就是通过实体之间的互相作用来实现数据和知识的共享。实体之间的相互作用,根据实体的不同性质可能有不同的形式。首先,每个实体与哪个或哪些实体相互作用是可以确定的,它们精确反映了相应的现实情况。在某种极端的情况下,任何实体可以与其他所有实体互相作用。第二,为了在互相作用的实体之间分享必要的数据和资料,就必须建立相应的信息结构。第三,只能在需要了解的范围内共享信息,以确保私密性和精确性。第四,假设所有信息通信都能得到保障,也就是说,一旦发送者传播信息,就能确保该信息被接收器接收到。因此,实体之间的数据和信息共享就意味着连接实体的通信网络也是信息分享算法的一个组成部分,而通信网络的拓扑结构则是由互相作用的性质决定的。
- ⑤ 算法正确性的证明:就其本质而言,控制和协调算法,通常涉及上百个自治实体,它们依靠非集中控制的智能体同步或异步地运行来控制实体的活动,这似乎很难为人们所理解。为了确保在该算法控制下能真正实现运输系统的准确性、正确性和安全性,就必须对算法的正确性进行严格的证明。基本上来说,对算法正确性的证明必须能确保系统正确运行,也就是说,执行的算法必须能准确地反映现实(如事故),不能和现实存在任何的不一致,同时还必须保证系统能实现唯一的目标。尽管在整个系统范围内从其他实体获取信息和数据的过程会受到滞后性的影响,但是由于每个制定决策的实体都需要利用一部分信息,因此正确性的证明显得特别重要。
- ⑥ 鲁棒性:我们一直期待异步分布式算法能成为一个高稳定性系统,不会像集中式系统那样容易受到自然灾害和人为因素的破坏。每一个地理上分散的实体都是一个决策单元,而一次灾害影响到所有实体的可能性是很小的。因此,即使有一个或更多的异步实体失效,其余的实体也很可能没有完全受到影响,系统仍可能在降级模式下继续发挥作用。为使算法在部分失效的情况下仍能运行,就必须在各实体及其异步交互的设计中充分考虑意外情况的处理。

- ⑦性能:在该算法的控制下,多个处理器的并行使用能大大提高系统性能。该算法能够利用系统中已有的并行处理器的程度就反映在它的性能指标上。尽管每一个运输系统都可能需要各自特定的性能标准,但还是存在两个可能同时适用于所有算法的标准。第一是"可扩展性"(将在第4章详细介绍),反映该算法继续扩展其功能的能力,也就是说,即使系统规模扩大,构成系统的实体数量增加,该算法仍能达到预期的性能目标。由于整个系统中决策层面的所有计算会被分配给系统中的每一个实体,因此,即使随着系统规模的扩大需要不断提高计算能力,但此时计算引擎的数量也会相应增加。假设通信网络按照一定的比例扩大,此时可用的计算能力和所需要的计算能力之比仅仅只会因受临界部分的影响而略有降低,这意味着其主要性能目标的实现几乎不受影响。第二,假设有一个机制能够确定所有特定的实际问题的全部性能,那么它反过来就可以作为评价任一算法性能的理想指标(第5章和第6章会详细阐述这个指标)。性能评价就可以在交通部门的投资和运营决策中发挥巨大的作用。
- ⑧ 稳定性:尽管算法是交通运输系统的组成部分之一,但在运营过程中,它很可能发生无法预料的变化。稳定性,是指在典型的不稳定的工作环境下算法将会有什么样的表现,这将在第7章中作详细说明。
- 10) 仿真是必不可少的环节:对于当今很多运输系统来说,规模的扩大和复杂度的提高意味着需要用大量的变量和参数来定义系统,对系统的价值观念的理解也会产生巨大的差异,此外,由于分析工作受到各种限制,还会导致在操作行为上也产生很大的不同。ITS 可能很复杂,而模拟和进行大规模异步分布式仿真或许是最合乎逻辑的,通常也是客观地研究 ITS 的唯一方法。模拟与仿真有很多优点:首先,在开发原型系统前,它可以用非常经济的成本检测出设计上的错误。第二,在模拟系统的运行过程中可以找出潜在的问题,包括一些不常见的或者是难以捉摸的问题。第三,对仿真结果的分析可以用于评价目标系统结构的性能,并找出提高系统性能的潜力所在。
- 11)对运行错误进行不断的检查:由于其本身的性质,复杂的系统往往很可能包含一些设计错误。这些错误在运行期间的出现是没有规律的,并且经常能逃过检测,但它们往往是非常有害的。为解决这一问题,ITS 架构必须不断地自动检查和接受用户的错误报告,并把它报告给系统管理层,这样才能及时修正和完善系统。
- 12)对 ITS 原则与社会规范之间的不一致性进行不断检讨:任何 ITS 架构都必须合乎一定的逻辑原则,而当它和已有规范出现不一致时,一定要仔细分析。例如,事实上,在整个美国,只要有高载客率(HOV)车道的地方,HOV车道一般都是在道路最左边的位置。而由于道路最左侧的车道一般都被设计成快车道,这就可能会造成一种错觉:HOV车道是一条快车道。另一件事也证明了这一点:通常

情况下,非 HOV 上快速行驶的车辆会使用 HOV 车道去超越快车道中速度较慢的 车。而 HOV 车辆, 为了进入这条位于最左侧的车道, 将不得不从位于右侧的匝道 汇入公路,然后穿过快车道上快速行驶的车流,才能进入 HOV 车道。如果这事发 生在一个拥挤的日子里,就会让人们误以为 HOV 车辆的司机打算开快车。很明显 这个假设是非常错误的, 因为许多 HOV 车辆装载的可能是一个集体旅行的家庭, 他们宁愿选择限速行驶也不想发生意外。第二个例子,许多公路的起源可追溯到工 会成立前的 1700 年。为了从城镇 S 开车到 T. 每个人都会被建议沿着公路 H 行驶, 因为公路 H 会将他直接引到一条主要的州际高速公路上。虽然 H 在地图上是一条 直路、除了那条州际高速公路之外、并没有与其他公路的交叉口、但是一名司机在 实际行驶时,竟然在三个不同的地点遇到分岔路。这些分岔路的一个共同特点是, 即使是沿着分岔路中最明显的支路前进,他(她)还是会感觉走错了路。事实是, 要想继续在 H 上行驶,就必须在分岔中比较不明显的那条支路上向左或右急转弯, 而不是直接前行。虽然分岔口上设有交通标志,但是几乎都已破损,或者被茂密的 植被所遮蔽。据推测, 这些支路的历史可以追溯到几百年前, 那时, 道路进入公路 H 段时就会转来转去以避免穿过居民房、教堂等。显然,数以千计的司机沿着公路 H 行驶时都会误时或迷路,过去 50 多年间累积的延误已经达到了数百数千小时。 如今已经很难去评判这种浪费,也很难合理地去判断使用 ITS 能不能改变这种状 况,这些问题都必须以科学、客观的态度去解决。

3.2 通过建模和仿真科学地验证 ITS 的设计

在以传统方式理解现实中运输系统行为的过程中,我们尝试了通过精确方程去拟合系统行为的分析模型,并成功地用数学方法解决了这些问题,这种方法还将继续有效地为其他许多学科服务。然而,对于当今很多交通运输系统而言,规模在不断扩大,复杂性也逐步提高,这意味着我们需要用大量的变量和参数来定义系统,对系统的价值观念的理解也会产生巨大的差异,此外,由于分析工作受到各种限制,可能导致在操作行为上也会产生很大的不同。加州大学伯克利分校的先进运输与高速公路研究所(PATH)的研究人员曾经为智能车路系统(IVHS)[26]提出了一个架构,其中,一辆或多辆汽车被组织成离散的车队,在某些特殊的车道上行驶。这种车道类似于 HOV 车道,使车队能以很高的速度在高速公路上行驶。当车辆进入到运输网络并确定其最终目的地后,系统就会通过网络给它分配一个常规路线。然而,当 PATH 研究人员想利用控制论原理建立一个体系结构的分析模型时,他们发现,这是一个含有大量不确定状态的复杂系统。将来可能遇到的系统预计将会更加复杂,这就意味着只有模拟和大规模的仿真才是最合乎逻辑的,而且常常是客观研究 ITS 的唯一方式。

模拟是指用计算机可执行的形式将一个系统表现出来,其根本目的是在主机上 进行模拟、尽量准确、真实地体现包括所有组成部分在内的目标运输系统的体系结 构。仿真是指在给定的输入条件下,在主机上对目标系统的设计进行模拟操作.并 收集和分析仿真结果。模拟和仿真的好处很多:首先,在开发出原型前,能够以最 少的费用检测出设计上的错误。第二,在模拟系统的操作过程中可以发现潜在的问 题,包括一些不常见的或者无法预料的问题。第三,对仿真结果的分析可以帮助我 们对目标系统架构的性能进行评价,找到使其性能提高的潜力所在。与过去不同的 是,如今的计算机运算速度和精确度都不断提高,这促进了高逼真度的运输系统模 型的发展,从而能快速产生相当准确的结果。反过来,这又可以使系统架构的设计 人员能很快了解到主要参数的变化对系统性能的影响。在某些情况下,甚至能实时 或更快地获得这样的研究结果,系统设计或许可以就此实现质的飞跃。在大多数情 况下、系统由于受到外部压力的影响、可能产生无法预料的变化、这些变化可以被 快速地模拟并得到相应的系统参数值,将参数值再次输入系统后,就能使系统成功 具备抵御外部压力的能力。简单来说,模拟允许玩"如果"的游戏,也就是允许 设计者和使用者构想各种假设、于是、未来就可以在低成本和安全的环境下对拟定 的解决方案和战略进行脱机测试。最后,新设计的性能指标可以很容易地帮助我们 更深入地了解系统的行为和性质。如何确定适当的交通需求输入方式的问题将在第 4~8 章中讨论。

第4章 DARYN: 铁路网络的 分布式决策算法

4.1 概述

铁路网络通常是由数千英里的轨道和大运量的机车组成。通常,为了提高效率并实现模块化处理,会把轨道分为一个个单元,每一个单元都由系统单独控制。因此,火车从位置 A 运行到 B 的过程中可能会通过多个轨道单元。在某段时间内,可能有两辆或更多的列车需要使用同一段轨道,但在某个确定时间,只能有一列火车可以占用这段轨道,因此铁路系统的主要目标就是给火车分配轨道,以避免火车碰撞,并实现对铁路资源的最优利用。

传统的方法利用了集中调度的原则来实现这些目标。在集中调度模式下,调度 员事先通过单处理计算引擎知道每列火车的目的地。此外,调度员会定期接收每列 火车的当前位置、速度和状态等信息,也包括系统中每段轨道是否被占用的信息。 调度员依次分析这些数据,并且根据费用函数计算每列火车接下来应该走的路线。 因此,是调度员控制了每条轨道的状态以及每列机车随后的运行轨迹,以达到系统 资源使用整体优化的目的。

图 4.1 所示的是一个铁路网,四段轨道 T_1 、 T_2 、 T_3 、 T_4 连接 4 个站 A、B、C、D。集中调度由图 4.1 中的 DPU 模块来完成,它通过实线分别与轨道 $T_1 \sim T_4$ 分别相连,其中,实线表示永久性的通信线路设备。每条线路上信息的流动都是双向的,即轨道状态输入给 DPU,反过来 DPU 给轨道下达新的指令。火车 R_1 从起点到终点站的运行过程中,会通过 DPU 建立一个临时通信线路(可能是一个无线电线路),一旦火车抵达终点站,此线路就会被取消。假设在某个瞬时,火车 R_1 、 R_2 、 R_3 分别占用轨道 T_3 、 T_1 、 T_2 。这时,图 4.1 中细实线所示的临时线路就表示火车和 DPU 之间信息的双向流动。每隔一段时间, R_1 、 R_2 、 R_3 和 T_1 、 T_2 、 T_3 、 T_4 都向 DPU 传送相关的信息。时间间隔是根据轨道的长度、列车速度和 DPU 信息处理速度、通信线路的延迟以及该决策所需要达到的精度确定的。DPU 依次读取、分析这些数据,并且根据费用函数计算 R_1 、 R_2 、 R_3 随后应该运行的轨迹。费用函数可能会考虑使用一组轨道的实际成本、列车的优先权和未来可能需要考虑的成本因素等。DPU 做出决策后,依次将这些决策传达给火车 R_1 、 R_2 、 R_3 ,并允许它们做出适当的运行决定。这个过程一直持续到所有列车到达最终目的地时,该系统程序才

会终止。为了简化该过程,时间间隔可以与列车依次到站的时间相对应。图 4.2 是描述集中调度算法的伪代码。

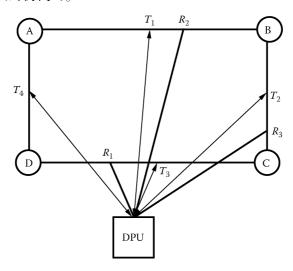


图 4.1 传统的集中调度方法

```
main()

{
    open file to read information on all stations;
    open file to read information on all tracks;
    open file to read information on all randomly generated train information;

interval = 1;

while not(all trains arrive at destinations) {
        for this interval, read speed and position of every train;
        for this interval, read status of every track;
        determine subsequent tracks for every train through cost function evaluation;
        permit the trains to move to the the next tracks;
        interval < - next interval;
        verify whether all trains have reached destinations;
    }
}</pre>
```

图 4.2 集中调度算法

Graff 和 Shenkin [27] 根据预先确定的目标集合描述了传统算法的仿真过程,其中,只要火车到达一个节点(例如一个中间站),算法就会立即计算后续的运行轨

迹。南太平洋运输公司也使用这个算法。作者们指出如果轨道和列车的数目分别在 15 和 25 以内, 那么该算法在 IBM 个人计算机上运行时, CPU 所需要的计算时间就 还在合理范围内。Fukumori^[28]提供了一个人工智能方法,利用时间表作为特定偏 序集来解决调度问题。传统算法的最新研究进展还包括列车、轨道与调度计算机之 间复杂的通信策略。Rao 和 Venkatachalam^[29]提出了一个实验系统,采用微处理器 来实现控制点之间的互锁、监测轨道电路,以及轨道电路与当前占据该轨道的机车 之间的互动等,其目的是降低事故发生的概率。Coll、Sheikh、Avers 和 Bailey[21]对 北美先进的列车控制系统 (ATCS) 进行了详细说明, 该系统中对列车的轨道进行 配置的过程是由调度程序单独控制的。ATCS 的创新性在于沿轨道嵌入了应答式计 算单元以监测所需的数据,并利用车载计算机执行由调度程序发出的分配指令,该 系统同时配有一个计算机之间先进的通信结构。Kashyap^[30]阐述了电子通信在印度 铁路轨道管理中的作用, Vernazza 和 Zunino[31]提出了一种铁路交通控制的分布式 智能方法,即整个轨道系统被分别分到几个决策中心 (DC) 中。当火车 R, 占用决 策中心 DC_i 管辖范围内的一条轨道时, DC_i 就负责 R_i 的调度。当 R_i 需要使用由另 一个决策中心 DC_k 控制的轨道时, DC_i 必须和 DC_k 协商 R_i 临时优先使用该轨道的 问题。参考文献「21-30〕中这些方法的一个主要缺陷就在于它们都只适用于集中 调度,这种调度模式对于大型系统来说太慢了,而如果是对一个实时系统,其控制 机构将是无效的。因为如果要将信息从每列火车及其行驶的轨道传送到调度中心, 一定会减缓通信速度,这是因为大量信号通过线路传播必然会产生延迟,同时,调 度程序读取每块数据也需要一定的时间。可以想象,这种延迟有可能会使火车毫无 必要地停靠在车站上[2]等待调度指令,甚至可能导致火车相撞。参考文献[30] 中的方法虽然比早期的算法有所改善, 但是它的局限性在于①能使用的决策中心数 量有限以及②缺少对实际情况的模拟和详细的仿真研究。

本章剩下的部分将提出一种新颖的方法 DARYN, 用来解决早期方法中的某些局限性。4.2 节详细介绍了 DARYN 算法以及在松耦合并行处理器上进行模拟的情况。4.3 节描述了 DARYN 在 ARMSTRONG 上的运行情况。4.4 节介绍了从建模、仿真到对 DARYN 的性能进行详细评价的过程。

4.2 DARYN 方法

4.2.1 算法

通过对集中式算法的分析,我们知道要使用调度程序计算每列火车的路线的主要原因,是因为在某个瞬间可能有两列或更多的火车需要使用同一条轨道。如果由完全独立的非协同的计算单元进行计算,就可能导致火车相撞。DARYN提出一种

试图消除碰撞的解决方案,其原理是在以必要的最小同步并行执行算法的自然实体中分配整个决策进程时,要求所有的决策保持一致性。这种情况下,才能最大限度地利用并行处理的优势。这个"自然实体"指的是在系统中确实存在的物理实体。

在 DARYN 算法中,会将决策过程分配给系统中已嵌入处理器的站台和机车。每列火车和每个车站都同时独立地执行自己的决策子任务,只要服从必要的同步,并不断保持一致性,那么整个系统的执行速度就会更快。与机车相关联的处理器处理整个网络的轨道布局。由于轨道实际布局改变的频率非常低,因此把车载计算机储存的这些信息作为其初始化的一部分是实际可行的。本书假设车载计算机总是存有整个轨道的布局,尽管存储量看起来似乎很大,但是只需要使用 32GB (还在不断增加)以上的 U 盘就可以很容易且经济地满足这个要求。假定大部分轨道布局是稳定的,当轨道布局由于故障、日常维修或其他情况产生微小的变化时,火车很容易就可以从邻近的站台计算机读取到这个变化。根据这一变化,火车 R_i 上的车载计算机 TC_i 重新评估费用函数,为 R_i 再确定一条优化路线。为了实现这个目标,车载计算机还必须和专门控制轨道的站内计算机相互联系,因为每条轨道都由一个独立的车站计算机专门控制,所以一个站附近的轨道实际上都由这台车站计算机控制。

初始化阶段,每条轨道都是空的,每列火车停在始发站,火车的运行路线还没 有确定。初始化后,每列火车完全了解整个网络的布局状况,包括轨道的位置和占 用轨道的列车位置,并且每台车站计算机也都知道自己控制的轨道的情况。火车 R_i 通过评估费用函数确定了自己的最佳路线后,向控制该最佳路线上第一条轨道 的车站计算机发出一条请求信息。该请求以消息的形式,由 R_i 发送到当前站 S_i , 如果 S_i 是 R_i 需要使用的这条轨道的管理者,它会对所有这样的同类请求进行检 查,根据先到先得的原则和优先权情况,制定解决方案。也就是说,是由车站来决 定所有要求使用某条轨道的火车可以进入该轨道的最小时间。只有和最小时间对应 的那条请求会被批准,其他的都会被拒绝。对原始请求的回应由 S_i 传给 R_i 。如果 S_i 不是火车要求使用的那条轨道的管理者,请求会被重新发送给正确的车站计算机 S_k , S_k 发出的回应通过 S_i 返回到 R_i 。 当火车 R_i 使用某条轨道的请求获得批准时, 火车就会沿该轨道向它的最优路径子集中的下一站前进。但是,如果请求被拒绝, 车载计算机必须重新评估次优路线,并且为申请新路线上的第一条轨道而重复上述 过程。如果一列火车的轨道请求在始发站或任意一个中间站被一再拒绝,车载计算 机还是得不断发出请求(这很可能是一个潜在的问题),直到火车获准占用下一条 轨道,它才能从它当前所在的车站开出。这样才可以保证决策的绝对一致性。当火 车到达下一站时,它又重复同样的协商过程。这里,每个车站的计算机都有两个功 能:①接受或拒绝列车占用它所控制的轨道的请求:②帮助火车向相应的车站计算 机发送路线请求、并反向回应。这个过程一直持续到所有火车都到达目的地为止。

因此,在 DARYN中,车载计算机只有当火车停靠在站台时才能发出请求。可 以想象、车载计算机与车站计算机通过光学、红外线或其他方式互联互通、火车一 进站,该通信过程就会启动。由于火车可以高达 200km/h 的最大速度行驶,而一 般车站的长度为200m, 所以站台与列车之间最小接触时间约为3.6s, 它至少是火 车与站台的数字计算机协商所需的 CPU 时间(毫秒)的几倍。因此,大多数情况 下,火车与车站计算机在协商过程中甚至不用真正地停在站台。此外,一旦原来计 算的最佳路线中某条轨道的使用请求被拒绝,车载计算机必须在火车驶出站台范围 之前重新计算出一条新的最佳路线。从逻辑上讲,有人可能会赞成在火车起动之 前,允许它请求为其预留最佳路线中两条或更多的连续轨道。虽然这种做法可能对 制定有效的长期规划更有利,但是也可能会由于太早保留轨道,或者由于轨道被过 早锁定,而影响到其他火车的最优路径计算,从而导致系统的整体性能降低。因 此, DARYN 算法只考虑未来, 并要求只有立即要使用的轨道才能预留。此外, 虽 然一般情况下费用函数的性质会比较复杂,但在本章中,我们假定了一个简单的费 用函数,即如果每条轨道的使用成本相同,在它上面行驶的每列火车就有相同的优 先权。假设火车网络的坐标是一个 X-Y 坐标,其中一列火车要在 X (水平) 和 Y(垂直) 方向上分别移动 $M \times N$ 个单位,费用函数就会强制火车在 $M \times N$ 值较大 的那个方向上先移动。当M = N时,任意选择X或Y方向都可以。

DARYN 算法中的一个必要条件是每个车站和火车都要配备有与一个绝对标准时钟同步的时钟。在初始化时,所有时钟都必须同步重置为 0。这样就确保了所有请求、回应和决策在时间上的一致性。

图 4.3 的网络是 DARYN 算法的一个例子。在图 4.3 中,A~D 的车站通过轨道 $T_1 \sim T_4$ 连接。假设 A 的车站计算机控制轨道 T_1 和 T_4 ,而 B 和 D 的站台计算机分别控制轨道 $T_2 \sim T_3$ 。而 C 的站台计算机不控制任何轨道。车站计算机之间通过沿着轨道铺设的通信线路进行通信。假设,在某个时刻,系统内有 4 列火车 $R_1 \sim R_2 \sim R_3$ 和 $R_4 \sim N$ 车 R_1 和 $R_2 \sim R_3$ 为别在 t=0 和 t=5 时到达站 A,且它们都要开往 C。火车 R_3 和 $R_4 \sim N$ 为别在 t=0 和 t=5 时到达站 C,且它们都要开往 A。DARYN 中,假设火车 $R_1 \sim R_3 \sim R_4$ 确定它们的最佳路线分别为 $\{T_4, T_3\} \sim \{T_2, T_1\} \sim \{T_1, T_2\}$ 和 $\{T_3, T_4\}$ 。因此, $\{T_4, T_3\} \sim \{T_2, T_4\} \sim T_4\}$ 的信息给站计算机 A 预留轨道 $\{T_4, T_4\} \sim T_4\}$ 的信息给站计算机 B。这两个请求都获准,然后火车将会沿各自预定的轨道移动。假定轨道的长度和火车的速度是: $\{T_4, T_4\} \sim T_4\}$ 不自己时间到达站 D, $\{T_4, T_4\} \sim T_4\}$ 的信息给站计算机 B。在 $\{T_4, T_4\} \sim T_4\}$ 不自己时间到达站 D, $\{T_4, T_4\} \sim T_4\}$ 不自己时间到达站 D, $\{T_4, T_4\} \sim T_4\}$ 不自己时间到达站 B。在 $\{T_4, T_4\} \sim T_4\}$ 不自己的路线,假定它决定向 A 发出一个 $\{T_4, T_4\} \sim T_4\}$ 不可轨道 $\{T_4, T_4\} \sim T_4\}$ 不是一直被占用直到 $\{T_4, T_4\} \sim T_4\}$ 不是一直被占用直

A 请求 T_1 轨道。这个过程一直持续到 t=6,此时 A 批准 R_2 的请求。然后, R_2 会在 t=8 时经由 T_1 轨道到 B。类似的,在 t=5 时,列车 R_4 向 D 发出一个请求 T_3 轨道的信息,然后请求被批准。因此, R_4 通过 T_3 向 D 移动,然后会在 t=8 时到达 D。同时,列车 R_1 在 t=7 时到达 D,并且向 D 发出请求轨道 T_3 的信息,这个请求被拒绝。由于 R_1 没有选择,它不得不在节点 D 等待,直到在 t=8 时获得 D 发出的可用轨道 T_3 的批准。这个过程一直持续到 4 列火车都到达目的地为止。可以看出路线决策是由各自的车载计算机决定的,而轨道分配是由站台计算机执行的,因此,该算法的处理能力会明显高于传统算法。

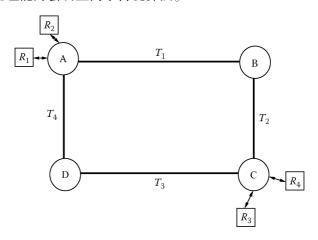


图 4.3 DARYN 算法示意图

决策的计算过程相当复杂,由于 DARYN 将决策任务分发到铁路网的每一个物理实体(机车及车站)中,并在各自独立的处理器上进行计算,所以整体数据处理能力会大大高于在单处理器上执行的传统算法。此外,在 DARYN 算法中,火车和所在车站之间的通信速度是最快的,而和邻近车站之间的通信是最差的。因此,DARYN 算法的总通信时间会远低于每列火车和轨道都将信息发送到固定的调度计算机的传统算法。最后,随着火车和轨道数量的增加,该系统规模也会增加,计算的要求和计算引擎的数量也会随之增加,但是 DARYN 算法中总CPU 时间的增长速度很可能会比相应的传统算法要慢得多,这体现了 DARYN 算法的极佳性能。

4.2.2 避免死锁的依据

DARYN 是一种绝对不会有死锁现象的异步、分布式算法。

多列火车可能同时请求争夺同一条单轨,所有这些请求最终被传送到单独控制 这条轨道的站台计算机。当收到一个或多个请求时,站台计算机必须根据火车的到 达时间做出决定,然后在有限的时间内为每列发出请求的火车生成并传出指令—— 批准或拒绝。站台计算机的回应是根据它已经收到的且尚未处理的请求总数计算 的,它在系统中是独立于其他所有实体的。因此,此时站台计算机的决策过程是 集中式的,而不是分布式的。站台计算机只是在有限的时间内计算与火车之间的 来往信息。在起终点之间,每列火车会不断地向车站发出请求并等待回应。在从 车站收到相应的回应之前,火车不会向任何车站发出第二次请求。每列火车请求 使用某段轨道的决策也是独立于其他火车和车站计算机的。在从车站计算机获得 使用某条轨道的授权后,火车在其行驶期间将成为这条轨道的独占者。因此,这 个过程中完全不存在对循环的依赖性,这样,一方面消除了形成死锁的可能性, 另一方面也同时确保了安全,因为这样可以保证两列或更多的火车不会在同一条 轨道上行驶或相撞。此外,由于系统内火车数量有限,并且每列火车在起终点之 间都会经过有限量的轨道,所以 DARYN 算法保证了每列火车都能在有限的时间内 抵达终点。这个证明是建立在假设每台站计算机只处理有限量火车请求的基础 上的。

4.2.3 在松耦合并行处理器上模拟 DARYN

为了详细分析 DARYN 算法的属性,我们在一个松耦合并行处理器 ARMSTRONG [32] 上对该算法进行模拟。这是个很合适的测试平台,因为它的结构——建立在特定通 信协议基础上的并行处理器——和列车与车站计算机之间在地理上的分布状况极为 相似, 并行的列车和车站计算机正是通过特定的信息进行通信的。ARM-STRONG^[32]有 68 个高性能的 MC68010 (10MHz) 处理器, 它们在一个超立方体结 构中通过高速通信线路连接。每个处理器可以同步或异步地执行应用程序。此外, 独立处理器上的执行程序可以通过明确的、高层次的通信元进行通信。在这项研究 中,某个给定的铁路网络上,每台车站计算机在 ARMSTRONG 中都表示成节点。 两台车站计算机之间的信息渠道是通过 ARMSTRONG 上相应节点之间的软件协议 来模拟的。在理想的情况下,人们希望系统中每台车载计算机都能表示成 ARM-STRONG 处理器,这就必须在车载计算机和列车目的车站的站台计算机之间动态地 建立协议。同时,车载计算机和前一站的车站计算机建立的协议必须删除,因为很 可能永远都不会再使用它。虽然在理论上 ARMSTRONG 支持协议的动态分配和重 新分配,但是这样的分配结果往往是不可靠的,所以无法使用。在此研究中,车载 计算机理论上是用迁徙过程表示的。当一列火车 R, 停留在站 A 时, 相应的车载计 算机(对应于 ARMSTRONG 上相应的节点)与车站计算机的计算进程同步执行决 策过程。当火车从 A 移到 B 时,上一节点的进程被禁用,车载计算机将各种重要 参数以消息的形式传达给 B。在 B 上,通过相应的 ARMSTRONG 节点,启动车站 B 计算机,同步执行一个新的可执行程序。车站计算机的计算进程是由R,传达给B 的参数决定的。于是,列车不断地在 ARMSTRONG 节点(对应列车行驶的站)上重复这种行为,直到到达目的地。也就是说,列车的参数一直在不断更新,并随着列车的运行,不断通过网络在节点间传送。这个模型的缺点是,车载计算机有可能无法通过独立、并行的 ARMSTRONG 处理器来模拟。此外,代表车站的节点必须将它的计算能力按车站和车载计算机分为两部分。ARMSTRONG 节点的这种双重身份明显降低了 DARYN 算法的性能。理想模式下的 DARYN 应该是能在 ARMSTRONG 并行处理器上控制每列火车。

在每一个对应于铁路网中车站的 ARMSTRONG 节点上,都需要存储整个网络的完整信息。当一列火车到达该车站时,车载计算机程序就会读取这些信息。而每台车站计算机则完全了解系统中每条轨道的真正占用者,这是基于这样的假设:轨道布局的变化频率非常小,并且任何这类变化的信息都可以迅速地通过网络进行传播。车载计算机发出的轨道请求,最初传达到火车当前所在位置的车站计算机中。然后,该消息被传送到实际控制火车希望使用的那条轨道的车站计算机上。在收到两个或更多的轨道使用请求后,车站计算机要判断哪个是最早请求时间,并且批准与最早请求时间对应的那列火车的请求。此时,所有其他轨道使用请求都被拒绝。如果有不止一台车载计算机符合最低请求时间条件,车站计算机将任意批准它们其中一个的请求。

由于 DARYN 是异步的,因此每台车载计算机都有自己的时间计数器。最初,所有的车载计算机启动时,它们的时间计数器都初始化为 0。随着模拟的进行,车载计算机会将计算火车请求的时间和火车移动的时间增加进来,火车在某条轨道上所花费的时间就等于该轨道长度除以火车的速度。因此,异步算法的优点就是能并行计算,同时简化了同步大型系统中由于存在大量火车所产生的复杂性。

在 ARMSTRONG 中,通信过程采用无闭塞基元。一旦建立了一个协议,数据在节点间的传送过程如下:发送节点通过 ipc_send(information_path,data_1,data_2,…,data_n)命令将数据发送到接收节点。发送程序会立即从收到信息的操作系统获得响应,如果信息传送成功,发送节点会收到一个 1,否则,就会收到 0,意味着传送失败。接下来发送程序会再次尝试发送信息。由于该系统是异步的,接收节点可能不知道发送者发送数据的确切时间。每隔一段时间,系统就运行一次 ipc_select(information_path)命令,检查数据是否一直沿着指定的路径传输。如果是,接收节点就可以通过 ipc_receive(information_path,data_1,data_2,…,data_n)读到数据,否则,接收者就不能执行 ipc_receive 命令。ipc_select 也是无闭塞的,也会立即得到正面或负面的响应。

为简化问题,将轨道放置在X(水平)和Y(垂直)方向,节点放置在长方形矩阵中,这确保了每列火车的轨道请求算法尽可能简单。每列火车首先通过计算

X-Y 坐标系中当前位置和最终目的地之差来确定它的轨道请求。如果 Y 方向离目的地的距离大于 X 方向,火车会试图在 Y 方向移动,从而接近最终目的地。类似地,如果 X 方向离目的地的距离大于 Y 方向,火车就会试图在 X 方向移动。就这样,火车找到合适的轨道请求后,将其提交给控制该轨道的相应的节点处理器。一旦收到确认或拒绝的回应,火车就会考虑下一步动作。如果火车请求得到批准,它会沿着批准的轨道移动。如果火车的请求被拒绝,它会重新计算第二个轨道选择,并再次提交请求。火车的第二选择是基于它试图移动的方向,而不是它的第一选择。这个过程一直持续到列车的某条轨道请求被批准为止。

ARMSTRONG 系统的仿真也是异步的,很有可能在同一时间从不同的节点发出两个或更多的信息,但是它们到达目的地的时间却有很大差异。这种现象导致在仿真中引入了"时间"问题,可以这样描述:假设,如图 4.4 中的铁路网所示,两列火车都向前驶向纽约站。火车 A 在轨道 2 上行驶,而火车 B 在轨道 3 上行驶。假设抵达纽约站后,A 和 B 都希望在轨道 1 上继续行驶。在现实中,可能会发生下列情况。轨道 2 比轨道 3 长很多,假设两列火车以同样的速度行驶,火车 B 会比火车 A 先抵达纽约站。因此,纽约站的站计算机很可能会授予火车 B 轨道 1 的行驶权限。然而,在仿真中,没有模拟行驶所需的时间,因为希望测量的只是决策过程所需的 CPU 时间,这样就可能会发生下列问题:包含火车 A 参数的信息比火车 B 先到达纽约站。也就是说,信息在 ARMSTRONG 中是异步传播的,火车 A 的"电子"的速度可能比 B 快,因此,纽约站的车站计算机可能会先赋予火车 A 使用轨道 1 的权限,这就会和实际情况不一致了。但我们要强调的是,这个问题纯粹是由于 ARMSTRONG 的硬件限制而引起的。

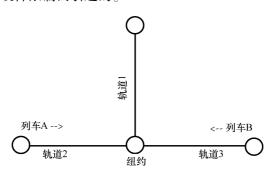


图 4.4 分布式仿真中一种潜在的时间问题

为了解决这个时间问题,火车从一个站到下一站所需的时间是通过按比例延迟来模拟的,因而,火车参数实际传播到下一个节点的时间延迟基本上与在 ARM-STRONG 上的时间延迟相等。

4.3 在 ARMSTRONG 上执行 DARYN

本节介绍在 DARYN 执行过程中不同的功能单元的数据结构及其伪代码。

每个模拟站台的 ARMSTRONG 节点中都包含着一些独一无二的信息,如节点的唯一标识号(ARMSTRONG 操作系统分配给节点的标识符)、与该节点连接的轨道数目、节点间的连接协议以及用于模拟现实计算的延迟时间等。这些信息都存储在数据结构 node_info 中。每个节点也在通信路径中存储信息,如它们的唯一标识号,即 ARMSTRONG 操作系统分配给它们的标识符,以及两个链接列表。第一个链接列表包含输入此节点的全部协议;第二个链接列表包含此节点全部输出协议的详细信息。

假设在某次模拟执行过程中,车站计算机控制它西边和南边的所有轨道。可以注意到,由于轨道网络是长方形的,有的车站计算机可能不控制任何轨道。车站计算机控制的每条轨道上的每个节点都有两个列表,第一个列表包含当前对这个轨道的预定请求,这个列表在每个运行周期中都要进行处理,然后被清除;第二个列表存储当前占用该轨道的火车(如果有的话)的唯一出入口。

最后,在通信过程中节点使用了两种结构。第一种结构对应于任意两个节点之间的消息。不论消息的类型如何,ARMSTRONG都要将每条信息扩展到指定的标准长度。由于仿真中要使用的信息类型不止一个,所以数据结构必须是所有这些信息类型的超集。可以想象得到,两个节点之间的消息很可能包含由于类型的差异而造成的一些空白字段。第二个结构对应于每个节点的输出缓冲区。由于使用了无闭塞通信基元,信息传输有时候可能会失败。在这种情况下,必须将信息存储在输出缓冲区中,以便于稍后尝试重新传输。缓冲区保存信息直到传输成功,此时将其复件从输出缓冲区中删除。

该程序也用一个数据结构来存储车载计算机的参数。假定火车具有某些参数,那么当它通过 ARMSTRONG 的节点时,每个节点就会拥有关于目前停在该节点的火车的各种参数。

ARMSTRONG 处理器的外部主机对模型进行初始化,并在每个相应的 ARMSTRONG 处理器中加载可执行程序的副本。在"main"函数中,节点首先读入由主处理器传送给它的数字标识符。然后,各节点都要执行函数"initial",从外部输入文件中读取信息。函数"initial"要执行两个动作。第一个动作是与相邻节点建立适当的通信路径。为了达到这个目的,它要使用一个函数调用子程序 initialize_protocols。该子程序从输入文件中读取所有相邻节点的标识符,然后调出 ARMSTRONG 操作系统给相邻节点分配的标识符,并进行比较。根据假设,如果一个节点的标识符数值比相邻节点的小,该节点就要发送路径标识符给相邻节点。相反,如果它的标

识符数值比相邻节点的大,该节点就要等待接收从相邻节点发来的路径标识符。这个过程为以明确的、一致的模式建立协议提供了保障。第二个动作是,用函数 initial 读取该节点其他方面的相关信息,如邻近轨道和来自这个节点的火车(如果有的话)。这个过程的伪代码如图 4.5 所示。

```
main()
  initial(train_array);
                                   initialize protocol connections, node
                                   information, & originating trains
  k_get_uptime(&starttime);
                                      initialize starting time of simulation
  simulating = 1;
  while (simulating = = 1)
     if (counter = = end_value) stop simulation;
     train function (train array):
                                          compute train requests for this track
     read_inputs;
                                 scan input protocols for incoming messages
     track_function;
                            process reservations for all tracks controlled by node
     clear buffers:
                        send output messages to other nodes
     traveling_trains;
                         verify whether trains have arrived at the node
initial (train_array)
  open external input file;
  initialize_protocols;
  read group, node identifier, track coordinates and update structure node_info
  read track direction, identifier, length, nodes connected to track for every track
          and update sub-structure node_info. tracks_connect[ti].
  if (track direction = = 3 or 4) update sub-structure node_info. tracks_control.
  read train destination, speed for any train originating at node & update the
          link-list train_array.
```

图 4.5 主程序和初始化流程

初始化程序完成后,每个节点都启动一个循环执行过程,该过程将在模拟结束时终止。这个循环嵌套在一个 while 循环中,有五个函数: read_inputs, train_function, track_function, clear_buffers 和 traveling_trains。接下来详细介绍这些内容:

1) read_inputs 函数:这个函数(见图 4.6) 不断检查所有输入信息的输入协议。它利用一个 for 循环语句不断扫描输入协议的链表。循环的第一步是通过一个 ipc_select 命令来检查协议中未确认的数据。如果结果是正确的,就通过 ipc_recv 命令读取消息。然后,将消息按不同类型进行分类。消息类型分为三种:①另一个 节点上的火车发出的轨道请求;②从此节点上某列火车发出的请求被接受或拒绝的信息;③进入该节点的火车信息。根据消息的不同性质,调用不同的子函数执行进一步的处理任务。这些子函数包括:

图 4.6 read_inputs 流程

- ① add_request:这个子函数负责增加一个由该节点之外的火车提出的请求,要将该请求添加到适当的轨道预留列表中。它通过一个 for 循环语句对由节点控制的所有轨道进行扫描,并将新请求增加到预留链表的末尾。
- ② receive_confirmation:如果从其他节点收到回应,这个子函数就负责更新当前节点上的火车的被确认状态。这里用一个 while 循环扫描火车的链表,并修改相应的确认域。
- ③ new_train:这个子函数负责将列车的参数增加到当前节点上火车的链表中。首先用一个 while 循环确定火车链表的尾端,然后把新的火车参数加到尾端上。

