

# 蚁群算法原理及其应用

Ant Colony Algorithms: Theory and Applications

段海滨 著

科学出版社

北京

## 内 容 简 介

本书系统、深入地介绍了蚁群算法的原理及其应用，力图概括国内外在这一学术领域的最新研究进展。全书共包括 10 章，主要内容包括蚁群算法的思想起源、研究现状及机制原理；蚁群算法的复杂度分析；蚁群算法的收敛性证明；蚁群算法参数对其性能的影响；蚁群算法的参数选择原则；离散域和连续域蚁群算法的若干改进策略；蚁群算法在多个优化领域的典型应用；蚁群算法的硬件实现技术；蚁群算法与其他仿生优化算法的比较与融合；蚁群算法的研究展望；最后还在附录部分给出了基本蚁群算法的程序源代码和相关网站。

本书内容取材新颖，覆盖面较广，深入浅出，系统性强，注重理论联系实际，力求使读者能较快掌握和应用这一新兴的仿生优化算法。

本书可作为计算机科学、控制科学、人工智能、管理科学等专业高年级本科生、研究生和教师的参考书，也可供理工科其他专业的师生参考，还可供利用计算机从事智能优化的科技人员阅读和参考。

### 图书在版编目(CIP)数据

---

蚁群算法原理及其应用/段海滨著. —北京:科学出版社,2005.12

ISBN 7-03-016204-8

I. 蚁… II. 段… III. 智能控制—算法 IV. TP273

---

中国版本图书馆 CIP 数据核字(2005)第 098539 号

---

责任编辑:王淑兰 赵卫江/责任校对:柏连海

责任印制:吕春珉/封面设计:王 浩

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

<http://www.sciencep.com>

双青印刷厂 印刷

科学出版社发行 各地新华书店经销

\*

2005年12月第一版 开本: B5 (720×1000)

2005年12月第一次印刷 印张: 29

印数: 1—3 000 字数: 552 000

**定价: 48.00 元**

(如有印装质量问题, 我社负责调换<新欣>)

销售部电话 010-62136131 编辑部电话 010-62130750 (TB06)

# 序

仿生优化算法是人工智能研究领域中一个重要的分支，其中包括模拟生物界中自然选择和遗传机制的遗传算法、模拟蚂蚁群体觅食行为的蚁群算法以及模拟鸟类群体捕食行为的微粒群算法等。

蚁群算法最初由意大利学者 Dorigo M 于 1991 年首次提出，其本质上是一个复杂的智能系统，它具有较强的鲁棒性、优良的分布式计算机制、易于与其他方法结合等优点。如今这一新兴的仿生优化算法已经成为人工智能领域的一个研究热点。目前对其研究已渗透到多个应用领域，并由解决一维静态优化问题发展到解决多维动态组合优化问题。如今在国内外许多学术期刊和重要国际会议上，蚁群算法已成为交叉学科中一个非常活跃的前沿性研究问题。

段海滨博士多年来一直从事蚁群算法方面的研究工作，并在该领域有着丰厚的研究积累。该书包含了作者在蚁群算法理论及应用方面的研究成果，同时也吸纳了国内外许多有代表性的研究进展。

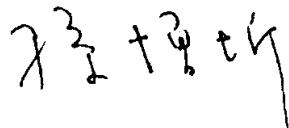
该书在系统研究蚁群算法的机制原理、理论分析及其在离散域和连续域的若干改进策略的基础上，阐述了蚁群算法在多个优化领域的典型应用，探讨了蚁群算法的硬件实现，研究了蚁群算法与其他仿生优化算法的融合策略，展望了蚁群算法的发展方向。

为了便于读者学习和研究，该书在附录部分给出了基本蚁群算法的程序源代码及相关的网站资源。因此，该书不仅具有较高的学术价值，而且对工程应用也具有较好的参考价值和指导意义。

该书取材新颖，覆盖面广，结构合理。内容阐述深入浅出，条理清晰，注重理论联系实际，具有前瞻性和创新性，较好地体现了在这一研究领域的最新进展。

该书可作为高等院校和科研院所的计算机科学、人工智能、控制科学、管理科学、系统工程、电力电子、机械工程和生命科学等专业的广大师生及科技工作者的学习参考书。

目前国内系统地介绍蚁群算法的专著还比较匮乏，该书的出版在一定程度上弥补了这个不足。相信它的出版将对蚁群算法的发展和应用起到积极的推动作用。



2005 年 10 月

## 前　　言

蚁群算法是一种最新发展的模拟昆虫王国中蚂蚁群体觅食行为的仿生优化算法，该算法采用了正反馈并行自催化机制，具有较强的鲁棒性、优良的分布式计算机制、易于与其他方法结合等优点，在解决许多复杂优化问题方面已经展现出其优异的性能和巨大的发展潜力，近几年吸引了国内外许多学者对其进行了多方面的研究工作。国际著名的顶级学术刊物《Nature》曾多次对蚁群算法的研究成果进行报道，《Future Generation Computer Systems》和《IEEE Transactions on Evolutionary Computation》分别于2000年和2002年出版了蚁群算法特刊，在布鲁塞尔每两年召开一次的蚁群算法国际研讨会进一步促进了这一智能计算领域的学术交流，从而使这种新兴的仿生优化算法展现出勃勃生机，其应用范围完全可与遗传算法相媲美。目前，蚁群算法已成为国际智能计算领域中备受关注的研究热点和前沿性课题。

本书是作者多年来在对蚁群算法理论及其应用所进行的一系列深入研究的基础上撰写而成的，同时吸纳了国内外许多具有代表性的最新研究成果。全书内容取材新颖，覆盖面较广，深入浅出，注重理论联系实际，力图体现国内外在这一学术领域的最新研究进展。本书可作为计算机科学、控制科学、管理科学等专业高年级本科生、研究生和教师的参考书，也可供理工科其他专业的师生参考，还可供利用计算机从事智能优化的科技工作者阅读和参考。