当读取并正确处理了所有新信息后,节点需要等待一段延迟时间 comm_delay。这个延迟就是前面讨论过的通信延迟。延迟过后,每个节点再次为新信息检查每个输入协议。如前所述,如果结果是正确的,就根据信息类型再进行分类。

2) train_function 函数:这个函数(见图 4.7)模拟在当前节点的火车行为。

利用 while 循环扫描火车链表,检查每列火车的确认状态。确认状态有三种可能值,对应于不同的值,依次执行不同的子函数。

```
train_function( train_array)
     initialize train_ptr to the dummy entry of train link-list, train_array;
     initialize prev_ptr to the pointer prior to the train_ptr;
     while (searching through entries, if any, in the train link-list)
        if(train has not completed its travel) {
            call check if done to verify whether train has reached destination;
            if (train not located at this node)
                  switch(train's confirmation status)
                    confirmation = -1: train is ready to compute its first choice
                                    call compute_1
                                    call send_request to submit request for track
                    confirmation = 1 train is ready to compute its second choice
                                    call compute_2
                                    call send_request to submit request for track
                    confirmation = 2: train has received confirmation for a track
                                    call move to simulate propagation of train
  advance train_ptr and prev_ptr to subsequent items in the link-list;
```

图 4.7 train_function 流程

- ① 确认值为-1:这表示火车的最近期的次优选择已经被拒绝了。因此,火车必须重新计算它的第一选择,并且再次提交它。子函数 compute_1 按照前面叙述的步骤重新计算火车的最优轨道请求。轨道请求重新计算后,调用子函数 send_request再次提交请求。显然,如果火车请求使用的是该节点西边或南边的轨道,当前节点就控制这条轨道,所以就在该节点处提交请求。如果火车请求使用的是该节点北边或东边的轨道,该请求就会被转送到控制所请求轨道的节点中。如果 send_request 已经准备好向另一个节点发送请求,就会调用子函数 send_message_to_buffer,将轨道请求发送到接收节点输入缓冲区的链接列表中。
- ② 确认值为1:在这种情况下,火车的最优选择已被拒绝。因此,必须重新计算次优选择并再次提交请求。子函数 compute_2 计算火车的次优选择,并且通过执

行子函数 send_request 将次优选择中的轨道发送到正确的轨道预留列表中。可以想象,在只能单向移动的地方,火车不可能有次优选择。此时,火车不会计算次优选择的轨道请求,而是继续不断地、定期提交它的最优轨道选择请求。

- ③ 确认值是 2: 这表示火车的轨道请求已经得到确认。执行子函数 move 和函数 send message to buffer,将火车参数传到后续节点的输入缓冲区中。
- 3) track_function 函数:这个函数处理轨道预留列表中的轨道预留信息。首先,确定某火车希望使用的轨道目前是否已被其他火车占用。如果是已被占用,将告知该火车不能使用该轨道,直到占用轨道的火车开走了为止。当轨道可用时,确定预留列表中的所有请求中的最早请求时间,并且批准对应该时间的火车使用该轨道,同时拒绝其他所有请求。于是,轨道预留列表中所有请求所对应的火车都获得了它们的确认状态。这里用一个 while 循环来扫描火车的链接列表。如果火车在当前节点上,就更新火车的链接列表。如果火车在另一个节点上,就将正确的确认或拒绝信息传送到对应的节点中。鉴于轨道请求信息中是包括返回路径的,利用这些信息,并使用子函数 send_message_to_buffer 就可以将确认/拒绝消息发回给请求者。
- 4) clear_buffers 函数:这个函数(见图 4.8)负责清空输出缓冲区中任何未确认的、未处理的信息。这里使用一个 while 循环对输出协议链表中的每个输出协议进行扫描。如果输出缓冲区中有一个消息是未被确认的,就利用函数 ipc_send 发送出去。当消息发送成功,即返回代码值为 0 时,该消息就会从输出缓冲区中删除掉。如果传送失败,该信息将会被继续保留,以便未来再次传送。

```
clear_buffers()
{
    while(processing every entry in the output buffer) {
        send message on appropriate output protocol;
    }
}
```

图 4.8 clear_buffers 流程

5) traveling_trains 函数: 当火车被允许使用它选择的轨道时,它会沿该轨道驶向后续车站。然而,火车的参数传递到下一个节点时肯定会延迟,而延迟的时间长短与轨道的实际长度与火车的速度之比成正比。这个函数(见图 4.9)以下列方式达到预期效果:火车的原始参数,会以消息的形式原封不动地传到下一个节点,但是,直到火车实际在节点的等待时间达到了延迟量乘上一个比例系数后的值时,火车才会计算后续的路径。也就是说,当火车的参数到达后续节点时,该函数首先读取 ARMSTRONG 节点计时器上的时间值(例如 v_1)。稍后,当计时器的当前值 v_1 超

过 v_1 的延迟量乘上系数后的值时,节点才记录列车的存在信息并允许列车重新计算前进的路线。

图 4.9 traveling_trains 流程

DARYN 是用 C 语言编写的,长度大约是 1500 行。这个程序在编译器优化指令 "-O4" (4 级) 下的 SUN3/60 工作站中进行编译,并且在 ARMSTRONG 系统中运行。

4.4 DARYN 的性能

本节将详细介绍在 ARMSTRONG 系统中做过的许多实验,以评估 DARYN 算法的性能。在实验中一共有 4 个铁路网络,分别包括 4、6、9 和 12 个站点,这些站点都被模拟为 ARMSTRONG 的节点。要模拟更大些的网络是很困难的,因为ARMSTRONG 的硬件限制了协议的数量以及同时打开所有节点相关文件的指针数量。这 4 个网络如图 4.10 所示。本节还将介绍 DARYN 的局限性。

为了评估 DARYN 的性能,我们提出了三项措施。首先,将模拟过程所需的 CPU 时间与传统的集中式方法需要的 CPU 时间相比较,根据不同的网络规模、列车数量和列车速度来计算加速因子。加速因子是指完成分布式仿真所需的 CPU 时间与单处理器仿真所需的 CPU 时间之比。由于 DARYN 是异步的,对终端的检测过程会非常复杂,所以我们可以这样近似地认为:系统中可能会有些行驶路线特别长的列车,它的速度是专门设定的,以确保其他所有列车都能比它先到达目的地,我们以下称其为专列。所以,完成整个过程所需的全部 CPU 时间是由这种长线路专列在线路上运行所花费的时间决定的。某专列从一个处理器开出,到达目的地,也

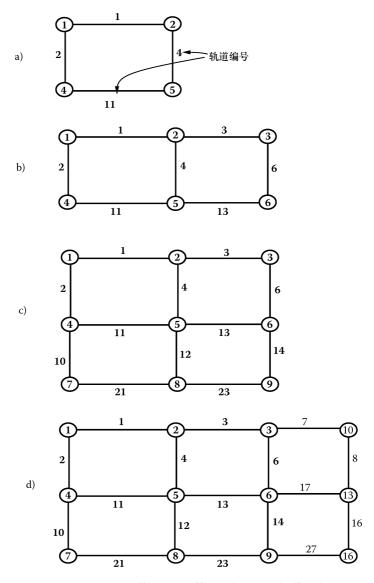


图 4.10 用于评估 DARYN 算法性能的 4 个轨道网络

就是 ARMSTRONG 中另一个节点。我们可以将火车开出时起始节点的计时器的值作为火车的起动时间参数 start_time。虽然节点是异步的,但是它们的计时器都在初始化时复位到零,并且假设它们的频率差异非常小。start_time 和专列到达目的节点的时间之差,就是所需的 CPU 时间。

第二项措施,是基于这样一个依据:如果 DARYN 算法具有可扩展,那么一列

火车以固定的速度运行固定的距离所需的 CPU 时间,就不会明显地受到整个系统内火车数量和网络规模的影响,因为网络规模和火车数量的任何增长,一定会伴有相应比例的计算引擎数量上的增加。CPU 的运行时间可能对系统中运行的火车数量有些影响,但不会影响车站的数量,因为数量增加后的车载计算机还是可以和原定数量的车站计算机实现交互。分布式仿真的这项措施主要是为了和相应的集中处理器的执行情况相对比。

第三项措施是测量"闲置时间"。一列火车在所经过的所有节点上都要与车站计算机互动以获得轨道使用权,那么它在所有节点上停车且等待分配的累计时间就是"闲置时间"。闲置时间也包括它计算未能执行成功的最佳路径所需的时间。这项措施也是针对相应的集中处理器的执行情况实施的。

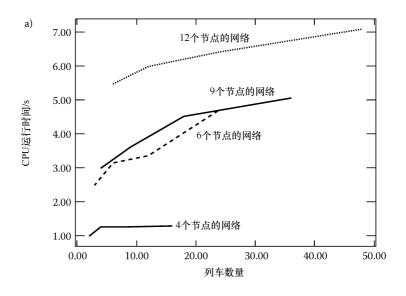
传统方法中造成集中调度通信延迟的一个很明显的原因是因为需要定期读取每列火车的位置和速度,以及每条轨道的状态。这在单处理器上模拟时会产生约0.7ms的延迟。

回顾一下,为了解决分布式仿真的时间问题,我们曾经把实际行驶时间当做延迟来模拟。在分布式和集中处理器仿真中,这些延迟都以同样的方式减缓了仿真速度,这就意味着我们能够得到一致的 CPU 运行时间,即决策时间与行驶时间之和,而不仅仅只能得到决策时间。因为这项研究的主要目的是对 DARYN 中分布式决策的性能进行评估,所以接下来我们要做这样一些事情:这两种仿真的 CPU 运行时间都可以通过按比例改变行驶延误,并通过绘图得到。两种情况下观察到的图形都是线性的,这意味着,行驶延误按比例线性地减缓了两种仿真的速度。在行驶延误等于0时,CPU 运行时间可以由这些图直接推算得出。在单处理器的情况下,可以看到推算出的数据与另一个不模拟行驶时间延误时的仿真结果完全一致。

此外,据观察,集中处理程序在 Sparc1 + 工作站上执行的速度比 ARMSTRONG^[32] 系统中每个处理器的速度都快大约 8. 56 倍。因此,必须将所有集中处理器的原始数据 都按比例扩展 8. 56 倍,然后才能与分布式仿真数据进行比较。

首先,考虑一个由 4 个节点和 4 条轨道组成的铁路网。其中,变量为列车的数量,取值范围从 2~11 时,分别记录相应的 CPU 运行时间。第二、三和第四个网路分别包括 6 个站和 7 条轨道、9 个站和 10 条轨道以及 12 个站和 17 条轨道。在每个网络中,变量都是列车的数量,其取值范围分别为 {3,24}、{4,36} 和 {6,48}。此外,对于所有的实验,在火车的起点和终点的选择上都考虑了地理位置分布的均匀性。火车的速度是随机分配的。我们可以观察到 CPU 运行时间的变化特征在所有情况下都是一致的。从图 4.11a 中可以看出,在全部 4 个网络中,推算出的分布式仿真需要的 CPU 运行时间都是火车数量的函数。图 4.11b 显示出这一特性在传统算法中也是类似的。可以看出 DARYN 图的斜率比传统方法小很多。图

4.11b 的 CPU 运行时间指的是 Spare 1 + 工作站的标量数据,即它们反映的是原始数据乘以 8.56 的值,以便于和图 4.11a 中相应的数据进行比较。



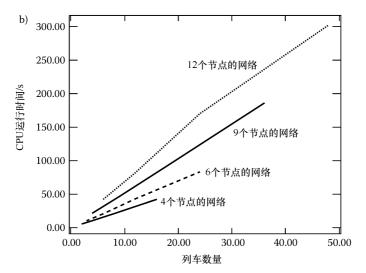


图 4.11 4 个网络中 CPU 运行时间与列车数量的函数关系图 a) DARYN 算法 b) 传统方法

图 4.12 表示 4 个网络中分布式算法相对于传统算法的加速因子与火车数量之间的函数关系。加速因子等于分布式仿真与集中处理器仿真的 CPU 运行时间之比。 当有 48 列火车和12 个站时,加速因子是 43。这个加速仅用了 12 个处理器来实现, 并不违反任何热力学原理。它仅仅反映了这样一种情况,即随着问题规模的扩大, 集中处理器会因为通信延迟的不断增大而导致运行速度越来越慢。

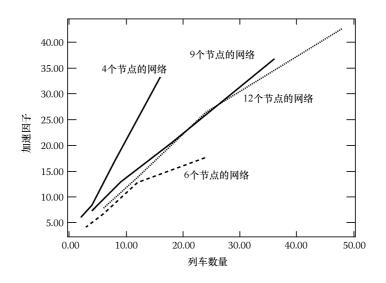


图 4.12 4个网络中加速因子与列车数量的函数关系图

图 4. 13 体现了算法性能的第二个种测量方式。一列火车以固定的速度行驶固定的距离所需要的 CPU 运行时间应该在不同条件下分别注明,因为火车的数量和网络规模都可能会不同。图 4. 13a 是分布式仿真算法需要的 CPU 时间,而图 4. 13b则是集中处理器需要的 CPU 时间。可以看出,图 4. 13b 的斜率很大,而图 4. 13a 的相对小很多,这意味着分布式决策优于集中式决策。此外,随着铁路网规模的增长,正如所预期的一样,随着时间的推移,DARYN 算法性能退化相对较慢,这意味着 DARYN 有很强的成长潜力。

最后,在4个网络中按不同的列车数量分别绘制特长专列的累计闲置时间曲线。闲置时间是指当火车停靠在某站点计算最优路径、决策,以及与车站计算机交互以获得所选择的轨道许可的过程中,闲置在该车站的时间。闲置时间也是衡量DARYN效率的一个指标,因为它反映了宝贵的列车资源的浪费情况。图 4.14a 表示的是 DARYN 算法中累计闲置时间与列车数量的关系,而图 4.14b 表示的是传统算法中累计闲置时间与列车数量的关系。虽然这个图不太容易看,但是很明显图 4.14a 中的闲置时间比图 4.14b 要少得多。

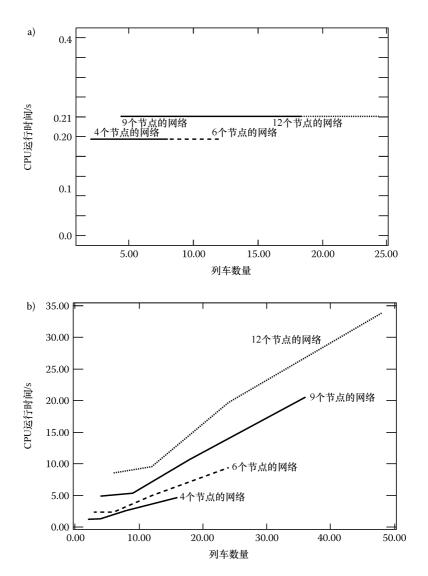


图 4.13 4 个网络中一列火车的 CPU 的运行时间与网络中列车数量间的函数关系图 a) DARYN 算法 b) 传统方法

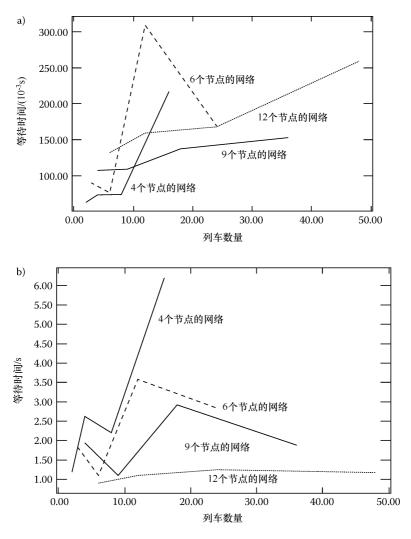


图 4.14 4 个铁路网中专列的累计空闲时间与列车数量的函数关系图 a) DARYN 算法 b) 传统方法

4.5 DARYN 算法的局限性

目前所知的 DARYN 算法的局限性在于它不能给火车分配优先权。车站计算机 完全是根据请求的到达时间来区分火车的请求,而不是根据火车的重要性。但是在 现实世界中,超高速火车,或装有易腐货物的火车都可能需要比那些没有携带易腐 烂货物的普通火车优先通过。只需要在 DARYN 中引入优先权的概念,就可以很容 易地解决这个问题。DARYN 也缺乏对交叉口和紧急情况进行管理的在线控制功能。为了解决这个问题,在交叉口或轨道的其他重要节点上,应该给 DARYN 提供额外的输入信号,紧急情况的信息也许可以依靠各车站来解决。交叉口的信号维护机制应该和目前在车站使用的机制是类似的。DARYN 的另一个明显局限性在于它不能处理列车当前所在位置前方的拥塞和瓶颈状况,这可能会影响它的性能。

第5章 RYNSORD: 一种新型的基于软预留 机制的铁路网络分散式控制算法

5.1 引言

当今世界,铁路网遍布全球。从 1825 年第一条常规货运铁路建成以来,铁路运输一直在客、货运输中占主导地位。世界上许多大国(如美国、俄罗斯、中国、印度等)及欧洲经济共同体都从分布广泛而运价经济的铁路网络中获益良多,而在其他一些像日本这样面积较小但人口稠密且财力雄厚的国家,铁路网络通过有效运输人和货物极大地促进了国民经济的发展。截至 1987 年,美国[33]总共已有 249412mile 铁路轨道,在一年内可以利用 5989522 节载重车皮完成总共 1249075534 车英里的周转量。客运服务方面,客运周转量达到了 3782470 人英里,净周转量达到 1357097 人英里。在日本,东日本铁路公司(坂本隆司,1987 年 12 月)一共拥有 7500km 铁轨,每天开行火车 12000 班次,运送旅客 16亿人。一般来说,为了提高列车运营效率、实现模块化和提高安全性,会把整个轨道分为若干独立单元,每个单元由系统单独控制。因此,一列火车从 A 开往 B 的过程中可能会经过几段轨道。一般情况下,在任何时刻,一段轨道只能被一列火车所占用,假如出现两列或更多的火车在某时刻要求使用同一段轨道的情况,就需要铁路网管理系统来为火车分配轨道,以达到两个目的:①避免列车相撞;②使铁路资源得到最优化利用。

除了在第 4 章中已经详细分析过的现有文献中铁路网集中式调度过程以外,ASTREE^[34]铁路交通管理系统还管理着一个分布式数据库,该数据库包括最新、准确及全面的路线布局数据和火车进程信息;该系统可以使用数据库中的信息自动或协助操作人员决定路线的设置和对火车的控制,这个设置会被记录到到路边的设备和机车中。Hill、Yu 和 Dunn^[35]在铁路网中模拟了电磁干扰,Ayers^[36]提出了在先进的火车控制系统(ATCS)中利用错误校正码可以实现可靠的无线链路通信。Sheikn、Coll、Ayers 和 Bailey^[37]提出在移动通信过程中信号衰减的问题。Hill^[38]提出了能使火车位置信息得到有效传送的编码方案,而 Shayan、Tho 和 Bhargava ^[39]提出了利用 Reed-Solomon 编解码器对 ATCS 进行改进的策略。美国铁路协会(George H. Way Jr. ,1992 年 2 月)指出,分布式算法可以提高火车调度的效率,而可能会对路径选择产生影响的几个社会经济因素包括所有权、轨道通过能力、加速能力、

等级、曲率、限距、乘务区以及经营协议等。如第4章指出,DARYN是一种新型的分布式算法,但它在使用前向单元方面存在缺陷。也就是说,在任何时候,它只能预留列车当前位置前方的下一条轨道。因此,它无法利用列车当前位置前方的拥堵信息来规划后续的路线,这就可能会导致列车运营效率的低下。

本章介绍的 RYNSORD 算法,它解决了前面描述的传统方法的主要局限性,标志着在铁路调度方面的一个重大进步。RYNSORD 研究了"前向"的概念,即预留当前位置前方的 N 条轨道,以提高资源和轨道的利用率并缓解拥堵。它还引入了一个新概念——软预留,其特点是在预留轨道上有更大的灵活性,而在传统的硬预留技术中,某一时刻的预留请求要么被批准,要么就被拒绝。

本章的其余部分安排如下: 5.2 节详细描述了 RYNSORD 算法的原理; 5.3 节描述了在一个由精确、真实的实验平台上模拟 RYNSORD 的过程, 这个实验平台由 70 台 SUN Spare 工作站联网组成, 并设置为一个松散耦合并行处理器; 5.4 节介绍了算法主要的实施问题; 5.5 节首先给出了实际铁路网在随机交通输入刺激下用 RYNSORD 进行仿真的运行数据, 然后对此进行了详细的性能分析。

5.2 RYNSORD 算法

RYNSORD 对铁路网来说是一个全新的算法,它有下列特点:首先,该算法是 一种非集中算法, 因为网络中所有列车路径的计算任务是由系统中的所有列车和站 点共同完成的。路径的确定过程是动态的,即当列车在驶向终点站的过程中依次连 续地选择轨道时,考虑了系统中其他列车的轨道需求。RYNSORD 在现代通信网络 技术方面也有独特的表现、其中的每一个移动设备、例如火车、都具有智能信息采 集功能、并且能完全独立自主地决定自己的运行路线。所以、要确保列车在铁轨上 安全地运行,就要求列车在上轨道运行之前,先从该轨道的管理节点处获得确定的 预留保证。想象一下, 一列火车在启程之前, 如果坚持要求为其预留所选择的路线 上从起点到终点的每一段轨道,就可能会因为使用了过时的信息,过早将列车前方 较远处的轨道锁定,而不能随着时间的推移而选择可能出现的更好的路线。与其相 反, RYNSORD 算法每次只考虑前方 N 段轨道的预留。系统中每列火车都要求以 N 段轨道为间隔为其进行预留、特别要求列车在开始一段新的运行之前、必须要获准 使用沿着既定路线上通向终点方向的后续 N 段轨道。在后续 N 段轨道的预留过程 中,RYNSORD提出了一个新颖的概念——软预留。在传统的硬预留中,一列火车 会在特定的时间段内发出 N 个连续信号请求预留 N 段轨道,而与轨道段相对应的 控制车站批准还是拒绝预留请求,是取决于在请求的瞬间各段轨道是否空闲。这 样, 当一列火车请求在 t, ~ t, 时间内使用一段轨道时, 即使车站知道该轨道只会被 占用到 t_1+1 ,接下来的时间都空闲,但它还是会拒绝批准。这导致火车不得不尝

试申请备选轨道,而备选轨道上的情况可能会更糟糕。因此,假如火车已经从站点了解到所有轨道被占用的情况,那么它就可能会选择等待1个单位时间,然后再选择较好的解决方案。所以在硬预留模式中,车站控制点采用的是僵化的二选一方式。相对于这种方式,RYNSORD提出软预留的做法:即只是为每列火车指定到达后续N段轨道所对应的站点的时间,该时间一定是单调增加的。这种情况下,如果轨道是可用的,控制节点就可以在火车提出请求的时刻批准轨道的预留,如果轨道暂时不可用,控制节点可以在超出请求时间后轨道变成可用的最早时刻批准预留。这一特征将使RYNSORD具有高效率和高鲁棒性,并能降低整个系统中灾难性事故发生的可能性。

在 RYNSORD 中,假定一个铁路网中包括一系列铁路车站(也称为节点),车站之间由很多有一定长度的铁路轨道相连。每个车站都配有一个计算引擎和通信设施,每两个车站之间由一条轨道连接,车站之间有双向通信链路,即每条轨道都是双向的。此外,每列火车都配备有车载计算机和通信装置,当火车到站时,就会立即启动与相应站点计算机之间的通信。而在火车移动过程中,RYNSORD 并不需要火车与火车或火车与车站之间进行通信。每个轨道区间由一定长度的轨道和独立控制该轨道的车站节点构成。节点 X 与 Y 之间的轨道由 X 或 Y 控制,控制节点独立负责与各列火车之间就该轨道的使用进行协商。RYNSORD 这些特点对保证安全和防撞是至关重要的。图 5.1 是一个简单的铁路网,从 A 至 F 的所有站之间由这个部分连通网络中的各轨道区间段连接起来。图 5.1 中也显示了相应轨道的长度和控制点。

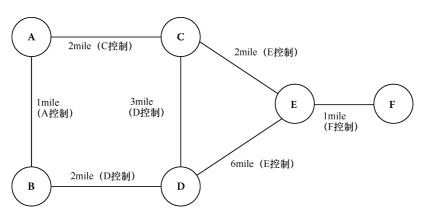


图 5.1 RYNSORD 中的一个铁路网示例

火车可以随时随刻、从任一车站,异步地进入到 RYNSORD 中。每列火车都有它自己的始发站、终点站和最高速度。一般来说,火车确切的行驶路线(从始发站到终点站经过的轨道序列)以及据此产生的到达时间都是由 RYNSORD 动态决定

的。但是,系统会给某些特殊的中途停靠站授权,将火车的整体路径分解成多组较短的路径,并在每一组路径的调度过程中连续使用 RYNSORD。也就是说,如果期望的路线是 $A \rightarrow C \rightarrow E$,那么在 RYNSORD 调度下先行驶 $A \rightarrow C$,然后在 RYNSORD 调度下行驶 $C \rightarrow E$ 。但如果过度使用这种中间站,也可能会影响系统的整体性能,因为 RYNSORD 的优势主要体现在能动态地控制火车路径,以最大限度地提高工作效率,优化资源配置,并避免拥堵。

如前面所述,RYNSORD中一个很重要的概念是前向性,定义为一列火车要通过预留申请与车站协商后续将要使用的轨道段的数量。前向性反映了火车试图为后续行驶预留资源,这些资源包括火车到达终点站前可能需要使用的一系列轨道区间段。

进入 RYNSORD 系统后,每台车载计算机首先会确定起点和终点间的最短路线。这就是所谓的"主要路线",其计算是基于各站间的距离,对主要路线的决策过程中并没有考虑拥堵情况。然后车载计算机会确定"次要路线",其组成轨道与主要路线的轨道应该是不重叠的,但存在一种例外的情况:在极少数情况下,如果有一段(或多段)轨道是网络中的一部分与另一部分相连的唯一通道,那么主要路线和次要路线就必须共用这一段(或多段)轨道。例如,在图 5.1 中,路段 E→F 是任一以站 F 为起点或终点的路径中必不可少的组成部分,因此该线路既在主要路线上,也在次要路线上。

接下来,火车从主要路线和次要路线中提取对应于前方 N 个轨道区间段(前向值 = N 时)的站点,将要发向这些站点的预留请求打包并启动预留请求。一个预留请求包由一个连续站点列表和每个中间站的预计到达时间组成。到达时间是根据当前时间、火车速度、车站之间的轨道区间长度计算的,并假设火车不会在中间站等待,也就是说,中间 X 站的离开时间就等于该站的到达时间。当然,火车在起点站还是要停留,在那些为预留前方 N 段轨道而发出请求的车站内也是要停留的,在这些站内,到达和离开的时间是会影响到轨道要预留的时间长短的。因此,对于轨道区间段 $X \rightarrow Y$ 来说,火车必须为在一定的时间段(离开 X 的时间,到达 Y 的时间)内使用该轨道进行预留申请。

火车将预留包发送到列表中的第一个车站。如果这个车站是第一段轨道的控制者,那么它将和列车之间就这段轨道的预留请求进行协商。假设火车需要预留的时间段是 (t_1,t_2) ,如果车站确定在这个时间内轨道不会被占用,就会批准预留。相反,如果所请求的时间段内轨道已经被其他火车占用,显然不会批准进行该预留。然后,车站会计算 t_2 之后最早的时间段,并且为新的时间段 (t_3,t_4) 预留轨道。时间段的长度根据轨道长度和火车速度进行计算,它将覆盖预留包中的第一个时间段,而使用后续轨道的时间段也被相应修改。如果第一个站不是第一段轨道区间的控制者,预留包就会被转发给第一段轨道区间的控制者。第一段

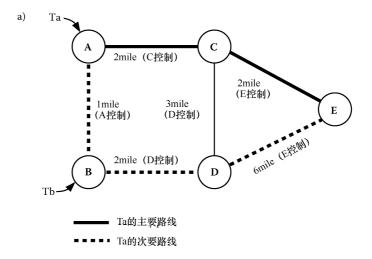
轨道区间的预留完成后,预留包被发送到控制下一段轨道的车站。如前所述,预留过程启动,当第二段轨道的预留时间段确定后,这一阶段的预留结束。如此,这个过程一直持续到所有 N 段轨道都获得预留时间段后才停止。随后,修改后的预留包沿已启动预留程序的路线返回到火车所在地。主要路线和次要路线会同时执行这个过程。

当一列火车收到它在主要和次要路线上的预留请求回应时,它也有可能不选择能以最短的时间到达后续第N 段轨道末端车站的路线为最佳路线(如 R1)。因为虽然主要和次要路线都通向最终目的地,且沿着路线 R1 能较早到达后续第N 段轨道的末尾,但这并不就能保证火车能更快地到达其最终目的地。因此,对于每一条主要和次要路线来说,火车会考虑到达后续第N 段轨道末尾的时间加上沿着最短路线从第N 段轨道末尾到达最终目的地的时间之和。假设主要和次要路线上的这个时间和分别用 TT_1 与 TT_2 来表示,火车就会选 TT_1 与 TT_2 中值比较小的路线。若 TT_1 = TT_2 ,则火车会任意选择,一般情况下会选择主要路线。然后,火车对没有被选中的路线生成一个预留删除请求,并传送到车站,以解除相应的轨道预留。

因为火车是沿着 N 段轨道从一个站到下一站前进,这就保证了相应轨道的使用时间与之前批准的预留时间是一致的。然而,如果一列火车早于预计到达时间抵达某车站,而下一段轨道可用的时间段也足够长,车站可以立即允许该火车前进。否则,火车就要撤回对下一轨道段的初始预留时间段,并对其进行修改。当一列火车抵达一个站时,也可能会发现下一段轨道的可用时间比它请求的时间段更早,这是因为当火车在执行最初预留请求时,如果决定选择另一条路线,这段轨道就会由于原预留请求被删除而空置出来。因此,之前批准的这段轨道的预留时间段是这列火车行驶时间的一个上限。如果有多列火车争夺由于这列火车离开而空出来的一段轨道,那么在车站已经等待了最长时间的火车就会得到最高的优先权。

为了理解 RYNSORD 的操作,看图 5. 2a 和图 5. 2b 中的铁路网。除了没有节点 F 外,它们和图 5. 1 完全相同。假设两列火车 Ta 和 Tb 同时进入系统,t=0 时分别在节点 A 和 B。Ta 和 Tb 都开往车站 E。图 5. 2a 描述了 Ta 从起点 A 到终点 E 的主要路线和次要路线的计算。图 5. 2b 描述了 Tb 从起点 B 到终点 E 的主要路线和次要路线的计算。图 5. 2a 和图 5. 2b 中,实线和虚线分别表示主要路线和次要路线。

假设前向值 N=2,火车 Ta 和 Tb 从主要路线和次要路线中提取与前方 N=2 段轨道相关的车站。在这个例子中,假设 Tb 的主要路线是根据轨道的数量,而不是根据从出发点到终点的英里数来选择的。车站的主要和次要路线是



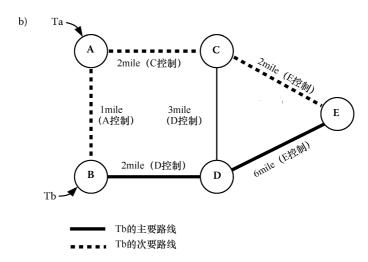


图 5.2 RYNSORD 中主要路线和次要路线的计算 a) 火车 Ta b) 火车 Tb

Train Ta:

Primary path station list: $A \longrightarrow C \longrightarrow E$ Secondary path station list: $A \longrightarrow B \longrightarrow D$

Train Tb:

Primary path station list: $B \longrightarrow D \longrightarrow E$ Secondary path station list: $B \longrightarrow A \longrightarrow C$

假设火车速度为 1 mile/min, Ta 和 Tb 将会生成预留请求包,并激活

Train Ta:

Primary path:

arrival at A at time 0
departure from A at time 0
arrival at C at time 2
departure from C at time 2
arrival at E at time 4

[no departure since E is the final destination]

Secondary path:

arrival at A at time 0
departure from A at time 0
arrival at B at time 1
departure from B at time 1
arrival at D at time 3

no departure since D is the last station in the station list

Train Tb:

Primary path:

arrival at B at time 0
departure from B at time 0
arrival at D at time 2
departure from D at time 2
arrival at E at time 8

[no departure since E is the final destination]

Secondary path:

arrival at B at time	0
departure from B at time	0
arrival at A at time	1
departure from A at time	1
arrival at C at time	3

[no departure since D is the last station in the station list]

图 5.3a~d 描述了 RYNSORD 中火车传送预留请求包和各车站处理预留请求包的操作过程。在图 5.3a 中, Ra1 和 Ra2 表示 Ta 传送的关于主要和次要路线的预留

请求包; Rb1 和 Rb2 表示 Tb 传送的相应预留请求包。在图 5. 3a 中,Ra2 请求站 A 在时间 (0, 1) 内预留轨道 $A \rightarrow B$ 。由于在时间段 (0, 1) 内轨道 $A \rightarrow B$ 是空闲的,这个请求被顺利批准。由于站 A 不能控制轨道 $A \rightarrow C$,Ra1 就不能利用站 A 请求预留 $A \rightarrow C$ 。Rb1 和 Rb2 都不能利用站 B,因为站 B 既不能控制 $B \rightarrow D$,也不能控制 $B \rightarrow A$ 。

图 5. 3b 表示预留包传送到下一站的过程。这里,Ral 和 Rbl 分别预留了时间段(0,1)内轨道 A→C 和(0,1)内的轨道 B→D。Ra2 什么都没做,因为 B 不是轨道 B→D 的控制者。当 Rb2 试图为时间间隔(0,1)预留轨道 B→A 时,因为火车 Ta 已经预留了这个时间段,因此请求失败。这样,站 A 就为火车 Tb 预留了下一个可用的时间段(1,2)。火车 Tb 更新了它的次要路线预留包,缩写为 [B @ 0/1] [A@ 2/X] [C @ X/X],这表示在时间 0~1 内火车 Tb 在站 B 等待,在时间 2 时前进到站 A,同样在时间 2 离开站 A,在时间 4 到达站 C。火车 Tb 不能预留 C 以后的轨道区间,因为它的前向值为 2,这可用子字段 " [C @ 4/X]"表示,X表示未知值。每一个简写的预留包的子字段都可以表示为 [站名@ 到达时间/离开时间(X 未知)]。

图 5. 3c 表示预留包的后续传播。Ral 和 Rbl 分别为时间段(2, 4)与(2, 8)成功预留了相应的轨道 $C \rightarrow E$ 和 $D \rightarrow E$ 。Ra2 不能为(1, 3)时间段预留轨道 $B \rightarrow D$,因为火车 Tb 已经预留了(0, 2)上的 $B \rightarrow D$ 。火车 Tb 可以预留的时间段为(2, 4),这个预留可以简写为 $[A@\ 0/0]$ $[B@\ 1/2]$ $[D@\ 4/X]$ 。Rb2 成功为(2, 4)时间段预留了轨道 $A \rightarrow C$ 。

因为前向值为 2,此时,Ra1 至 Rb2 的所有预留包已经成功预留到了最后一段轨道区间,并且分别从站 A 和 B 将相应预留情况返回给了火车 Ta 与 Tb。在站 A,火车 Ta 发现通过主要路线到达终点站 E 的总时间是 4,而通过次要路线到达 D 就需要 4 个时间单位,到达终点站 E 所需的额外行驶时间至少是 6 个时间单位,这意味次要路线的总行驶时间是 4+6=10 个时间单位。显然,火车 Ta 选择主要路线,即 $A \rightarrow C \rightarrow E$ 为最佳选择,然后将为 Ra2 所做的预留删除请求传送到车站。对于火车 Tb,主要路线需要 8 个时间单位到达终点站 E,而次要路线需要 4 个时间单位到达 C,到终点站 E 所需的额外行驶时间至少是 2 个时间单位,即次要路线的总行驶时间是 4+2=6 个时间单位。因此,火车 Tb 选择次要路线,即 $B \rightarrow A \rightarrow C$,并向车站传送一个预留删除请求,解除它之前在主要路线 Rb1 上获得的预留。

图 5.3d 表示当火车 Ta 和 Tb 开始运行,并且预留删除请求已经被处理之后 RYNSORD 的情况。应该注意的是随着火车 Ta 在轨道 $A \rightarrow B$ 上的预留被删除,火车 Tb 可获准比它预留的时间段 (1,2) 更早地使用这段轨道。

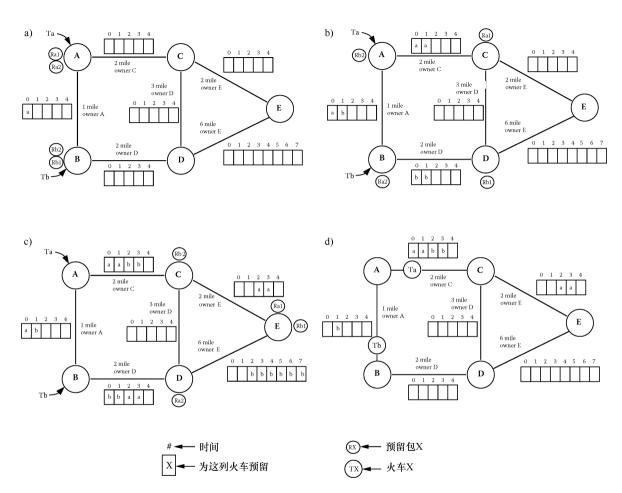


图 5.3 预留操作的初始化和进行过程

5.3 在精确、实时、并行处理的实验台上模拟 RYNSORD

RYNSORD 最重要的贡献就是把网络中所有火车的路径计算任务分配给系统的所有实体——火车和站点来完成。因此,在现实中,对于一个有 N 列火车和 S 个车站的铁路网,协调式计算引擎总数就是 N + S。为了解 RYNSORD 的性能及不同因素对它的影响,我们首先对它进行模拟,然后在一个并行处理实验台上进行仿真,该实验台由配置有松散耦合并行处理器的工作站联网组成。结合实验台进行的有真实际上与真实情况下的执行非常类似。为了模拟一个真实的系统,必须有一定数量的火车,而每个站点都用一个工作站表示,火车被模拟为任务,并由车站的基本工作站执行。当一列火车停在一个主站时,它的计算是由车站的基本工作站完成的,它与其他车站的通信也是通过这个车站进行的。当一列火车从当前车站(A)行驶到另一个车站(B)时,基本工作站将车站 A 的火车-任务进行封装,以消息的形式传送给车站 B,并在车站 B 中重新显示为基本工作站的一项火车-任务。因此,火车在仿真中是以电子速度而不是物理速度移动的,而火车的计算和通信任务由主站的基本工作站执行。

尽管火车能以约 120mile/h 的速度运行, 但是实验台的快速计算机能使仿真以 比现实快很多倍的速度运行,这有助于对 RYNSORD 中不同参数值进行快速评估。 仿真过程中的基本时间单位称为1个时间步长,它是仿真中火车移动的最佳分辨指 标。任一火车的最小行驶距离被限制在1个时间步长内,同时,如果没有得到必要 的预留批准,火车就需要在车站等待至少 1 个时间步长。在目前 RYNSORD 的实施 过程中,实际运行的每个时间步长值设置为 1min, 主要是因为: ①即使是最快的 火车, 在 1 min 内的行驶距离也仅为 2 mile, 明显小于 50 mile 的最短轨道长度; ②为 处理后续 N 段轨道的预留, 车站与火车之间所有必要的计算和电子通信都可以在 1min 内完成。—列火车可能需要几分钟才能驶过一条单轨,而信息的传播和计算 只需大约 10ms。RYNSORD 允许火车异步地进入系统,即在不同的时间段内进入。 除此之外、火车本身是独立的、因而它们的决策是异步执行的。而且、实验台是由 具有不同时钟速度的异构工作站组成的。因此, 为了保证系统中火车传播的一致性 和准确性, RYNSORD 要求每个站点和火车的时间步长值同步。例如,在 RYNSORD模型中,尽管处理器和通信链路速度可能不同,但是 Ta 必须在 Tb 之前 到达 E。上述要求就可以确保 Ta 和 Tb 在实际时间 12:01PM 和 12:05PM 分别到达 终点站 E。RYNSORD 的同步是利用空消息[40,41]的一个异步分布式离散事件模拟技 术来实现的,这里就不详细介绍了。