全书共包括10章，第1章是引论，主要介绍了蚁群算法的思想起源和研究现状；第2章阐述了基本蚁群算法的机制原理、系统学特征、数学模型及具体实现，并对其复杂度作了深入分析；第3章对蚁群算法的收敛性问题进行了深入研究和理论证明；第4章通过大量的实验数据分析了不同蚁群算法模型中诸多关联参数对其性能的影响，并给出了其主要参数的选择原则；第5章和第6章分别对离散域和连续域蚁群算法的若干改进策略进行了研究；第7章阐述了蚁群算法在多个优化领域的典型应用；第8章介绍了蚁群算法的硬件实现技术；第9章将蚁群算法同目前比较流行的几种仿生优化算法做了比较，并给出了蚁群算法与这些仿生优化算法的融合策略；第10章展望了蚁群算法的研究方向和发展前景；最后，还在附录部分给出了基本蚁群算法的程序源代码和相关网站资源，并附上了书中出现的中英文词汇对照及缩略语。全书的10章内容（连同附录）基本构成了一个完整的封闭体系。

值此，作者非常感谢王道波教授、高镇洋教授、陈宗基教授、张平教授、周锐教授、郭锁凤教授、温卫东教授、孙久厚教授、夏品奇教授以及朱家强博士

后、罗德林博士在本书撰写过程中所给予的支持和帮助；特别感谢中国人工智能学会副理事长、清华大学智能技术与系统国家重点实验室孙增圻教授在百忙之中认真审阅了书稿，提出了许多宝贵的修改建议，并欣然为本书作序。没有他们的惠教、支持与鼓励，就不可能有本书稿的诞生。另外，在写作过程中参考了大量的最新文献，这里也向这些文献的作者们致以诚挚的谢意！衷心感谢培养我的母校——南京航空航天大学，特别感谢南京航空航天大学研究生院对本书出版的全额资助，也非常感谢北京航空航天大学自动化学院领导和老师们对我的支持和帮助，同时，也感谢科学出版社王淑兰等编辑在本书出版过程中所付出的辛勤劳动！

目前国内系统研究蚁群算法的书籍和资料还十分匮乏，我非常希望能献给大家一本既有理论又重视实践的好书，尽管罄尽全力，但囿于水平，且时间紧促，书中错误和不妥之处在所难免。凡此，恳请各位专家、学者及广大读者不吝指正。

段海滨

2005年8月

# ABSTRACT

In ant societies, the activities of the individuals, as well as of the society as a whole, are not regulated by any explicit form of centralized control. On the other hand, adaptive and robust behaviors transcending the behavioral repertoire of the single individual can be easily observed at society level. These complex global behaviors are the result of self-organizing dynamics driven by local interactions and communications among a number of relatively simple individuals. The simultaneous presence of these and other fascinating and unique characteristics have made ant societies an attractive and inspiring model for building new algorithms and new multi-agent systems. In the last decade, ant societies have been taken as a reference for an ever growing body of scientific work.

Among the different works inspired by ant colonies, the ant colony algorithm (ACA) is probably the most successful and popular one. ACA is a novel bio-inspired optimization algorithm, which simulates the foraging behavior of ants for solving various complex combinatorial optimization problems. This book clearly defines ACA and its complexities, and presents both the most significant theoretical achievements and the state-of-the-art of ACA applications, especially the hardware realization of ACA. This book is broadly divided into 10 chapters and is organized as follows.

**Chapter 1** starts with a description of the biological characteristics ants. On the basis of the introduction of the idea origins of ACA, the development of ACA is illustrated.

**Chapter 2** presents a well-structured definition of basic ACA, and detailed implementation process and complexity analyses of basic ACA are also presented in this chapter.

**Chapter 3** is dedicated to discuss the in-depth convergence proofs for specific classes of ACAs.

**Chapter 4** presents detailed experimental analyses on the effect of pertinent parameters and ant colony behaviors in ACA, and an effective “three-step” method for optimum configuration of pertinent parameters in ACA is concluded in this chapter.

**Chapter 5** is devoted to the explanation of improvement strategies of ACA

in discrete space optimization, and this description takes advantage of the traveling salesman problem (TSP).

**Chapter 6** is devoted to the explanation of improvement strategies of ACA in continuous space optimization.

**Chapter 7** presents the state-of-the-art of ACA typical applications in various fields. The main application principles, that is, rules of thumb to be followed when attacking a new problem, are identified and discussed in this chapter.

**Chapter 8** reports on what is currently known about the hardware realization of ACA.

**Chapter 9** presents a systematic comparison and detailed combination of ACA and other bio-inspired optimization algorithms.

**Chapter 10** outlines some ongoing and most promising research trends in ACA.

Finally, there are four appendixes, which are programs of basic ACA, web sources about ACA, terminology (Chinese-English) and a piece of poetry extolling ACA.

This book is intended primarily for (1) advanced undergraduate and graduate students in computer science, cybernetics, management science and other related majors; (2) academic and industry researchers in artificial intelligence and computational intelligence; (3) practitioners willing to learn how to implement ACA to solve various combinational optimization problems.

# 目 录

<b>第 1 章 绪论</b> .....	1
1.1 引言 .....	1
1.2 蚂蚁的生物学特征 .....	2
1.3 蚁群算法的思想起源 .....	8
1.4 蚁群算法的研究进展.....	11
1.5 本书的体系结构.....	15
1.6 本章小结.....	18
参考文献 .....	18
<b>第 2 章 基本蚁群算法原理及其复杂度分析</b> .....	24
2.1 引言.....	24
2.2 基本蚁群算法的原理.....	24
2.3 基本蚁群算法的系统学特征.....	26
2.4 基本蚁群算法的数学模型.....	29
2.5 基本蚁群算法的具体实现.....	36
2.6 基本蚁群算法的复杂度分析.....	39
2.7 基本蚁群算法的性能评价指标.....	42
2.8 本章小结.....	42
参考文献 .....	43
<b>第 3 章 蚁群算法的收敛性研究</b> .....	45
3.1 引言.....	45
3.2 图搜索蚂蚁系统 (GBAS) 的收敛性研究 .....	45
3.3 一类改进蚁群算法的收敛性证明.....	59
3.4 GBAS/tdev 和 GBAS/tdlb 的确定性收敛证明 .....	65
3.5 基本蚁群算法的 A. S. 收敛性研究 .....	72
3.6 一类分布式蚂蚁路由算法的收敛性研究.....	76
3.7 基于分支路由和 Wiener 过程的蚁群算法收敛性证明 .....	81
3.8 一种简单蚁群算法及其收敛性分析.....	84
3.9 遗传-蚁群算法的 Markov 收敛性分析 .....	90
3.10 一类广义蚁群算法 (GACA) 的收敛性分析 .....	93
3.11 本章小结 .....	97
参考文献 .....	98

<b>第4章 蚁群算法的实验分析及参数选择原则</b>	100
4.1 引言	100
4.2 蚁群行为和参数对算法性能影响的实验分析	100
4.3 蚁群算法参数最优组合的“三步走”方法	116
4.4 本章小结	117
参考文献	117
<b>第5章 离散域蚁群算法的改进研究</b>	119
5.1 引言	119
5.2 自适应蚁群算法	119
5.3 基于去交叉局部优化策略的蚁群算法	125
5.4 基于信息素扩散的蚁群算法	130
5.5 多态蚁群算法	135
5.6 基于模式学习的小窗口蚁群算法	139
5.7 基于混合行为的蚁群算法	144
5.8 带聚类处理的蚁群算法	148
5.9 基于云模型理论的蚁群算法	153
5.10 具有感觉和知觉特征的蚁群算法	157
5.11 具有随机扰动特性的蚁群算法	166
5.12 基于信息熵的改进蚁群算法	169
5.13 本章小结	172
参考文献	172
<b>第6章 连续域蚁群算法的改进研究</b>	175
6.1 引言	175
6.2 基于网格划分策略的连续域蚁群算法	176
6.3 基于信息量分布函数的连续域蚁群算法	179
6.4 连续域优化问题的自适应蚁群算法	183
6.5 基于交叉变异操作的连续域蚁群算法	187
6.6 嵌入确定性搜索的连续域蚁群算法	190
6.7 基于密集非递阶的连续交互式蚁群算法(CIACA)	194
6.8 多目标优化问题的连续域蚁群算法	201
6.9 复杂多阶段连续决策问题的动态窗口蚁群算法	205
6.10 本章小结	209
参考文献	209
<b>第7章 蚁群算法的典型应用</b>	212
7.1 引言	212
7.2 车间作业调度问题	212

7.3 网络路由问题 .....	225
7.4 车辆路径问题 .....	238
7.5 机器人领域 .....	249
7.6 电力系统 .....	258
7.7 故障诊断 .....	268
7.8 控制参数优化 .....	272
7.9 系统辨识 .....	282
7.10 聚类分析 .....	290
7.11 数据挖掘 .....	297
7.12 图像处理 .....	302
7.13 航迹规划 .....	306
7.14 空战决策 .....	310
7.15 岩土工程 .....	315
7.16 化学工业 .....	319
7.17 生命科学 .....	323
7.18 布局优化 .....	327
7.19 本章小结 .....	331
参考文献 .....	332
<b>第8章 蚁群算法的硬件实现 .....</b>	<b>343</b>
8.1 引言 .....	343
8.2 仿生硬件概述 .....	343
8.3 基于 FPGA 的蚁群算法硬件实现 .....	346
8.4 基于蚁群算法和遗传算法动态融合的软硬件划分 .....	360
8.5 本章小结 .....	371
参考文献 .....	372
<b>第9章 蚁群算法同其他仿生优化算法的比较与融合 .....</b>	<b>374</b>
9.1 引言 .....	374
9.2 其他几种仿生优化算法的基本原理 .....	374
9.3 蚁群算法与其他仿生优化算法的异同比较 .....	382
9.4 蚁群算法与遗传算法的融合 .....	385
9.5 蚁群算法与人工神经网络的融合 .....	390
9.6 蚁群算法与微粒群算法的融合 .....	398
9.7 蚁群算法与人工免疫算法的融合 .....	402
9.8 本章小结 .....	410
参考文献 .....	410

<b>第 10 章 展望</b>	414
10.1 引言	414
10.2 蚁群算法的模型改进	414
10.3 蚁群算法的理论分析	415
10.4 蚁群算法的并行实现	416
10.5 蚁群算法的应用领域	417
10.6 蚁群算法的硬件实现	418
10.7 蚁群算法的智能融合	418
10.8 本章小结	419
参考文献	419
<b>附录 A 基本蚁群算法程序</b>	421
A.1 C 语言版	421
A.2 Matlab 语言版	426
A.3 Visual Basic 语言版	432
<b>附录 B 相关网站</b>	439
<b>附录 C 基本术语（中英文对照）及缩略语</b>	441
<b>附录 D （词一首）鹧鸪天·蚁群算法</b>	447

# CONTENTS

<b>Chapter 1 Introduction</b> .....	1
1. 1 Introduction .....	1
1. 2 Biological Characteristics of Ants .....	2
1. 3 Idea Origins of Ant Colony Algorithm(ACA) .....	8
1. 4 Development of ACA .....	11
1. 5 Outline of the Book .....	15
1. 6 Summary .....	18
References .....	18
<b>Chapter 2 Principles and Complexity Analysis of Basic ACA</b> .....	24
2. 1 Introduction .....	24
2. 2 Principles of Basic ACA .....	24
2. 3 Systematic Characteristics of Basic ACA .....	26
2. 4 Mathematical Model of Basic ACA .....	29
2. 5 Implementation of Basic ACA .....	36
2. 6 Complexity Analysis of Basic ACA .....	39
2. 7 Performance Criteria of Basic ACA .....	42
2. 8 Summary .....	42
References .....	43
<b>Chapter 3 Convergence Proofs for ACAs</b> .....	45
3. 1 Introduction .....	45
3. 2 Convergence Proof for the Graph-based Ant System (GBAS) .....	45
3. 3 Short Convergence Proof for a Class of Improved ACA .....	59
3. 4 Deterministic Convergence Proof for GBAS/tdev and GBAS/tdlb .....	65
3. 5 A. S. Convergence Proof for Basic ACA .....	72
3. 6 Convergence Results for a Class of Distributed Ant Routing Algorithms .....	76
3. 7 Branching Process and Branching Wiener Process based ACA .....	81
3. 8 Convergence Analysis for a simple ACA .....	84
3. 9 Markov Convergence Analysis for the Combination of ACA and Genetic Algorithm (GA) .....	90