前面提到的假设,即与一个预留请求有关的所有的消息通信和决策的过程,都必须在一个时间步长内完成,有这样几层含义:如果车站 A 的一列火车 Ta 在时间

步长 t_1 启动一个预留包,并将其传播到其他合适的车站(X, Y, Z, …),预留包就必须在该车站内进行处理,并且在车站 A 和车站(X, Y, Z, …)中的每个基本工作站的时间步长值增加到 t_1 +1 之前返回到车站 A 的 Ta 中。为了实现这个目标,RYNSORD 采用了一种特殊的同步节点,该节点与所有站点相连,它用于监测:对所有在车站外发出的预留请求的必要的通信和回应,是否能在允许的站点增加时间步长值之前完成。RYNSORD 中这种特殊的同步节点是试验台上并行仿真的一个产物,在现实中并不存在。在一个实际的铁路网中,电子通信和计算需要10~100ms,这段时间内最快的火车也只行驶了 18ft (1ft = 0. 3048m) 的轨道。因此,在实际中,通信和计算基本都是瞬时完成的。

每个车站站点和特殊的同步节点的基本功能被分别封装在图 5.4 和图 5.5 所示的伪代码中。

```
while simulation is not finished
         send out reservation requests
         while not done
            process incoming trains
            process incoming reservation requests
            process incoming reservation responses
            if received responses to all reservation requests
               send done to synchronization node
            if received update from synchronization node
               set done to true
    update internal time
          图 5.4 火车站点的基本功能伪代码
   while simulation is not finished
             while not done {
                 if received done from station node
                    increment count
                 if count is equal to number of stations
                    set done to true
          send update to all stations
          update internal time
          图 5.5 特殊的同步节点功能伪代码
```

RYNSORD 需要交流 4 类消息:首先是要表示由火车启动和发出的"预留包";第二是模拟火车从一个站到下一个站进行消息封装和传播过程;第三,如果一列火车提早到达轨道前端的车站,为了能比预定时间更早地行驶到下一段轨道,允许该火车和控制站点进行协商;第四是车站允许火车立即行驶在某段轨道上。假设预留包的传递没有任何延迟,即一旦一个车站收到预留包,就在同一步长时间内立即对其进行处理;最终被批准的预留包也要在同一步长时间内返回到始发站。在始发站,火车在初始化时是没有预留包的,然后,该火车产生并发送预留包,一旦收到已批准的预留包,火车就会选择预留包中的一个作为最佳选择,于是就有了自己的预留包。然而,这个预留包只适用于随后的 N 段轨道,而且火车必须不断重复这个过程直到它到达最终目的地。下面对组成预留包的 5 个字段进行解释。

预留包:

- 1) 车站列表: 完整的车站列表包括每个车站的车站编号、到达时间和离开时间。
- 2) 状态: 预留的状态——①预留,如果预留包沿着车站列表中的路线向前推进,就是在试图预留;②删除,如果预留删除包正在向前传播,就是在解除先前批准的预留;③接受,如果预留包返回到起始站就接受预留。
- 3) 火车编号:火车的唯一编号。id = (起始站 × 100000) + 火车被引入系统的时间。
- 4) 火车速度: 当原始请求不能得到满足,车站必须修改火车的预留时间时,就必须知道速度信息。利用速度信息计算通过轨道的行驶时间,用以确定下一个预留时间段。
 - 5) 预留编号: 预留的唯一编号。

由于火车从一个站到下一个站可能需要很长的行驶时间,因此假设"火车包"不是瞬间传播的。事实上,每个火车包都已被盖上了时间戳值,这个值表示它预计到达终点站的时间步长。一旦收到一个火车包,站点就会将其存储在缓冲区,直到火车包的时间戳值等于车站的时间步长值时。然后假设火车已经到达站点,并启动下一步处理。一个火车包由6个字段组成,下面对其列举并详细说明。

火车包:

- 1) 时间戳,火车预计到达接收站的时间。
- 2) 火车编号:火车的唯一编号。编号 = (起始站 × 100000) + 火车被引入系统的时间。
 - 3) 起点:火车的始发站。

- 4) 终点:火车的终点站。
- 5) 路线:火车行驶轨道的顺序列表,该列表用于进行数据收集与分析。
- 6) 预留:与这列火车有关的预留包。

由于火车可以要求取消先前批准的沿某条路线的轨道预留请求,所以当火车决定选择备用路线时,抵达车站的火车可能会发现其下一条要使用的轨道是空闲的。为了提高轨道的使用效率,在可能的情况下,应该允许火车沿着轨道继续行驶。为实现这个目标,当一列火车在它预定离开那个节点的时间之前到达该车站,它就会产生并传播一个"等待包"给这段轨道的管理车站。一旦收到等待包,对应的车站就会对火车进行排队。在每个时间步长内,车站检查轨道是否空闲,并说明轨道空闲的时间步长数(如 Q)。然后,车站从排队的火车中选出一列(如果有的话)等待时间最长的火车,这列火车有可能在 Q 时间步长内成功通过该轨道。车站发送一个"许可包"给这列火车,允许它立即使用轨道,并从队列中删除对应的进入请求。这种等待包由 4 个字段组成,如下所示,而许可包只包含一个字段。

等待包:

- 1)火车编号:火车唯一的编号。编号 = (起始站 ×100000) + 火车被引入系统的时间。
- 2) 火车速度: 计算火车通过下一段轨道的行驶时间时必须要知道火车速度。
 - 3) 等待开始时间:火车比预定离开时间早到达车站后的排队时间步长。
- 4) 位置:火车等待的地方,轨道的头部或尾部处的车站。拥有该轨道的车站利用这个信息可以指引许可包的传递方向(必要时)。

许可包:

火车编号:火车唯一的编号。编号 = (起始站 × 100000) + 火车被引入系统的时间。

为了更好地理解 RYNSORD 中火车的分布式动态路径计算,以及不同参数对 RYNSORD 性能的影响,我们使用一个图形化终端来实现 RYNSORD 操作的可视化显示。该图形支持以下功能:

- ①开发和编辑铁路网。
- ② 观看仿真运行的重复放映。
- ③ 在运行时,监测并与仿真互动。
- ④ 查看关于输入流量、结果和运行一次仿真时的实时表现的统计信息。
- 运行实时显示器也显示以下参数:
- ①火车的位置。

- ② 在各个车站处理的预留累计数。
- ③ 在各个车站等待的火车数量。
- ④ 沿各个路段传播的预留累计数,按类型分为预留、删除、接受。
- ⑤ 已经传播到每段轨道的火车累计数。

图 5.6 是图形界面的一个截屏图,它详细显示了图 5.8 中有 50 个车站的铁路网络。在图 5.6 中,每个车站被标记为其名称的前三个字符,并赋予唯一编号。通过这个界面很容易增加或删除车站和路段,该图形程序还能自动、正确地重置 RYNSORD 以准确执行仿真。图 5.7 给出了一个实际模拟运行的屏幕截图。火车用圆圈表示,位于其行驶轨道的顶部,并通过它们各自对应的编号、起点和终点来区别。在实际模拟运行过程中,用户可以和任一车站相互通信,以获取所有需要的数据结构和信息,并可以在任一时间步长,从任一车站生成并将火车引进系统中。

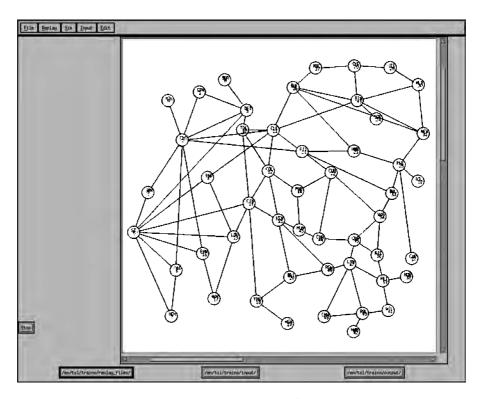


图 5.6 图形界面的一个截屏

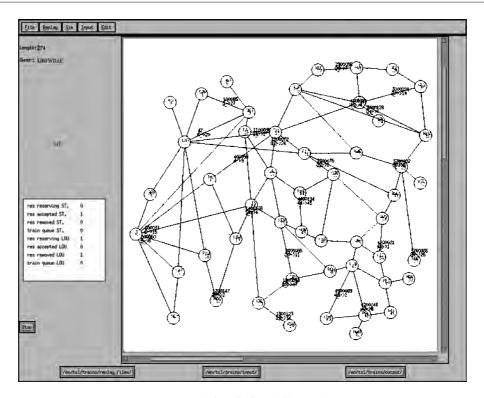


图 5.7 一个实际仿真运行结果的截屏

5.4 算法的执行问题

RYNSORD 模型和仿真器用 C 语言编写,在一个基于 UNIX 工作站的异构网络上执行,该网络通过一个 10Mbit/s 的以太网连接,并且配置为一个松散耦合的并行处理器。工作站使用免费却有版权的 Linux 操作系统,该系统提供英特尔 486DX2/66 和奔腾处理器,还包含 SUN OS4.1.2 和 SUN Solaris 5.3 操作系统的 SUN Spare1、Spare2 和 Spare10 工作站。站点和火车通过程序进行模拟,并且通过 TCP/IP 进行通信。关于每个车站及停靠在该站点的火车在内的代码段大约是 1700 行 C 程序,网络代码约 1000 行,也是 C 程序代码。仿真器由 GNU C, gcc 编译,效果设置为"好",在低优先级背景下执行。我们要说明的是到这些工作站可以为控制台的用户完成大部分主要工作。如果 50 个 SUN Spare10 工作站同时运行,进行单次仿真实验的平均时间约为 2h。

RYNSORD 是一个建立在最高级通信协议 TCP/IP 上的应用程序。根据定义, ISO-OSI 层模型开始于会话,它在两个或更多的机器之间通信时负责所有

必需的数据转换工作。由于英特尔 80X86 机器采用 Little Endian 模式而 SUN Spare 机器采用 Big Endian 模式^[42],所以在异构网络工作站中要实现必要的转换数据,就必须使用 n-to-h l(网络到主机长整型)、n-to-h s(网络到主机短整型)、h-to-n l(主机到网络长整型)和 h-to-n s(主机到网络短整型)等实用程序^[43]。

在这一章中,我们根据美国东部现有的主要铁路情况,选中美国东部铁路网的一部分(如1994年兰德麦克纳利商业地图集所示)作为研究对象,并在其中加入一些额外的轨道来代表一些次要铁路路段。图 5.8 给出了这个典型铁路网的情况,该网络由 50 个主要车站、84 个轨道段区间和总长 14469 mile 的轨道组成。在图 5.8 中,网络的模型在 RYNSORD 上建立,仿真在 50 个工作站的网络上执行,并且一个车站对应于一个工作站。

为了得到有代表性的运行结果,许多实验中随机产生输入火车并将其引入到 RYNSORD 中运行。铁路交通密度,是指每单位时间步长内被引入到系统的火车数量,对该密度的选择是根据一年 365 天都使用美国东部铁路轨道的货运列车的实际数量获得的。本章的实验选择了三种铁路交通密度——低、中、高。在一个时间步长内(即实际的 1 min)生成一列火车的概率,对于低、中、高交通密度来说,分别设为 0.075、0.15 和 0.30。对于每列在车站始发的火车,其速度是随机生成的,取值范围为 60 mile/h。除了始发站是随机选择的一个车站外,最终目的地的选择,也是通过对始发站以外的每个车站分配相等的权重后随机产生的。地理上的优势在选择过程中不起作用。对应于大城市的车站更有可能产生较高的交通密度,我们从图 5.8 中选择了一组 9个具有"高流量"的车站,包括芝加哥、底特律、圣路易斯、费城、纽约、华盛顿、匹兹堡、哥伦布和辛辛那提。对于和这些城市相对应的车站,输入火车的交通密度将增加一倍。另外,在选择火车最终目的地的过程中,这些城市的车站权重是其他城市的两倍。

根据不同的情况,针对不同的参数,我们对这个有代表性的铁路网仿真了150 多次。每次仿真都执行10080 个时间步长,它对应于实时操作的一周时间。如前所述,执行一次典型的仿真实验大约需要 2h,而 50 个工作站如果单独运行,最长的运行时间往往需要 7h。在每一次仿真运行的过程中,火车输入都是一个常数,火车速度也是一定的,该速度是在仿真开始时就设定好的。表 5.1 分别给出了低、中、高输入火车密度下被引入系统的火车累积数和所有火车行驶的累积总里程估计值。这一估计值的确定,是建立在每列火车都是沿着从起点到终点的最短路径行驶这一假设基础上的,所以这个值并不一定在每种情况下都是对的。

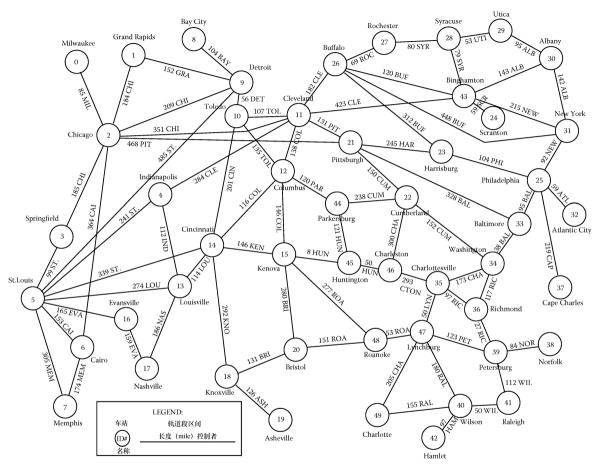


图 5.8 一个典型的铁路网:美国东部铁路网络中由 50 个车站组成的子网

输入火车密度	引入 RYNSORD 的火车 累积数量	所有火车行驶的 累积总里程估计值/mile		
低	484	288860		
中	869	497658		
高	1772	1033123		

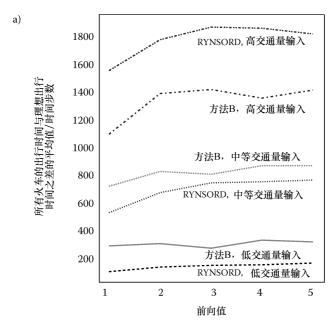
表 5.1 输入的火车交通参数

5.5 仿真数据和性能分析

为了了解 RYNSORD 的性能,首先要确定独立参数和主要的性能评价指标,然后在实际铁路网络上进行大量的仿真实验,如图 5.8 所示。独立参数包括:①进入系统的火车数量;②火车的密度,即火车进入系统的频率;③使用的参数假设值。为了客观评价软预留的作用,本章还执行了另一个分布式路径算法,称为算法 B。除了采用传统的硬预留策略之外,算法 B 在其他各方面都与 RYNSORD 类似。火车首先为主要路线发出硬预留请求,车站会尽量依照时间顺序在需要的时间段内为火车预留所请求的轨道。如果请求成功,火车就会使用主要路线上被批准的轨道。相反,如果每条轨道都很忙,预留请求就会被拒绝,火车立即返回始发站。在这种情况下,火车会接着为次要路线发送硬预留请求,如果这也失败了,火车再次为主要路线发起请求之前,必须至少等待一个时间步长。这个过程一直持续到火车能够获得预留为止,然后沿着被批准的轨道继续前进。所以,火车在成功获得预留批准前必须在车站等待,只有预留请求被成功批准后,火车才能沿着 N 段连续轨道向前移动。

主要性能评价指标有:①火车的行驶时间,即所有火车抵达目的地的平均时间;②抵达终点站的火车比例;③火车使用的中继段(轨道)数的分布;④火车行驶的中继段(轨道)数与前向值的函数关系;⑤每列火车的行驶时间与它们进入仿真的时间的函数关系;⑥轨道利用情况。此外,为了了解在全部实体中分配整体路径计算和通信任务的重要性,我们另外定义了三个性能评价指标,包括:⑦火车执行计算时间的分布;⑧车站处理预留的数量分布;⑨车站间链路的最大通信速率。

图 5. 9a 是所有火车的出行时间与理想出行时间之差的平均值和前向值的函数关系图。每列火车的理想行驶时间作为一个参考值,它是指当整个系统中只有一列火车,并沿着最短路线前进到终点站而不受其他任何火车的干扰时,该火车需要的行驶时间。显然,由于实际系统中一定会有其他火车出现,所以某列火车有时并不能成功获得轨道预留,也不一定能按最短路线行驶。图 5. 9a 显示了 6 条曲线,对应在 3 种不同密度下 RYNSORD 和算法 B 的情况。对于中、低输入交通密度,RYNSORD 的性能一直优于算法 B。图 5. 9a 揭示了当前向值增加时,火车的平均



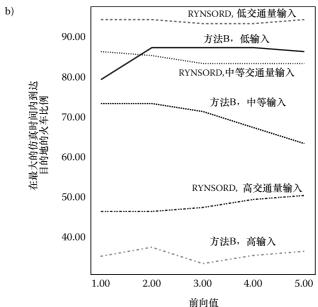


图 5.9 所有火车的出行时间与理想出行时间之差的平均值与前向值的函数关系 (图 a) 以及在最大仿真时间内到达其目的地的火车比例 (图 b)

行驶时间也会有不同程度的增加。对于高交通密度情况, RYNSORD 的性能表现相对于算法 B 要差一些,这个异常现象可以由图 5.9b 中的曲线来解释。图 5.9b

描绘了仿真终止前抵达终点站的火车比例,同样对应在3种不同密度下 RYNSORD和算法 B 的情况。密度的增加会引发更大的拥堵,从而导致能够完成 全部行程的火车比例降低。同样,无论是在哪种情况下,RYNSORD 仿真中火车 抵达终点站的比例都比算法 B 大,这体现了软预留的优越性。此外,由于在高 交通密度下相对于 RYNSORD 来说, 算法 B 中抵达终点站的火车数量明显较少, 造成图 5.9a 中算法 B 对应的行驶时间曲线不包括行驶到更远目的地的火车行驶 时间,因此该曲线是扭曲的。

图 5.10 显示了火车的中继段(轨道)的分布情况,即在低输入交通密度情况 下,利用中继段(轨道)抵达终点的火车数量与中继段(轨道)数量之间的关系, 轨道的数量在1~20之间。图5.10中有5条曲线,一条对应于理想状况,两条分 别对应于前向值分别为 2 和 4 的 RYNSORD 结果, 另外两条对应前向值为 2 和 4 时 算法 B 的结果。这里的理想状况是指:将每列火车都看做系统中唯一的火车,火 车按计算出的理想路线行驶的情况。在实验台上模拟的实际条件中,大多数火车都 极可能无法为它们理想路线中的每段轨道获得预留,因为系统中存在的其他竞争火 车也可能需要这些轨道。可是,与此推论截然相反的是,从仿真获得的图中却显示 中继段轨道数量的分布非常接近于理想状况,也就是说,尽管有484列火车竞争轨 道. 但是 RYNSORD 在软预留策略下的分布式、动态路径算法产生的结果仍接近于

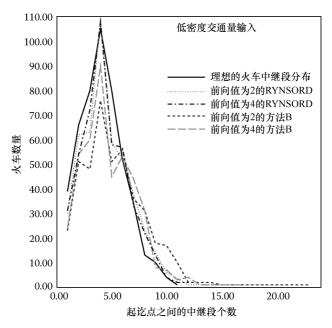


图 5.10 在 RYNSORD、方法 B 和理想状态下 的火车实际中继段(轨道)数量分布图

理想状况。造成这种状况的原因如下:学术界一般认为,尽管分布式算法可能更快地产生结果,但分布式解决方案的质量还不能达到集中式算法结果的质量。这种看法源于一个事实,即当地的计算引擎在使用分布式算法进行决策的过程中,只被允许访问全局数据中的一小部分。而严密的 RYNSORD 仿真结果无疑证明了这种想法并不具有一般性。RYNSORD 表明,在某些情况下,分布式算法也可以非常快速地产生高质量的解决方案。作者目前就正在研究一种新的数学结构,希望能从问题的集中式描述中提取出相应的分布式算法。图 5.10 也证明了 RYNSORD 的软预留是优于算法 B 的硬预留的。

图 5. 11a~c 中的曲线对比了低、中、高交通密度条件下不同前向值的 RYN-SORD 和理想情况下的中继段轨道数量的分布情况。图 5. 11a~c 中各自的每一条 RYNSORD 曲线都稍有不同,这表明前向值对中继段轨道数量的分布影响不大。此外,随着交通密度的增加,中继段轨道数量的分布越来越偏离理想状况,这意味着轨道竞争的加剧使得火车不再是沿它们从起点到终点的最短路线选择轨道。在图 5. 11a~c 的每个图中,对应于前向值为 2 的图都显示出一个看起来小却不容忽视的迹象:较少的火车的数量却对应了过多的中继段轨道数。这是由折返行驶造成的,即当火车试图与车站协商一条合适的路线以到达目的地时,它可能会在两个或更多的车站间折返行驶。尽管这看起来可能会降低轨道的利用效率,但图 5. 9a 的结果却表明火车在前向值为 2 的情况下,能更快到达目的地。随着前向值的增加,出现火车折返行驶的情况会大幅减少。

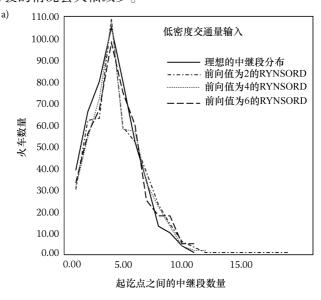


图 5.11 低、中、高交通密度的输入条件下 RYNSORD 的 中继段轨道数量实际值分布与理想分布的对比

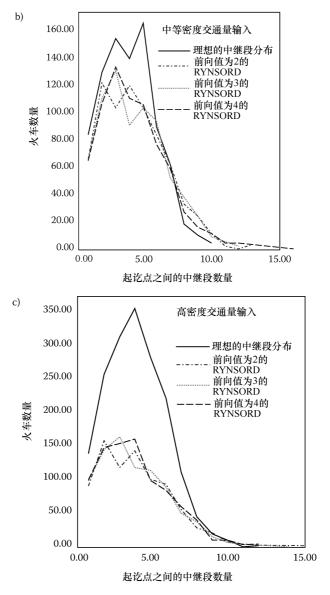


图 5.11 低、中、高交通密度的输入条件下 RYNSORD 的中继段轨道数量实际值分布与理想分布的对比(续)

图 5.12 说明了在三种不同的输入交通密度条件下,RYNSORD 和算法 B 的前向值大小对中继段(轨道)平均数的影响。对于每列火车,要计算理想的中继段轨道数,并将它当作比较火车实际使用的中继段轨道数的标准。在RYNSORD下,三种输入密度情况下的曲线看来都收敛到一个较小的数,它略小于方法 B 的收敛

值,这再次证明了软预留的优越性。此外,对图 5.12 中每种情况的低前向值状态进行考察,发现平均中继段轨道数都明显高于理想的中继段轨道数。这证实了之前的发现:低前向值会使火车在运行过程中频繁转换轨道,造成火车在整个过程中要使用更多数量的轨道。

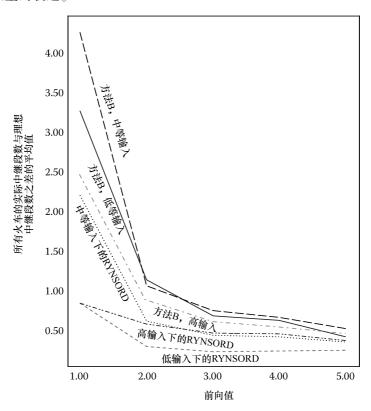
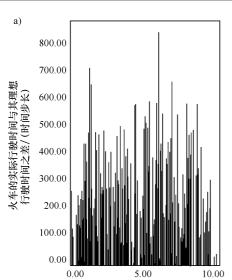
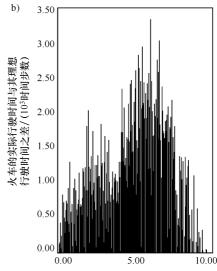


图 5.12 所有火车的实际中继段数与理想中继段数之差的平均值与前向值的函数关系

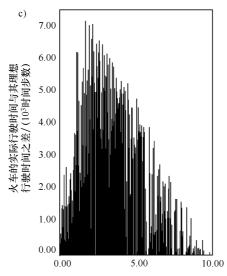
图 5. 13a~c 描述了所有到达目的地火车的数组关系 [(实际行驶时间与理想行驶时间之差),火车进入系统的时间]。图 5. 13a~c 分别对应于低、中、高输入交通密度情况。一般来说,随着越来越多的火车竞争轨道,火车会需要更多的时间到达目的地。这可以从图 5. 13a~c 中 Y 轴上增加的时间步长刻度单位上看出。对于低输入交通密度,大多数火车到达目的地需要的行驶时间与其进入系统的时间无关,所以图 5. 13a 中的分布相对均匀。而图 5. 13b 则在较高的进入时间值处,即进入系统的时间步长为 10000 处出现一定程度的中断,这种情况在图 5. 13c 中更加严重。这反映了一个事实:在较高的输入密度下,一列火车 Ta 比另一列火车 Tb 更晚进入系统,而火车 Tb 却可能比 Ta 需要更多的行驶时间,并且,在某些情况下,它甚至可能无法在允许的最大模拟时间内成功完成旅程。显然,为了形成一个中断最



火车进入系统的时间/(103时间步数)



火车进入系统的时间/(103时间步数)



火车进入系统的时间/(103时间步数)

图 5.13 前向值为 4 时, RYNSORD 中火车的实际行驶时间与其理想行驶时间之差, 与火车进入系统的时间之间的函数关系

a) 低交通输入密度 b) 中等交通输入密度 c) 高交通输入密度

少、稳定而连续运行的系统,必须选择一个适当的输入交通密度。

图 5.14a、b 显示了轨道的使用结果,即算法 B 和 RYNSORD 中每段轨道被火

车使用的累积次数。轨道区间段编码为1~84。虽然大部分轨道段的使用都是合理的,体现出了资源利用的有效性,但还是有一些轨道段有过高的使用率,反映出火车目的地选择的随机性和对"高流量"车站的选择偏好。可以观察到,图 5.14b中的轨道使用率在一般情况下要比图 5.14a 的高,这意味着 RYNSORD 软预留优于算法 B 的硬预留。

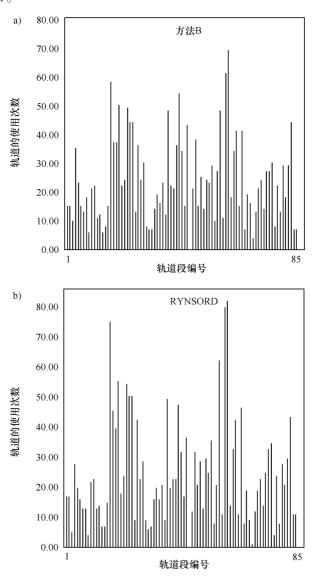
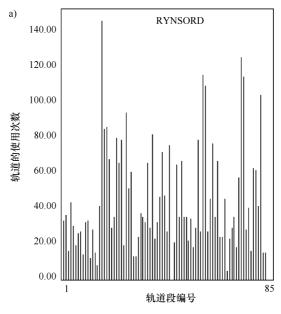


图 5.14 低输入交通密度、前向值为 6 时轨道的使用次数分布 a) 方法 B b) RYNSORD

图 5.15a、b显示了高输入交通密度时的轨道利用情况,它显然大大高于图 5.14b 中低输入交通密度时的利用率。然而,从图 5.15 可以看出,RYNSORD 的轨道利用率并没有受到前向值选择的显著影响。



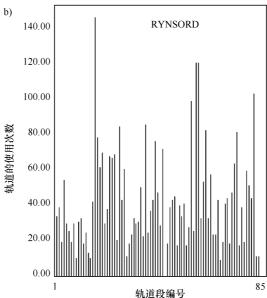


图 5.15 高输入交通密度时的轨道使用次数情况 a)前向值为2 b)前向值为4

图 5. 16 显示了为所有站间火车进行路径计算的部分任务的分布情况。为所有火车计算路径的任务中,一个重要组成部分就是预留处理。图 5. 16a 对应的前向值为 2, 图 5. 16b 的前向值为 4。前向值为 2情况下,车站间的计算负荷分配值略高于前向值为 4 的情况。尽管在低前向值情况下,每列火车都要更频繁地执行Dijkstra^[44]的最短路计算,但在某个特定时间,它们预留的车站可能更少。

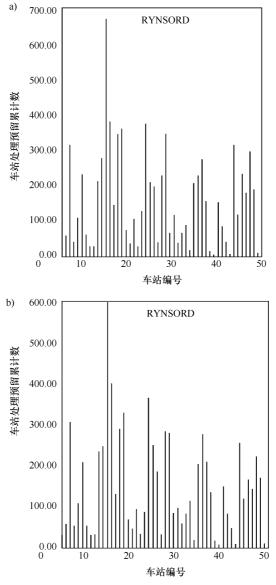


图 5.16 在中等输入交通密度下, RYNSORD 中车站处理的预留数分布情况 a) 前向值为2 b) 前向值为4

图 5. 16a和图 5. 16b 都强调了要将全部路径计算任务有效地分配给所有车站节点这一初始目标。预留处理分布的不均匀主要是由火车目的地的随机性造成的,这又反过来影响了火车的路径计算。

图 5.17 给出了完成火车间全部路径计算任务中剩余部分的时间分布情况。全部路径计算任务的另一个主要组成部分就是火车通过执行最短路算法计算主要路线和次要路线。图 5.17a 对应于前向值为 2 的情况,图 5.17b 对应于前向值为 4 的情

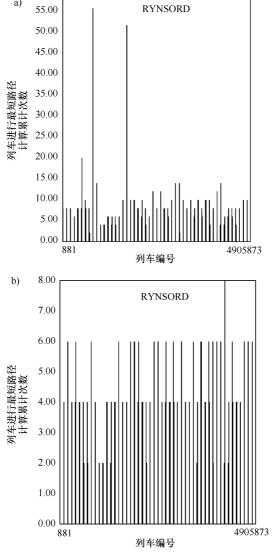


图 5.17 在中等输入交通密度情况下, RYNSORD 中列车计算量分布情况 a) 前向值为2 b) 前向值为4

况。图 5. 17a 和图 5. 17b 也都强调了 RYNSORD 的初始目标是在所有火车间有效地 分配全部路径计算任务。在图 5. 17a 和图 5. 17b 中,除了少数火车外,大多数火车 的计算量负荷均匀,这样就实现了在所有火车中均匀分配计算任务的目标。图 5. 17a 的计算量明显高于图 5. 17b,因为在低前向值的情况下,火车需要更频繁地 进行最短路计算。

表 5.2 和表 5.3 提供了一些从仿真中采集的数据,以帮助我们理解前向值对主要性能指标的影响。"平均时间"是指火车实际行驶时间与理想行驶时间的相对数,它的值等于 {[(每列火车的实际行驶时间-理想行驶时间)对所有火车求和]÷火车的总数}。表 5.2 是低输入交通密度情况下的数据,表 5.3 则对应中输入交通密度。在表 5.2 中,对于高前向值,火车的平均行驶时间会增加,与其有关的平均等待时间也相应增加。平均等待时间的增加是由于 N 值的增加,增加了要预留的轨道数,火车就必须在启动预留请求的主站等待更长时间,同时也需要与更多的车站通信。但是,平均中继段轨道数却随着前向值的增加而减少,而火车折返行驶的次数减少幅度更大。低前向值和高前向值对各性能参数影响的对比在表 5.3 中表现得更明显。

前向值	路段使用 率 (%)	总中继段 轨道数	总折返行驶 的次数	每列火车使用的 平均中继段数	平均时间	平均 行驶里程	平均等待时间 /时间步数
1	27	2313	321	5. 039	122. 11	620	113
2	27	2053	17	4. 483	151. 91	627	119
3	27	2026	5	4. 443	164. 24	619	137
4	27	2029	3	4. 450	170. 56	627	136
5	27	2046	2	4. 458	182. 01	620	151

表 5.2 低输入交通密度下不同前向值对 RYNSORD 性能参数的影响比较

表 5 3	中等输入交通密度下不同前向值对 RYNSOI	PD 性能参数的影响比较

前向值	路段使用率(%)	总中继段 轨道数	总折返行驶 的次数	每列火车使用的 平均中继段数	平均时间	平均 行驶里程	平均等待时间 /时间步数
1	52	4656	1274	6. 151	545. 94	647	615
2	48	3428	144	4. 565	688. 95	619	647
3	46	3194	12	4. 357	759.77	610	711
4	45	3186	9	4. 341	769. 62	604	728
5	45	3137	3	4. 274	780. 01	603	737

对于低前向值情况,火车的平均行驶时间和平均等待时间明显较低。然而,火车的平均行驶里程、平均中继段轨道数和路段的使用率却较高。此外,火车折返行

RYNSORD 的一个基本目标是通过将总体路径计算任务分配到全部实体中去,而尽可能使网络内部的通信量减到最少。图 5.18 显示了每个站间通信链路的最大通信速率。鉴于仿真的分辨率为 1 个时间步长或实际操作的 1 min,这里提出的数据分辨率也限制为 1 min。在图 5.18 中观察到的数据传播的最大速率是 500 B/min,这通过传输速率为 9600 bit/s 或 19200 bit/s 的商业无线调制解调器可以很容易地实现。这样也就实现了 RYNSORD 的一个基本目标。相比之下,集中式算法理论上需要更高的通信速率,这意味着通信系统接口的成本会十分昂贵。

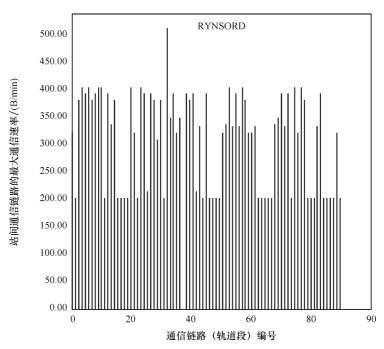


图 5.18 站间通信链路的最大通信速率

5.6 RYNSORD 的局限性

目前 RYNSORD 实施过程中的一个局限性在于它不能模拟轨道发生意外故障的情况。可以想象得到,轨道故障很可能会导致局部的严重拥堵,并会蔓延到网络的其他部分。虽然 RYNSORD 允许火车使用拥堵信息重新规划路线,但是它在发生轨道故障时的性能表现还需要进一步研究,这个问题将会在第7章中进行讨论。

第6章 DICAF: 一种智能车路 系统的分布式、可扩展结构

6.1 引言

根据美国智能运输协会的研究^[24],美国的路面交通正走在一个重要的十字路口。2 亿辆汽车严重阻塞了全国 4000000mile 铺装道路,并且阻塞状况还在不断恶化。事实表明,以修建更多的道路来缓解拥堵的传统观念在经济和环境上都已行不通。美国每年因拥堵带来产能下降、能源浪费、机动车辆损坏以及由于车辆怠速导致排放物持续增长等方面的损失为 1000 亿美元(Frank Kreith, 2000 年 5 月 16 日)。此外,仅在 1993 年,交通事故就造成 40000 人死亡、5000000 人受伤。针对这些问题,美国国会于 1991 年通过了综合地面运输效率法案(ISTEA),该法案的主要目标就是要建立一个经济上高效、环境上可持续发展的、能充分利用能源完成人货运输的国家交通运输系统。为此,美国运输部启动了智能车路系统(IVHS)计划。IVHS 并非为了解决通行能力问题,其目的在于协助并引导驾驶员避开交通瓶颈,推行一些合理的强制性管理措施,如拥挤收费等。道路交通问题并不仅仅出现在美国,实际上,在欧洲、日本这些人口稠密的地区和国家,交通问题更加严重。欧洲高效交通和安全计划(PROMETHEUS)^[45]、日本的先进车辆交通信息与通信系统(AMTICS)计划^[46]与美国的 IVHS 非常相似,并且几乎是同时推出。

事实上,每位驾驶员们几乎都对当前州际及州内的高速公路系统状况感到不满,该系统存在的问题是导致拥堵的主要原因。King^[47]指出驾驶员在不断寻找正确的路径方面造成的浪费,在价值上相当于非经营性车辆全程里程的 6.4% 和总时间的 12%,这些费用高达百万美元。罗得岛运输部(RIDOT)^[48,49]估计,所有车辆小时损耗的 60% 都来源于交通事故、车辆延误以及其他路面小事故,这些事故都是动态且无法预知的。其他州的运输部也有类似的报道。Perters、McGurrin、Shank和 Cheslow^[50]认为: ITS 体系必须将道路的车辆处理能力提高 30%,才能避免拥堵的继续恶化。造成拥堵的其他原因还包括路面维护、建设以及其他特殊事件等这样一些可预测因素。一般来说,人们普遍相信,准确获取道路相关信息是解决交通问题的一剂良药。

我们来考虑以下三个典型场景:①假设一个司机从 24 号出口附近进入一条高速公路,行驶不到 1 mile 就遇到堵塞,堵塞情况十分严重,但车辆排队只延伸到 22

号出口。如果司机在进入高速公路之前就已经知道这个信息,他/她就可以很容易地选择其他迂回道路,在避开拥堵路段后再进入高速公路。②一个驾驶员经过了一个加油站,行驶 20mile 以后才发现燃油不够,此时大部分加油站已经关门。如果这个驾驶员一开始就意识到这个问题,他/她肯定会在经过最后一个加油站的时候就加满油了。③一个驾驶员要在星期天从新泽西回到位于罗得岛的家中。现在是下午1点并开始下雪,根据天气预报,傍晚7点后还将有大雪。假定驾驶员的必经之路 I-95 畅通无阻,他大约需要在 5h 内行驶 240mile,才能在 6点左右到达目的地。而实际情况是,当司机一进入 I-95 州际公路就发现路面堆满积雪,就连停到路边上清除挡风玻璃上的冻雨都很困难。汽车平均行驶速度只有 25mile/h,这个司机最终被困在大雪中 10h 后才回到家中。在上述这些情形下,如果驾驶员能提前获得准确的信息,就可以合理计划行程,这样不仅能保证行驶的安全和经济,也能避免造成拥堵。

除了在交通管理与控制方面有大量的研究成果外,近期的文献也报道了许多在 ITS 方面的研究进展。Haver 和 Tarnoff^[51]构建了一个新的、有效的交通管理系统, 该系统能利用微处理机和本地局域网实现在线信号优化。Fenton 和 Mayhan^[52]报告 了他们针对自动高速公路系统中的控制概念和全方位控制器的研究和进展情况。 Powell^[53]总结了当前用于对汽车运输公司驾驶员配置进行优化和实现动态车队管理 的主要工具,这里的动态车队管理是指对取货和卸货计划以及路径的管理。他发现 传统的车辆路径问题是非常复杂的数学问题、目前用于解决该问题的各项技术区别 不大。Batz^[54]报告了TRANSCOM的使用情况,它是一个实时交通信息系统,纽约/ 新泽西大都市区中14个交通运输管理机构正在使用该系统,它也被用于物流行业。 当事故发生时,TRANSCOM 可通过一条长度为80 个字符的消息报告事故发生的时 间、地点以及解决事故需要的预计时间,各汽车运输公司获取该信息后,分析该事 故对他们所有车辆的影响,并将处理信息及时传送出去。该系统目前正在评估过程 中,人们期待它能有效提高车队的生产能力、改善车辆行驶环境以及提高客运服务 质量。Roper 和 Endo^[55]总结了圣莫尼卡智能交通走廊项目的进展情况、这个项目 的主要目的是更充分地利用各种道路设施来平衡交通流,并提议建立一个集中的城 市快速路交通控制中心 CENTRAL,该中心通过各种渠道采集交通数据,并以广播 的方式,通过车载频道25向途中车辆实时传递信息。车载处理器每分钟从控制中 心接收一次包含有路段拥挤数据的广播信息,并从这些信息中提取拥堵发生的位置 和车辆行驶方向等数据。在控制中心,有一个专用的工作站对每辆测试车辆进行追 踪,以监测车辆的路径选择和驾驶员的转向情况。这个计划预计能使每百万辆车的 年出行总时间降低 15%, 使高速公路的总行驶时间降低 12%, 并使高速公路平均 速度从 15~35mile/h 提高到 40~50mile/h。

在交通事故管理计划[48,49]中, RIDOT 希望能通过交通广播向公众发布可预知

的可能会引起拥堵的动态信息(如道路施工等)。为了能够获得精确的信息,RIDOT计划利用计算机、调制解调器和传真机组成的系统,把可视化的空中交通巡逻信息、视频摄像信息、RIDOT 地面浮动车采集信息、公共安全监控信息以及紧急车辆获得的信息等以一个标准的信息交换格式进行信息整合,还将建立一个 24h 免费电话事故报告机制和一个免费 "SP" 手机专线。除此之外,RIDOT 还计划对现有的交通检测线圈技术进行改进,并希望能研究出更好的事故高频区检测方案。

Kremer、Hubner、Hoff、Benz 和 Schafer^[56]提出了一个 IVHS 专用短程移动无线 通信网络,类似于 LAN's,并介绍了一个仿真系统 MONET3,该系统可以在上百个 工作站组成的网络中对协议进行评估。这个网络最重要的一个优点就是大多数交通 数据可由当地提供而不需要使用全球通信。Sakagami、Aoyama、Kuboi、Shirota 和 Akeyama^[57]提出了一个在多路径环境下根据多波束天线接收到的电波角度来确定汽 车位置的方法。当 GPS 信号被高楼阻隔时,这个方法可以帮助我们追踪到 GPS 信 号丢失后的车辆。Hussain、Saadawi 和 Ahmed [58] 提出了一种通过实验性的高空红外 光学系统来检测和监督交通状况的方法。这种方法不仅能成功地检测出汽车数量, 还能适应各种天气,具有较好的经济成本效益。Kim、Liu、Swarnam 和Urbanik^[59] 提出了一个区域交通控制系统 (ARTC), 在这个系统中, 信号控制器之间通过频 繁交换交通流信息,能有效控制拥堵的发生。该系统成功地对定时控制策略进行了 优化、并提高了系统的容错能力。根据相关原理、Von Tomkewitsch 提出了 ALI-SCOUT^[60],一个安装在车载计算机中的动态路径引导系统。汽车通过安置在信号 灯处的特殊的红外通信信标接收来自控制中心计算机的路径引导信息: 交通控制中 心的计算机则根据当前的交通状况确定路径树,即最佳路径;车载计算机接收路径 树信息后,根据自己的终点选择合适的路径。报告中用大量篇幅讨论了与红外通信 信标有关的关键技术问题,并指出该方法已经在700辆汽车上试验过。然而却没有 为该系统提供任何性能评价方法,也就是说我们并不能确定该方法是否能有效地扩 展,因为利用中央计算机产生路径树的方法很有可能会阻碍 ALI-SCOUT 的扩展。 Denney 和 Chase [6] 报告了分布式处理方法在圣安东尼奥市中心区交通系统中的使 用情况,其结果是产生了一种经济、可靠的开放式结构。Kline 和 Fuchs [62] 指出, 到当高速公路标志牌上符号标志的可见性明显高于相同尺寸的文字时,为了避免 过高的空间频率,可以通过使用光学模糊(如低通量)设计符号标志的方法加 以改进。Robertson 和 Bretherton^[63]描述了对交通信号进行实时优化的 SCOOT 方 法,它能使信号及时地自动适应新的车流模式。Bernard[34]提出了 ASTREE 铁路 交通管理系统,该系统使用了最新的、精确的分布式数据库以及路线布局和列 车进程的综合表达方式。然而,ASTREE 做出决策仍然是集中式的,虽然它使 用了分布式数据库中的信息。ASTREE 可以自动做出决策,也可以协助人工操作 员在路径设定和列车控制等方面做出决策。这些设定将会被下载到路边设备和机 车上。

加州大学伯克利分校的先进运输与公路研究所(PATH)的研究人员已经为IVHS设计了一种架构^[26],在这种架构里,一辆或多辆汽车会被组织成一些离散的车队,这些车队在当前高速公路的某个专用车道上[类似于高占有率(HOV)车道]高速行驶。当一辆车进入高速公路网络并告诉系统其目的地时,IVHS系统会通过网络把这辆车安排到适当的路径中。虽然这个方法已经测试成功^[64],但是仍有一定局限性:在高速公路上穿越一群低速车辆而进出 HOV 车道是危险的,而且多数驾驶员不喜欢被迫进行高速驾驶。Shladover 和他的同事们^[65]总结了他们关于车辆自动控制横向(转向)和纵向(调节间距和速度)驾驶的研究成果。Von Aulock^[66]报告了由卡尔斯鲁厄科技大学教授 Hiersche 主持的关于德国高速公路上使用自动诱导系统的可行性分析结果,研究表明:现有的高速公路和桥梁由于经济原因不可能专门为自动车辆引导系统进行新建或改建,因此,车辆以 120km/h 的速度沿着仅5m 的交织带进出系统时,很有可能会导致交通事故。

美国近期公布了由休斯飞机公司、劳拉联邦系统、罗克韦尔国际公司和韦斯廷 豪斯电气公司领导的 4 个国家级体系结构发展团队提出的 ITS 设计指南。休斯飞机 公司的方法是建立中央交通管理中心 (TMC),负责检测和分析事故,发布实时交 通信息和路径参数,并通过匝道流量控制、信号配时调整等手段来缓解拥堵。TMC 在区域处理器的协助下完成上述功能,区域处理器主要控制 TMC,并负责与车辆 之间进行交互的信标之间的通信。劳拉联邦系统的方法是利用全集成交通运输系统 的概念,考虑模块化和灵活的子系统,并支持开放式标准界面。罗克韦尔国际公司 提出了一个多层结构,并在应用层为每个界面推荐了界面标准。它也提到了 TMC 的建立,但有意避开了 TMC 的具体设计和配置情况。韦斯廷豪斯团队建议使用 TMC 和交通控制中心,为车辆提供集中式路径引导和其他信息。对于这些已提出 的体系结构,一个要考虑的重要问题是:这些结构都还需要科学实验、数学检验或 者模拟仿真的结果来支持。通过与联邦公路局(FHWA)的合作, Nynex 公司[67] 已经成功开发了一个交通管理系统,在这个系统里,利用集中式的 TMC 来为测试 车辆服务,这些测试车辆上装有专门的蜂窝式电话。运行测试车辆获得的初步数据 表明: TMC 为每辆车提供路径引导指令所需要的平均通话时间为 5~10min。 NYNEX 研究的车辆总数很小,而研究结果表明花费太高,不利于获得消费者的普 遍认可。一个更严重的问题是,通话长度还会随着与 TMC 连接的车辆数的增加而 显著增加。德国 VanGrol 和 Bakker 的研究证实了 NYNEX 对集中式 TMC 的研究成 果,即 TMC 执行动态交通管理和短期交通预测的速度也会逐渐跟不上需求的增加 速度。Igbal、Konheim 和 Ketcham^[69]指出:对公路走廊沿线的常发性和非常发性拥 堵的预测精度,受到走廊本身的一些静态特性以及已获得数据的高变化特性的制 约。Jing、Huang 和 Rundensteiner^[70]不仅认识到同时对大型交通网络中大批路径进 行计算的困难性,还认识到在计算机内存里存储大量待计算路径也是很困难的。他们提出了一种局部路径分层式编码方法,该方法可改进计算性能,提高空间的利用率。Talib、Love、Gealow、Hall、Masaki 和 Sodini^[71]提出把一个专用的高像素的并行处理器芯片安装到计算机上,使 ITS 系统能快速处理微光图像。Ziliaskopoulos、Kotzinos 和 Mahmassani^[72]提出了在 CRAY 超级计算机上快速执行最短路径算法的技术。为辅助进行出行路径选择,Rouphail、Ranjithan、EI Dessouki、Smith 和 Brill^[73]提出为出行前的路径规划开发决策支持系统,该系统能为根据路段行程时间定义的网络生成最多的不同路径。中央 TMC 的建立和维护是复杂而昂贵的。例如,明尼阿波利斯的 TMC^[74]有 48 个 17 in(1 in = 25.4 mm)显示器,控制着 354 个匝道测量仪,它从安置在高速公路旁并通过光纤连接的 142 个闭路电视摄像机接收数据,由 37个员工管理。Upchurch、Powell 和 Pretorius^[75]报告了沿亚利桑那州菲尼克斯市 256km 的干线走廊部署的一个闭路电视摄像网络的建设情况,该网络耗资 4200 万美元。

Chang、Junchaya 和 Santiago^[76]描述了一个在连接机 CM-2 上建立的交通仿真系统,并指出其性能有非常好的发展前景。Junchaya 和 Chang^[77]声称他们建立的仿真系统具备强大的路线处理能力,可以用来模拟驾驶员的路径选择行为。它们能够用16384 个处理器在 3.5 min 内以 1s 的间隔模拟 30 min 内 32000 辆汽车的运行。假设CM-2 是单指令多数据(SIMD)结构,即在步模式中每个处理器都执行一样的指令,这使得处理器变得非常简单,但这种仿真系统不能模拟复杂的、同时发生的、自主的以及车辆的个体行为。而且,SIMD 机器在本质上是同步的机器,因此它们自然不能模拟现实中的异步交通运输系统。

为了理解之前的研究成果,考虑一个高速公路系统如图 6.1 所示,它包括编号为 1~12 的 12 条高速公路,以及一个集中式 TMC。TMC 通过永久性链路与 12 条高速公路连接,如图中实线所示,它将现状信息传输到 TMC 中,并将匝道控制命令返回给高速公路路段。信息交换的频率是由交通流量大小决定的,该频率决定了任一给定时间现状系统中 TMC 的知识的准确性。当图 6.1 中编号为 1 的车辆进入该系统时,它与 TMC 之间建立一个临时通信连接,如图中虚线所示,并告诉 TMC 它的目的地。然后 TMC 充分考虑系统中其他汽车的数量、目的地,并根据在车辆总旅行时间最短条件下均衡使用资源的目标,做出路径选择,并将该路径选择信息发送给车辆 1。

上述研究成果存在一个明显的局限性:对于一个单一的集中式 TMC 来说,连续地为高速公路上行驶的上千辆汽车播报它们所需要的每一条数据信息,既不合逻辑,也不可行。而且,由于无线电发射器的电力限制、崎岖的地形等其他原因,一个单一的集中式 TMC 不太可能为大面积区域提供有效的服务。此外,单一的集中式单元不可能随时随刻都能获得每条公路路段上最精确的信息。而通过使用大量相对自主的本地通信单元却能很好地达到所需的精度。如果美国交通部将集中式

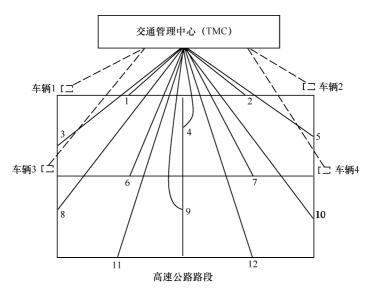


图 6.1 使用集中式交通管理中心的 IVHS 结构体系

TMC 作为 IVHS 的基础,假设未来汽车数量增加,拥堵也将相应增加,在未来,对复杂出行服务的需求、驾驶员对其他高速公路相关信息的需求量也将不断增长,这些都很可能会要求 IVHS 系统进行彻底的重新设计,因为单处理器计算机具有本质上的缺陷,不能满足这种要求。一个 TMC 由一系列计算引擎组成,即它一次只执行一个程序子任务,而且不管这个子任务有多复杂、多强劲,所以它的表现势必随着与之相连的汽车数量的增加而越来越差。

另一个局限在于:目前的研究要求 TMC 在了解所有相关道路拥堵状况的条件下能够完全转移和控制交通流。在某些特定情况下这可能有用,但是有时可能会引起驾驶员的不满。这里存在另外一些问题:第一,如前所述,我们可以肯定地说,TMC 不可能在高速公路所有路段上都一直能获得准确和最新的信息,尤其是当路段数量和汽车数量都很庞大的时候。第二,TMC 为车辆选择的路径可能不是最理想的,因为 TMC 无法考虑到每个驾驶员的特殊情况。比如,一个 TMC 指示某驾驶员走 I-95 公路穿过罗德岛,当前该司机在 5 号出口附近,为了避免 14 号出口附近的交通事故,系统指令他转道至 I-495 公路。相反,这个驾驶员由于开了夜车非常疲惫,如果他只是被告知事故,而不是被命令换道,他可能会选择直行至 12 号出口,然后花 1h 在餐馆享受一顿早餐,再继续前行。到那时,事故可能已经处理完毕。在这个充满独立思维的国家,人们普遍偏好那些能提供个性化选择的技术,而不喜欢那些统一的解决方案。Kawashima^[46]观察到:在日本,将 AMTICS 和 RACS融合在一起,能够通过车载计算机而非中央计算机实现路径诱导,但同时也指明这还仅仅是个概念模型。Dailey、Haselkom 和 Meyers^[69]准确观测到 ITS 的一个关

键技术是如何把动态数据实时发布给大量被授权的用户。他们建议使用一种异步分布式的客户/服务器结构,这种结构依靠一种自主的可重复使用的硬件设施来实现。Hall^[78]提出先进的出行者信息系统必须以引导用户使用可替代路径为目标,尽可能避免交通流的非均衡行为,提高用户出行的信心,保障用户出行的舒适性。

近年来,在美国许多城市,尤其是西雅图、休斯敦和洛杉矶等地,都会定期采集各条州际高速公路路段上的拥堵信息,并在网络上显示出来^[79]。一般利用 5 类信息(George H. Way Jr., 1992 年 2 月)就可以将拥堵状况全面地显示在地图上,即完全自由行驶的信息、拥挤严重的、一般拥挤的、走走停停的以及无法获得数据的信息。因为每辆汽车自己很难自动获取信息,所以这个方法提出了对 DICAF 的基本需求,这也是本章的中心议题。然而,这里存在一个严重的问题,就是存在信息延迟,即信息产生和使用时间之间存在差异,此差异对这类高度动态信息的时效性和准确性都有较大影响。

我们认识到这些问题并提出使用一种分布式策略,即 DICAF。DICAF 使用了与第 4 章和第 5 章提出的异步分布式算法相同的基本原理,包括使用局部性数据来计算决策,并随时随地地将变化,也就是新的信息,传送给系统中需要该信息的其他实体。Clark 和 Daigle^[80]对计算机仿真在交通工程中的重要性研究进行了综述,并强调了它在新的思路、算法和交通控制系统的发展和评价过程中的重要作用。本章剩余部分组织如下:6.2 节详细介绍了 DICAF 算法;6.3 节描述了 DICAF 在一个精确的、真实的、并行操作试验台上的仿真情况。6.4 节展示了 DICAF 在实验台上执行的细节;6.5 节阐述了对一个有代表性的交通网络在随机输入流量下的仿真情况,并对其性能表现进行了详细分析。

6.2 DICAF: 一种新颖、分布式、可扩展的 IVHS 方法

由于车辆的路径选择问题在交通运输行业应用非常广泛,同时在数学上又非常复杂,所以这可能是交通运输领域中最复杂的问题之一^[53]。Bodin 等^[81]回顾了过去 40 多年中 700 多篇参考文献,证明了这个问题的复杂性。正如之前提到的一样,对于单一的集中式 TMC 来说,连续不断地为在公路上行驶的数千辆汽车提供每辆车所需的公路信息既不合逻辑也不可行。而且,由于无线电发射器电力资源有限、崎岖的地形以及其他原因,单一集中式的 TMC 不太可能有效地为大面积区域提供好的服务。此外,还有一个众所周知的问题是,单一集中式单元并不是总能获得每个公路路段上的准确信息,只有通过大量相对自主的本地通信单元,才能较好地达到所需的精度。

DICAF 认识到这些问题并提出使用分布式体系结构,在该结构中,数据收集、

处理、信息传播以及决策等全部任务都被分配给 IVHS 系统中各个组成部分来共同完成。DICAF 的基本理念就是在实体之间智能化地分配决策任务,同时尽可能使本地计算最大化,全局通信任务最小化,最终实现系统的强鲁棒性,提高系统的生产能力。这种理念的一个直接产物就是良好的可扩展性,即随着车辆和公路路段的增加,系统规模的扩大,系统将继续正常运行并能保持良好的性能。DICAF 采用分布式体系结构的目的是通过向驾驶员及时提供准确而充分的公路数据来影响其对路径的选择,帮助他/她及早规划备选方案,而不是硬性规定驾驶员必须按他所憎恶或抗拒的系统指定路径行驶。Allen、Ziedman、Rosenthal、Stein、Torres 和 Halati^[82]认为在仿真模拟研究中,导航系统的性能对驾驶员路径转换行为的影响十分明显,好的系统能使驾驶员更充分地预知交通拥堵的发生。实验室中对驾驶员进行的仿真研究也表明当出行者能获得更多、更好的出行信息时,其驾驶的总里程数就会随之减少。

为实现上述目标,这项研究提出以下三点要求:①对构成交通系统的实体——车辆和路段,要了解并分析其基本需求;②确认实体之间需要通过通信来交换的必要信息;③确定适用于所有实体的一般模型。一般模型必须能够根据输入实体的有限信息做出独立决策,且每个决策必须与其他决策保持"一致",并共同遵守效率和安全性的总目标。本章提出的方法将会仔细确认决策和实体间通信的基本规则,以适应各种可能的情况。根据定义,由于不存在不必要的干扰和同步性,异步分布式算法理论上应能使并行处理获得最佳效果,它能最大限度使用来自不同实体的分布式信息。在一个交通运输系统的实际操作过程中,不同时间点的结果会是某些参数的函数,这些参数包括驾驶员意向的转变、汽车故障、路况变坏、交通事故等,以及其他不可预知的情况。由于单个实体不可能在任意时刻都能处理整个交通系统中准确和完整的信息,所以异步式方法允许每个实体在不影响系统安全有效地利用资源这一总体目标的前提下,尽可能快速地处理信息。异步式方法还能够识别并充分利用每个实体自身特有的能力。

目前的高速公路系统由两个部分组成——公路路段区间和车辆。公路路段区间是指高速公路上连续两个出口之间的几条路段组成的区间。本章所分析的公路指的是一般意义上的高速公路。在高速公路路段上,车辆不能随意改变路径。也就是说,假设2、3号出口间的一段公路路段区间上发生了严重的交通拥堵,正驶向2号出口的汽车可以先从2号出口离开而选择另外一条路径。如果这辆车没能从2号出口离开,就只能行驶至3号出口了。DICAF结构中介绍了高速公路的第三个组成部分——高速公路基础设施,这是该系统实现其目标的关键。基础设施分为不同部分,其中各个分布式交通管理中心(DTMC)在各自负责的地区范围内采集并传播信息。假如每辆车都能从 DTMC 获得所需要的信息,我们就认为这些车辆是自治的,然后车辆就可以利用这些信息进行决策。信息可以有很多格式。假设一辆车在

运输途中需要非常特殊的信息,它就向当前位置所在区域的管辖者 DTMC1 提出请求。这种请求可能不仅包含当地信息,比如附近某个高速公路路段区间上的拥堵信息,还可能包含非本地信息,比如离汽车当前位置非常远的高速公路路段区间的驾驶环境和天气,甚至可能是远离它当前位置的高速公路沿线的银行、邮局、医院、餐馆的地址及营业时间等。为回应这一信息请求,DTMC1 会立即向请求车辆传送数据。而如果所请求的信息超出其管辖范围,DTMC1 可通过网络先从正确的 DT-MC 处获得数据,然后再将其传送给请求车辆。本章着重论述路径引导的关键参数,即拥挤信息。而获得银行、邮局、医院及高速公路沿线餐厅地址和营业时间等信息的问题不在本章的研究范围之内。

在 DICAF 系统中,DTMC 的确切位置和数量是以下 4 类参数的函数: ①请求服务的汽车数量; ②车辆与 DTMC 间的平均通话时间,即传送信息所需的时间; ③所要求的服务等级; ④通信机制的服务范围,这取决于车辆和 DTMC 之间是使用无线通信还是红外信标通信。在无线通信机制中,有些最基本的选择包括标准的蜂窝式电话、蜂窝数字分组数据系统(CDPD)、被美国联邦通信委员会(FCC)搁置的专门为 FHWA 的 IVHS 定制的 220MHz 的无线电广播,以及其他设备。Kamali^[84]对能用于 ITS 的众多无线通信技术进行了全面综述。Sodeikat^[85]报告称,在德国 LISB 地区进行的演示实验成功证明了路边短程红外信标能支持高达 500bit/s的传输速度。DTMC 内网的设计也是交通流量和所要求的服务水平的函数。如果在城市区域内需要较高的带宽,那么一个中等带宽的网络就足以满足郊区的服务需求。

驾驶员的路径选择可能会受到天气、紧急情况、道路条件、疲劳、车辆条件以及其他主观和客观问题的影响。虽然 DICAF 给予每个驾驶员充分的自由,但是为了客观地评价 DICAF 算法,该方法还是假定每个驾驶员的基本目标是以最短时间到达最终目的地。当然也存在某些例外情况,特别是对于旅游观光或其他原因的出行。由于路线上的一个或多个路段区间可能会遇到严重的拥堵,因此驾驶员并不一定非得使用距离最短的路径。为了协助驾驶员,本章记录了一些观察和判断拥堵的方法。作为第一近似值,给定公路路段区间上行驶车辆的平均速度是个很好的测量指标,因为它能很好地反映路段区间上的生产能力;然而平均速度并不能帮助我们获得公路路段区间上行驶车辆的总量,这个量直觉上是一定会受到拥堵影响的;虽然平均车辆间距看上去能够反映路段区间上车辆总数的多少,但是就其自身而言,它既不能很好地度量流量,也不能用于判断拥堵情况,例如,它不能区分所有车辆都以50mile/h或100mile/h速度行驶这两种情况,因为这两种情况中车辆都一直会保持相同的间距;同样的道理,任何一个时刻公路路段上的车辆总数也不能体现是否发生了拥堵。因此,本章提出了一种新的判断拥堵的概念——公路路段拥堵程度的(C. M.),它能综合反映公路路段区间上车辆总数、车辆的平均速度对拥堵程度的

影响:

C. M. =
$$\frac{\sum \text{ r in Tyure}}{\sum \text{ r in y in d}} \times \left(1 - \frac{X}{Q}\right)$$
 (6.1)

式中, X 是某时刻公路路段区间上的车辆总数; Q 是路段区间上允许的最大车辆 数, $Q = 0.5 \times N \times \frac{D}{L}$, 其中 $N \times D \times L$ 分别是车道数、路段区间长度以及每辆车的 平均长度。D和L是估算值。在交通工程文献中,常用的两个术语"车头距离" 和"间隙"都可以用时间和距离来衡量和表达。车头距离指的是前车和后车车头 之间的差距,而间隙指的是前车车尾和后车车头之间的差距。在某些分析中,特别 是在路面街道车流在高峰时期产生严重拥堵时,如果前车是卡车,这个评价指标是 非常有用的。在本书的分析研究中, D和L把车头距离和间隙都考虑在内了。根据 推测,高速公路路段区间理论上在任一时刻最多能容纳 $N \times \frac{D}{I}$ 辆车,此时所有车辆 必须以相同的速度行驶。但作为工程估计值、Q、即高速公路路段区间在任一时刻 允许容纳的最大车辆数应近似等于为 $0.5 \times N \times \frac{D}{L}$ 。本章中假定所有路段的车道数 都一样。在驾驶员手册中通常可以发现这样一个经验法则:如果前后相邻车辆之间 有一个车身长的间距,就可以判断此时的车速为 10mile/h, 这一假定没有在上述 C. M. 公式中反映出来的原因是: 最大车辆数必须能够反映路段区间在任一时刻物 理上所能容纳车辆的最大值,而这个物理上的最大车辆数很明显是由路段长度和车 辆长度严格定义的,与车辆速度没有必然关联。所以,相比之下,式(6.1)提出 的定义更具一般性,因为这里采用"最大车辆数",就包含了严重交通拥堵的情 形,同时,它还允许出现比经验法则估计值更高的车辆密度。如果不考虑拥堵情 况,就很难使用 DICAF 仿真分析系统在极端情况下的行为。经验法则,正如字面 上的意思、仅仅是一种依靠经验的估计方法、既没有任何正式的理论基础、也不能 严格服从,只能作为参考,在极度拥堵的情况下,它通常是不适用的。

选择式 (6.1) 中定义的另一个合理依据是,在一般情况下,每辆车的速度都是不相同的,一个众所周知的证据就是,通常左侧车道会被设计为高速车道,而右侧车道一般是为低速车辆设计的。此外,在任何高速公路上,高速车辆超过低速车辆都是非常常见的事情。当路段区间上的汽车数量较少时,即使有许多车辆由于各种原因而低速行驶,高速车辆仍可以保持较高的行驶速度,因此,整条路段区间上的平均速度仍可能较高。而当车辆数增加,车辆间距减少时,这种情况会发生较大改变。高速车辆超越低速车辆会变得越来越难,而且也不安全,导致高速车辆不得不减速,最终使路段的总平均速度下降。因此车辆数的不断增长会使路段拥堵情况

恶化,C. M. 值减少。在发生了严重交通事故的情况下,平均速度可能下降至 0,C. M. 值也会降至 0。同样,当 X 值等于 Q 值时,C. M. 值也是 0。假定车辆都以不同的速度行驶,就意味着所有车辆最终将达到一个静止状态。本章假定车辆的速度服从一个均值为 μ 、方差为 σ 的正态分布,它是由特定路段区间情况和驾驶时间特性等因素决定的,同时指出当 C. M. 的绝对值较高时,拥堵水平较低,而较低的 C. M. 值表示较高的拥堵水平。最大 C. M. 值由路段允许的最大车速决定,最小值是 0。显然,式 (6.1) 中关于 C. M. 的定义并不适用于所有车辆以相同速度行驶的情况。有人曾假定车辆速度服从著名的泊松分布,但要注意的是,虽然泊松分布更容易进行数学处理,但是正态分布更容易生成表格,使 DICAF 仿真能够更顺利进行。还要指出的是,仿真过程假设在某个典型场景中进行,要将全年不断行驶的车辆,与一天中某个时间以及一年中某一天内行驶的车辆进行对比,为此,本研究中使用了连续的正态分布。Kreyszig [86] 进一步指出,对于较大的 N 值,即一次实验中需要独立操作的次数较多的情况,二项分布可由正态分布估计得到,而泊松分布只是二项分布中的一种特定情况。

正如本章前面提到的,大多数传统的路径引导方法采用的是二元准入策略,即根据拥堵程度决定允许进入或拒绝进入。众所周知,二元策略^[87]通常会导致武断的决策。与之相反,在本章中,C. M. 值会逐渐演变成一个路段区间上车辆数和平均速度的函数。C. M. 是一个连续函数^[86],因为对于任意给定路段区间,该值都可以被定义或假设为(0mile/h,路段允许的最大速度值)之间的任意值。无论拥堵水平如何,都不会有车辆被拒绝进入该路段区间,除非 X 值等于 Q 值。我们期望尚不在该路段上的车辆都能避开 C. M. 值较低的路段,而转向走 C. M. 值较高的路段。因此,如果在对每辆车进行路径决策时,不采用武断的做法,路段上车辆的分布就可能更加均衡一些。

图 6.2显示了一个高速公路系统的 DICAF 体系结构,该系统包含 12 条高速公路路段区间,以及 9 个吸引车流进人系统的交叉口。DICAF 把高速公路系统分成 9 个区域,每个区域都由唯一的 DTMC 分别控制。虽然图 6.2 中 DTMC 都正好安装在交叉点上,但是在现实世界中,它们可能会被安装在所控制路段附近的某一个地方。途中,DTMC1 控制路段 1 和 3,DTMC2 控制路段 2 和 4,DTMC8 控制路段 12,DTMC9 不控制任何路段。所有 DTMC 都是通过一个广域网连接,网络中的链路平行于高速公路路段。车辆一进入系统,就与当地 DTMC 建立联系,获得相关路段的 C. M. 值,然后它会以快速到达目的地为目标选择路径。当车辆到达下一个 DTMC 时,会重新计算路径,这一过程不断循环直到到达最终目的地为止。某一路段的 C. M. 值在控制该路段的 DTMC 内是最准确的,但当车辆向离它距离较远的其他 DTMC 请求数据时,C. M. 值可能会由于延迟而导致精确度降低。因此,车辆在驶向目的地的途中逐渐获取越来越准确的路段 C. M. 值,其结果是,车辆通过持续改

进路径, 最终实现效率的最大化。

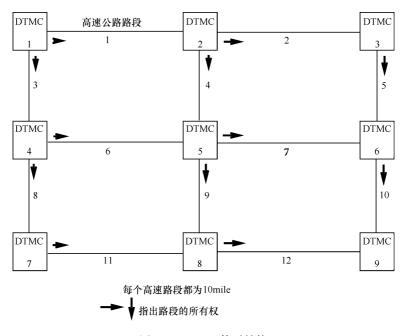


图 6.2 DICAF 体系结构

每辆车都拥有一个高速公路系统的完整的静态拓扑结构数据,即路段的数量、长度以及路段之间的连通性。然而,车辆并不知道路段的 C. M. 值,因为这个值是动态的。本章假定静态信息不会发生变化,且不考虑施工或事故导致的道路损坏问题。当车辆在某个 DTMC 的管辖范围内时,它会从中获取相关路段的 C. M. 信息,然后,车辆执行修正的 Dijkstra 最短路径算法^[88],其目的是为了选择一条从当前位置到终点的估计行驶时间(ETT)最小的路径。一个路段区间上某辆车的 ETT 的值由下式表示:

式 (6.2) 中,由于以下原因,ETT 只能是一个估计值:对后续路段 C. M. 值的测量是动态的,如果车辆并未在该路段终止行程,则该路段的 C. M. 值可能并不等于车辆在该路段上实际行驶过程中计算 EFT 时使用的 C. M. 值。此外,由于 C. M. 值仅是拥堵水平的一个指标,本章作如下假设:如果车辆准备开始在该路段上行驶,且 C. M. 值低于车辆的期望车速,可允许车辆以不超过 125%的 C. M. 值,或自身的期望速度行驶。如果 C. M. 值高于期望车速,只允许车辆以期望车速在该路段上行驶。

反过来,每个 DTMC 都会计算它所控制路段的 C. M. 值,然后通过泛洪算

法^[88],将该值传递给其他 DTMC。为提高效率,只有当某路段上的 C. M. 值与其他部分所传递过来的值不同时,该 C. M. 值才能被传递出去。当一个 DTMC 接收到来自其他 DTMC 的 C. M. 值时,它会更新 DICAF 系统中的本地记录,并利用泛洪算法^[88]将其传递出去。本地记录中的相关信息也会被传递给它管辖范围内的请求车辆。DTMC 利用路段上的车辆数和任一给定时刻下该路段上的车辆速度来计算 C. M. 值。同样考虑到效率问题,只有在有一辆或多辆车进出该路段时,该算法才会被激活。当车辆进入某 DTMC 的管辖区并要求信息时,该 DTMC 会为进入的车辆登记以激活 C. M. 的计算。当车辆离开 DTMC1 的管辖区进入 DTMC2 管辖区时,后者一旦登记了该车辆的进入,就会第一时间通知 DTMC1。然后,DTMC1 会重新触发 C. M. 的计算。

每个 DTMC 和车辆的功能分别在图 6.3 和图 6.4 中用伪代码表示出来。

```
While (simulation is not complete) {
   Check for flooding message from neighbors
   If (a message with higher sequence number) {
     Update C. M. of segments for other DTMCs stored locally
     Propagate the message to other neighbors
   Check for vehicles entering any highway segment it owns
   If (a vehicle has entered) {
     Communicate with the vehicle, download C. M. measures
     Upload vehicle's actual speed
     Determine when vehicle will exit segment based on its actual speed
     Add vehicle to the wait-to-leave list
     Update C. M. of the affected segment
     If (change in C. M. exceeds a specified threshold)
         Send flooding message to other neighbors
  Check the wait-to-leave list
  If (a vehicle exits the segment)
    Update the C. M. of the affected segment
    If (the change in C. M. exceeds a specified threshold)
       Send flooding message
```

图 6.3 DICAF 中 DTMC 的功能

```
While (destination not reached yet) {

If (entered the jurisdiction of a DTMC) {

Communicate with the DTMC, download appropriate C. M. values

Compute maximum permitted speed

Set actual speed equal to maximum permitted speed

Upload actual speed to DTMC

Compute best route and execute travel

}
else {

continue travel
}
```

图 6.4 车辆的函数

DICAF是可以扩展的,即随着系统的发展,车辆数和 DTMC 数量的增加,DICAF仍能继续运行,且性能也不会减弱。随着系统规模的扩大,计算车辆路径的任务也大大增加,计算实体的数量也会同比例增加,仍可以共同完成该任务。相反,传统的 TMC 在类似的条件下,连续不断地执行算法的集中式计算机很快就会被不断增加的计算负荷压垮。 TMC 需要昂贵的中央超级计算机,而 DICAF 却是经济的,每个 DTMC 都配备有便宜的最新式微处理器,DTMC 计算单元的负担明显轻很多。此外,DTMC 中的计算机维护费用也非常低。DICAF 中的 DTMC 可以是全自动、无人操纵和独立存在的,非常类似街角的交通信号控制器。车辆的计算单元由廉价的微处理器,比如英特尔 8086、摩托罗拉 6809 等组成。与传统的 TMC 类似,如果出现更强大的处理器,DICAF 中的 DTMC 计算机也可以很快更新。DICAF 非常稳定可靠,如果一个或多个 DTMC 无法运行,剩下的 DICAF 系统还会继续运行,而传统的 TMC 如果无法运行,则会导致整个系统完全瘫痪。在无法运行的 DTMC 管辖区内的车辆会被暂时排除在 C. M. 值计算之外,但当车辆进入其他正常运行的 DTMC 管辖区时又会恢复正常的操作。

6.3 在一个精确的并行处理试验台上进行 DICAF 的现实模拟

DICAF 的主要贡献在于通过网络把全部车辆的路径计算任务分配给 DICAF 中的所有实体——车辆和 DTMC 来共同完成。因此,事实上,对于一个包含 V 辆车和 D 个 DTMC 的 DICAF 系统来说,共同参与计算的单元总数是 V+D。为了理解 DICAF 的性能和不同因素对它的影响,我们首先要对 DICAF 建模,然后在一个并行处理试验台上进行仿真,该试验台是由配置松耦合并行处理器的工作站网络构成

的,在试验台上进行的仿真非常类似于实际操作。为了能对一个拥有一定数量汽车的实际系统进行真实仿真,每个 DTMC 都表示成工作站,车辆则被模拟成任务,然后由工作站在 DTMC 指挥下来运行。当车辆沿着高速公路路段行驶时,它的计算就由控制该路段的 DTMC 中的工作站来进行。每辆车都表示成一个以消息形式从这个 DTMC 转移到下一个 DTMC 的进程。同时车辆信息用一个数据结构来表示,该数据结构的字段中存储了车辆的重要参数,这些字段包括推荐速度、实际速度、起点和终点。当车辆从当前的 DTMC(A)行驶到另一个 DTMC(B)时,A 工作站中对应进程的重要参数就被压缩成消息传递给 B 工作站,并在 B 工作站中以一个进程的形式再次出现。因此,在仿真过程中车辆是以电子速度而不是物理速度移动的。对路段上的行驶过程来说,车辆路径的计算子任务由控制该路段的 DTMC中的工作站来完成。因此,DICAF 能够以比实际快很多倍的速度对高速公路系统进行模拟。

本章假定了一个典型高速公路系统是以网格形式组织的。DTMC 安装在每个交叉口处,虽然看起来它应该可以控制最多 4 个方向的路段,即与最多 4 个路段连接,但在实际中它最多只可能控制位于它东、南两个方向的路段。对于安装在高速公路系统网格图底部的 DTMC 而言,是没有南向路段的。同样,对于安装在公路系统网格图最右端的 DTMC 来说,是没有东向路段的。因此,它们能控制的最大路段数是一致的。

每个 DTMC 都以下列数据结构表示它所控制的高速公路路段区间的重要信息:根据 DTMC 在网格中的位置,用第 1 个字段 "index"存储 DTMC 的唯一标识符。而第 2~5 个字段与 DTMC 东面的高速公路路段区间相关,第 6~9 个字段与 DTMC 南面路段相关。对于东向路段,C. M. 值以 mile/h 为单位存储到字段 "eastCM"中。如果路段不存在,该字段会被赋值为 -1。"eLastSent"存储的是某路段的最新 C. M. 值,该值刚刚被传递给了其他 DTMC,"eSeq"指的是该值的序列号。字段"eCar"存储的是当前行驶在路段上的车辆数。最后4 个字段 southCM、sLastSent、sSeq 和 sCar 分别与 eastCM、eLastSent、eSeq 和 eCar 相对应。

```
Type def struct DTMC {
    int index;
    float eastCM;
    float eLastSent;
    int eSeq;
    int eCar;
    float southCM;
    float sLastSent;
```

```
int sSeq;
int sCar;
```

除此之外,每个 DTMC 都不断地为 DICAF 中所有其他路段更新 C. M. 值。这种更新是根据从其他 DTMC 中周期性接收到的泛洪消息而进行的。当一辆车与一个 DTMC 连接时,为了确定路径,它会下载一部分数据库。为提高内存的使用效率,每个 DTMC 都有一个规格为 MAX DTMC 的指针矩阵,如下所示:

typedef struct DTMC * DTMC_PTR;

DTMC_PTR DTMC [MAX_DTMC];

在初始化过程中,为了存储 DICAF 所模拟的高速公路系统中所有路段的 C. M. 值,会动态分配一些额外的内存。

DICAF 使用两种消息类型: 一种是将车辆信息封装起来, 并模拟车辆从一个 DTMC 转到其下一个 DTMC 的过程,另一种是为路段传递更新了的 C. M. 值。被用 来封装车辆信息的数据结构中,字段"origin"存储车辆出发地 DTMC 的标识符, 而字段"dest"存储终点 DTMC 的标识符, 车辆的唯一标识符存储在"id"中。该 标识符由生成车辆的 DTMC 分配,是由 DTMC 标识符和该 DTMC 产生车辆的序列 号组合而成的。字段"update"和"from"存储的信息用于指引接收 DTMC 更新关 于 C. M. 的本地信息以及与路段有关的其他信息。如果车辆当前行驶路段属于该接 收 DTMC 的管辖范围,就会被标上"update"标志。字段"from"是用来区分 DT-MC 的东向或南向路段是否需要更新的标志。字段"inserted"包含车辆进入该路段 时间,字段"appSec(以s为单位)"和"appMSec(以ms为单位)"共同存储了 车辆按计划应该出现的时间,即到达终点 DTMC 的时间。可以证明,它可以根据 路段长度和车辆沿路段行驶的速度计算获得。字段"trvSec(以s为单位)"和 "trvMsec(以 ms 为单位)"则共同存储了车辆从起点到达当前位置的实际累计行驶 时间,字段"idealTrvSec(以s为单位)"和"idealTrvMSec(以ms为单位)"共 同存储的是车辆从起点到终点需要的理想行驶时间,即理论上的最少运行时间。字 段"speed"存储的是车辆沿当前路段行驶的速度,字段"prefSpeed"是指期望 车速。

```
typedef struct Vehicle {
    int origin;
    int dest;
    int id;
    int update;
    int from;
    long inserted;
```

```
appSec;
long
       appMSec;
long
       trvSec:
long
long
       trvMsec;
       idealTrvSec:
long
long
       idealTrvMSec;
float
       speed;
float
        prefSpeed;
```