3.10	Convergence Analysis for a Class of Generalized ACA (GACA) .....	93
3.11	Summary .....	97
	References .....	98
<b>Chapter 4 Experimental Analysis of ACA and Optimum Configuration</b>		
	<b>Principles of Pertinent Parameters in ACA .....</b>	100
4.1	Introduction .....	100
4.2	Experimental Analysis of the Effect of Pertinent Parameters and Ant Colony Behaviors in ACA .....	100
4.3	“Three-step” Method for Optimum Configuration of Pertinent Parameters in ACA .....	116
4.4	Summary .....	117
	References .....	117
<b>Chapter 5 Improved ACAs in Discrete Space Optimization .....</b>		119
5.1	Introduction .....	119
5.2	Self-adaptive ACA .....	119
5.3	Cross-removing and Local Optimization Strategies based ACA ...	125
5.4	Pheromone Diffusion based ACA .....	130
5.5	Polymorphic ACA .....	135
5.6	Model-learning based Little-window ACA .....	139
5.7	Hybrid Behavior based ACA .....	144
5.8	Improved ACA with Clustering .....	148
5.9	Cloud Models Theory based ACA .....	153
5.10	Sensational and Consciousness ACA .....	157
5.11	Improved ACA with Random Perturbation Behavior .....	166
5.12	Information Entropy based ACA .....	169
5.13	Summary .....	172
	References .....	172
<b>Chapter 6 Improved ACAs in Continuous Space Optimization .....</b>		175
6.1	Introduction .....	175
6.2	Gridding Partition based ACA .....	176
6.3	Trail Distributed Function based ACA .....	179
6.4	Self-adaptive ACA for Solving Continuous Space Optimization Problems .....	183
6.5	Cross and Mutation Operation based ACA .....	187
6.6	Improved ACA with Embedded Deterministic Searching .....	190

6.7	Dense Hierarchy Continuous Interacting ACA .....	194
6.8	Improved ACA for Multiobjective Optimization Problems .....	201
6.9	Dynamic-window-search ACA for Complex Multistage Decision-making Problems .....	205
6.10	Summary .....	209
	References .....	209
<b>Chapter 7</b>	<b>Typical Applications of ACA .....</b>	<b>212</b>
7.1	Introduction .....	212
7.2	Job-shop Scheduling Problem (JSP) .....	212
7.3	Network Routing Problem .....	225
7.4	Vehicle Routing Problem (VRP) .....	238
7.5	Robot Field .....	249
7.6	Power System .....	258
7.7	Fault Diagnosis .....	268
7.8	Parameter Optimization .....	272
7.9	System Identification .....	282
7.10	Clustering Analysis .....	290
7.11	Data Mining .....	297
7.12	Imagine Processing .....	302
7.13	Route Planning .....	306
7.14	Air-combat Decision-making .....	310
7.15	Geotechnical Engineering .....	315
7.16	Chemical Industry .....	319
7.17	Life Science .....	323
7.18	Layout Optimization .....	327
7.19	Summary .....	331
	References .....	332
<b>Chapter 8</b>	<b>Hardware Realization of ACA .....</b>	<b>343</b>
8.1	Introduction .....	343
8.2	Overview of Bio-inspired Hardware .....	343
8.3	FPGA Implementation of ACA .....	346
8.4	Hardware/Software Partitioning based on Dynamic Combination of ACA and GA .....	360
8.5	Summary .....	371
	References .....	372

<b>Chapter 9 Comparison and Combination of ACA and Other Bio-inspired Optimization Algorithms .....</b>	374
9. 1 Introduction .....	374
9. 2 Principles of Several Typical Bio-inspired Optimization Algorithms .....	374
9. 3 Comparison of ACA and Other Bio-inspired Optimization Algorithms .....	382
9. 4 Combination of ACA and GA .....	385
9. 5 Combination of ACA and Artificial Neural Network(ANN) .....	390
9. 6 Combination of ACA and Particle Swarm Optimization(PSO).....	398
9. 7 Combination of ACA and Artificial Immune Algorithm(AIA).....	402
9. 8 Summary .....	410
References .....	410
<b>Chapter 10 Prospects .....</b>	414
10. 1 Introduction .....	414
10. 2 Model Improvement of ACA .....	414
10. 3 Theoretical Analysis of ACA .....	415
10. 4 Parallelization of ACA .....	416
10. 5 Application Fields of ACA .....	417
10. 6 Hardware Realization of ACA .....	418
10. 7 Intelligent Combination of ACA .....	418
10. 8 Summary .....	419
References .....	419
<b>Appendix A Programs of Basic ACA .....</b>	421
A. 1 C Language Version .....	421
A. 2 Matlab Language Version .....	426
A. 3 Visual Basic Language Version .....	432
<b>Appendix B Web Sources about ACA .....</b>	439
<b>Appendix C Terminology (Chinese-English) and Abbreviations .....</b>	441
<b>Appendix D (Poetry) Zhe Hu Tian • ACA .....</b>	447

# 第1章 绪论

## 1.1 引言

自然界一直是人类创造力的丰富源泉，人类认识事物的能力来源于与自然界的相互作用之中。自然界中的许多自适应优化现象不断给人以启示：生物体和自然生态系统可通过自身的演化就使许多在人类看起来高度复杂的优化问题得到完美的解决。近些年来，一些与经典的数学规划原理截然不同的、试图通过模拟自然生态系统机制以求解复杂优化问题的仿生优化算法相继出现（如遗传算法、蚁群算法、微粒群算法、人工免疫算法、人工鱼群算法、混合蛙跳算法等），大大丰富了现代优化技术，也为那些传统最优化技术难以处理的组合优化问题提供了切实可行的解决方案。伴随着模拟自然与生物机理为特征的仿生优化算法时代的悄然兴起，一些仿生优化算法已在经典 NP-C 问题的求解和实际应用中显示出强大的生命力和进一步发展的潜力。