每个 DTMC 中的工作站的工作过程都在图 6.5 中以虚拟代码的形式表示出来。

```
main {
    Synthesize the DICAF network;
    Generate vehicles and include in wait-to-leave list;
    While (simulation time is not expired) {
        Check incoming message(s)-may be encapsulated vehicle or C. M. update;
        Process incoming message-process vehicle or forward C. M. updates;
        Check wait-to-leave list;
        Process vehicle(s) off from the list;
        Propagate updated C. M. values;
}
```

图 6.5 DICAF 中的"main"流程

在图 6.5 中,前两句是与初始化过程有关的。在初始化期间,DTMC 中的工作站通过软件链路来相互连接,以使最后形成的网络能与实际路网相对应。此外,DTMC 生成的车辆都拥有唯一的标识符和随机分配的终点。当期望车速为随机值时,服从正态分布。生成的车辆先放在 DTMC 的等待车辆列表中,并根据车辆进入路段或者沿着正确路段离开 DTMC 的时间先后顺序进行组织。DICAF 仿真可以一直持续进行,直到仿真时间结束为止。仿真步骤分为以下几个阶段:第一阶段,读取输入的信息,并利用函数 Process_Message 进行处理,如图 6.6 所示。在这个函数中,根据获得的泛洪消息产生新的消息,即更新的 C. M. 值,在更新了本地数据库后,该消息就继续传递到其他 DTMC 中。如果输入的消息是一辆封装后的车辆,它在当前的 DTMC 中会以车辆-进程的形式重新出现,并被加入到等待离开的车辆清单里。第二阶段,检查等待离开的车辆清单,以确定某辆车在当前时间下是否必须离开 DTMC。计算车辆路径,并以消息的形式封装起来,传递给下一个

DTMC。同时,受影响的高速公路路段的 C. M. 值要被重新计算并传递给其他 DT-MC,该过程一直持续到仿真过程终止为止。

```
Process_Message {
    if (Message relates to flooding)
        Receive_Flooding;
    else {
        if (Vehicle commences travel of segment owned by current DTMC) {
            Update segment;
        }
        Insert vehicle into wait-to-leave list;
    }
}

Receive_Flooding {
    if (Message sequence number > previous sequence number of the segment) {
        Update segment-update C. M. & advance sequence number;
        Forward flooding messages to all its neighbors;
    }
}
```

图 6.6 DICAF 中的 "process message" 流程

图 6.7 中的伪代码详细说明了函数 "check wait-to-leave list" 的组织过程。在仿真期间,如果当前时间就是等待离开车辆清单中的某辆车离开 DTMC 的时间,首先会从清单中抽出该车,然后利用路网中最新的路段 C. M. 值确定该车的路径,接着就将该路径消息传递给所选路径上的下一个 DTMC。对于某个给定的路段,车辆无论进入还是驶出这个路段,C. M. 值都会发生变化,这就使清单中包含了两类车辆—①标记为"正常"的进入该路段的车辆;②正驶出该路段的"影子"车辆。由于只有 DTMC 的管辖者才能为路段维护 C. M. 值,影子车辆的进出能帮助确保 C. M 值在车辆完成该路段行驶任务后得到及时更新。

如果更新后的路段 C. M. 值与之前的值完全不同,DTMC 的管辖者必须通过泛洪算法将信息传递给其他 DTMC,对应的函数是 "Update_segment"。在这个过程中,会给每条消息赋以一个序列号以避免重复传输。当消息从原 DTMC 传播到相邻的 DTMC 时,该 DTMC 使其序列号增大。当此条消息被另一个不同的 DTMC 接收时,只要该消息的序列号比上一个 C. M. 值对应的序列号更新,接受 DTMC 就会将此消息传递给与它相邻的 DTMC。该算法以伪代码表示,如图 6.8 所示。

```
Process_vehicle {
   if (entry refers to a "shadow" vehicle)
     Update segment since vehicle reaches the end;
   else {
     if (current segment used by vehicle is owned by DTMC)
        Update segment since vehicle exits the segment;
     Determine best route and the subsequent DTMC;
     if (this DTMC is the final destination of vehicle)
        Record relevant statistics in vehicle's data structure;
     else if (subsequent segment is owned by current DTMC)
         Determine the vehicle's speed on the segment;
         Update segment since vehicle enters the segment;
         Create a "shadow" entry for time when vehicle will complete travel
         Include entry in the list with shadow flag set
       else if (current DTMC does not own highway segment)
          set the "update" field in vehicle's structure
       Propagate the data structure to the next DTMC:
                       "Check wait-to-leave list"函数
    Update_segment
       if (vehicle exits highway segment)
          Decrement number of vehicles on highway segment;
       else
          Increment the number of vehicles on highway segment;
       Compute new C. M. value;
       Update local copy of the C. M. value;
       if (C. M. differs from previous value by a margin exceeding threshold)
          Propagate C. M. value through Send_flooding;
    Send_flooding {
       Increment sequence number associated with the segment;
       Initialize message with sequence number, segment ID, and C. M. value;
       Propagate message to all the neighbors;
                    DICAF 中 "Update_ segment" 函数
```

6.4 实施操作和调试

DICAF 模型和仿真器都是用 C 语言编写的,通过带宽为 10Mbit/s 的以太网连接,在基于 Unix 工作站的网络上执行,并配备有松散耦合的并行处理器。该系统包括 65 个装有 SUN Solaris 2.4 操作系统的 SUN Sparc10 工作站。关于每个 DTMC 以及在该位置进入系统的车辆在内的信息代码段大概为 4432 行,用 C 语言编写,网络代码约 1500 行,由 C ++ 语言编写。仿真器由 GNU C 编译器以及 gcc 进行编译,并以"nice"效用在较低的优先级上执行。DICAF 的执行会产生将近 4 ~ 9.5 MB 的数据,然后由解析器解析得到性能曲线图。解析程序包括 910 行 C 语言代码。每个仿真都与某个实验相对应,模拟高速公路上 24h 的真实交通情况。对于10 处理器和 51 处理器的两种 DICAF 系统,仿真时间都需要约 16 min。输入 DICAF 系统的车辆最大数为 45000 辆,试验台中有 65 个同步执行的 SUN Sparc10 工作站,每个都有 32 MB 内存和 42 4 MB 硬盘。此次研究共进行了 400 次仿真。根据记录,工作站在各自的控制台上为用户完成了最主要的工作。

异步分布式算法的一个特有的特征(以 DICAF 为例)就是调试极其困难。据估计,假设 DICAF 有将近 4000 行程序,那么调试一个有 50 个节点的 DICAF 仿真程序就相当于要调试 4000 × 50 = 200000 行 C 程序代码。然而,实际上问题还要更加严重,在操作过程中的任一时刻,每个处理器都可能只执行 DICAF 程序中的某一行,且每一步之间都没有任何相关性。人们一般就此认为对多重自治实体进行综合同步操作是非常困难的。可以想象,当人们打算在代码中添加一条 printf 语句来显示执行过程中的变量和结构的关键值时,却发现变量值的输出打印次数很快超过1000,并且还在迅速增加。与时间有关的问题要求更精确,调试起来也更加困难,这是因为如果在代码中加入 printf 语句,代码就会变得不连续。根据作者的经验,高效的调试需要对整个程序有极为深入的了解,并要求调试人员具有高度的专注力、责任心和耐心。

异步分布式算法同样对操作系统的精确性和正确性有较高要求。首先,这些算法要求能快速传递中小尺寸的消息;第二,消息的分布具有较高的突发性;第三,假设每个处理器除了计算和传递拥堵信息以外,还得为 45000 辆车进行路径选择计算,那么消息分布的突发性将达到极限,这将会导致频繁地写入缓冲区,使其长期处于饱和状态。据观察,有时候仿真会意外终止,错误显示为无法写入、通道断裂或写入数据错误等。还有其他一些难以解释的错误,包括成功打开,处理器却无法将数据写入文件等。

在仿真过程中,车辆收集并存储自身的出行信息,包括路径、C. M. 值以及出行时间等。当车辆到达终点 DTMC 时,数据即被写入到一个文件中。解析器根据

此文件生成性能表现统计数据,包括车辆在整个行程中的平均速度、实际出行时间、理论上的最短出行时间、车辆实际速度分布、随车辆标识符和车辆进入 DICAF 时间而改变的出行时间差别率等。本章在后面会给出出行时间差别率的定义。此外,解析器还可以将 C. M. 值分别作为路段数和仿真时间的函数,分别绘制函数曲线图。车辆数在路段上的分布、车辆和 DTMC 之间的消息数、DTMC 之间的消息数等也同样可以由解析器获得。为更好地进行各种评价,解析器还会算出各个指标的平均值和方差。

6.5 仿真结果和 DICAF 的性能分析

DICAF 要达到的目标包括:①所有车辆都能快速到达目的地;②能高效利用包括高速公路路段区间在内的所有资源。此外,DICAF 还要建立一个可扩展、实用而经济的系统,该系统对每个 DTMC 的通信和计算需求都非常小。进一步来说,对于某个给定的高速公路系统以及在一定时间段内真实的交通分布而言,DICAF中任何一对起讫点之间的任意车辆所需出行时间一定是一致的,并与车辆在该时间段出现的时间无关。

为了对 DICAF 的性能表现作系统阐述,本章中首先定义了几个独立参数以及主要的性能评价指标,并根据各种典型的交通条件进行了大量的仿真实验,如图 6.1 和图 6.9 的两个高速公路系统所示。图 6.1 的高速公路系统包含 9 个 DTMC 和 12 个路段;图 6.9 包含 50 个 DTMC 和 85 个路段。每个 DTMC 控制它东边和南边的路段。假设所有路段长都是 10 mile,车速是随机确定的(后面再详细说明)。

独立参数包括①进入系统的车辆数;②车辆输入密度,即车辆被输入 DICAF 中的频率:③车速的分布。

主要性能评价指标包括:①到达终点的车辆出行时间分布;②到达终点的车辆比例;③出行时间的均值和标准差,它是交通密度的函数;④选定路段的 C. M. 值分布,它是仿真时间的函数;⑤不同交通密度下各路段 C. M. 值的平均值;⑥个体车辆的出行时间,是它们进入 DICAF 时间的函数;⑦在每条路段上的车辆平均数。上述指标分别对应不同的车辆输入密度。除此之外,还有两种方法可以更好地检查 DICAF 在车辆出行时间最短方面的性能表现:第一种⑧是每辆车在出行过程中实际平均车速的分布,它与期望速度相关。第二种评价方法的目的则是为了能帮助我们更好地理解期望车速的分布对 DICAF 的影响,这可以通过测量在 μ 值较高、 σ 值相同以及 μ 值较高、 σ 值较小两种情况下车辆的平均出行时间而获得。这两种情况下的车辆平均出行时间被分别称为评价指标⑨和⑩。

为了预测 DICAF 分配整个通信任务的结果,下面定义另外两个性能指标,包

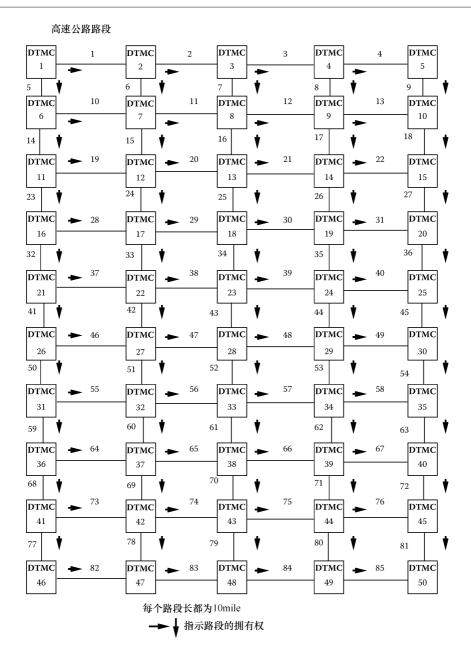


图 6.9 DICAF 模拟的第二个典型高速公路系统

括: ⑩每个 DTMC 到车辆的最大数据传送率; ⑫泛洪消息总数的分布, 接收发送均由 DTMC 负责, 是仿真时间的函数。泛洪消息是来自其他 DTMC 的, 带有其他路段 C. M. 值信息。

在 DICAF 中,车辆通过一个数据结构来表示,并以电子速度,而不是较低的实际车速,从一个 DTMC 传递到另一个 DTMC。因此,DICAF 仿真能够比实际路段上车辆的运行快很多倍,这就使我们能在许多有代表性的情况下对 DICAF 的表现进行详细研究。DICAF 在设计上比实际路段运行快 90 倍,就意味着仿真 1s 相当于实际路段运行 90s,24h 路段实际运行就只需要(24×60×60÷90)s=960s(16min)的仿真时间。这在 DICAF 中是以下列方式实现的:车辆从之前的 DTMC 到达现在的 DTMC 仅需几毫秒。然而,车辆到达终点 DTMC 后会有意延迟一段时间,延迟的时间值等于实际高速公路上的行驶时间除以 90,这样就能保证模拟的准确性,保证 DICAF 中对与仿真方案有关的每辆车出行特性表达的精确性。

平均车辆长度假定为0.0025mile,根据式(6.1),可以知道任意路段上允许的 车辆最大数量为 $0.5 \times \frac{10}{0.0025} = 2000$ 辆。输入交通分布采用随机函数来产生 DICAF 中所有出现在 DTMC 处的车辆。每次仿真运行都相当于实际路段一天 24h 的运行。 据此,交通发生器在每次仿真过程中产生的车辆数相当于实际路段一天 24h 的车辆 数。在每个 DTMC 中,会给每辆产生的车辆随机分配一个终点 DTMC。在整个仿真 过程中, 进入任一 DTMC 处的车辆数都是时间的一个非单值函数。而车辆速度则 是利用 drand48 伪随机发生器函数随机产生的,并以特定的 μ 和 σ 值服从正态分 布。每辆车都被赋予一个唯一标识符,对于一个拥有9个DTMC的高速公路系统来 说,标识符按如下方法计算:(DTMC标识符×10000)+(1···DTMC处产生的最大 车辆数)。对于一个拥有 50 个 DTMC 的高速公路系统而言,标识符按如下方法计 算: (DTMC 标识符×1000000) + (1…DTMC 处产生的最大车辆数)。路段允许的 最大 C. M. 值是 100mile/h,允许的最高和最低车速分别是 90mile/h 和 30mile/h。 因此, 在本章中, 图 6.1 和图 6.9 显示的高速公路系统的大部分性能表现结果都是 在 μ = 50 mile/h、 σ = 10 mile/h 的条件下获得的。此外,图 6.1 中的公路系统在 DICAF 中对两种情形进行了仿真: ① $\mu = 65$ mile/h, $\sigma = 10$ mile/h; ② $\mu = 65$ mile/h, $\sigma = 5$ mile/h。图 6.1 中的高速公路系统要求有 10 个处理器并行、异步地进行处理, 而图 6.9 中的高速公路系统则要求有 51 个 SUN Sparc10 同时运行。输入 DICAF 中 的车辆数由交通密度函数控制,假设在图 6.1 中其取值分别为 3、6 和 8。交通密 度值 D 表示由每个 DTMC 中随机产生的,在仿真运行的每一秒钟内进入 DICAF 的 车辆数, 其范围为0~D。可以知道与密度取值3、6、8 相应的车辆数分别是8528、 21572 和 30284。

RIDOT 会根据 24h 内每个路段上的实际最大车辆数来确定主要高速公路的交通密度数据,这些高速公路包括 I-95、I-195 和 I-295 等。由于同一车辆可能行驶在不同的路段上,所以非常有必要根据交通密度地图^[89]来估算进入整个罗德岛高速公路系统的车辆总数。我们注意到,3车道的 I-95 高速公路承载了最多交通流

量,24h 最大流量达到了约 89302 辆。也就是说单车道路段上最大的 24h 标准交通流量接近 30000 辆。罗德岛是一个非常小的州,面积仅为 $40 \times 16 \text{mile}^2$ ($1 \text{mile}^2 = 2.58999 \times 10^6 \text{m}^2$),却拥有 I-95 这条至关重要的道路,因此本章假定 I-95 是所有行驶在罗德岛高速公路路段的车辆的必经之路。于是,进入罗德岛高速公路系统的最大车辆数将达到 30000 辆/天,也即 DICAF 中最高交通密度 8 所代表的实际交通情况。

在图 6.9 所示的 50 个 DTMC 的高速公路系统中,进入 DICAF 中的车辆总数是 45296,对应交通密度为 3. 仿真速度也比实际运行速度快 90 倍。

这里要注意,每辆车的产生都是随机的,即车辆的来源、目的地以及速度等都是随机变量。当产生并仿真的车辆数超过 45000 辆时,就可以认为所得到的结果能合理地、公正地评价 DICAF 的性能。我们进行了超过 400 次的仿真,尽管每次都有不同的种子值,但可以看到数据的一般行为大体相似,所以我们在这里只针对一次特定的仿真运行结果进行报告。还要注意, μ 值和 σ 值都不会被测量,而是不失一般性地由假设获得。 μ 的置信区间是 $\{\hat{x}-k\leq\mu\leq\hat{x}+k\}$ 。其中, \hat{x} 是计算平均值; $k=\frac{c\mu}{\sqrt{n}}$,c 与置信水平 γ 有关; σ 是测量方差;n 是样本数量。本章中,n 超过45000 意味着 k 值很小,即置信区间很小,即使是对较大的 c 值也一样,这说明具有较高的置信水平。

这个部分展示的性能表现数据,是根据每秒的仿真样本中的关键指标获得的。

图 6.10 展示的就是一个包含 9 个 DTMC 的高速公路系统在交通密度分别为 3、6 和 8 的情况下车辆期望速度的正态分布。X 轴表示车辆速度,从 30mile/h 到 90mile/h; Y 轴表示各种期望速度下的车辆数。显然绝大部分车辆希望行驶速度在 55mile/h 左右,几乎没有车辆希望以更高或更低的速度行驶。

为了对 DICAF 与传统的 TMC 方案的性能进行对比,一般认为需要开发一个采用集中调度的仿真器,并在单处理器上依次操作。在本研究中,没有开发这种单处理机仿真器的原因有以下两个:第一,要表示超过 30000 辆车的数据结构需要非常庞大的存储空间;第二,仿真器的执行速度有可能会非常低。在本章中,我们提出了一种评价 DICAF 绝对表现的方法:对于每辆车而言,完成出行的时间是按照从起点 DTMC 到终点 DTMC 的最短路径长度 (mile)除以车辆的期望速度来计算的。这个时间就是理想出行时间,因为只有当 DICAF 中仅有一辆车行驶,而没有其他车辆和它争夺资源的情况下才能实现。从绝对意义上来讲,一辆车的理想出行时间是它所能达到的最佳状态。当一次仿真运行完成时,每辆车所需要的实际出行时间是根据车辆从起点驶向各自终点的过程中车辆采集到的信息而获得的。对于车辆而言,实际出行时间是仿真过程中车辆出行路径上所经过的每个 DTMC 对之间行驶

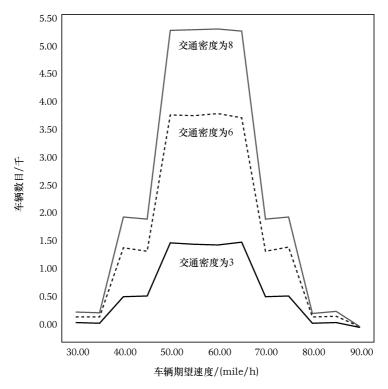
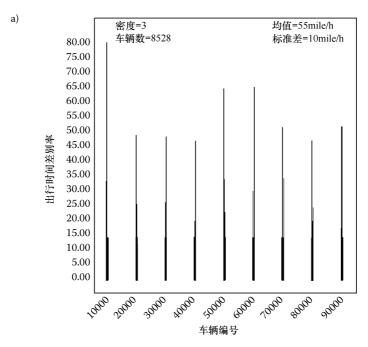


图 6.10 在交通密度为 3、6 和 8 的情况下车辆期望速度的正态分布

时间的累计和。某辆车的出行时间差别率这样计算: (车辆实际出行时间 – 理想出行时间)/(理想出行时间)×100%。根据定义,每辆车的出行时间差别率必须是正值,即大于或等于 0,并能反映车辆的出行时间超出绝对极小值的程度。图 6.11a~c分别表示在密度为 3、6、8 的情形下,每辆车的出行时间差别率。X 轴表示唯一的车辆标识符,范围从 10000 到 90000。平均值和标准差在密度为 3、6、8 的情形下分别是 {1.89%,6.09%},{1.85%,5.71%},{2.12%,6.14%}。所有车辆出行时间差别率的平均值从 1.89%增加到 2.12%,反映了由于有更多的车辆在路段上行驶而导致所有车辆总出行时间的增长。进一步来说,即使 DICAF 系统里有超过 30000 辆车,车辆的平均出行时间也仅比最佳时间高 2.12%。此外,最坏情况下的车辆平均出行时间,如图 6.11c 所示,也仅比最佳时间高 85%,也就是比理想出行时间的两倍稍低。

显然,这些发现证明了 DICAF 具有良好的性能。因此,尽管高速路段上有30284 辆车行驶,但 DICAF 中每辆车的自主分布式的动态路径选择结果却与理想情况十分接近。曲线图揭示了以下原因:在并行处理研究中,人们普遍认为分布式算法能更快得到结果,而从集中式算法中得到的解却是开放性的解。在分布式算法



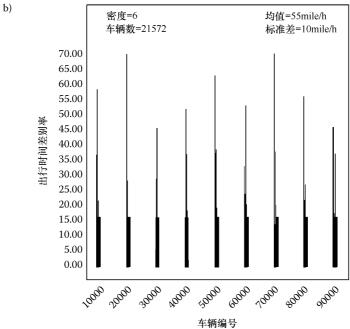


图 6.11 在密度为 3、6、8 的情形下, 所有车辆的出行时间差别率

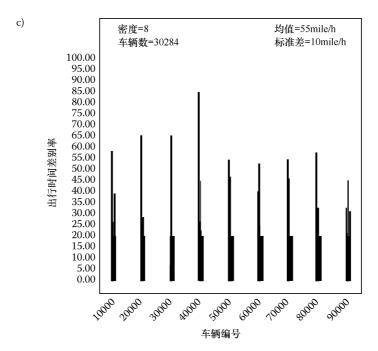


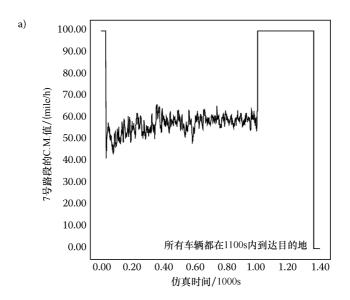
图 6.11 在密度为 3、6、8 的情形下, 所有车辆的出行时间差别率(续)

中,当地的实体在执行决策的过程中,只能连接系统的一小部分。虽然人们很担心不能连接系统全部内容可能会有问题,但 DICAF 的结果却显示其效果非常好。也就是说,可以证明,在某些情形下,DICAF 中运用分布式算法能在快速求解的同时得到高质量的解。据此,作者提出了一种从问题的集中描述中提取分布式算法的新的数学框架^[90]。

在 DICAF 的仿真运行全程,即在高达 960s 的仿真过程中,车辆一直处于 DT-MC 中。仿真过程能连续执行不多于 1100s,直到所有车辆到达其终点。

图 6. 12a~c分别显示了 7 号路段在交通密度为 3、6、8 的情形下 C. M. 值的变化情况。根据式 (6.1) 的定义, C. M. 值的变化是连续且一致的。选择 7 号路段的原因是它位于高速公路系统的中心,许多斜对角穿行的车辆都可能经过这个路段。初始化默认的 C. M. 值为 100mile/h,当仿真时间为 1100s 时,所有车辆完成行驶任务,而 C. M. 值再次增加到 100mile/h。正如所预料的那样,平均 C. M. 值在低交通密度时较高,在车辆总数较大时较低。

图 6.13a~c显示了整个仿真过程中所有路段在三种交通密度下的 C. M. 平均值。可以看出,路段的平均 C. M. 值都非常相近,范围从 54mile/h 到 57mile/h,这意味着 DICAF 中所有资源都被公平地使用了。



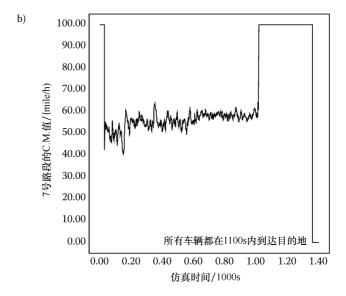


图 6.12 在交通密度分别为 3、6、8 的情形下 (分别对应图 a、图 b、图 c) 7 号路段的 C. M. 值与仿真时间的函数关系

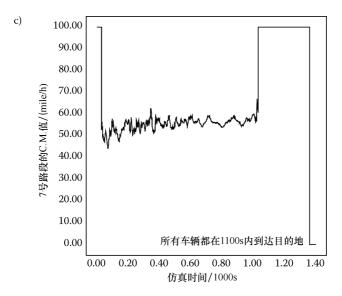


图 6.12 在交通密度分别为 3、6、8 的情形下 (分别对应图 a、图 b、图 c) 7 号路段的 C. M. 值与仿真时间的函数关系 (续)

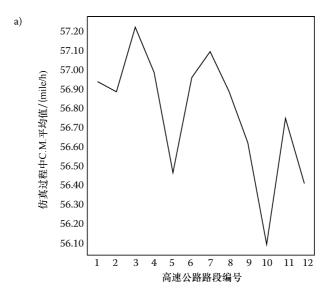
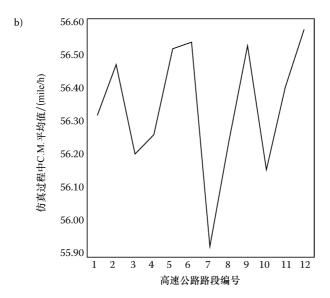


图 6.13 在交通密度分别为 3、6、8 (分别对应图 a、图 b、图 c) 时 DICAF 中所有路段上的 C. M. 平均值



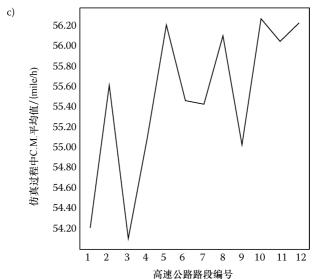
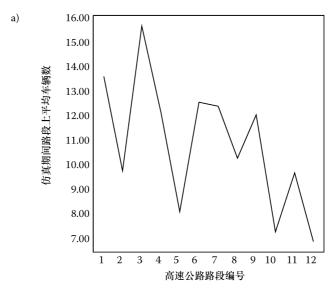


图 6.13 在交通密度分别为 3、6、8 (分别对应图 a、图 b、图 c) 时 DICAF 中所有路段上的 C. M. 平均值 (续)

图 6. 14a~c 通过平均车辆数曲线显示了仿真全过程中各个高速公路路段在三种交通密度下资源的利用情况。尽管所有路段都被合理利用了,反映出 DICAF 在成功利用所有路段资源上的优势,但是还是有某些路段上的车辆平均数远高于其他路段的情况,这主要是由于车辆的速度和终点的随机性而造成的。图 6. 14c 中每个路段的车辆平均数高于图 6. 14b,而图 6. 14b 中的数量又高于图 6. 14a,这是因为

较高的交通密度意味着更多的车辆需要利用路段资源完成它们的行驶任务。交通密度为3时,所有路段上的最大车辆平均数为16;而交通密度为6和8时,对应的数字分别为83和130。任一路段上的车辆平均数都可以通过对每个仿真秒中该路段上的车辆排队长度进行抽样得到。要将这些数据和路段最大通行能力联系起来,可以考虑下面这种情形:假设一个高速路段最多允许2000辆车,以0~90mile/h范



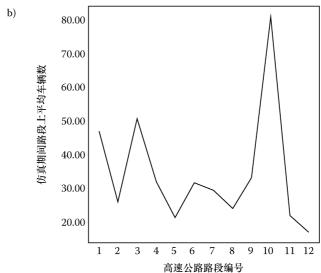


图 6.14 仿真全过程中交通密度分别为 3、6、8(分别对应图 a、图 b、图 c)时 高速公路路段上的平均车辆数

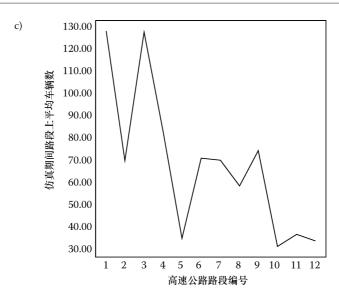


图 6.14 仿真全过程中交通密度分别为 3、6、8 (分别对应图 a、图 b、图 c) 时 高速公路路段上的平均车辆数 (续)

围内某个速度等速行驶。假设平均行驶速度为 90mile/h, 最短行驶时间等于 10mile/h÷90mile/h=399.9s, 相当于仿真时间为 4.4s。于是,对每一个仿真秒而言,每条路段上最大车辆数为: 2000 辆÷4.4s=454 辆/s。从图中可以看出 DICAF 中任意路段上的最大车辆平均数为 130,只相当于绝对最大值的 28%。因此,可以说,对于给定数量的车辆,DICAF 能将交通流量在所有路段上公平地分配,而不会像传统方式那样造成一部分路段极度拥挤的状况。

为了理解车辆进入时间对出行时间的影响,图 6. 15a~c显示了所有车辆的数组关系 [(一辆车的实际行驶时间与理想行驶时间之差与理想出行时间之比),车辆进入 DICAF 中的时间]。图 6. 15a~c分别对应交通密度为 3、6、8 的情况。图 6. 15a~c中黑点的增多反映出车辆的增多。除了图 6. 15c中 600~960s间的黑色区域以外,三幅曲线图的轮廓非常相似。这意味着除了在仿真开始阶段以外,无论车辆在仿真的什么时间进入 DICAF,车辆的行驶时间都保持不变。图 6. 15c中 600~960s间的黑色区域并不明显,它只是反映了一组可预计数量的车辆在仿真时间段 {600s,900s}中进入到 DICAF中,对于这些车辆而言,出行时间差别率为 0%~10%。然而,三幅曲线图中的负斜率都非常明显,它或者说明了图中右上角数据缺失的原因。这意味着尽管交通密度很高,DICAF 仍能成功地为所有车辆选择较好的路径。可以观察到每次仿真中所有车辆都能正确到达它们的目的地,图 6. 15a~c中的曲线同样也表明在交通密度为 3~8 的条件下,高速公路系统都是稳定的,也就是说它至少可以容纳 30284 辆车。

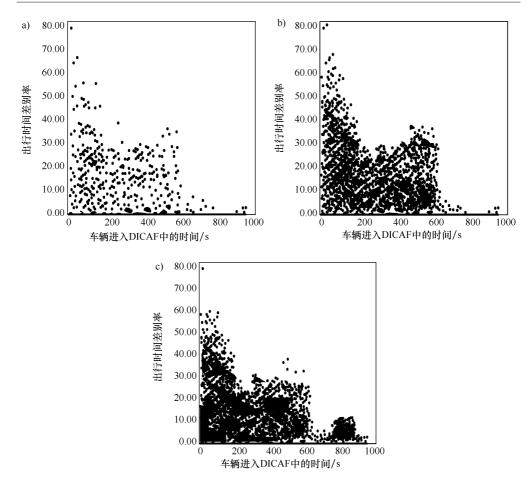


图 6.15 每辆车的出行时间差别率 [(车辆实际行驶时间与理想行驶时间之差/理想出行时间) ×100%],与车辆进入 DICAF 时间的函数关系 a)交通密度为3 b)交通密度为6 c)交通密度为8

在曲线图中,白天较早的时候进入 DICAF 的车辆似乎要比仿真过程中其他时间进入的车辆花费更多的行驶时间。这似乎与常理相违背,因为一般情况下早上车辆都比较少。我们可以通过检测进入 DICAF 的车辆的速度分布得到原因,该速度分布是仿真时间的函数,如图 6.16 所示。

图 6. 16 显示了进入 DICAF 中车辆的速度情况,范围从 30mile/h 到 90mile/h, 仿真时间从 0s 到 100s。从 100s 到 550s,速度带宽会缩小为 40mile/h 到 80mile/h,接下来,速度带宽会不断缩窄,直到第 960s 时,速度范围缩窄为 50 ~ 70mile/h。如果要使车辆速度服从正态分布(见图 6. 10),交通发生器程序必须首先建立对应于不同速度带宽 {30mile/h,35mile/h} 到 {85mile/h,90mile/h} 的"速度组",

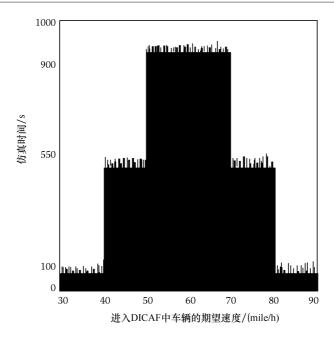


图 6.16 交通密度为 8 时进入 DICAF 中车辆的期望速度 和仿真时间的函数关系

然后分别针对三种交通密度情况给每组分配适当的车辆数。接下来,产生速度和目的地都是随机值的车辆,它们在仿真时间为 0s 时被分配至合适的组中。正如预期的那样,这些包括极限值 $\{30 \text{mile/h}, 35 \text{mile/h}\}$ 和 $\{85 \text{mile/h}, 90 \text{mile/h}\}$ 速度带宽的组首先被填满。然后,在仿真时间的 $100 \sim 550 s$,速度带宽为 $\{40 \text{mile/h}, 45 \text{mile/h}\}$ 、 $\{45 \text{mile/h}, 50 \text{mile/h}\}$ 、 $\{70 \text{mile/h}, 75 \text{mile/h}\}$ 、 $\{75 \text{mile/h}, 80 \text{mile/h}\}$ 的组也相继被填满。到此时为止,除了 $\{50 \text{mile/h}, 70 \text{mile/h}\}$ 这个速度区间外的所有组都被填满了。所以,在 $550 \sim 960 s$ 间,DICAF 中的车辆的速度分布在很窄的范围内。

从逻辑上来讲,车辆速度的下降很可能会导致路段 C. M. 值的下降。因此,除了那些低速车辆会花费较长的行驶时间以外,那些有着较高期望速度的车辆也会需要更长的行驶时间。随着仿真的进行,可以看到车辆的平均期望速度逐渐增加,从而产生更快的出行和更短的出行时间。

此外,比较图 6.17 和图 6.10 中的曲线,可以看出实际车辆速度分布与初始的期望正态分布有些相似,但是图 6.17 中的分布因为拥堵的存在而显得有些扁平。

图 6.17 显示了每辆车的实际平均速度,是用每辆车的总出行距离除以它们各自的出行时间来得到的。可以观察到,在整个仿真过程中,大多数车辆都以 30~80mile/h 的平均速度行驶,而超过 75mile/h 的车辆数非常少。平均速度集中在

60mile/h 左右,这意味着尽管有 30000 多辆车在争夺道路资源,在 DICAF 中大多数车辆还是能够有效地进行路径决策,并实现在较短出行时间内完成出行任务的目的。

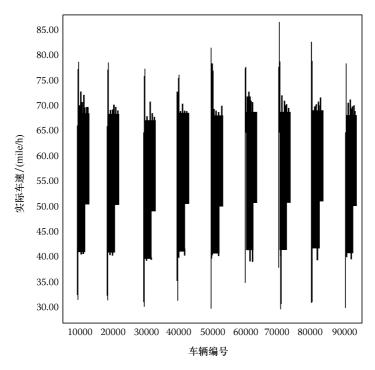
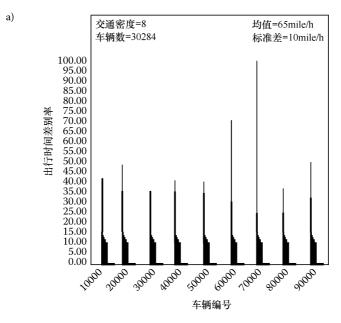


图 6.17 交通密度为 8 时 DICAF 中车辆的实际平均速度

在仿真刚开始阶段就进入 DICAF 中的车辆,与后来进入的车辆在出行时间差别率上的对比如图 6. 15 所示,图中可以看出较窄的速度带宽对所有车辆有效选择路径和实现更快出行有着重要的作用。为了检验这个假设,我们设计了在以下两种交通输入参数情况下进行的仿真: $\{\mu=65 \text{mile/h}, \sigma=10 \text{mile/h}\}$ 和 $\{\mu=65 \text{mile/h}, \sigma=5 \text{mile/h}\}$,交通密度选择为 8。图 6. 18a 和图 6. 18b 显示了所有车辆的实际出行时间与理想出行时间之差,与理想出行时间的比值。图 6. 18a 对应的交通发生器参数为 $\{\mu=65 \text{mile/h}\}$,不 $= 10 \text{mile/h}\}$,图 6. 18b 对应的交通发生器参数为 $\{\mu=65 \text{mile/h}\}$ 。在图 6. 18a 中,车辆总数为 30284,平均差和标准差分别为 1. 11% 和 4. 13%。在图 6. 18b 中,车辆总数为 30302,平均差和标准差分别为 0. 03% 和 0. 45%。

与图 6.11c 中相关数据的比较显示出,要提高出行效率,即使所有车辆的总出行时间更短,可以通过定义最大速度和最小速度,将速度限制在较狭窄的范围内而得到,而不是像当前政策那样仅以最大速度 55 mile/h 作为限制。可以想象,在现

有的高速公路系统中,车辆速度波动的主要原因是缺乏高速路段的信息,比如出口的位置和速度等,这些信息都是可以从 DICAF 中获得的。



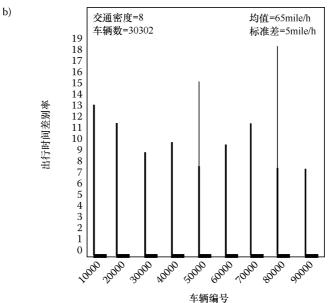
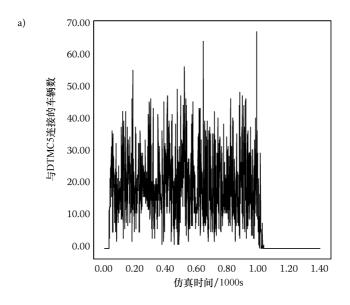


图 6.18 交通发生器参数为 $\{\mu = 65 \text{ mile/h}, \sigma = 10 \text{ mile/h}\}$ (图 a)、 $\{\mu = 65 \text{ mile/h}, \sigma = 5 \text{ mile/h}\}$ (图 b) 情况下 所有车辆的出行时间差别率 (续)

DICAF 的一个基本目标就是公平地将所有通信任务分配给所有实体。图 6.19a 展示了与 DTMC5 连接的车辆数的分布情况,它是仿真时间的函数。正如之前所分析的那样,选择 DTMC5 的原因是它位于高速公路系统的中心,很有可能位于许多车辆的必经之路上,所以,研究这个 DTMC 中与数据有关的通信问题可以反映可



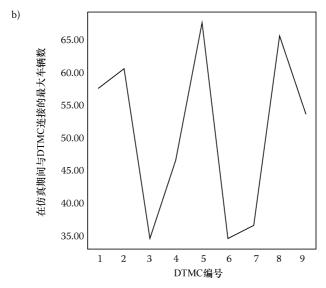


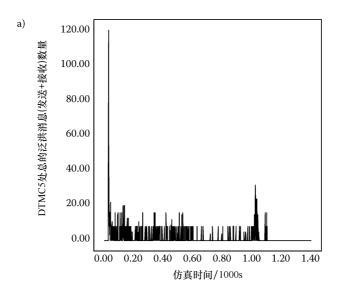
图 6.19 与 DTMC5 连接的车辆数和仿真时间的函数关系(图 a) 以及与 DTMC 连接的最大车辆数的分布(图 b)

能遇到的最坏的情况。图 6. 19a 还反映出任意路段上的车辆数,以及路段的 C. M. 值都是和时间高度动态相关。图 6. 19b 显示了与 DTMC 连接的最大车辆数的分布情况。图 6. 19a 和图 6. 19b 所展示的数据是通过对每个仿真秒中的排队长度进行抽样而获得的。当车辆与 DTMC 连接时,车辆会下载相应路段的 C. M. 值,平均信息长度为 96B。由于 DTMC5 中最大车辆数为 68,因此 DTMC5 与车辆间的最大通信速率等于 68 × 96B/仿真秒 = 6528/1. 5B/min = 4. 352B/min。这种通信速率很容易通过 9600bit/s 或 19200bit/s 的商业无线调制解调器来实现。

图 6. 20a 显示了 DTMC5 处发送和接收的泛洪消息的数量分布,它是仿真时间的函数。选择 DTMC5 的原因在前面也提到过。对于每个 DTMC 来说,DTMC 内部接收和发送的最大泛洪消息数量会被记录下来,并制成曲线图,如图 6. 20b 所示。可以观察到在每个 DTMC 中,最大泛洪消息数是与仿真开始同时发生的,原因如下:一旦车辆进入高速路段,所有 DTMC 都会计算其所控制的路段上的新的 C. M. 值。新的 C. M. 值很有可能与预先默认的 100mile/h 不同,因此会被立即传播给其他 DTMC。随着仿真的进行,C. M. 值逐渐增加,而消息泛洪的频率逐渐下降。图 6. 20b 的曲线是针对交通密度为 8,即有 30284 辆车在路段上行驶的情况,通过对仿真中每秒内的消息队列进行抽样得到的。每条泛洪消息包含 4 个字段,需要 16B。图 6. 20b 中,泛洪消息的最小和最大数量分别为 55 和 122,相应的数据通信速度为 55 × 16B/仿真秒 = 55 × 16/1. 5B/min = 586B/min,122 × 16B/仿真秒 = 122 × 16/1. 5B/min = 1301B/min,同样很容易通过 9600bit/s 或 19200bit/s 的商业无线调制解调器来实现。假设仿真过程需要 1s 或者实际的 1. 5min,那么这里显示的数据也是在 1. 5min 内的。

因此,图 6.19 和图 6.20 可以证明,DICAF 的一个主要目标,即建立一个不太昂贵的、成本效益比较高的高速公路体系,该目标已经实现了。相比之下,集中式算法在理论上需要更高的通信速率,这意味着需要价格更加昂贵的接口。

图 6. 21a 和图 6. 21b 显示了图 6. 9 所示的大型复杂的高速公路系统的模拟和仿真结果。这项研究的目的就是要检测 DICAF 中异步分布式算法在一个拥有 50 个DTMC 和 45296 辆自治车辆,同时在 51 个工作站上执行的大规模系统中的实用性。图 6. 21a 和图 6. 21b 中的曲线对应于交通密度为 3 时,实际高速公路 24h 的运营情况,交通发生器参数为 μ = 65 mile/h, σ = 5 mile/h。图 6. 21a 显示出均值和标准差分别为 0. 038% 和 0. 577%,这表明尽管有 45296 辆车在竞争路段资源,车辆平均行驶时间仅超过绝对最小值 0. 038%。图 6. 21b 表明,除了那些在仿真开始阶段就进入 DICAF 的车辆外,其他时刻进入 DICAF 的车辆都能成功到达终点,且平均行驶时间与进入 DICAF 的时刻无关。我们还有其他一些观察有 9 个 DTMC 的系统得到的曲线图,也显示了同样的行为特征,这里就不再赘述。



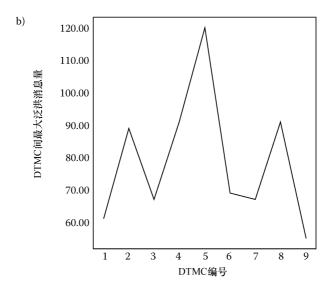
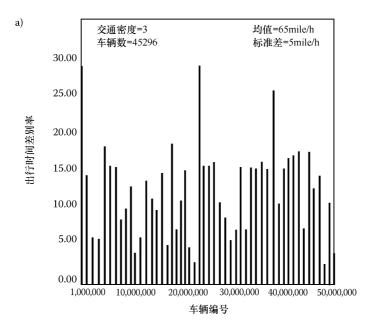


图 6. 20 DTMC5 处的 DTMC 间泛洪消息数量和仿真时间的 函数关系 (图 a) 以及每个 DTMC 中处理的 DTMC 间最大泛洪消息量 (发送+接收)(图 b)



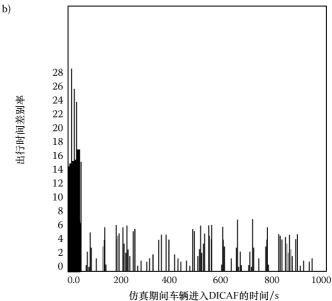


图 6.21 在 DICAF 中有 50 个 DTMC 的公路系统的仿真结果: 出行时间差别率 a) 对所有车辆 b) 与车辆进入 DICAF 时间的函数关系

DICAF 的缺陷

本章中没有考虑交通事故、突发事件和可能由此产生的拥堵问题。因此本章考 虑的拥堵问题仅仅是由 DICAF 中车辆速度和终点选择的随机性造成的。一般来说, 任何路段的上下行方向的 C. M. 值应该是不同的, 而 DICAF 中假设上下行方向的 C. M. 值相同。但这种假设是合理的、因为 DICAF 中车辆的目的地是随机的、所以 上下行车辆数量很可能基本相同,因此也就具有了相同的 C. M. 值。本章的目的是 针对一些有代表性的高速公路系统,研究 DICAF 的一般性能,而不是对特定的联 邦和州际公路的实际交通情况进行精确模拟。因此,我们假设交通分布与数据选取 日期没有关系。正如本章前面所提到的, 当用一个工作站模拟一个 DTMC 时, 每 辆车都模拟为一个进程。进程从一个 DTMC 传递到另一个 DTMC, 并由车辆所在地 管辖的 DTMC 中的工作站执行。选择该方法的主要原因是我们无法获得包括 45000 个相互连接的工作站的试验台。如果模型与实际不符,是有利有弊的事情。坏的一 面是不能详细模拟 DTMC 和车辆之间建立联系的过程。并且, DTMC 和车辆之间在 某特定操作系统下进行的数据交换在实际中会很慢,因为仿真过程中交换的是本地 的数据结构。好的一面是本章展示的实验数据都可以保存在 50 个工作站中, 这些 工作站承担了45000多辆车中每辆车的路径计算任务。由于在实际中是由每辆车的 计算引擎计算它自己的路径、所以基于 DICAF 开发的系统在实际中的性能表现很 有可能会大大超过本章在仿真中预测的值。

第7章 RYNSORD 在干扰下的稳定性

7.1 稳定性

根据文献,物理学和控制系统理论中对稳定性的研究是最全面的。Stewart 在 1992 年翻译了 A. M. Lyapunov 的"运动稳定性的一般问题^[91]",并在前言中论述了定义稳定性的动机。Stewart 指出:Lyapunov 认识到存在很多不同的稳定性概念——即存在许多不同的方式,将"运动中微小的干扰会导致微小的变化"这一思想形式化。从工程学到政治学,这个一般的概念已被广泛应用于各个领域。然而,几乎在所有情况下,该定义都正被迫地不断适应它所应用的领域。本章中对稳定性的研究同样是自由的,但是会尽量围绕"运动中微小的干扰会导致微小的变化"这一原始动机。

Chen^[92]描述了控制理论中的三类稳定性:①有界输出稳定性;②临界稳定性;③渐近稳定性。下面将会逐一介绍。对于每种类型,控制理论允许通过微分方程、状态空间和转移函数模型等来进行定义。

1. 有界输出稳定性

有界输出(BIBO)稳定性是指对于任意一个有界输入,输出也是有界的。有界函数是指它的值大小总是在某些常数形成的范围内。这个定义可以直接应用到RYNSORD^[93,94]中,因为任何时间,在任一车站或一组车站,输入可以用进入系统的火车数量表示,而输出则是由输入的每列火车离开系统,即抵达终点的时间组成,也可被看做是一个时间的函数。要使RYNSORD 达到BIBO 稳定,对于任何给定的有界输入函数,输出函数也必须是有界的。这只有当系统中每列火车都在有限时间内抵达终点时才能成立,这也正是RYNSORD的一个重要性质,该性质接下来会被用于确定系统的稳态输入率。然而,我们在后面将会看到,对于RYNSORD来说,人们总是有可能选择一个会导致无界输出的输入率。在一般情况下,基于BIBO稳定性的选择并不适当,因为不是所有的ADDM系统都非得给自己建立一个明确的输入一输出关系的。不过,BIBO 仍是RYNSORD的一个重要性质,我们将在稳定状态分析中对它作更详细的研究。

2. 临界稳定性

临界稳定性一般参考 Lyapunov 的稳定性概念进行定义, Fuller^[91]指出这一定义的提出起源于 Lyapunov 对天文学问题的兴趣。对许多问题来说,一次扰动可能没

有完全消散,并且会在一定范围内一直持续,例如,在围绕一个点的圆形轨道上,一个质子围绕中心点不断振荡,这样就实现了临界稳定。这个稳定性定义非常类似于 RYNSORD 中采用的临界稳定性的定义。

3. 渐近稳定性

与临界稳定性相反,渐近稳定性的干扰最终会消散。Letov^[95]在解释 Lyapunov的第二定理时指出,在渐近稳定性下,受干扰的运动随着时间推移到无穷大时会收敛到一个无干扰状态。控制系统工程师们发现这个定义非常有吸引力,它构成了本章中对"强稳定性"进行定义的基础。

令 Casavant [96] 感到遗憾的是很难将控制理论直接运用到分布式系统中,因为常规使用的数学方法难以适用于分布式系统,这主要是由于分布式系统中存在复杂的交互作用,很难通过一组微分方程或转移函数来描述它们的一般非线性性质,除非先将假设条件大大简化。与之相反,在本章所采用的方法中,虽然评价的精度是取决于假设简化的影响有多大,但是控制理论在分析 RYNSORD 这样的 ADDM 系统的性质时也是非常有价值的。许多控制理论的基本概念也适用于严谨的数学基础理论不适用的地方。

在分布式系统的理论研究中,稳定性的讨论总是与自稳定性、正确性,即避免 死锁、鲁棒性、容错和服务质量等性质有关。

1. 自稳定性

1973 年 Dijkstra 开始研究分布式系统中的自稳定性。他写道:"我将一个系统称为'自稳定性'时,不论其初始状态是怎么样的,它都可以保证在有限的步骤内到达一个合法的状态"^[97]。因此,自稳定性意味着系统具有足够强的鲁棒性,使其能从非法状态中恢复到合法状态。Dijkstra^[98]根据优先权定义了合法状态,而对优先权的判断是基于进程本身的状态及其近邻的状态。然而,很难找到符合这个合法状态和优先权定义的现实系统。另一个问题是,这个定义是基于对特定状态的识别,而识别全局状态哪些是合法的,哪些不合法,这是一个很困难的问题,因为对于一个大型复杂的系统而言,各种状态的组合数量可能是巨大的。

在不考虑优先权的条件下,Awerbuch 等^[99]给出了一个更一般化的定义: "自稳定性将下列直觉上的目标进行了形式化:尽管灾难性故障长期都存在,但一旦灾难性故障停止时,系统应能稳定到无需人工干预的正确行为上来。"这里,灾难性故障被定义为"全局状态被严重破坏"的状态。这个定义比 Dijkstra 的定义更加实用。如果我们将"正确行为"定义为一个合法状态,这两个定义就是一样的。本章中提出的定义也可以看做是这个直觉目标的一种形式化,但并不只局限于状态的破坏。Awerbuch 的工作着重于非交互式系统和方法的自稳定性,通过定期检查正确性,无论何时只要发现了一个错误,就会重新执行一次整个程序。与此相反,RYNSORD系统是不断与环境相互作用的,基于实际的考

虑,不能停止,也不能重新执行。

2. 稳定性质

与分布式系统模型相关的是由 Chandy 和 Lamport [100] 定义的稳定性质。他们指出:如果 y 是一个判断参数,并且是分布式系统 D 的一个稳定性质,那么在 D 的计算中,一旦 y 为真,计算的其余部分也都为真。包括 Venkatesan 和 Garg [101,102] 在内的其他研究人员却使用稳定"判断参数"而不是稳定性质。然而,这个稳定性质的定义与一般情况不同,一般情况下所说的"性质"会被定义为系统的判断参数,而这里的定义是指全系统的稳定性。例如,在参考文献 [100] 中,死锁被看做是一个稳定性质。相反,在本章中,如果一个系统被死锁,它会导致系统的不稳定。这个明显的矛盾反映了这样一个观点:系统的稳定性是建立在系统中的错误是有限的基础上的,而死锁系统是一种无限(无边界)错误的情况。这里存在的一个假设是系统不会因为设计的原因形成死锁,就像 RYNSORD 和最实用的系统那样,所以死锁是不稳定系统的一个特征。

3. 鲁棒性

鲁棒性是指尽管发生一些变化,系统也能够保持正确行为的能力。Schreiber^[103]根据错误的类型来区分鲁棒性和故障弱化的行为:鲁棒性处理的是输入的错误,而故障弱化行为是由系统中的故障造成的。Stankovic^[104]提出了一个不同的定义:"在计算机科学文献中,鲁棒性通常是指一个系统处理故障的能力"。目前的争议在于定义的范围。Schreiber 的定义仅限于考虑输入中的错误,而 Stankovic 是从"故障"的角度进行定义的。Meyer^[105]确定了分布式实时系统的 4 个属性:①并发性;②及时性;③容错;④故障的可降解性。在控制理论中,一个鲁棒系统是指尽管处于受干扰状态,但仍能正确执行的系统。本章既研究输入模式变化对系统性能的影响,也研究故障对系统性能的影响。干扰并不一定就意味着故障,但是干扰代表正常运行环境发生了一些变化。所以,这里涉及的定义包括鲁棒性、故障弱化行为和降解性能。

4. 容错

容错注重的是系统对故障的承受力,这与本章所研究的问题根本不同。稳定性 注重的是发生故障后的表现,而不是如何从故障中恢复。

5. 服务质量

在参考文献 [106] 中,Garg 等定义了分布式应用程序的稳定性。他们从稳定性表现这一角度出发,采用提供给用户的服务质量(QoS)作为其性能指标。他们定义了一个稳定的分布式应用程序,它的 QoS,在包括干扰在内的所有时间内一直会受到约束。该定义具有局限性,因为,第一,QoS 属性与 RYNSORD 和其他系统无关;第二,受干扰期间的错误可能是无界的。

7.2 RYNSORD 稳定性的正式定义

本节正式介绍 RYNSORD 稳定性的概念,并提出几个定义。本章的目的是根据一个性能标准来定义 RYNSORD 的稳定性,从而为一个处于动态变化的环境中的系统提供性能保证。Ferrari^[107]把性能定义为系统有多好的一个指征,当然该系统已先被假定是正确、有效的。

RYNSORD 的一个重要概念是需要一个量化的误差评价标准,我们称之为用户定义的可测量的量。对这个作为误差评价标准的量来说,要满足三个要求:第一,它是一个可量化的值;第二,从概念上,它反映了系统对某些理想状态的偏差,所以理想状态也必须可以量化;第三,用户希望使误差最小化。

1. 定义1

误差量:系统性能的定量测量标准可以表示为误差量 = | 理想值 - 实际值 | 。 理想值与实际值都必须是可测量的或可计算的。

分布式系统的平衡或稳态被简单地定义为误差总是被限制在有限常数范围内的运行环境(即系统输入和资源的组合等)。这个限制的确切范围一般不会具体给出,除非是要求必须小于某个小于无穷大的常数。

2. 定义2

稳态:如果一个系统处在一个稳态中,系统的误差 e 总是可以被定义为 $e < K < \infty$,其中 K 是一个任意常数。

定义稳定性的重点在于一个处于稳定状态中的系统,当环境发生变化(称为干扰)时会发生什么,这种变化在真实世界中是不可避免的。这些变化可分为两类:系统级的干扰和输入干扰。系统级的干扰一般被称为故障或失败,它包括所有形式的硬件故障以及局部状态的强制中止。输入干扰是指输入 RYNSORD 系统的方式或速率上的变化。干扰的意思是它违背了原有的假设。一个假设通常被描述为稳态运行环境中的一个特征。虽然可能有多重干扰同时影响系统的情况,但是本章只考虑环境中有一个干扰,即单一变化的情况。

3. 定义3

干扰:干扰是指对某种假设的违反,可以根据以下因素定义:违反行为的性质、适用范围和两个时间值: $t_{\rm pert}$ 是指干扰发生的时间, $t_{\rm pert_end}$ 表示系统变化中止的时间。 $t_{\rm pert_dur} = t_{\rm per_end} - t_{\rm pert}$ 被定义为干扰持续时间。

4. 定义4

稳定性:如果一个系统处于稳态中,而这时发生了一个干扰,随着 $t\to\infty$,它会恢复到稳定状态。 K_1 是原始稳定状态的边界, K_2 是最终稳定状态的边界。如果 $K_2 < K_1$,则系统是强稳定的,否则就是临界稳定的。

强稳定和临界稳定之间的区别非常重要。假定一个强稳定的系统处于稳定状态,且发生一个干扰,随着该干扰的不断重复进行,该系统最终将返回到一种更好的,或至少是与原始状态相同的稳定状态,这取决于误差范围的大小。相反,随着干扰的进行,一个临界稳定系统可能会在一个更糟糕的误差范围上产生一个新的稳定状态,并且,随着该干扰的不断重复进行,还可能出现越来越大的误差范围。在最坏的情况下,一个周期性干扰就可能使系统变得不稳定,也可能会导致误差在连续的干扰间振荡。

本章定义了两个相关的稳定性类型,它们可以根据相应的干扰类型划分为输入 稳定和系统级稳定。

5. 定义5

输入稳定:与输入干扰有关的稳定性,即输入率、分布或输入幅度的稳定程度。

6. 定义 6

系统级稳定:与系统级干扰有关的稳定性,该定义包括一切除了输入干扰以外的干扰,例如元件故障(链路和节点)以及元件退化(丢失信息)等。

7.3 RYNSORD 稳定性分析模型

为了进行稳定性分析,首先要为 RYNSORD 的分布式离散事件仿真建模。为了模拟真实铁路系统的运行情况(该系统应有一定数量的火车),每个站节点用一个工作站表示,火车被模拟为由车站下属工作站执行的任务。当一列火车位于主站时,其计算由车站下属工作站执行,它与其他车站的通信也通过这个工作站完成。当一列火车从当前站(A)驶向另一个车站(B)时,A的下属工作站就将相应的火车任务进行封装,以消息的形式传送给B,并在B的下属工作站中重新显示为一个火车任务。因此,火车在仿真中是以电子速度而不是物理速度移动的,火车的计算和通信子任务都是在主站的下属工作站中执行的。

虽然火车的行驶速度可达到 120mile/h (192km/h),但是试验台中的快速计算 引擎能使仿真的速度比现实要快许多倍。仿真中的基本时间单位是时间步长,它决定了火车行驶的最佳方案。对于选择时间步长的原因,读者可以参考文献 [93]和 [94]。RYNSORD 允许火车异步地进入系统,即以不规则的时间间隔进入。此外,火车本身是自主的,因此,它们的决策也是相互异步地执行的。

执行问题

为进行这项研究,和第5章一样,我们选择了美国东部铁路网的一个子集,并增加了一些额外的轨道,用来表示一些次要的铁路路段。图7.1表示这个有代表性的铁路网络,由50个大型车站、84个轨道区间段以及总长14469mile的铁轨组成。图7.1中网络的模型是在RYNSORD中建立的,每个车站作为一个进程。此外,所

有实验的前向值都设为3。

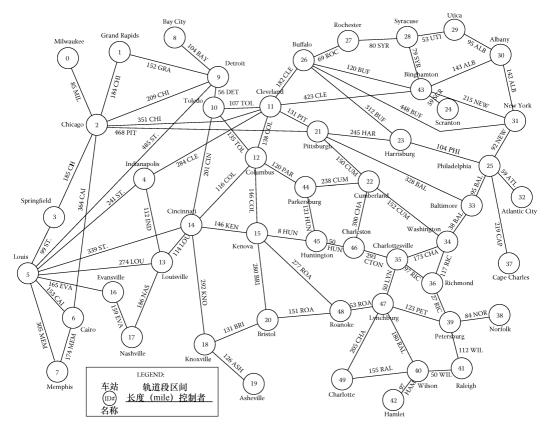


图 7.1 一个有代表性的铁路网: 由美国东部铁路网的 50 个车站组成的子网

7.4 RYNSORD 的稳定性分析

7.4.1 稳定性分析的误差标准

衡量 RYNSORD 性能的主要标准是火车抵达目的地所需的行驶时间和在每个车站等待的平均火车数量。由于稳定性是通过行为表现来度量的,因此,我们在这章中提出两个误差标准: Ⅰ和Ⅱ,这两个标准可以反映性能表现和标准及基准值之间的偏差。本章提出了一个新基准——理想的性能表现值。每列火车的理想行驶时间就是当整个系统中没有任何其他竞争车辆时,火车通过最短路线从起点到终点所需的行驶时间。任何时间,在任何车站等待的理想火车数量是0。虽然这些理想值不可能在没有任何火车竞争的情况下实现,但是它们都是绝对最小值,并非常适合作

为基准。误差标准I表示为

误差 = | 实际行驶时间 - 理想行驶时间 |

其中火车的实际行驶时间就是火车进入系统的时间与到达其目的地的时间之 差,这一过程中是有其他竞争火车出现的。

误差准则Ⅱ表示为

误差 = | 在一个车站等待的实际火车数量 - 0 |

虽然这两个误差标准的目的都是要实现系统的整体性能最佳,但在某些情况下它们可能会不一致。为了使火车的行驶时间最小,第一个误差标准可能会支持火车在沿着最佳路线上的某个特定车站等待,直到前方轨道可以供其行驶为止。相反,第二个误差标准却可能会支持火车在最佳路线被占用时,通过选择更长、更慢的路线使火车继续前进。虽然 RYNSORD 是根据出行时间最少来选择路线,并不直接考虑在车站等待所花费的时间,但是这两个误差标准都显示出相似的稳定性质,也都反映出这样一个事实:平均等待队列长度和每列火车的行驶时间之间是相关的。

实际执行的 RYNSORD 是一个连续不断运行的系统,但是在本章中,仿真过程中假定是有开始和结束的。仿真启动时,系统中并没有火车,此时将时间步长设为1。然后该系统连续运行直到时间结束,大部分实验的仿真时间相当于 17280 个时间步长或 12 个操作日。在仿真启动后,刚进入系统的火车很少会遇到竞争,这似乎体现了这个系统的卓越性能。但是在 RYNSORD 的性能分析中,我们没有考虑这些火车。当仿真在一个预定的时间步长终止时,可能还会有火车在系统中行驶。这些火车被标记为已进入,但还没到达自己的终点,它们也不会被考虑到性能数据中去。在计算结果中,RYNSORD 只考虑了那些已成功到达目的地的火车。

7.4.2 稳态分析

由于干扰引起的系统误差分析与其正常行为相关,所以必须首先确定系统的稳态行为。本节介绍 RYNSORD 的稳态分析过程,并确定一个关键量作为输入交通分布。在 RYNSORD 中货运列车比旅客列车有优势,本章假定列车进入 RYNSORD 在时间上呈均匀分布。与脉冲交通模型不同,均匀分布很可能意味着以某个固定水平使用网络,这会使资源得到高效的利用。在每个车站,在每个时间步长内有一列火车从该站始发的概率定义为输入率。对于每一列在该站始发的火车而言,火车速度是随机产生的,范围为 60~100mile/h(96~160km/h)。始发站是随机选择的一个站,而最终目的地则是通过给每个始发站以外的车站分配相等的权重后随机产生的。地理上的优势在选择过程中不起任何作用。由于对应于大城市的大型车站更有可能出现高交通流密度,所以在图 7.1 中确定了一组 9 个 "高流量"车站——芝加哥、底特律、圣路易斯、费城、纽约、华盛顿、匹兹堡、哥伦布和辛辛那提。对应于这些城市的车站,输入火车交通密度设定为 0.3,远高于系统的最高稳态输入

率。然而,稳态分析结果将告诉我们,这些高流量车站的存在并不妨碍系统实现全局稳定。而且,在选择火车目的地的过程中,这些城市车站分配到的权重为其他车站的两倍,这表明它们比其他城市更可能被选中。

可采用反复试验法来确定稳态条件。RYNSORD 在不同的输入率情况下进行了仿真,表 7.1 总结了在不同输入率下输入火车的平均数量。图 7.2a~c 把误差标准 I 作为某列火车进入系统时间的函数。在图 7.2a 中,误差没有随着时间的增加而持续增加,因此,可以认为 RYNSORD 在输入率为 0.125 时表现出稳态行为。与此相反,图 7.2c 对应的输入率为 0.175,此时误差明显随着时间的增长而增长,从而反映出非稳态行为。当输入率为 0.140 时,如图 7.2b 所示,RYNSORD 既显示出受约束的有界行为,也显示出依赖于特定随机输入的增长,反映出这个输入率就是有界和无界之间的临界值。正如预期的那样,不同的稳态条件表现出不同的误差边界值,图 7.3 显示了误差标准 II 对应的稳态输入率范围为 0.05~0.125。

交通输入密度	出人系统火车总数	完成运行火车总数	完成运行的火车 平均误差/时间步数	最大误差/时间步数
0. 050	453	439 (97%)	67. 95	439
0. 100	858	822 (96%)	183. 18	822
0. 125	1093	1048 (96%)	309. 65	1387
0. 140	1250	1156 (92%)	710. 25	3207
0. 175	1506	1292 (86%)	1115. 62	5324

表 7.1 用于 RYNSORD 稳态分析的输入交通参数

7.4.3 对输入率的干扰和稳定性分析

与任何现实世界系统一样,RYNSORD 是为了在稳定状态中运行而设计的,但在运行时可能会遭遇不可预测的情况,例如输入率的快速波动等。因此,最常见的对 RYNSORD 中输入率的干扰通常包括输入率在很短的时间段内突增等。随着输入率增长幅度和干扰持续时间的增加,对 RYNSORD 中稳态工作点的选择很可能会影响其稳定性。人们期望能将 RYNSORD 设计成一个强稳定系统,即随着干扰的终止,在有限的时间内,最终至少能返回到原始稳定状态的系统。许多实验都是根据不同的初始稳态工作点设计和进行的,它们都能在不同的干扰情况下,达到一次 RYNSORD 系统的稳态点。然后,才会随着仿真的进展和分析去衡量误差准则 I 和 II 。

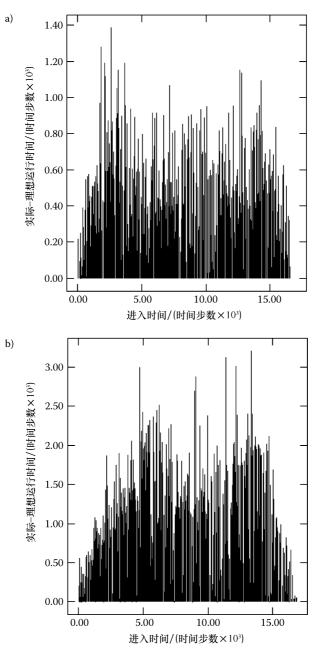


图 7.2 不同输入率情况下每列火车的误差标准 I 和火车进入时间的函数关系 a) 输入率 =0.125 b) 输入率 =0.14

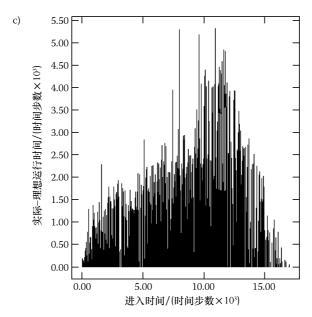


图 7.2 不同输入率情况下每列火车的误差标准 I 和火车进入时间的函数关系(续) c) 输入率 = 0.175

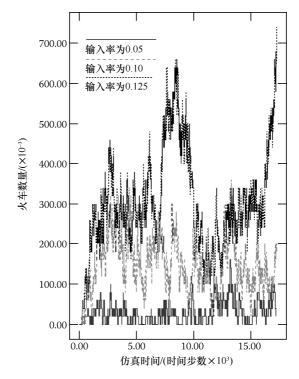


图 7.3 不同稳态输入率下误差标准 Ⅱ 与仿真时间的函数关系

表 7. 2 总结了三种不同的输入率干扰下的系统特性。这三个干扰的幅度设计的目的是要使 RYNSORD 成功超过稳态点。在图 7. 4a~c中,分别对应于三种情况,把每列火车的误差标准 I 作为它进入系统时间的一个函数。图 7. 5 描述了三种情况下的误差标准 II 与仿真时间之间的函数关系。图 7. 4a 和图 7. 5 中所有的误差标准都随着干扰的出现而即时增加。然而,随着时间的推移,所有三种情况中的误差幅度都会下降,而 RYNSORD 最终都返回到初始的稳态点,因此, RYNSORD 就输入干扰而言是强稳定的。

输入率	干扰系数	干扰开始时间/时间步数	干扰持续时间 /时间步数	进入系数 火车总数	完成运行 火车总数
0. 05	+0.5	5760	720	1230	1219 (99%)
0. 125	+0.5	5760	720	2955	2911 (99%)
0. 125	+3.0	5760	720	3755	3443 (92%)

表 7.2 对输入率的干扰和系统特性

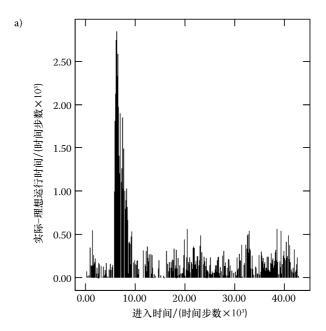


图 7.4 每列火车的误差标准 I 随火车进入时间变化的函数关系 a)输入率为 0.05,干扰系数为 0.5

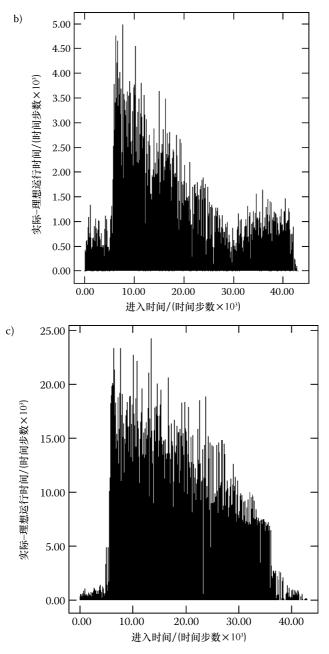


图 7.4 每列火车的误差标准 I 随火车进入时间变化的函数关系 (续) b) 输入率为 0.125,干扰系数为 0.5 c) 输入率为 0.125,干扰系数为 3.0

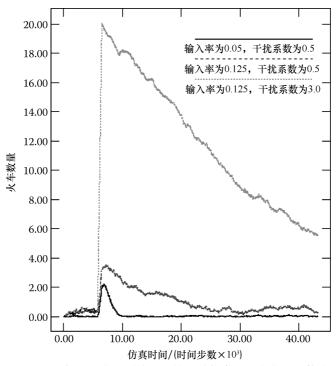


图 7.5 误差标准Ⅱ在不同输入干扰下随仿真时间变化的函数关系

7.4.4 对系统特征的干扰和稳定性分析

RYNSORD 的基本结构假定每列火车能够与相应的车站进行通信,车站彼此之间也可以进行通信。虽然 RYNSORD 算法没有明确考虑轨道故障的可能性,但是当一个或多个轨道正在使用或无法使用时,火车能够确定替代路径。本节介绍在这种故障于扰情况下 RYNSORD 算法稳定性研究的成果。

7.4.4.1 对车站之间和火车与车站之间通信的干扰

车站之间和火车与车站之间的互动是 RYNSORD 中最关键的通信任务,没有这些,火车就不能成功地预留轨道,也不能朝着它们的目的地行驶。为满足正确性的要求,除非火车关于某段轨道的预留请求已被明确批准,否则火车不能在该段轨道上行驶。如果一列火车起动的预留请求还在等待回答,火车绝不能尝试使用请求中的轨道。因此,这里有意引入干扰影响预留过程,但是不会影响 RYNSORD 的正确性。

这些干扰的特征为:消息从一个站传到另一个站的过程中永远不会到达目的地。此外,车站与火车之间的通信也不会到达接收器。如果发生这种情况,RYNSORD中的火车会做出如下行为:当一列火车没有收到对于它的预留请求的答复时,它会选择使用可用的备用路径,而不是无限期地等待回应。如果它

的两个预留请求都没有收到回应,火车会暂时改变它的前向值并重新发出预留请求。火车传送预留请求的两条通信链路都不通,这种情况几乎是不可能发生的,火车最近的活动会确保得到至少一个预留回应。而一旦这件不可能的事情发生了,火车会在车站等待,并且在下个时间步长根据重设的前向值重新开始它的预留请求。一般来说,计算机通信故障通常都是比较短暂的,本文推想等待一个时间步长是合理的,在这个时间内通信链路可能就恢复了,而火车不用着急地去迂回绕路。人们还认识到两个车站之间的通信故障并没有对于轨道区间本身的用途造成破坏,该故障仅仅影响到那些想通过不是该轨道控制者的车站实现预留的火车,因为信息永远不会到达该轨道的控制车站,它才是唯一有能力批准预留的车站,从控制该轨道的车站驶出的火车才可以请求并使用该轨道。

许多实验可用于测试稳定性:各种不同的通信链路故障、选择不同的故障持续时间直至永久性故障或使用不同的流量输入率值等,其目的都是分析通信干扰对RYNSORD 的影响,并确定使RYNSORD 在干扰下能保持稳定的流量输入率。在第一个试验中,两组分别由 3 条和 8 条链路组成的集合分别出现故障,这些链路依次通过其两端的车站来识别。如果有一条链路是将某车站和网络其他部分相连的唯一通信线路,就要特别小心,避免它出现故障。3 条链路以上的组合称为高流量链路,选择 8 条链路的组合说明我们期望考虑故障在整个网络上分布的情况。仿真时间为 17280 个时间步长,分别以 0.05 和 0.125 作为稳态输入率。故障在时间步长为 5760 时引入,持续时间为 1440 个时间步长。因此 $t_{\rm pert}$ = 5760,而 $t_{\rm pert_end}$ = 5760 + 1440 = 7200。选择故障持续时间为 1440 个时间步长,是因为它相当于一整天的实际操作时间,也提供了足够的维修时间。在另一组试验中,通信链路出现永久故障,即 $t_{\rm pert}$ = 5760, $t_{\rm pert_end}$ = ∞ 。

- 3 条链路的集合包括巴尔的摩 (34)、华盛顿 (33)、底特律 (9)、托莱多 (10)、罗阿诺克 (48) 和林奇堡 (47)。
- 8条链路的集合包括克里夫兰 (11)、哥伦布 (12)、罗切斯特 (27)、锡拉丘 兹 (28)、圣路易斯 (5)、底特律 (9)、威尔森 (40)、罗利 (41)、夏洛特维尔 (35)、里士满 (36)、纽约 (31)、费城 (25)、诺克斯维尔 (18)、布里斯托尔 (20)、帕克斯堡 (44) 和亨廷顿 (45)。

表 7.3 总结了 RYNSORD 的性能表现结果,并证明了就有限时间的故障而言,RYNSORD 是强稳定的。由于通信故障能被快速修复,所以仿真结果是令人满意的。不过,对于在两种链路集合中存在的永久性干扰,在流量输入率为0.05 和 0.125 时,RYNSORD 分别表现为临界稳定和不稳定。显然,临界稳定和不稳定的临界值是流量输入率、轨道故障数量和具体的轨道故障的函数。图 7.6a~图 7.9b 是 8 条故障轨道集合的情况,显示了在不同的流量输入率和干

扰持续时间下的误差标准Ⅰ和Ⅱ的变化情况。有人指出误差标准Ⅱ映射了误差标准Ⅰ的行为,3条故障轨道集合的结果与8条轨道集合的结果类似,在这里不作论述。

故障链路数	系统基本输入率	干扰时间/时间步数	干扰持续时间/ 时间步数	稳定等级
3	0. 05	5760	1440	强稳定
3	0. 125	5760	1440	强稳定
3	0. 05	5760	œ	临界稳定
3	0. 125	5760	œ	不稳定
8	0. 05	5760	1440	强稳定
8	0. 125	5760	1440	强稳定
8	0.05	5760	œ	临界稳定
8	0. 125	5760	α	不稳定

表 7.3 通信干扰下的性能表现

然而,对两种故障轨道集合的误差标准Ⅱ进行对比分析的结果(见图 7.10)揭示了下列情况:图 7.10 中的数据对应于较低的流量输入率和永久性故障。虽然两种情况下 RYNSORD 都是临界稳定的,但 3 条链路故障集合的最终稳态点更糟糕,即相对于 8 条链路故障的集合,其误差临界值更大。结果表明,具体的链路故障原因(如拥挤的程度)比出现故障的链路数量更重要。此外,仿真终止后进行的离线分析结果表明,使用一个或以上的 3 条链路组合的火车总数为 201 列,这些链路被用在它们的最短路线上。相反,只有 167 列火车会利用一个或以上的 8 条链路集合。因此,稳定分析可能有助于找出那些即使出现故障,也可能不会影响 RYNSORD 性能的通信链路。

7.4.4.2 与轨道区间段有关的干扰

发生事故、故障或者遭到破坏之后,当然也可能是由于维护的原因,某段轨道可能暂时无法使用。虽然 RYNSORD 并没有在设计过程中就制定出处理这种问题的办法,但是本章假设:一旦发生故障,轨道区间段的两个端点上的车站在一个时间步长内,即实际操作的 60s 内就知道了该故障的发生。此外,故障发生时正行驶在某轨道区间段上的火车将继续行驶,并安全到达另一端。在端点的车站将通过取消所有预留来阻止其他火车使用该轨道,并强制命令所有受影响的火车重新选择路径。

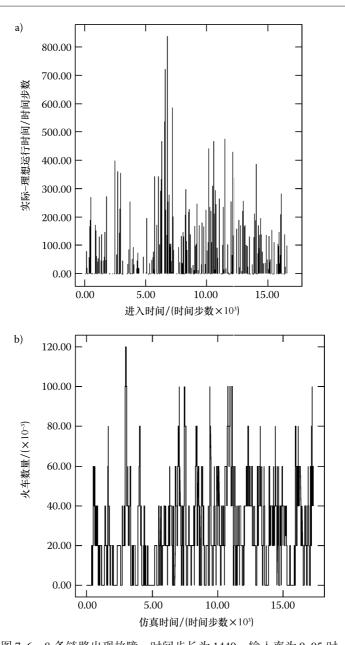


图 7.6 8条链路出现故障,时间步长为1440、输入率为0.05时 每列火车的误差标准 I 与车辆

进入时间的函数 (图 a) 以及每列火车的误差标准Ⅱ和仿真时间的函数 (图 b)

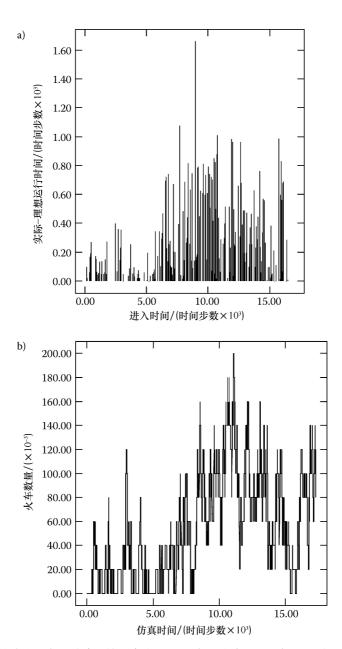


图 7.7 8条链路出现永久故障,输入率为 0.05 时每列火车的误差标准 I 随进入时间变化的函数 (图 a) 以及每列火车的误差标准 II 随仿真时间变化的函数 (图 b)