生物学家通过对蚂蚁的长期观察研究发现，每只蚂蚁的智能并不高，看起来没有集中的指挥，但它们却能协同工作，集中食物，建起坚固漂亮的蚁穴并抚养后代，依靠群体能力发挥出超出个体的智能。蚁群算法 (ant colony algorithm, ACA) 是最新发展的一种模拟昆虫王国中蚂蚁群体智能行为的仿生优化算法<sup>[1]</sup>，它具有较强的鲁棒性、优良的分布式计算机制、易于与其他方法相结合等优点<sup>[2, 3]</sup>。尽管蚁群算法的严格理论基础尚未奠定，国内外的相关研究还处于实验探索和初步应用阶段，但是目前人们对蚁群算法的研究已经由当初单一的旅行商问题 (traveling salesman problem, TSP) 领域渗透到了多个应用领域，由解决一维静态优化问题发展到解决多维动态组合优化问题，由离散域范围内的研究逐渐拓展到了连续域范围内的研究<sup>[4, 5]</sup>，而且在蚁群算法的硬件实现上也取得了很多突破性的研究进展，从而使这种新兴的仿生优化算法展现出勃勃生机和广阔的发展前景。

本章首先阐述了蚂蚁的社会形态，随后分析了蚂蚁群体“寻找食物”、“任务分配”及“构造墓地”三种社会行为，接下来在总结真实蚂蚁和人工蚂蚁之间异同点的基础上，阐述了蚁群算法的思想起源，紧接着评述了蚁群算法的发展现状，同时列举了部分具有代表性的蚁群算法及其应用情况，最后给出了本书的组织结构以及各章所包括的主要内容。

## 1.2 蚂蚁的生物学特征

蚂蚁又称“玄驹”、“蚍蜉”或“状元子”，是一种既渺小而又平常的社会性昆虫。在分类上，蚂蚁属于节肢动物门、昆虫纲、膜翅目、蚁科，它在昆虫界种类最多、生存量最大。蚂蚁的起源可追溯到一亿年前的恐龙时代，那时地球上就有蚂蚁的祖先繁衍。随着地球生态环境的变迁，身躯庞大的恐龙早已灭绝了，而身躯细小的蚂蚁在弱肉强食、物竞天择的自然界依靠集体的力量和顽强的生命力一直生存繁衍到今天，而且形成了家庭兴旺的蚂蚁王国。目前，全世界的蚂蚁约有 260 属，16000 多种，其中已命名的蚂蚁有 9000 多种。据生态学家估计，地球上蚂蚁总重量约占陆生动物总量的 10%；科学家曾做过计算，蚂蚁能把比自己重 1400 倍的食物拖回家去，能举起超过自身体重 400 倍的物体，无愧于动物世界中的“大力士”这一美誉。

### 1.2.1 蚂蚁的社会形态<sup>[6~9]</sup>

蚂蚁在 8000 万年之前就建立了自己的社会，而我们人类只有 5000 余年的文明史。人类的许多城市都有不少都市问题，可是小小的蚂蚁却能建立起组织完好的复杂“城市”。有许多“蚂蚁城市”往往由 5000 万个成员组成，而在人类社会人口较为稠密的城市也不过才 1000 多万人。

蚂蚁有四种不同的蚁型，即蚁后、雄蚁、工蚁和兵蚁。不同的蚁型有不同的分工：蚁后被称作“蚂蚁女王”，其生殖能力很强，主要职责是产卵、繁殖后代和管理蚂蚁家族，她可根据食物多少决定自己是否生育，还可根据筑巢和建立新群体所需的蚂蚁数量来调节其“人口”结构；雄蚁具有发达的生殖器官，其主要职能是与蚁后交配；工蚁是没有生殖能力的雄性，其主要职责是建造巢穴、采集食物、伺喂幼蚁和蚁后等；兵蚁的头比较大，上颚发达，可以粉碎坚硬食物，在保卫群体时即成为战斗的武器。

化学通信是蚂蚁采取的基本信息交流方式，自然界中的蚂蚁会分泌一种化学刺激物——信息素(pheromone)。蚂蚁的许多行为受信息素调控，如蚁后分泌名为“女皇物质”的信息素来控制工蚁的发育。遇警时，信息素可刺激蚁群兴奋，具有使蚁群按计划执行某项任务的作用。蚂蚁特有的控制自身环境的能力，是在其社会行为不断发展的过程中获得的。

虽然“蚂蚁王国”是个极端母性的社会，但“蚂蚁王国”的蚂蚁们彼此都是平等的，就连“蚂蚁女王”也仅仅具有生儿育女的作用。虽然蚁后是其他蚂蚁的母亲，但她不具有任何领导地位。所有的家庭成员都相互平等，各尽其责，为家族的兴旺贡献力量。蚂蚁社会是家族社会，在这个社会中，所有的家庭成员都是

“一家人”。而且蚂蚁都是“利他主义”者，只要符合集体利益的需要，它们可以献出自己的生命。蚂蚁对同类极其友善，且习惯把食物送给饥饿的伙伴吃。每只蚂蚁都有一个附属胃，当它的同伴需要时，可把附属胃中储存的食物献给对方。可以说，“蚂蚁王国”是一个永远协调一致的“共和国”。

### 1.2.2 蚂蚁的群体行为

尽管蚂蚁个体比较简单，但整个蚂蚁群体却表现为高度机构化的社会组织，在许多情况下能完成远远超过蚂蚁个体能力的复杂任务<sup>[10]</sup>。这种能力来源于蚂蚁群体中的个体协作行为，其群体行为主要包括寻找食物、任务分配和构造墓地等三种。

#### 1.2.2.1 寻找食物

自然界中，蚂蚁的食物源总是随机散布于蚁巢周围。我们只要仔细观察就可以发现，经过一段时间后，蚂蚁总能找到一条从蚁巢到食物源的最短路径。

单只蚂蚁的能力和智力非常简单，但它们通过相互协调、分工、合作完成不论工蚁还是蚁后都不可能有足够的能力来指挥完成的筑巢、觅食、迁徙、清扫蚁穴等复杂行为，比如蚂蚁在觅食过程中能够通过相互协作找到食物源和巢穴之间的最短路径<sup>[11, 12]</sup>。在现实生活中，我们总可以观察到大量蚂蚁在巢穴与食物源之间形成近乎直线的路径，而不是曲线或者圆等其他形状，如图 1.1(a)所示。蚂蚁群体不仅能完成复杂的任务，而且还能适应环境的变化，如在蚁群运动路线上突然出现障碍物时，一开始各只蚂蚁分布是均匀的，不管路径长短，蚂蚁总是先按同等概率选择各条路径，如图 1.1(b)所示。蚂蚁在运动过程中，能够在其经过的路径上留下信息素，而且能感知这种物质的存在及其强度，并以此指导自己运动的方向，蚂蚁倾向于信息素浓度高的方向移动。相等时间内较短路径上的信息量就遗留得比较多，则选择较短路径的蚂蚁也随之增多，如图 1.1(c)所示。不难看出，由于大量蚂蚁组成的蚁群集体行为表现出了一种信息正反馈现象，即某一路径上走过的蚂蚁越多，则后来者选择该路径的概率就越大，蚂蚁个体之间就是通过这种信息交流机制来搜索食物，并最终沿着最短路径行进，如图 1.1(d)所示。

Goss S 等曾于 1989 年做了著名的“双桥”实验<sup>[13]</sup>，如图 1.2 所示。其中，图 1.2(a)是“双桥”实验的真实照片，图 1.2(b)给出了“双桥”实验中非对称桥的尺寸，图 1.2(c)和图 1.2(d)分别为实验进行 4 分钟和 8 分钟时的情况。该实验表明，蚁群最终会选择一条从巢穴到食物源(觅食区)的最短路径。

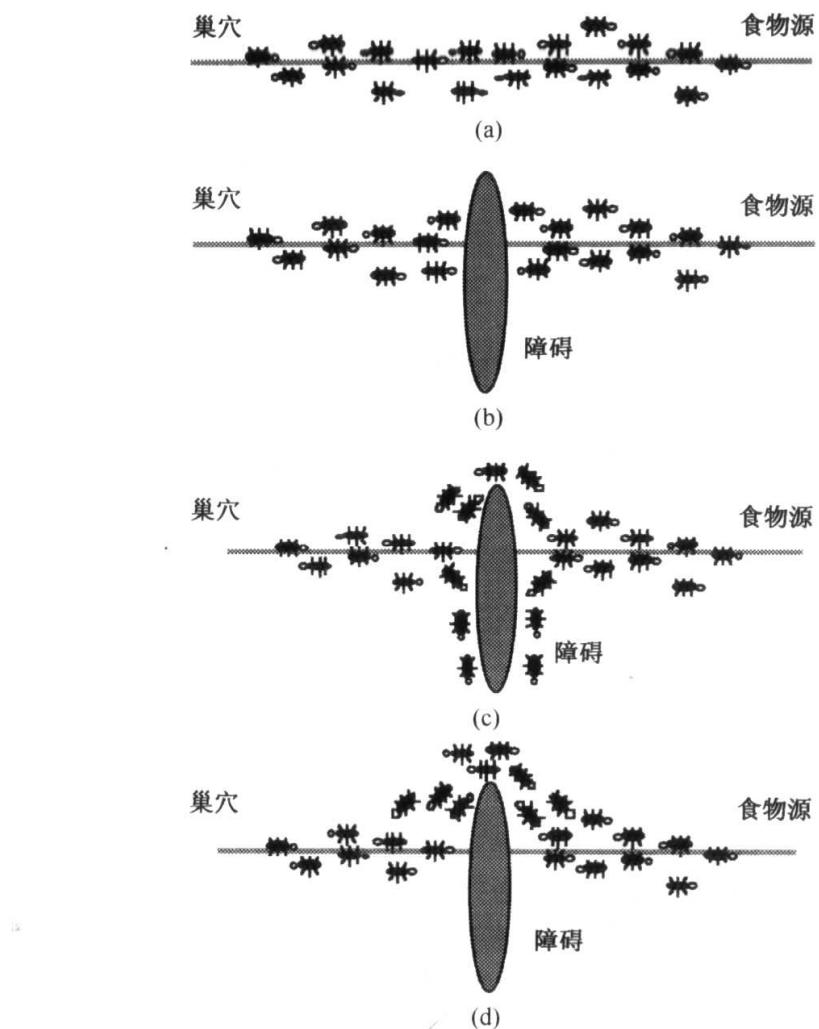


图 1.1 现实中蚁群寻找食物的过程

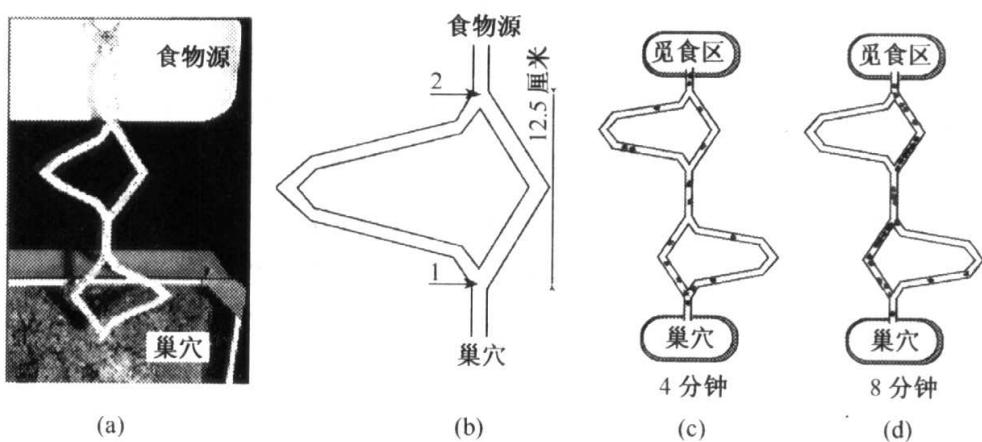


图 1.2 蚁群觅食的“双桥”实验

Goss S 等通过进一步研究，给出了“双桥”实验的数学模型<sup>[13]</sup>。首先，假设在非对称桥上的信息量与过去一个时间段内经过该桥的蚂蚁数目成正比；其次，假设某一时刻蚂蚁按照桥上残留信息量的多少来选择其中某座桥，经过该桥的蚂蚁数目越多，则该桥上的残留信息量就越大。该实例中，假设短桥为 A，长桥为 B， $A_m$  和  $B_m$  分别表示经过桥 A 和桥 B 的蚂蚁数目 ( $A_m + B_m = m$ )，则当所有  $m$  只蚂蚁都经过两座桥之后，第  $m+1$  只蚂蚁选择桥 A 的概率为

$$P_A(m) = \frac{(A_m + k)^h}{(A_m + k)^h + (B_m + k)^h} \quad (1.2.1)$$

而选择桥 B 的概率为

$$P_B(m) = 1 - P_A(m) \quad (1.2.2)$$

式中，参数  $h$  和  $k$  用以匹配真实实验数据。第  $m+1$  只蚂蚁首先按照公式 (1.2.1) 计算选择概率  $P_A(m)$ ，然后生成一个在区间  $[0, 1]$  上均匀分布的随机数  $\varphi$ ，若  $\varphi \leq P_A(m)$ ，则选择桥 A；否则选择桥 B。