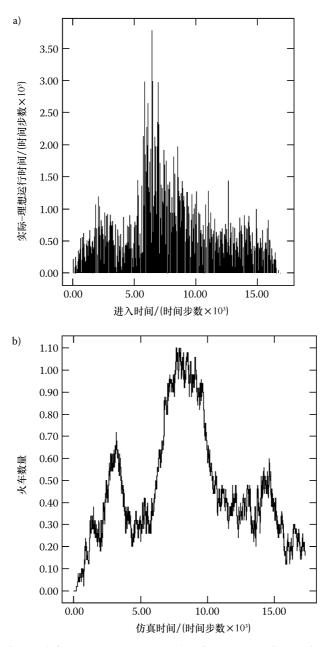


图 7.8 8条链路出现故障,时间步长为 1440、输入率为 0.125 时每列火车的误差标准 I 随进入时间变化的函数 (图 a) 以及每列火车的误差标准 II 随仿真时间变化的函数 (图 b)

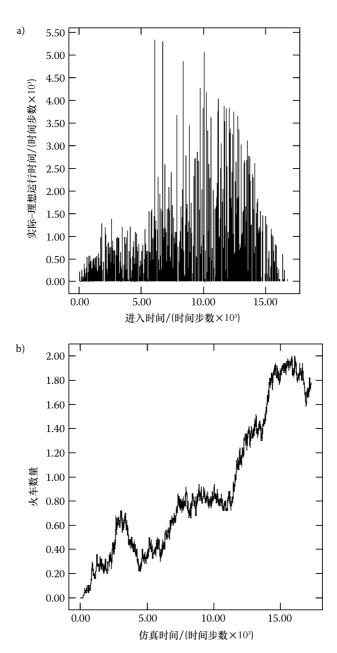


图 7.9 8条链路出现永久故障,输入率为 0.125 时每列火车的误差标准 I 随进入时间变化的函数 (图 a) 以及每列火车的误差标准 II 随仿真时间变化的函数 (图 b)

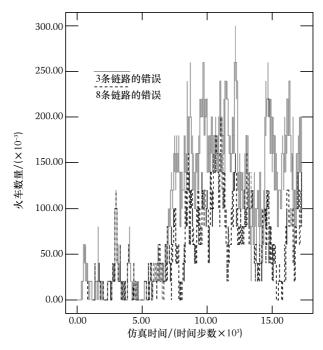
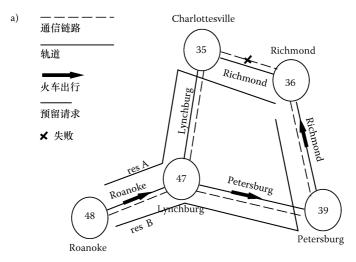


图 7.10 输入率为 0.05 的情况下,分别有 3条和 8条链路永久 失效时,每列火车的误差标准Ⅱ随仿真时间变化的情况

虽然轨道故障对火车产生的影响看起来与通信链路故障的影响类似,但是实际却有很大不同。图 7.11a 和图 7.11b 都是整个网络的一部分。图 7.11a 说明了夏洛特维尔(35)和里士满(36)之间的一个通信链路故障的情况,而图 7.11b 显示的则是夏洛特维尔(35)和里士满(36)之间的轨道故障情况。考虑一列火车在通信链路出现故障之前的某个时间,从罗阿诺克(48)进入系统,终点站是里士满。当链路正常时,预留请求得到答复,火车会选择其请求的路线 A。无论此时的通信链路情况如何,火车在夏洛特维尔和里士满之间的轨道区间段上正常行驶是可以保证的。相反,在图 7.11b 中,火车启动后,轨道发生故障,火车到达夏洛特维尔时,轨道仍未修复。此时火车被迫重新选择一条新的路径,甚至在通过帕克斯堡到达里士满之前,可能需要返回到林奇堡。这个例子似乎说明轨道故障可能会极大影响 RYNSORD 的性能,但接下来的分析将证明通信故障的影响可能更加严重。

我们设计了一个试验,其中轨道如本章之前所描述的情况一样:有2个分别由3条和8条轨道区间段组成的集合发生了故障,分别对应于由3条和8条通信链路组成的集合发生故障。表7.4所示轨道发生故障时的性能表现与通信受到干扰时的性能表现(见表7.3)完全一致。除了某些关键因素不同外,误差标准图也非常类似于通信干扰的情况。图7.12a描述了在高流量输入率下,8条链路(或轨道)的

集合发生故障,且故障持续时间为有限的 1440 个时间步长时,通信链路故障和相应的轨道故障的误差标准 II 变化情况。虽然 RYNSORD 在这两种干扰情况下都是强稳定的,图 7.12a 中两条曲线变化也非常类似,但还是可以看出通信干扰的误差幅度明显高于轨道干扰。图 7.12b 则显示在低流量输入率下,8 条链路(或轨道)集合发生永久性故障的情况下,误差标准 II 的变化情况。虽然在两种情况下,RYNSORD 都是临界稳定的,且图形的变化也是极为相似的,但是同样可以看到通信干扰对应的误差幅度远远高于轨道干扰。一种可能的解释是:对于一列火车而言,只有当故障轨道真正无法被一列火车使用时,通信链路的故障才会真正影响到火车竞争预留的能力,从而影响到火车使用轨道的能力。



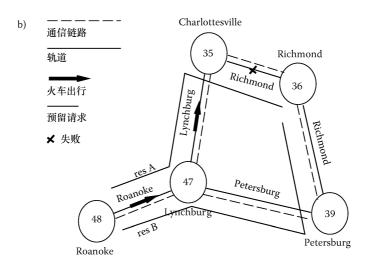


图 7.11 通信链路故障的影响(图 a)以及轨道区间段故障的影响(图 b)

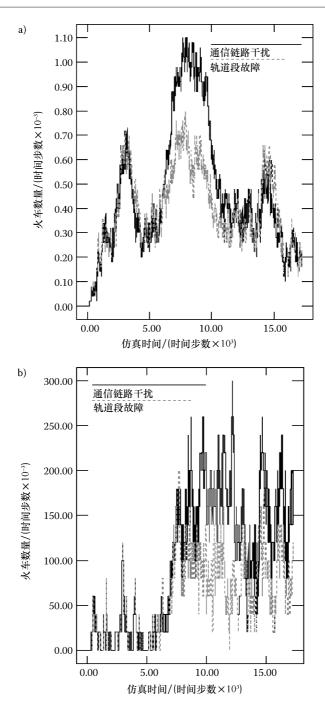


图 7.12 误差标准 II 在通信链路干扰和轨道干扰情况下随仿真时间变化的情况 a) 高输入率和有限干扰时间 b) 低输入率和永久性干扰

故障链路数	系统基本输入率	干扰时间/时间步数	干扰持续时间/ 时间步数	稳定等级
3	0.05	5760	1440	强稳定
3	0. 125	5760	1440	强稳定
3	0.05	5760	oc	临界稳定
3	0. 125	5760	oc	不稳定
8	0.05	5760	1440	强稳定
8	0. 125	5760	1440	强稳定
8	0.05	5760	oc	临界稳定
8	0. 125	5760	oc	不稳定

表 7.4 轨道干扰情况下的性能结果

研究的局限性

RYNSORD 的稳定性分析表明:在面对时间有限的流量输入率干扰和轨道区间段故障干扰时,它是强稳定的。而在遇到永久性干扰时,稳定性的判断取决于干扰开始之前的流量输入率。然而,在遇到通信链路故障时,RYNSORD 的稳定性表现较差,这部分的算法需要重新进行设计以提高抗干扰的能力。

第8章 ITS设计中的建模和仿真技术

8.1 引言

正如第3章所解释的以及第4~7章所体现的那样,ITS设计的一个重要特征是每个实体(比如机车、汽车等)都自带计算引擎,且在运输系统中由异步分布式算法控制。每个实体都有着明确定义的计算和交流需求,并被看做是一个异步自主式进程。在这些实体中,有些进程静止不动,有些进程在运输系统内移动。移动进程的具体移动方式由运输系统的特征和实际输入的数据来决定。由于每个移动过程都是自主进行,即每个移动实体都基于自身的行为、外界输入的刺激以及与静止实体间的互动来决定移动形式,因此移动形式非常复杂。每个移动或静止的实体都具有自己独特的计算和交流需求特征,而且,移动是异步进行的,即它开始移动的时间段不规则且不可预知。最后,在许多运输系统中,移动和静止的实体数量非常大,采用分布式可扩展的模拟和仿真技术非常有必要。

本章剩余部分组织如下: 8.2 节介绍两种互相竞争的用于 ITS 模拟和仿真的移动策略; 8.3 节详细描述了两种策略中的软件技术; 8.4 节详细介绍了在并行处理器试验台上执行仿真的过程; 8.5 节展示了在典型网络的实际输入条件下, 使用两种仿真技术所获得的结果, 并对其进行对比分析。

8.2 ITS 设计中移动策略的虚拟和物理的处理过程

本章假定 ITS 具有如下特征:①静态实体的数量相对合理,而移动实体的数量是庞大的,范围为 10~100s;②系统规模很有可能会随时间推移而不断扩张;③静态实体在地理位置上是分散的,而移动实体是自主的,这意味着每个移动进程的移动形式都是独特的且不可预知的;④静态进程之间能通过静态互联网直接联系,移动进程之间却假定为不需要直接进行通信,原因如下:第一,假设移动进程的数量很庞大,在任意两个实体之间提供直接通信很可能需要巨额的开销,这不利于系统的扩展;第二,系统巧妙地设计了分布式算法,使静止节点在必要时,能在移动过程之间起到协调作用。当然对于某些特殊的运输系统而言,在移动进程之间提供通信功能还是必要的。

因此,ITS 的计算机模型应包括能在计算引擎上执行的静态和移动进程,以及能促进静-静实体间和动-静实体间交流的机制。每个进程都有自己的控制线程,因此它们相对于系统中其他进程而言是自主和异步的。这些进程的处理能力是由系统本身的性质决定的。静态进程从其他静态进程和动态进程中获取必要信息,紧接着又被其他移动进程下载和使用。当连接静态进程的静态网络能实现永久互联时,移动进程则会以不规则的时间间隔与相应的静态进程实现动态异步连接或断开连接。当一个移动进程 M_i 选择与静态进程 S_i 脱离联系而与静态进程 S_k 建立联系时,对于所有合理的 i 和 k 值而言,就产生了一个移动。

在一个实际运作的运输系统中,每个静态或移动进程都由自身的计算引擎和设备来启动与其他进程的交流。因此,假定在一个 ITS 仿真过程中,每个静态和移动过程都与自己的计算引擎连接,这在逻辑上是合理的。然而,在许多并行处理试验台(包括本章用到的)中,可用的处理器可能比静态和移动进程总数少得多,这导致在试验台中通过进程来表达移动实体时会出现两种基本策略:虚拟和物理进程移动策略,本章接下来将会详细介绍。还有第三种方法,它是物理进程移动策略的一个变体,即在初始化时,每个静态实体和移动实体之间就建立了联系。当某个移动实体需要与某个特定的静态实体相互作用时,就会使用对应的连接。但在平时,这个连接是空闲的。但是,一旦取消了对动态建立和破坏连接的限制,每个 UNIX 进程中对开放连接数量的限制就会成为该方法的一个弱点。本章不会讨论到这个方法。

8.2.1 虚拟进程移动策略

一般来说,合乎逻辑的选择是将数量相对适中的静态实体表示成实际进程,并按一对一的原则,把它们分配给并行处理试验台上的处理器,而用虚拟进程表示移动实体。一个静止节点代表一个位于特殊地理位置的实体。一个虚拟进程在必要时会在处理器之间移动,它的计算需求会由其相对应的静态节点的主机来执行。根据定义,一个虚拟进程不能与某个处理器永久相连,有时它会与一个静态节点的处理器相连,执行其计算需求,并暂时给它分配一个实际进程状态。这种策略被称之为虚拟进程移动(VPM),实际上我们已在第4章和第6章讨论过这种策略。在第5章和第7章中,我们对VPM策略进行了适当的修改,用UNIX进程表示静态和移动实体,而UNIX进程是可以在有任意数量处理器的试验台上运行的。因此,在第5章和第7章中,需要的处理器数量不一定等于静态实体的数量,这与第4章和第6章的情况不同。虽然前文中作过描述,但本章仍会将VPM作为模拟和仿真ITS的一般方法做更详细地介绍。处理器之间相连的结构与静态实体间的拓扑结构相同,这种结构一开始就被初始化,并在整个仿真过程中保持不变。

VPM 中的虚拟进程就像一个操作系统中的"索引",然而,它并不像"索引"那样包含代码、存储单元、堆栈指示器以及程序计数器,一个虚拟进程只包含执行程序所必需的基本参数,而具体是哪些参数是由应用程序决定的。举例来说,在模拟和仿真 IVHS 时,移动车辆的参数包括车牌、模式、制造商、现行速度、设计速度、位置、方向、起点和终点。当一个移动实体位于一个静态节点时,它出现在那个节点,就在那里显示为一个实际进程,由该节点的主机执行并计算。移动实体可以利用静态节点和它自己所包含的相关信息来决定自己接下来的行动,其中就可能包括移动到另一个静态节点的决策。然后,相应的虚拟进程及其所有参数会迁移到移动实体再次出现的静态节点处。因此,一个移动实体的行为是完全独立的,不受所在的静态节点以及在相同的静态节点驻留的其他虚拟进程的影响。而且,在任何时刻,都可能会有一个或多个虚拟进程在一个静态节点处停留,争夺计算和通信资源。因此,可能需要利用调度程序来给各进程安排计算和通信设备。静态进程和虚拟进程之间的通信只需要通过复制缓冲区就可以很容易地实现。

图 8.1 描述了一个有三个静态实体的简单的 ITS 设计,用静态进程 SP-1、SP-2、SP-3 来表示。仿真包括以下三个处理器:处理器 1、处理器 2 和处理器 3。在给定的时刻,虚拟进程 VP-1~VP-5 都停留在图 8.1 所示的三个处理器上,而 VP-6 从处理器 2 迁移到处理器 1。在迁移过程中,首先终止对应于停留在处理器 2 处的 VP-6 的实际进程,其基本参数被压缩到一条消息中,然后这条消息又被传递给处

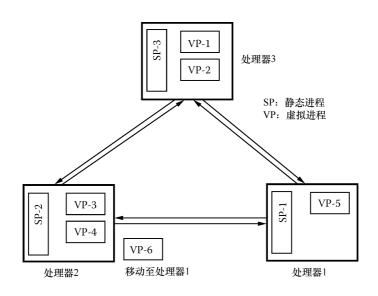


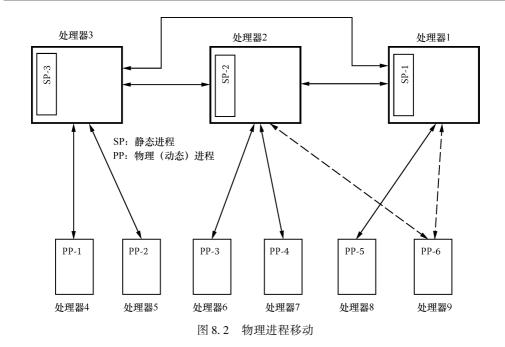
图 8.1 虚拟讲程移动

理器 1,并在处理器 2 处解码,最后,在处理器 2 处形成了与 VP-6 相对应的新的实际进程。每个处理器都必须执行静态进程、至少一个虚拟进程、调度程序以及通信基元。我们通过一项严密的测试发现了 VPM 的两个重要特征:第一,一般来说,对于一个从静态节点 A 移动到静态节点 B 的移动实体来说,A 点和 B 点必须是直接连通的。虽然这样也许可以降低系统复杂程度,但是也有可能产生更复杂的迁移。第二,在任意时刻,一个静态节点处的移动实体数量都受到操作系统允许处理的最大进程数的限制。

8.2.2 物理进程移动策略

尽管第4章和第5章中成功应用了 VPM, 但是它还是有些重要的局限性, 并很可能对模拟未来的 ITS 造成困难。随着移动实体数量的增加,对主处理器 计算和通信资源的争夺愈演愈烈,这可能会大大降低仿真速度。为了应对这种 局限性,本章提出了一种竞争式方法,物理进程移动(PPM)方法。在 PPM 中,每个进程(静态或动态的)都配置一个特殊的处理器。进程一旦在系统中 启动,该处理器就开始抢占资源,当进程结束时,抢占停止。当一个移动进程计 划在运行中与一个静态节点联系时,就先在其处理器之间动态地建立一个通信协 议,然后才开始信息交换。此后,当移动进程要与另外一个不同的静态进程互联 时,就要先解除旧的协议,然后再建立新的连接。一个移动进程可以在任何时刻 与某个单一的静态进程保持连接。因此, PPM 策略是对现实模拟得更精确的模 型。而静态进程之间的静态互联网络仍然采用与 VPM 中的一样的方式。显然, 每个移动进程的计算需求都由它本身的处理器执行, 在移动实体的计算需求量很 大的地方, PPM 可能会有比 VPM 更高效的处理能力。与 VPM 不同的是, 即使在 A 和 Z 之间没有直接连通的情况下, PPM 中一个移动进程也能够容易地从静态节 点 A 迁移至 Z。PPM 的主要优势在于每个进程都只使用一个处理器。但是,由于 现在的试验台技术有限,还不能广泛使用配有1000台处理器的试验台,因此使 用 PPM 策略的 ITS 仿真只能局限于中等规模的移动计算网络。PPM 对动-静态进 程之间的通信成本也有上限限制,这一通信过程包括动态通信协议下的精确消息 的交换。

图 8.2 描述了在图 8.1 所示的简易 ITS 设计中 PPM 的使用情况。除了模拟静态进程的三个处理器之外,处理器 4 到处理器 9 构成了 6 个物理进程,PP-1 到 PP-6 是对应于 6 个移动进程的处理器。在图 8.2 中,静态进程和移动进程之间的实线表示某时刻下正在使用的协议,虚线表示正在建立或者已经断开的协议。因此,VP-6 和处理器 1 或 2 之间的虚线反映出物理进程 PP-6 正在从静态节点 SP-2 向 SP-1 移动。



8.3 进程移动策略中的软件技术

在仿真初始化期间, VPM 和 PPM 中连接静态进程的静态网络就已经建立了。 正如之前所提到的那样、每个静态进程被分配一个处理器或者工作站、称为节点。 在运行期间、进程先利用工作站标识符打开一个唯一的外部输入文件。这个文件包 含节点与其他静态节点之间的连接情况。然后,进程开始利用伯克利套接字协议, 一次一个地建立点对点连接。在两个处理器之间建立点对点的连接时,发起进程会 执行 "connect", 而相应的接收进程会执行 "accept"。每个连接都是半双工的, 即 网络图中的有向边, 网络图中还可能会有多重相互嵌套的圈。"connect"是无阻塞 的,需要接收进程标识符作为参数,而"accept"是阻塞的,并能从任一处理器接 收任一 "connect" 命令, 这可能导致网络在初始化时就出现死锁。为了应对这个 问题,一般采用下面的算法。下层节点都有自己唯一的标识符,当一个节点(标 识符为X) 处的静态进程需要与另一个节点(标识符为Y, Y > X) 处的静态进程 连接时, X 总是执行 "connect", 而 Y 则执行 "accept"。在内存中完成了网络的设 置后,每个节点会对静态和移动进程进行初始化运行,这个过程在 VPM 和 PPM 策 略中是不同的。图 8.3 中的伪代码说明了在每个节点处建立静态互联网的函数。在 图 8.3 中, L1 行代码首先将大于当前节点标号的节点放进系统静态节点集合中, 然后从 L2 行开始的代码段试图通过执行 "connect"命令,在当前节点和系统静态

节点集合中的每个节点之间一个一个地建立连接,直到成功建立所有连接为止。从 L3 行开始的代码段则是关于当前节点和系统中标号小于当前节点的其他节点之间 建立连接的过程。

```
void make_connections()
L1: let t be the set of nodes with great identifier values;
L2: while (number of elements in t = 0)
          create socket;
          choose a node from t;
          execute a connect system call to the selected node;
          if(fails) {
            close socket;
       } else {
            send my identifier value to the selected node;
            save the socket descriptor;
            remove the node from t:
L3: create socket;
     set up address (sockaddr_in) structure;
     bind the socket using the address structure;
     call listen relative to the socket;
     let f be the set of nodes with lower identifier values;
     while (number of elements in f = 0)
       execute an accept system call;
       if(fails) {
            continue:
       } else {
            accept returns new socket;
            read the remote node's identifier value from the new socket;
            save new socket descriptor;
            remove the node from f;
close original socket;
```

图 8.3 节点处初始化过程中建立静态互联网络的伪代码

8.3.1 VPM 中的软件技术

在 VPM 中,每个移动实体都由一组参数表示,这组参数构成一个结构体。结构的大小是应用对象的函数且为动态的。为了对 VPM 和 PPM 进行比较,假定每个移动实体结构中都包括以下字段(见图 8.4):第一个字段是实体的标识符(标号);第二个字段反映了计算需求;第三个字段将实体必须与主静态进程进行交换的其他消息数进行压缩;第四个字段存储了实体移动模式中剩余的跳频数。

一个移动单元的计算负荷用一个 100~10000000 之间的整数表示,它用一个 "for loop"循环来实现,即"for loop"循环中的迭代次数就等于计算负荷值,也 等于利用迭代循环来仿真实际计算时间的执行次数。在 VPM 中,在任一时刻下,为了争取同一个控制线程,可能会有一个或多个移动实体在同一个静态节点处驻留。为了保证每个移动实体都能公平地使用该控制线程,仿真提出采用时间分割的方法,即每个虚拟进程每次只执行 100 次迭代,然后该进程将自动放弃控制。

```
struct VPM_entity{
    int id;
    int load;
    int message_per_hop_remain;
    int hops_remain;
}
```

图 8.4 VPM 中每个移动实体的结构

移动实体到达节点之后,就被看作一个实际进程,并在调度程序列表中排队等候。调度程序把时间空当安排给各进程,并在时间空挡中执行移动实体的主体程序。图 8.5 中的伪代码表示正通过迭代进行仿真的移动实体的程序主体。迭代按 100 次为一组执行。在图 8.5 中,从第 L1 行开始的代码段检查是否还有迭代没有执行。如果是肯定的,移动实体应至少执行 100 次迭代;如果是否定的,则意味着移动实体已经完成所有预定的迭代。下面标号 L2 的代码段检查是否还有需要进行交流的信息未被传递。如果是,移动实体将通过写入动态地址来模拟完成该信息的交流;如果不是,则意味着所有预定的信息已完成交流。标号 L3 的代码段检测所有预定的迭代和信息交流都已完成的情况。然后就可以确定移动实体的下一个迁移,并将该实体传送至下一个目标节点。

```
void wake_mobile_entity(struct VPM_entity)
{
L1:if(load > 0) {
          do minimum(100,load)for-loop iterations;//time slice is 100//
          decrement load;
}
L2:if(message_per_hop_remain > 0) {
          write to a variable location to emulate a message;
          decrement messages_per_hop_remain;
}
L3:if(load equals 0 AND messages_per_hop_remain equals 0) {
          determine stochastically where this entity must migrate;
          migrate the process and remove it from the scheduler's list;
}
```

图 8.5 移动实体在节点处作为进程出现

在迁移过程中,移动实体的参数被压缩成一条消息。其地址和大小通过调用"write"函数传递给操作系统,该程序将信息写入相应的发送套接字并执行转换过程。

在接收端,节点在0超时条件下调用"select"函数选择到达的移动实体。该选择仍具备调用单一函数对多个套接字连接进行监测的能力。消息到达时,节点将其看作一个实际进程,并将其放进调度程序列表中排队等候。图 8.6 中的伪代码解释了节点的这个功能。在图 8.6 中,标号 L1 的代码段检测是否有一条封装了到达的移动单元的新消息。如果检测到该消息,节点就会读取该消息并生成相应的进程。

```
VPM_entity check_migrate()

struct timeval timeout;
timeout. tv_sec = 0
timeout. tv_usec = 0
fd_set fdvar;
FD_ZERO(&fdvar);
FD_SET(socket,&fdvar);
L1:if(select(socket + 1,&fdvar,0,0,&timeout))
    VPM_entity new_entity;
    read(socket,&new_entity,sizeof(VPM_entity));
    return new_entity;
} else {
    return 0;
}
```

图 8.6 节点拦截封装了移动实体的消息

调度程序对静态进程,以及可能在主处理器驻留的一个或多个移动进程进行循环调度。在程序执行时,移动实体首先离开等候队列,执行完成后,移动实体或者重新进入调度列表排队,或者被标记为要移动到另一个节点的实体。由伪代码编写的调度函数如图 8.7 所示。在图 8.7 中,标号 L1 的代码段反映了调度程序让移动实体队列中的第一个元素出列,并执行它的过程。如果这个实体已经完成了迭代并标记为准备迁移,并被代码段 L2 检测到,调度程序就会以消息的形式封装并输出这个实体。否则,这个实体会重新回到节点处的移动单元列表进行排队等候。

```
void schedule()
{
L1:dequeue the first element from the queue of waiting mobile entities;
    call wake_mobile_entities() for this mobile entity;
L2:if(mobile entity migrated) {
      write(socket,&mobile_entity,sizeof(VPM_entity));
      free the structure corresponding to the migrated mobile entity;
L3:} else
      enqueue the mobile entity back into the queue of waiting mobile entities;
}
```

图 8.7 每个节点处的调度程序

节点处执行的主程序将上述子程序整合到一起,描述了整个实施过程,如图 8.8 所示。首先,数据结构按代码段 L1 所表述的那样进行初始化,然后,代码段 L2 利用封装在外部数据文件中的信息压缩移动实体。程序的剩余部分将会按照适当的顺序反复执行下面两个操作:①检查是否有一个新的移动节点从邻近的静态节点迁移过来。如果是,利用网络消息将这个移动实体再现成一个实际进程,然后将其插入调度程序的队列中,准备在将来某一时刻执行。②执行调度程序,将分割的时间空隙分配给排队的移动节点。代码段 L3 和 L4 分别描述了操作①和操作②。

```
void main ()
{
L1: initialize data structures for scheduler, etc.;
L2: synthesize mobile entities at this location;
    loop until end of simulation {
L3: call check_ migrate;
    if (new mobile entity has arrived from an adjacent stationary entity) {
        assign the mobile entity to the scheduler;
    }
L4: execute schedule () to schedule the execution of the mobile entities;
}
```

图 8.8 每个节点处的主程序

8.3.2 PPM 中的软件技术

1. 移动实体

初始化阶段,每个移动实体都与一个处理节点有关,这个节点负责执行迭代循环、信息传递以及迁移路径的计算。与静态节点有关的信息转换需要用到网络协议。

在迁移过程中,必须首先终止移动进程和静态进程之间原有的连接。正常情况下,会在连接的任意一端执行"close"系统调用来完成这个任务。然而,如果两个处理器执行系统调用不同步,连接就只能部分关闭,而当进程试图写入程序时,系统会生成一个 SIGPIPE 信号。为了避免这种我们不期望的情况发生,两端需要配合良好。在本章中,当一个连接必须终止时,首先由移动节点发送一条终止命令到静态节点。然后,在两端进程都执行"close"系统调用后,移动节点会等待静态节点的认可。得到认可后,移动节点才会与目标静态节点建立新的连接。图 8.9 中的伪代码描述了迁移、切断连接以及重新连接等函数。其中迁移函数会调用 disconnect 和 try_connect 函数。

```
void migrate(stationary node)
{
    call disconnect() to terminate connection;
    determine stochastically where this process will migrate;
    try_connect(stationary node);//attempt connection to the new stationary entity/
}
void disconnect()
{
    send disconnect command to the current stationary node;
    wait for an acknowledge signal;
    call close to terminate the connection;
}
int try_connect(stationary node)
{
    call connect() to attempt connection;
    if(success) |
     send id of this mobile entity;
    return OK;
} else |
    return ERROR;
}
```

图 8.9 PPM 中每个移动实体的迁移、断开连接和重新连接函数

图 8.10 所示的 mainLoop(),整合了上述所有子函数,以表达 PPM 中移动节点处的全局函数。首先,系统会从外部数据文件中读取静态节点的网络拓扑结构(如代码段 L1 所述),在仿真过程中利用该信息决定与迁移有关的问题,然后,代码段 L2 用于创建内部数据结构。接着,系统会重复执行以下三个操作:①执行迭代循环以模拟实际计算过程;②与当前所连接的静态节点交换模拟数据元;③检查是否所有数据元已完成交换以及是否执行了所需要的迭代循环次数。如果是,那么移动实体将随机确定一个新的静态节点,并启动迁移过程。从代码段 L3、L4 和 L5 开始的程序分别表示了任务①、②和③。

```
void mainLoop()
L1: read static network topology from external data file;
L2: initialize internal data structures;
     establish connection with the default stationary node;
     loop until end of simulation
        if (load > 0)
L3:
            do minimum (100, load) for-loop iterations;
            decrement load:
L4:
        if(messages_per_hop_remain > 0)
            write( socket, default_message, length( default_message) );
            decrement messages_per_hop_remain;
L5:
        if(load equals 0 AND messages_per_hop_remain equals 0) {
            reset the values of load and messages_per_hop_remain;
            determine a new target stationary node, stochastically.
            call migrate();
```

图 8.10 PPM 中对应于每个移动实体的主程序

可以看出,对于 VPM 示例来说,图 8.9 和图 8.10 所示的程序段是被封装在静态节点中并由静态节点执行的,而在 PPM 示例中,该程序段是由每个移动实体进程自己执行的。同时,图 8.9 中切断连接和重新连接的函数在 VPM 中没有对应的部分,而在 PPM 中,这两个函数却是构成移动单元的基础。

2. 静态实体

一个静态节点有以下三项职能:①从移动实体接受连接请求;②与移动实体交换数据:③接受移动实体的终止请求。

```
void mainLoop( ) {
read the static network topology from the external data file;
establish connections to other stationary nodes using make_connections();
create and bind special socket to permit mobile nodes to connect;
struct timeval timeout;
timeout. tv\_sec = 0
timeout. tv\_usec = 0
loop until end of simulation
    fd_set fdvar;
    FD_ZERO(&fdvar);
    FD_SET(mobile node connection socket, &fdvar);
L1:
           if(select(mobile node connection socket + 1,0,0,&timeout)) {
               issue accept system call;
               read mobile node id and save new socket descriptor and id;
         for (all mobile node sockets)
               FD_SET(socket, &fdvar);
         select(highest descriptor value + 1,&fdvar,0,0,0))
          read message from socket;
          switch (message type)
L2:
             case disconnection;
                 send acknowledge signal;
                 close the socket:
                 remove the socket handle from storage;
            break;
L3:
                case data transfer;
                      read message;
                      discard message;
              break;
```

图 8.11 PPM 中每个静态节点的主程序

假设移动实体对某移动节点到某静态节点的连接进行异步动态地初始化处理,那么每个静态节点都必须提供出入点,使移动节点可以对连接进行初始化。为了获得出入点,每个静态节点会与一个专门的套接字绑定在一起,并定期通过执行"select"系统调用来确定是否有移动节点需要与之连接。图 8.11 中的代码段 L1 可以完成这个任务。如果有需要连接的移动节点,静态节点就会执行"accept"系统调用,并读取和存储标识符,然后开始与该移动节点建立连接。一旦连接上,就有两种消息需要传递:数据元和切断连接的请求。数据元可以在移动节点和静态节点之间进行交换,而在本章中,这些数据是模拟的,很容易丢失。当移动节点打算断开连接时,它会给静态节点传送切断连接的消息。反过来,静态节点会反馈一个认可信息,并执行"close"系统调用以切断连接。图 8.11 的代码段 L2 和 L3 都是关于如何收取切断连接的请求和如何转换数据元的。

如图 8.11 所示, PPM 中由每个静态节点执行的程序与图 8.8 不同,图 8.8 中的程序既不仿真也不执行与移动实体有关的活动,而是从移动实体中接受连接请求和断开连接的请求。

8.4 实施情况

VPM 和 PPM 策略都是在拥有 65 + SUN Sparc 10/40 工作站的试验台上实施的,这些工作站配置成一种松耦合的并行处理器。每个工作站都配置有 32MB 的内存和 Solaris 2. 3 的操作系统,它们通过带宽为 10Mbit/s 以太网相互连接。除此之外,设计的代码可以同时在 SUN OS 4. 1. 3 和 Linux 操作系统^[108]环境下执行。编程语言为 C + + ,长度约 2000 行,由 GNU g + +编译。用户在控制台操作程序,而代码进行后台处理,本文展示的仿真数据都是在夜间,网络负荷很小的时候获得的。

8.5 仿真结果和性能分析

为了比较分析 VPM 和 PPM 的性能,这两种策略都在同一个并行处理试验台上进行建模和仿真。试验台在非常接近实际情况的环境下,进行了大量实验设计和操作。一个实际移动计算网络的关键参数包括:静态网络的规模,即静态网络中的静态节点数量;静态网络的互联拓扑结构、移动实体数量、移动实体计算量、移动实体和静态实体每次交换的信息量以及移动实体的移动形式等,仿真过程中会采用一些独立变量来表示这些参数。评价性能的一个关键指标是完成试验台上所有处理程序所需的最长时间。在实验过程中,实体数量是受限制的,因为试验台只有65个工作站。假定静态互联网络拓扑是全连通时,静态节点的数量范围是5~10个,移动实体的数量范围是5~50个。一个移动单元的计算负荷用100~100000000 之间的

某个整数表示,形成一个简单的"for loop"循环(见图 8.5 和图 8.10),这意味着"for loop"循环中的迭代次数就等于计算负荷值,也等于迭代循环模拟实际计算时间的操作次数。假定一个移动实体和一个静态实体之间的数据元交换量为 1~100,其中,每个数据元素的虚拟长度为 128B,这与第 4 章使用的消息尺寸是一样的。消息的交换也参考图 8.5 和图 8.10 中提到的方式,其中也展示了每个移动实体的移动形式:①随机的;②唯一的,即所有移动节点的移动形式都是相互独立的;③异步式的,即移动实体会以不规则的时间间隔发生迁移。这里对移动单元唯一的限制是实体不能立即与它刚刚连接过的静态节点重新连接。在仿真过程中,除另外说明的情况外,每个移动实体会与静态节点连接或切断连接的总次数是 1000 次。

假设并行处理试验台的处理器都是异步的,执行速度不同,时间也不同步,那么仿真过程中事件执行的顺序一般也与实际操作中不同。有时人们会采用一些不同的同步技术,来保证事件执行顺序的有效性。但是使用这种技术会加大仿真中人为的痕迹,产生与实际不符的仿真结果。这些同步技术还会大大降低仿真速度,对 VPM 和 PPM 都造成相似的影响,因此本章设计实施 VPM 和 PPM 时一律不使用这些技术。但是要注意的是,在第5~7章实施 VPM 时,是使用了同步化技术的。

仿真一共运行了 200 次,每次大约需要 1000s, 65 个工作站同时执行,仿真中会产生几 MB 数据。当一个移动实体与一个静态实体相连时,首先它会在计算负荷值范围内执行运算,然后与静态进程交换数据。当每个移动实体都完成了规定数量的连接和断开后,仿真终止。用于评价的仿真时间包括建立协议连接所需的时间、运算时间、交换数据的时间以及断开连接的时间。

图 8. 12a 分别表示了 VPM 和 PPM 中仿真时间随移动实体的计算负荷变化的情况。静态节点数设定为 10,移动节点数分别为 5、10 和 50。每次数据元交换量设定为 1,且每个移动实体总共要连接和断开 1000 次。这些曲线显示:当计算负荷增加时, VPM 中的仿真次数增长很快,而 PPM 中的仿真次数仍保持相对稳定。对于 5、10 和 50 个移动实体而言, PPM 的仿真次数分别为 110s、300s 和 710s,且一直保持不变。在计算负荷值较小时, VPM 有较好的表现,因为 PPM 中连接和断开的花销很大。对于 5 个移动实体而言,当计算负荷超过 20000 时,PPM 的表现优于 VPM。同样,对于 10 个移动实体,当计算负荷超过 50000 时,PPM 的表现也超过了 VPM。更进一步来说,PPM 成功地对 50 个移动实体进行仿真时,计算负荷值高达 100000,需要 710s,而 VPM 所需要的运行次数则远远超过 100000 次。当每个中继站交换的数据量设定为 10、有 5 个静态节点和 5 个移动实体时,图 8. 12b 显示 PPM 与 VPM 的比较结果也是差不多的。因此,对于较小的数据元交换量、中等的移动实体数量、计算负荷值在 100000 以下的情况,PPM 策略具有较好的可扩展性,并且表现出更优越的性能。

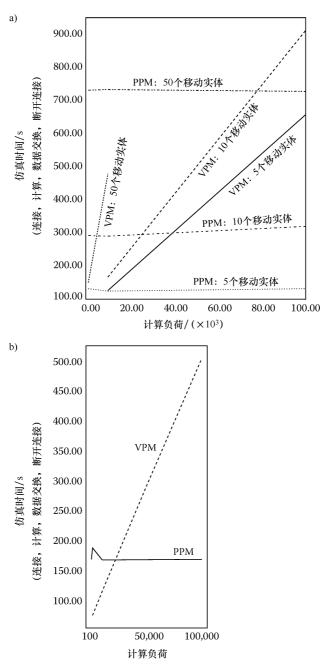


图 8.12 VPM 和 PPM 中的仿真时间是移动实体计算负荷的函数 (静态节点总数为 10,中继站每次交换的数据元个数为 1,移动单元要完成 1000 次连接和断开) a)每次交换 10 个静态节点和 1 个数据元 b)每次交换 5 个静态节点和 10 个数据元

图 8.13 重新组织了图 8.12a 中的数据,在计算负荷值分别为 1000~100000 的各种情形下,将仿真时间作为移动节点数的函数来画图,其目的在于揭示移动实体数量变化时,计算负荷值对 PPM 和 VPM 表现的影响。可以观察到,计算负荷值较小(在1000~10 000 之间)时, VPM 性能比 PPM 好;超过 100000 时,PPM 性能较好。PPM 中的曲线差不多重叠在一起,这表明 PPM 中更多数量的处理器均匀高效地分担了高负荷的计算任务。

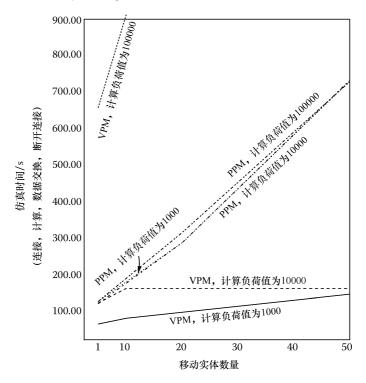


图 8.13 VPM 和 PPM 中的仿真时间是不同计算负荷值情况下移动实体数量的函数 (静态节点数为 10,每次交换的数据元个数为 1,移动单元要参与 1000 次连接和断开)

图 8. 14a 和图 8. 14b 绘制了 VPM 和 PPM 中仿真时间随着每次数据元交换数量而变化的情况。移动实体数设定为 10。图 8. 14a 和图 8. 14b 分别将计算负荷值设定为 10000 和 100000,分别考虑静态节点数量为 5 和 10 的情况。VPM 曲线显示其仿真时间增长与数据元交换量的增长保持基本一致,因为复制缓冲区是非常快速的。然而,我们观察到,10 个静态节点的 VPM 曲线比 5 个静态节点需要更多的仿真时间,这似乎是违反常理的,因为人们一般认为 10 个处理器,即 10 个静态节点去完成移动实体的计算任务要比 5 个处理器,即 5 个静态节点要快。出现这样的观察结果的原因在于,在每个静态节点中,将处理器标识符转换为套接字描述符号的函数