对蚁群觅食过程的计算机动态模拟如图 1.3 所示<sup>[14]</sup>，其中食物源 1、食物源 2 及食物源 3 是随机放置在蚁巢周围且与蚁巢距离不等的 3 个等量食物源，觅食的最终结果是距蚁巢最近的食物源的路径上聚集的蚂蚁数目最多，其觅食过程体现了蚁群从无序到有序的动态优化过程。

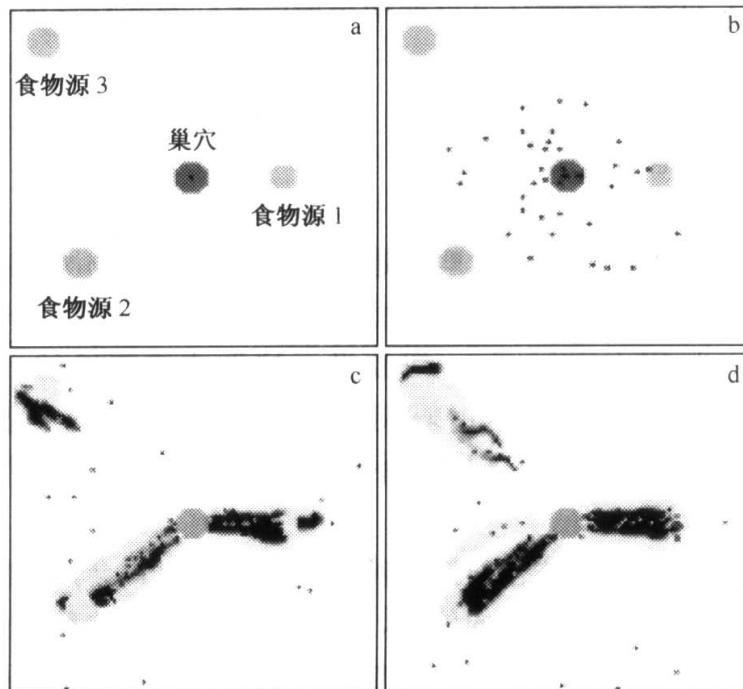


图 1.3 蚁群觅食过程的计算机动态模拟

### 1.2.2.2 任务分配

任务分配是蚂蚁群体行为中又一个非常显著的特点。蚁群中各个蚂蚁都有不同的分工，每只蚂蚁对不同任务的响应也是不完全相同的，但是它们可通过角色的协调和相互的协作来很好地完成许多任务。蚂蚁的任务分配可大致分为两个层次：第一个层次的划分一般可分为从事繁殖的个体和从事日常工作的个体；又可对从事日常工作的个体作进一步划分，即可分为寻找食物的蚂蚁和建筑巢穴的蚂蚁<sup>[15]</sup>。蚂蚁是在没有任何指导信息的情况下进行上述分配的，在这种任务分配过程中存在着一种科学的动态平衡。

蚂蚁对任务响应的行为与现实中生产调度、动态任务分配等问题很相似。当正在执行任务的蚂蚁受到外界环境所赋予的新任务时，相当于受到一个刺激。当蚂蚁受到的刺激较小时，蚂蚁仍然保持原有的状态；而当蚂蚁受到的刺激大于其激活阈值时，蚂蚁就会去处理相应的新任务。当新任务不被处理时，它就会开始影响蚂蚁的行为。随着时间的推移，其紧迫程度也越来越高，紧迫程度高的新任务就使得蚂蚁不能忽视它，有时候蚂蚁可能需要放下正在进行的任务转而去处理更加紧迫的工作。当新任务被处理后，这一新任务的紧迫度就会下降，从而不会引起蚂蚁的注意。环境中的新任务可能动态地产生，因此蚂蚁也相应地在环境中移动，以便及时响应新任务的出现。

### 1.2.2.3 构造墓地

生物学家通过长期观察发现，蚂蚁通常构筑一个固定场所的墓地，尽管蚂蚁并不知道关于蚂蚁整体的任何指导性信息，但它可将死去的同伴安放在一个固定的区域。Chrétien L 用 *Lasius niger* 蚂蚁做了蚁群构造墓地的实验<sup>[16]</sup>；Deneubourg J L 等也用 *Pheidole pallidula* 蚂蚁做了类似的实验<sup>[17]</sup>。实验证实某些种类的蚁群的确能够构造蚁穴中的墓地，并将分散在蚁穴内各处的蚂蚁尸体堆积起来。

另外，观察还发现，蚁群在安排不同蚁卵的位置时，会按照蚁卵大小不同而分别将其堆放在蚁穴周围和中央的位置，我们可把蚁群的这种行为同构造墓地行为一起称之为蚁群的聚类行为。图 1.4 是真实蚁群聚类行为的实验结果，四张照片分别是实验初始状态、3 小时、6 小时和 36 小时的真实蚁群聚类情况。

Deneubourg J L 等进而提出了一种解释蚁群聚类现象的基本模型，这种模型主要是基于对单只蚂蚁“拾起”、“放下”物体的行为方式进行建模。一只随机移动的无负载蚂蚁在遇到一个物体时，如果该物体周围与之相同的物体越少，则“拾起”该物体的概率越大；反之，则“放下”该物体的概率越大。这样可以保