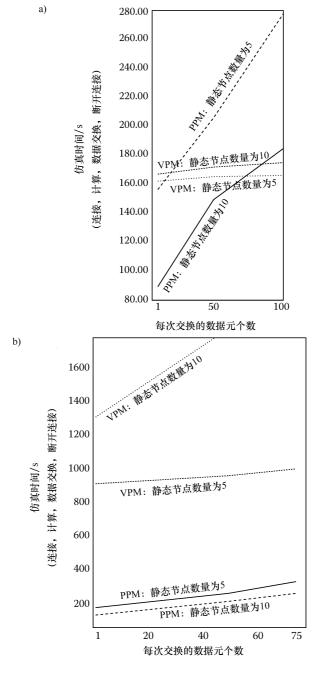


图 8.14 VPM 和 PPM 中的仿真时间与每次交换的数据元个数的函数关系 (移动单元数为 10,移动单元要参与 1000 次连接和断开) a)负荷值为 10000 b)负荷值为 100000

是通过一个链表来实施的。在一般常用的方法中,每次执行函数时,都要依次搜索一遍链表。假设静态节点是全连通的,那么套接字的数量就会迅速增加,这样就会导致有 10 个静态节点的系统仿真速度比有 5 个静态节点的系统慢。UNIX 效能评测器显示: 10 个静态节点的系统会在该函数上花费 40% 的仿真时间,而 5 个静态节点的系统则只花费了 23% 的仿真时间。由于处理器标识符与套接字描述符之间的绑定是固定的,人们已打算采用矩阵数据结构来提供更直接更快的访问路径。

然而,在 PPM 中,随数据元交换的是精确的消息,所以数据元交换量的增加意味着需要更多的仿真时间,PPM 曲线的斜率证明了这一点。对于 10 个静态节点的系统而言,PPM 仿真时间虽然是增长的,但是在每次交换数据元的数量达到 90 之前,一直都低于 VPM 的仿真时间。显然,PPM 中由 20 个处理器执行计算任务要比 VPM 中 10 个处理器来执行快得多。但是,当数据元素交换量超过 90 时,PPM 中由于传递精确信息产生的费用会超过拥有更多计算单元所带来的优势,这时,VPM 会在性能上超越 PPM。静态节点数为 5 时,PPM 中会配备 15 个处理器,而 VPM 中只有 5 个处理器。假设中等计算负荷值为 10000,那么 PPM 对 5 个静态处理器的连接和断开压力将持续增加,再加上高额的精确信息传递花费,都将导致 PPM 的表现劣于 VPM。也就是说,当每次数据元素交换量超过 10 时,PPM 会失去它的性能优势。

当计算负荷值较高,如为100000时,PPM中更多数量的处理器很可能会使它的表现优于VPM。图 8.14b的曲线进一步证实了这一预期,可以看出,在每次数据元素交换量从1增加到75的情况下都是如此。图 8.14a中PPM曲线的斜率比VPM曲线的斜率大,但是图 8.14b中,两种曲线的斜率是相近的。这是因为较大的计算负荷值抵消了PPM中昂贵的信息交流费用对其产生的影响。当有10个静态节点时,PPM 仿真过程会使用20个处理器,只有5个静态节点时,PPM使用15个处理器,并且和预期的一样,前者的性能要优于后者。

总体来说, VPM 和 PPM 在模拟和仿真实际的移动计算网络时都起到了极大的作用。PPM 从本质上来讲更切合实际情况,但是 VPM 也能以适当数量的处理器实现拥有大量移动实体的网络在试验台上的仿真。第 6 章已描述了在一个有 60 个 SUN Sparc 10 工作站的网络上,模拟有 45000 个自主车辆实体的 IVHS 的过程。相比之下,PPM 实施起来会比较困难,因为它需要大量的处理器,而且由于要处理大量的连接和断开,会使它的运行速度非常慢。当计算负荷以及移动实体数都很高时,PPM 无疑会更好一些。然而,与 VPM 相比,当系统需要移动单元和静态单元之间频繁进行数据交换时,PPM 的性能会差一些。我们希望在未来,随着试验台技术的大大改进,随着处理器数量的增加,连接和断开费用的降低,PPM 能得到更广泛的运用。

第9章 ITS 未来的问题

我们可以预知,未来的智能运输系统(ITS)将在两个重要方面取得显著进 展。首先,在不久的将来,智能运输方式的核心理念将超越汽车和火车运输的范 围,扩展到其他运输方式,如航空客、货运输,海运等以及个性化快速公交 (PRT) 系统等。在更远一些的未来,我们甚至可能看到 ITS 的发展延伸到星际旅 行。我们会在以下三个方面更深切地感受到人们对智能运输的需求: 行驶速度的增 加、出行者数量的显著增加以及出行者对精确和及时信息的需求增加,所有这些在 未来都是极有可能出现的。从科学和工程的角度来看,智能运输的进步将在理论和 技术创新方面都有所体现。然而,从普通旅客的角度来看,智能运输的真正进步和 直接获益之处在于不同运输方式之间实现无缝的一体化衔接。这种一体化衔接的结 果应该是:①旅客可以从系统中的任何位置获得世界上任何地方、任何一种运输方 式的准确信息:②旅客可以在任何时刻,甚至是在旅途中,对世界上的任何一种运 输方式实现有效的和动态的预订。信息的准确性和及时性对旅客建立对系统的信心 和信任是至关重要的,而这种信心和信任是可以相互传播的。旅客可以利用智能个 性化辅助决策系统处理所获得的信息,计算最有效的路径,或在所有不同的运输方 式 (航空、铁路、汽车、轮船等) 中重新设定路线。而重新规划出行的最常见原 因可能有:旅客的出行目的或需求发生改变,或者在当前预定的运输系统中发生了 预料外的延迟等。

其次,ITS 是一个非常复杂、昂贵的系统。ITS 一旦建立,人们就期望它能在较长一段时间内持续地提供服务,这一要求显然是合乎逻辑的。因此,绝对有必要对 ITS 发展建立一个正确的和全面的认识,对 ITS 的建设和扩展做好规划和部署,因为对该系统的需求必然会随着时间推移而增长。基于这个原因,同时考虑效率性和经济性,我们有必要利用目前所能采用的最实用的科学工具——行为模拟和异步分布式仿真技术,对 ITS 的体系结构和设计的具体细节进行深入研究。在异步分布式仿真中,运行一次高度复杂的系统仿真只需要几天时间,而目前的单处理器仿真系统可能需要数月时间。假定一项研究可能需要运行上百次仿真过程,就可以利用行为建模和异步分布式仿真来更好地理解这种复杂系统的行为,因此,行为建模和异步分布式仿真已成为发展未来 ITS 的不可缺少的工具之一。举一个例子,如果需要将美国某个地理区域内若干个交通管理中心连接起来,就可以对方案进行模拟和仿真,其结果可以为我们构建控制中心之间互联和信息交换的合理

结构提供有意义和有价值的参考,促使我们实现所设定的长期目标和高层次目标。投资数百万美元,建立一个在完成后无法实现目标和持续发展的系统,明显是不合理的。

未来 ITS 的研究范围会很广,一些其他待研究的重点领域包括:为 PRT 设计新的体系结构、开发嵌入式光纤,并将其用于高速公路以评估车辆的平均速度、把通信网络设计的基本原则融合到 ITS 设计中去等。

第 10 章 CD- ROM 中的 RYNSORD 仿真器 及其实验范围说明

这一章中,我们将说明随书附带的 CD-ROM 中的 RYNSORD 仿真器的设置和使用情况。与第 5 章和第 7 章中所描述的不同,本书提供的软件是为 Linux 操作系统而设计的。我们相信 Linux PC 的低成本和高可用性能使之成为构建分布式仿真平台的一个理想选择。系统硬件要求如下:

- 1) 奔腾级 CPU:
- 2) 32MB 内存;
- 3) 执行文件的磁盘空间约为 100KB, 但还需要额外的磁盘空间供输入和输出 文件使用;
- 4) Linux glibc ELF 系统——kernel 2.0.36 已经过测试, 但也可以使用其他版本:
 - 5) TCP/IP 网络可用。

此外,帮助文件假设终端 rsh 是一个可信任的主机,所以不要求使用密码。由于存在这个安全隐患,因此不推荐主机连接互联网。

如果在一个并行处理器配置中使用多个主机,可执行文件和输入文件都必须手动或通过 NFS 来分发。

10.1 安装

- 1) 将 CD 设为根目录 mount/dev/cdrom/mnt/cdrom
- 2) 将 rynsord tar 文件复制到设定目录 (例如/rynsord) cp/mnt/cdrom/rynsord. tar/rynsord 中
 - 3) 将该文件提取出来:
 - ① cd/rynsord;
 - 2 tar xvf rynsord. tar
 - 4) 显示为:
 - -rwxr-xr-x 1 ts1 ts1 10316 Jul 26 21:18 in_gen
 - drwxr-xr-x 2 ts1 ts1 1024 Ju1 26 21:41 input
 - -rwxr-xr-x 1 ts1 ts1 4296 Jul 26 21:39 input_merge

```
-rw----- 1 ts1
                            989
                                   Jul 26 21:55 network. 10
                   ts1
                                   Jul 26 21:44 network. out- > network. 10
1 rwxrwxrwx
            1 ts1
                             10
                   ts1
                             19396 Jul 26 21:18 out an
- rwxr- xr- x
          1 ts1
                  ts1
drwxr-xr-x 2 ts1
                             1024 Jul 26 21:56 output
                  ts1
-rwx----- 1 ts1 ts1
                            2252 Jul 26 21:53 script. 10
-rwx----- 1 ts1
                    ts1
                             11466 Jul 26 21:40 script_night_4. sh
- rwxr- xr- x 1 ts1
                            36296 Jul 26 21:17 tsw4
                  ts1
                            28360 Jul 26 21:17 tsv4
-rwxr-xr-x 1 ts1
                  ts1
输入与输出目录会被自动创建。
```

10.2 概述

每个火车站被模拟为一个 UNIX 进程 ("tsw4"),在仿真中被表示成铁路拓扑节点的车站组可以在任何可用的主机中运行,可以在每个主机上运行一个进程,也可以运行单个主机上所有的进程。这种仿真系统不是为 CPU 密集型的现代化机器群设计的,在一台主机上运行应该没有问题。请注意,目前提供的仿真系统仅能模拟 10 个车站。

不管有多少台主机,典型的操作方法是(以10个车站组成的拓扑结构为例,给定用户根目录为"/rynsord"):

- ① 创建相应的 network. out 文件。所有进程都查找文件"/rynsord/network. out"。
- ② 用户打开 10 个 x 客户端 (对不同主机,必要时使用 rsh 命令)。
- ③ 在每个 x 客户端执行下列操作:

cd ~/rynsord

setenv NETUID < uid >

setenv NETPORTNUM < portnum >

where uid and portnum are the unique identifier and portnumber for that node. A complete description of this is in the next section.

④ 每个节点的车站软件启动:

tsw4-t 10080 -1 4

the "-t" argument is the length of the simulation in simulated minutes, the "-1" argument is the lookahead in hops.

without these arguments, the default length is 10080, and the default lookahead is 1.

⑤ 最后, 打开第11个客户端:

cd ~/rvnsord

seteny NETUID 99

setenv NETUID < sync node portnum > tsy4-t 10080 这将启动让仿真开始的同步节点。

10.3 准备运行

10. 3. 1 NETWORK. OUT

这是一个描述网络拓扑结构的命令文件。提供的例子是一个具有 10 个节点的网络,命名为"network. 10"。如前所述,这个文件必须被复制或重命名为"network. out"。该文件的格式是

comments begin with #, everything following the # on that line is ignored.

two types of entries, node lines and link lines

node lines begin with "n" and have the following syntax

n < uid > < name > < unused > < hostname > < portnum >

<uid > is the unique identifier for this node, this is an integer and as it implies, must be unique throughout the network

< name > is a string with the name of the node---this is a read by scanf so the standard rules of no spaces, etc, holds true.

< unused > are unused, safest to set to "0"

< hostname > is the name of the host upon which this node will run.

The same value returned by "uname-n"

< portnum > is the portnumber upon which this node will listen for incoming connections. There must be a unique number for each individual node running on a given host. It also cannot conflict with any other running services. Use netstat-a|grep < portnum > to check for conflicts--- the IANA reports that the following software use the range of 5300--5320 which is used in the provided examples.

链接行以"1"开始, 语法如下:

 $1 < uid_A > | < uid_B > < owner > < unused > < distance >$

<uid_A > and < uid_B > are the unique identifiers of the endpoints

< owner > is the name of the endpoint that "owns" this link for reservation purposes

< unused > unused, safest to set to "0"

< distance > is the distance of this link in miles

注意每个网络都有同步节点,如 uid 99,它通过链路与其余的每个节点连接。 我们为这些链路设定一个虚拟的无限大距离,以防止任何路线真地使用到它。还要 注意,节点的控制者与这些链接不相关。

10.3.2 帮助文件

- 1) script. 10: 是一个非常简单的 csh 脚本, 用来打开 10 个客户端仿真程序, 同时, 为方便运行仿真, 对环境变量进行设置。script. 10 根据节点的 uid、名称和端口号进行设置, 使之对应于 network. 10。客户端的几何特性通过一个虚拟桌面设置, 推荐分辨率为 1024 × 768 的 X- windows (注意, x 偏移是从 1040 开始的), 这已在 fvwm2 @ 1024 × 768 中使用, 当然, 对几何参数做简单的修改是必要的。
- 2) Script_ night. 10: 是 script. 10 的一个变种,可用于以批处理模式运行仿真系统,对参数研究非常有用。

10.3.3 输入生成

输入文件存放在/rynsord/input,格式如下:

< id > < orig > < dest > < speed > < time >

where id is the unique train identifier,

orig is the origin stations uid,

dest is the destination stations uid,

speed is the train speed in mph,

time is the time at which the train is first scheduled at the origin.

程序"in_gen"是自动生成这些输入文件的途径之一。要运行该程序,只需要在/rynsord 目录下键入"in_gen",它将读取"network.out"文件,获得拓扑信息,然后提示用户输入一些参数,包括

Orig vol: The traffic volume (see chapter 4 for more information)

Max time: The time at which trains will cease to be asserted.

Enter spike time...: This allows the user to force a perturbation period in the form of higher than normal traffic. The three numbers required are time to begin, the duration, and the additional volume. If no spike is desired, answer "0 0 0".

得到这些信息后, in_gen 开始运行, 并生成输入文件。

10.3.4 输出文件

每个车站在/rynsord/output 上产生输出文件, 命名为 t_output_ < uid >。

这些文件的格式沿用了以第一个字母定义输出行类型的传统:

t:train information line

 $t: [\ < timestamp > \] \ id < train \ id > from < origin > time (\ < orig \ time > \ , < time \ left \ origin >)$

值得注意的是,每列火车的最终目的地是那个存储了火车信息的节点。对于这 列火车,会显示下列几行信息:

a < number of route computation made > < number of messages sent >

p[< A > - > < B > (< time arrive B > , < time leave B >)

this is the hop by hop path that the train took

w < time spent waiting >

graphs! xgraph

在每个输出文件末尾都有以下内容:

 $1[\ < A > - > < B > \]$ usage < timesteps > of < max time > per < percentage > trains < num trains >

this is a summary of the link usage for those links owned by this node

对于这条链路,下列信息也适用:

r < reservations attempted > < reservations made > < reservations removed > < diff > < traveled >

where < diff > is the total difference between the reserved time and the departure time and < traveled > is the number of times a train traversed this track segment.

This is a summary of simulated network usage.

< a > total number of integers sent

< b > maximum number of integers sent in any one timestep

< c > time at which < b > occurred

< d > minimum number of messages sent

<e>time at which <d>occurred

< f > number of integers sent for waiting messages

最后,在文件的末尾是10个统计数值:

s1: reservations originated from this node

s2: reservations accepted at this node

s3: reservations denied at this node

s4: reservations processed at this node

s5: number of reservations which are 'ideally' met

s6: number of times the primary path was used

s7: number of times the secondary path was used

s8: number of times a train was allowed to leave early

s9: total number of minutes saved by this

s10: number of trains processed by this node

程序"out_ an"是用来分析这些输出文件的。它会创建一个以". dat"为扩展名的文本文件,它会同时打开"xgraph"工具,该工具具有一定的绘图功能。

10.3.5 故障处理

10.3.5.1 如何知道仿真系统正在工作

仿真启动时包括两个阶段。首先是要完成网络的初始化。当一个节点的输出如下所示时,它就已经完成了这个阶段的工作:

fd 4 $\lceil 0 < - > 1 \rceil$ status 0

fd 5 [0 < -> 4] status 0

fd 6 $\lceil 0 < -> 99 \rceil$ status 0

它会暂停在这里、直到所有其他节点也完成网络初始化为止。

然后开始第二阶段,就是实况仿真。此时唯一可见的输出是在同步节点,随着仿真时间的推移,系统会不断提示"TIME UPDATE"信息。其他节点在处理火车和消息时,会定期给火车发送有关消息。

10.3.5.2 问题

如果不是所有网络节点都正确地完成了初始化工作,那么 network. out 文件和使用的环境变量都可能发生错误。要根据 network. out 文件的内容详细检查每一个客户端的仿真程序,还要核实在同一部机器里,相同的端口号是否被使用了两次。即使一切看起来都正常,还是可能存在争夺同一个端口的情况。可以用命令"netstat-a"(与 grep 一起使用)检查你选择的端口号上的冲突情况。

如果关闭 (Control-C 或 通过 "kill" 命令) 一个仿真 (重启前你必须关闭所有节点),再次使用相同的端口号前,可能会有一个时间延迟。"netstat-a" 命令会显示 TIME_WAIT 状态。在一段很短的时间后,就可以重新使用这些端口了。

如果运行一段时间后, 仿真看起来停滞不动, 一个可能的问题就是网络图并不 是完全连通的。

10.3.6 轨道和通信故障

我们提供的仿真器能够模拟单一轨道或通信链路的故障。要启动这个过程,只需将下面的命令行参数添加到所有节点中(同步节点除外):

-k < A > < B > < time to initiate > < duration >

to fail a track segment between uid A and B.

-f < A > < B > < time to initiate > < duration > to fail a communication link between uid A and B. 要注意,不能同时使用两个标记。

10.4 结论

本章提供的信息,连同附带的软件,足以使读者能使用RYNSORD 算法进行一些简单的试验。应该注意,这是一个研究软件,可能不是完全鲁棒的,这个软件绝对不是广告性的软件。如果你遇到问题或有什么意见,可以随时发送邮件到 sumit. ghosh@ ieee. org 与我们联系,我们将会尽力及时回复。

参考文献

- 1. S. Yamanouchi. Essential information systems for railways and intensive application of ADS technology—COSMOS and ATOS. In *Keynote Address: International Symposium on Autonomous Decentralized Systems*, pp. 2–9, Tokyo, October 1999.
- 2. W.A. Carley. Railroads test satellite positioning in effort to improve safety, efficiency. *The Wall Street Journal Interactive Edition*, June 29, 1998.
- 3. T. Lomax. Five most congested urban areas: LA, DC, SFO-Oakland, Miami, Chicago. Texas Transportation Institute, June 24, 2002. Vol. 34, No. 1, http://mobility.tamu.edu.
- 4. History Channel. The Crumbling of America. June 29, 2009.
- 5. S. Ghosh and T. Lee. *Principles of Modeling and Asynchronous Distributed Simulation of Complex Systems*. IEEE Press: Picataway, NJ, 2000.
- P. Heck and S. Ghosh. A study of synthetic creativity through behavior modeling and simulation of an ant colony. *IEEE Intelligent Systems*, 15(6):58–66, November/December 2000.
- 7. P. Heck and S. Ghosh. The design and role of synthetic creative traits in artificial ant colonies. *Journal of Intelligent and Robotic Systems*, 33(4):343–370, April 2002.
- 8. S. Ghosh. Creative thinking. *IEEE Circuits and Devices*, 20(1):52–62, January/February 2004.
- 9. D.T. Emerson. The work of Jagadis Chandra Bose: 100 years of millimeter-wave research. *IEEE Transactions on Microwave Theory and Techniques*, 45(12):2267–2273, December 1997.
- 10. D.E. Barrett. Traffic-noice impact study for least Bell's vireo habitat along California State Route 83. *Transportation Research Record*, 1559:3–7, 1996.
- 11. D.E. Barrett. California study addresses issues of roadway noise control for the benefit of endangered songbirds. *Road Management and Engineering Journal, TranSafety,* http://www.usroads.com/journals, pp. 1–6, October 1997.
- 12. The Learning Channel. Kansai International Airport. November 26, 2000, http://www.tlc.com.
- 13. Xcel Energy Colorado System. *PHEV Impacts on the Xcel Energy Colorado System: Executive Summary*. Colorado, December 31, 2008.
- 14. T. Markel and K. Bennion. Xcel PHEV/V2G Project. In National Renewable Energy Laboratory, October 24, 2008.
- W.L. Grossman. History of transportation. In *Colliers Encyclopedia*, CD-ROM. February 28, 1996.
- S. Simon (Interviewer). Big dig—Most expensive road in U.S. history [transcript]. National Public Radio, Weekend Edition, June 22, 1996.
- 17. B. Blide. Personal rapid transit study in Gothenburg, Sweden. *Transportation Research Record*, 1451:87–94, December 1994.
- 18. I. Andreasson. Vehicle distribution in large personal rapid transit systems. *Transportation Research Record*, 1451:95–99, December 1994.
- 19. E.J. Anderson. Dependability as a measure of on-time performance of personal rapid transit systems. *Journal of Advanced Transportation*, 26(3):201–212, Winter 1992.
- Raytheon Electronic Systems. Personalized Rapid Transport, PRT 2000. Technical report, Marlborough, MA, October 1996.

- 21. D.C. Coll, A.U. Sheikh, R.G. Ayers, and J.H. Bailey. The communications system architecture of the North American Advanced Train Control System. *IEEE Transactions on Vehicular Technology*, 39(3):244–255, August 1990.
- 22. N. Utamaphethai and S. Ghosh. DICAF, A high performance, distributed, scalable architecture for IVHS utilizing a continuous function—"congestion measure". *IEEE Computer*, 31(3):78–84, March 1998.
- 23. Federal Aviation Authority. http://www.faa.gov/freeflight/.
- 24. Intelligent Transportation Society of America and US Department of Transportation. *ITS Architecture Development Program Phase I.* ITS America: Washington, DC, 1994.
- 25. S. Ghosh and T. Lee. *Intelligent Transportation Systems: New Principles and Architectures.* CRC Press: Boca Raton, FL, January 2000, Reprinted June 2002.
- 26. P. Varaiya and S. Shladover. Sketch of an IVHS architecture. *PATH Research Report*, UCB-ITS-PRR-91-03, February 1991.
- S.M. Graff and P. Shenkin. A computer simulation of a multiple track rail network. In Sixth International Conference on Mathematical Modeling, St. Louis, MO, August 4–7, 1987.
- 28. K. Fukumori, H. Sano, T. Hasegawa, and T. Sakai. Fundamental algorithm for train scheduling based on artificial intelligence. *Systems and Computers in Japan*, 18(3): 52–63, 1987.
- V.P. Rao and P.A. Venkatachalam. Microprocessor-based railway interlocking control with low accident probability. *IEEE Transactions on Vehicular Technology*, 35(3): 141–147, August 1987.
- 30. S.K. Kashyap. Telecommunications on Indian Railways. *IETE Technical Review (India)*, 5(3):117–119. March 1988.
- 31. G. Vernazza and R. Zunino. A distributed intelligence methodology for railway traffic control. *IEEE Transactions on Vehicular Technology*, 39(3):263–270, August 1990.
- 32. J.T. Rayfield and H.F. Silverman. Operating system and applications of the Armstrong multiprocessor. *IEEE Computer*, 21(6):38–52, 1988.
- 33. Interstate Commerce Commission. Ninety-ninth annual report transport statistics in the United States, December 1987.
- P.H. Bernard. ASTREE: A global command, control, and communication system. *Transportation Research Record*, Transportation Research Board: Washington, DC 20418: 133–139, 1991.
- 35. R.J. Hill, S.L. Yu, and N.J. Dunn. Rail transit chopper traction interference modeling using the spice circuit simulation package. *IEEE Transactions on Vehicular Technology*, 38(4):237–246, 1989.
- 36. R.G. Ayers. Selection of a forward error correcting code for the data communication radio link of the advanced train control-system. *IEEE Transactions on Vehicular Technology*, 38(4):247–254, 1989.
- A.U.H. Sheikh, D.C. Coll, R.G. Ayers, and J.H. Bailey. ATCS—Advanced train controlsystem radio data link design considerations. *IEEE Transactions on Vehicular Technol*ogy, 39(3):256–262, 1990.
- 38. R.J. Hill. Optimal construction of synchronizable coding for railway track circuit datatransmission. *IEEE Transactions on Vehicular Technology*, 39(4):390–399, 1990.
- 39. Y.R. Shayan, L.N. Tho, and V.K. Bhargava. Design of Reed-Solomon (16,12) codec for North-American advanced train control-system. *IEEE Transactions on Vehicular Technology*, 39(4):400–409, 1990.

- 40. S. Ghosh and M-L. Yu. An asynchronous distributed approach for the simulation and verification of behavior-level models on parallel processors. *IEEE Transactions on Parallel and Distributed Systems*, 6(6):639–652, June 1995.
- 41. J. Misra. Distributed discrete-event simulation. *Computing Surveys*, 18(1):39–65, March 1986.
- 42. J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann: Palo Alto, CA, 1990.
- 43. W.R. Stevens. Unix Network Programming. Prentice Hall: New Jersey, 1993.
- 44. E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- 45. I. Catling and B. McQueen. Road transport informatics in Europe—Major programs and demonstrations. *IEEE Transactions on Vehicular Technology*, 40(1):132–140, February 1991.
- 46. H. Kawashima. Two major programs and demonstrations in Japan. *IEEE Transactions on Vehicular Technology*, 40(1):141–146, February 1991.
- 47. G. King. Driver performance in highway navigation tasks. *Transportation Research Record (TRB), National Research Council*, Washington, DC, 1093:1–11, 1986.
- 48. Rhode Island Department of Transportation. *State of Rhode Island Incident Management Plan.* State Office Building: Providence, RI, September 1992.
- 49. C. Levesque. Rhode to the future. *Traffic Technology International*, 38–42, June/July 1998.
- 50. J. Peters, M. McGurrin, D. Shank, and M. Cheslow. Estimate of transportation cost savings from using intelligent transportation system (ITS) infrastructure. *ITE Journal*, 67(11):42–47, November 1997.
- 51. D.A. Haver and P.J. Tarnoff. Future-directions for traffic management systems. *IEEE Transactions on Vehicular Technology*, 40(1):4–10, 1991.
- 52. R.E. Fenton and R.J. Mayhan. Automated highway studies at the Ohio State University—An overview. *IEEE Transactions on Vehicular Technology*, 40(1):100–113, 1991.
- W.B. Powell. Optimization models and algorithms—An emerging technology for the motor carrier industry. *IEEE Transactions on Vehicular Technology*, 40(1):68–80, 1991.
- 54. T.M. Batz. The utilization of real-time traffic information by the trucking industry. *IEEE Transactions on Vehicular Technology*, 40(1):64–67, 1991.
- 55. D.H. Roper and G. Endo. Advanced traffic management in California. *IEEE Transactions on Vehicular Technology*, 40(1):152–158, 1991.
- 56. W. Kremer, D. Hubner, S. Hoff, T.P. Benz, and W. Schafer. Computer-aided design and evaluation of mobile radio local area networks in RTI/IVHS environments. *IEEE Journal of Selected Areas in Communications*, 11(3):406–421, April 1993.
- 57. S. Sakagami, A. Aoyama, K. Kuboi, S. Shirota, and A. Akeyama. Vehicle position estimates by multibeam antennas in multipath environments. *IEEE Transactions on Vehicular Technology*, 41(1):63–67, February 1992.
- 58. T.M. Hussain, T.N. Saadawi, and S.A. Ahmed. Overhead infrared sensor for monitoring vehicular traffic. *IEEE Transactions on Vehicular Technology*, 42(4):477–483, November 1993.
- J.L. Kim, J-C.S. Liu, P.I. Swarnam, and T. Urbanik. The areawide real-time traffic control (ARTC) system: A new traffic control concept. *IEEE Transactions on Vehicular Technology*, 42(2):212–224, May 1993.
- 60. R. Von Tomkewitsch. Dynamic route guidance and interactive transport management with ALI-SCOUT. *IEEE Transactions on Vehicular Technology*, 40(1):45–50, 1991.

- 61. R.W. Denney and M.J. Chase. True distributed processing in modular traffic signal systems—San Antonio Downtown System. *Transportation Research Record*, 1324: 130–136, 1991.
- 62. D.W. Kline and P. Fuchs. The visibility of symbolic highway signs can be increased among drivers of all ages. *Human Factors*, 35(1):25–34, 1993.
- D.I. Robertson and R.D. Bretherton. Optimizing networks of traffic signals in realtime—The SCOOT method. *IEEE Transactions on Vehicular Technology*, 40(1):11–15, 1991.
- 64. W.C. Collier and R.J. Weiland. Smart cars, smart highways. *IEEE Spectrum*, pp. 27–33, April 1994.
- S.E. Shladover, C.A. Desoer, J.K. Hedrick, M. Tomizuka, J. Walrand, W.B. Zhang, D.H. McMahon, P. Huei, S. Sheikholeslam, and N. Mckeown. Automatic vehicle control developments in the PATH program. *IEEE Transactions on Vehicular Technology*, 40(1):114–130, 1991.
- 66. W.H. von Aulock. Smart cars may not be smart. IEEE Spectrum, pp. 17–18, March 1994.
- 67. H.D. Goldstein. *Field Test Report*. NYNEX Assurance Services: Elmsford, NY, May 1994.
- 68. H.J.M. VanGrol and A.F. Bakker. Special-purpose parallel computer for traffic simulation. *Transportation Research Record*, 1306:40–48, 1991.
- D.J. Dailey, M.P. Haselkorn, and D. Meyers. Structured approach to developing realtime, distributed network applications for ITS deployment. *ITS Journal*, 3(3):163–180, 1996.
- 70. N. Jing, Y-W. Huang, and E.A. Rundensteiner. Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 10(3):409–432, May–June 1998.
- 71. Z.A. Talib, N.S. Love, J.C. Gealow, G. Hall, L. Masaki, and C.G. Sodini. Massively parallel image processing system for intelligent transportation system application. In *Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems*, pp. 367–372, November 9–7, 1997.
- A. Ziliaskopoulos, D. Kotzinos, and H.S. Mahmassani. Design and implementation of parallel time-dependent least time path algorithms for Intelligent Transportation Systems applications. *Transportation Research Part C: Emerging Technologies*, 5(2):95–107, April 1997.
- N. Rouphail, S.R. Ranjithan, W. El-Dessouki, T. Smith, and D.E. Brill. Decision support system for dynamic pre-trip route planning. In *Proceedings of the 1995 4th ASCE International Conference on Applications of Advanced Technologies in Transportation Engineering*, pp. 325–329, Capri, Italy, June 27–30, 1995.
- 74. The Minnesota Department of Transportation. The Minnesota Department of Transportation's Traffic Management Center (TMC). Minneapolis, MN, January 1994.
- 75. J. Upchurch, D. Powell, and P. Pretorius. Model deployment of Intelligent Transportation Systems: The AZTech experience. In *Proceedings of the 1998 5th ASCE International Conference on Applications of Advanced Technologies in Transportation*, pp. 367–372, Newport Beach, CA, April 26–28, 1998.
- G.L. Chang, T. Junchaya, and A.J. Santiago. A real-time network traffic simulation model for ATMS applications: Part II—Massively parallel model. *IVHS Journal*, 1(3):243–259, 1994.
- 77. T. Junchaya and G.L. Chang. Exploring real-time traffic simulation with massively parallel computing architecture. *Transportation Research, Part C: Emerging Technologies*, 1(1):57–76, March 1993.

- 78. R.W. Hall. Route choice and advanced traveler information systems on a capacitated and dynamic network. *Transportation Research Part C: Emerging Technologies*, 4(5): 289–306, 1996.
- 79. Washington State Department of Transportation. *Seattle Area Traffic Conditions*. http://www.wsdot.wa.gov/regions/northwest/NWFLOW/ (accessed 1998).
- 80. J. Clark and G. Daigle. Importance of simulation techniques in its research and analysis. In *Proceedings of the 1997 Winter Simulation Conference*, pp. 1236–1243, Atlanta, GA, Dec 7–10, 1997.
- 81. L. Bodin, B. Golden, A. Assad, and M. Ball. Vehicle routing of vehicles and crews: The state of the art. *Computers and Operations Research*, 10(2): 1983.
- 82. R.W. Allen, D. Ziedman, T.J. Rosenthal, A.C. Stein, J.F. Torres, and A. Halati. Laboratory assessment of driver route diversion in response to in-vehicle navigation and motorist information systems. *Transportation Research Record*, 1306:82–91, 1991.
- 83. P. Davies and G. Klein. Field trials and evaluations of radio data system traffic message channel. *Transportation Research Record*, 1324:1–7, 1991.
- 84. B. Kamali. Wireless communications: The nervous system of intelligent transportation systems. *Applied Microwave and Wireless*, 9(5):6–6, September–October 1997.
- H. Sodeikat. Universal system architecture. In *Proceedings of the Pacific Rim Transtech Conference*, pp. 251–257, Washington State Conventional Center, July 25–28, 1993.
- E. Kreyszig. Advanced Engineering Mathematics. John Wiley and Sons: New York, 1967.
- 87. A. Bonde and S. Ghosh. A comparative study of fuzzy versus "fixed" thresholds for a robust queue management in cell-switching networks. *IEEE/ACM Transactions on Networking*, 2(4):337–344, August 1994.
- 88. D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall: Upper Saddle River, NJ, 1992.
- 89. Rhode Island Department of Transportation. *State Highway Map of Rhode Island—1993 Annual 24 Hour Average Daily Traffic.* State Office Building: Providence, RI, 1994.
- T. Lee, S. Ghosh, J. Lu, X. Ge, A. Nerode, and W. Kohn. A mathematical framework for asynchronous, decentralized, decision-making algorithm with semi-autonomous entities: Synthesis, simulation, and evaluation. *IEICE Transactions on Fundamentals*, E-83A(7): 1381–1395, July 2000.
- 91. A.M. Lyapunov. *The General Problem of the Stability of Motion*. Taylor & Francis: London, 1992.
- 92. C-T. Chen. Analog and Digital Control System Design: Transfer-Function, State-Space, and Algebraic Methods. Saunders College: New York, 1993.
- 93. T. Lee and S. Ghosh. RYNSORD: A novel, decentralized algorithm for railway networks with "soft reservation." In *Proceedings of the 48th IEEE Annual Vehicular Technology Conference (VTC'98)*, May 18–21 1998.
- 94. T. Lee and S. Ghosh. RYNSORD: A novel, decentralized algorithm for railway networks with "soft reservation." *IEEE Transactions on Vehicular Technology*, 47(4): 201–222, November 1998.
- 95. A.M. Letov. *Stability in Nonlinear Control Systems*. Princeton University Press: Princeton, 1961.
- T.L. Casavant and John G. Kuhl. Effects of response and stability on scheduling in distributed computing systems. *IEEE Transaction on Software Engineering*, 14(11):1578–1588, November 1988.

- 97. E.W. Dijkstra. Self-stabilization in spite of distributed control. In *Texts and Monographs in Computer Science*. pp. 41–46: Springer-Verlag: New York, 1982.
- 98. E.W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, November 1974.
- 99. B. Awerbuch and G. Varghese. Distributed program checking: A paradigm for building self-stabilizing distributed protocols. In *Proceedings of the 32nd IEEE Annual Symposium on Foundations of Computer Science*, pp. 258–267, Los Alamitos, CA, 1991.
- 100. K. Mani Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, February 1985.
- S. Venkatesan and B. Dathan. Testing and debugging distributed programs using global predicates. *IEEE Transaction on Software Engineering*, 21(2):163–177, February 1995.
- 102. V.K. Garg and B. Waldecker. Detection of strong unstable predicates in distributed programs. Department of Electrical and Computer Engineering, University of Texas, Austin.
- F.A. Schreiber. Notes on real-time distributed database systems stability. In *Proceedings* of the 5th Jerusalem Conference on Information Technology, pp. 560–564, Los Alamitos, CA, 1990.
- 104. J.A. Stankovic. Stability and distributed scheduling algorithms. *IEEE Transaction on Software Engineering*, SE-11(10):1141–1152, October 1985.
- 105. J.F. Meyer. Performability modeling of distributed real-time systems. In G. Iazeolla, P.J. Courtois, and A. Hordijk, ed., *Mathematical Computer Performance and Reliability*, pp. 361–369. Elsevier Science Publishers B.V.: North-Holland, 1984.
- P. Garg, S. Frolund, A. Shepherd, and N. Venkatasubramanian. Towards distributed applications stability engineering. In *Proceedings of the 5th California Software Symposium*, pp. 528–529, Irvine, CA, March 1995.
- D. Ferrari. Computer Systems Performance Evaluation. Prentice-Hall: Englewood Cliffs, NJ, 1978.
- Free Software Foundation, Inc. Linux Operating System. Technical report, Cambridge, MA, 1995.

本书特色

原书第2版中的新内容:

- 涵盖了全新的、高效的绿 色环保发展策略。
- 全面回顾和总结了ITS重要 的发展以及实践成果。
- 讨论了如何使用全新的视 野和元层原理解决现实世界中的 ITS问题。

原书第2版全面检视了近10年中ITS的成功和失败之处。原书第2版新的副标题反应出作者更侧重于对ITS取得进展有决定意义的元层原理的研究,并成功地利用智能/绿色技术构建了ITS的体系结构。这本书指出了一些必须要解决的具有挑战性的问题,只有这些问题得以解决,才能带来实际生活质量上的改善,并对我们的环境和土木基建系统产生积极的影响。

国际视野 科技前沿

国际电气工程先进技术译丛 传播国际最新技术成果 搭建电气工程技术平台

- 《智能运输系统:智能化绿色结构设计》(原书第2版)
- 《纯电动及混合动力汽车设计基础》(原书第2版)
- 《超高压交流地下电力系统的性能和规划》
- 《可再生能源系统——100%可再生能源解决方案的选择与模型》
- 《机电系统中的传感器与驱动器——设计与应用》
- 《碳捕获与封存》
- 《LED照明应用技术》
- 《零排放动力循环》
- 《柔性交流输电系统在电网中的建模与仿真》
- 《风电并网:联网与系统运行》
- 《可再生能源的转换、传输和储存》
- 《海底电力电缆——设计、安装、修复和环境影响》
- 《光伏技术与工程手册》
- 《风力发电的模拟与控制》
- 《风电场并网稳定性技术》
- 《智能电网中的电力电子技术》
- 《电接触理论、应用与技术》
- 《电磁屏蔽原理与应用》
- 《高效可再生分布式发电系统》
- 《电网保护》
- 《分布式发电——感应和永磁发电机》
- 《电力系统谐波》
- 《风能与太阳能发电系统——设计、分析与运行》(原书第2版)
- 《瞬时功率理论及其在电力调节中的应用》
- 《风力机控制系统原理、建模及增益调度设计》
- 《高压输配电设备实用手册》
- 《电力变流器电路》
- 《电力系统中的电磁兼容》
- 《超高压交流输电工程》(原书第3版)
- 《高压直流输电与柔性交流输电控制装置——静止换流器在电力系统中的应用》
- 《电磁兼容原理与应用》(原书第2版)
- 《电力电子技术手册》
- 《基于晶闸管的柔性交流输电控制装置》
- 《电力电容器》
- 《电力系统谐波——基本原理、分析方法和滤波器设计附习题解答》
- 《配电可靠性与电能质量》
- 《现代电动汽车、混合动力电动汽车和燃料电池车——基本原理、理论和设计》(原书第2版)



上架指导: 工业技术/交通运输

CRC Press
Taylor & Francis Group

地址:北京市百万庄大街225 电话服务

杜服务中心: (010)88361066 销售一部: (010)68326294 销售二部: (010)88379649 郵政编码: 100037 网络服务

[]户网: http://www.cmpbcok.com 教材网: http://www.cmpedu.com 封面无防伪标均为盗版 ISBN 978-7-111-37676-7

● ISBN 978-7-89433-443-5(光盘)

● 封面设计:马精明

定价: 58.00元