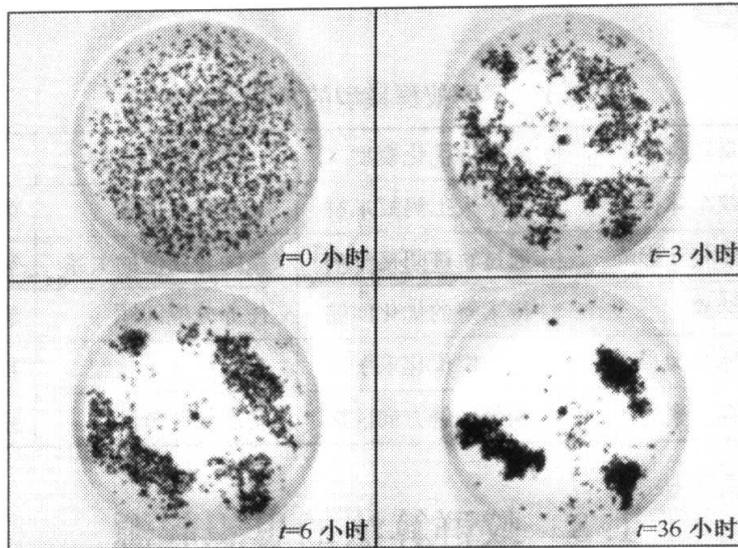


图 1.4 真实蚁群的聚类行为

证不破坏大堆的物体，并且能够收集小堆的物体。假设环境中只有一种类型的物体，则可定义由一只随机移动的无负载蚂蚁“拾起”一个物体的概率为

$$p_p = \left( \frac{k_1}{k_1 + f} \right)^2 \quad (1.2.3)$$

式中， $f$  表示蚂蚁附近的物体观察因子，反映了蚂蚁附近同类物体的个数， $k_1$  是阈值常数，若  $k_1 \gg f$ ，则  $p_p$  接近于 1（即当周围没有多少对象时，“拾起”一个物体的概率很大）；若  $k_1 \ll f$ ，则  $p_p$  接近于 0（即在一个稠密的聚类中，一个物体被移动的概率很小）。一只随机移动的有负载蚂蚁“放下”一个物体的概率为

$$p_d = \left( \frac{f}{k_2 + f} \right)^2 \quad (1.2.4)$$

式中， $k_2$  是不同于  $k_1$  的阈值常数，且当  $k_2 \gg f$ ，则  $p_p$  接近于 1；而当  $k_2 \ll f$ ，则  $p_p$  接近于 0。由此，蚂蚁的“拾起”和“放下”行为大致遵守相反的规则。

蚂蚁群体能以惊人的效率来有效地完成优化和控制的复杂任务，这不仅引起了生物学家和昆虫学家的极大兴趣。而且近些年也激发了计算机学家和系统优化学家的研究兴趣。他们发现通过对蚂蚁群体行为的深入研究可以引导他们从一个完全未知的领域来发展一些新型的、分布式的多智能体仿生算法。Dorigo M 等<sup>[1, 2]</sup>由蚂蚁群体的“寻找食物”行为得到启发而提出了蚁群优化模型；Bonabeau E 等<sup>[18]</sup>由蚂蚁群体的“任务分配”行为受到启发而提出了简单阈值模型；Deneubourg J L 等<sup>[17]</sup>由蚂蚁群体的“构造墓地”行为得到启发而提出了仿生制造模型。

模拟蚁群“寻找食物”、“任务分配”和“构造墓地”三种群体行为而诞生的“蚁群优化”、“简单阈值”及“仿生制造”三种蚁群模型的系统学特征可通过表

1.1 进行对比说明。

表 1.1 不同蚁群模型的系统特征比较

系 统	真实蚁群	蚁群优化模型	简单阈值模型	仿生制造模型
元素	蚂蚁个体行为	单个人工蚂蚁求解	个体完成任务	单元体拾取与堆积
结构	个体相互影响	通过轨迹间接影响	激励与阈值的关系	空间单元体分布情况
系统环境	自然界	待求解的优化问题	任务分配问题	数据分析问题
行为和功能	群体行为	自组织优化求解	自组织任务分配	自组织数据分析
系统的演化	生存、繁衍	朝最优解方向前进	动态任务分配	完成数据分析

## 1.3 蚁群算法的思想起源

根据蚂蚁“寻找食物”的群体行为，意大利学者 Dorigo M 等<sup>[1]</sup>于 1991 年在法国巴黎召开的第一届欧洲人工生命会议 (European Conference on Artificial Life, ECAL) 上最早提出了蚁群算法的基本模型；1992 年，Dorigo M 又在其博士学位论文中进一步阐述了蚁群算法的核心思想<sup>[2]</sup>。

### 1.3.1 人工蚂蚁与真实蚂蚁的异同比较

#### 1.3.1.1 相同点比较

蚁群算法是从自然界中真实蚂蚁觅食的群体行为得到启发而提出的，其很多观点都来源于真实蚁群<sup>[1]</sup>，因此算法中所定义的人工蚂蚁与真实蚂蚁存在如下共同点。

##### 1. 都存在一个群体中个体相互交流通信的机制

人工蚂蚁和真实蚂蚁都存在一种改变当前所处环境的机制：真实蚂蚁在经过的路径上留下信息素，人工蚂蚁改变在其所经路径上存储的数字信息，该信息就是算法中所定义的信息量，它记录了蚂蚁当前解和历史解的性能状态，而且可被其他后继人工蚂蚁读写。蚁群的这种交流方式改变了当前蚂蚁所经路径周围的环境，同时也以函数的形式改变了整个蚁群所存储的历史信息。通常，在蚁群算法中有一个挥发机制，它像真实的信息量挥发一样随着时间的推移来改变路径上的信息量。挥发机制使得人工蚂蚁和真实蚂蚁可以逐渐地忘却历史遗留信息，这样可使蚂蚁在选择路径时不局限于以前蚂蚁所存留的“经验”。

## 2. 都要完成一个相同的任务

人工蚂蚁和真实蚂蚁都要完成一个相同的任务，即寻找一条从源节点(巢穴)到目的节点(食物源)的最短路径。人工蚂蚁和真实蚂蚁都不具有跳跃性，只能在相邻节点之间一步步移动，直至遍历完所有城市。为了能在多次寻路过程中找到最短路径，则应该记录当前的移动序列。

## 3. 利用当前信息进行路径选择的随机选择策略

人工蚂蚁和真实蚂蚁从某一节点到下一节点的移动都是利用概率选择策略实现的，概率选择策略只利用当前的信息去预测未来的情况，而不能利用未来的信息。因此，人工蚂蚁和真实蚂蚁所使用的选择策略在时间和空间上都是局部的。

### 1.3.1.2 不同点比较

在从真实蚁群行为获得启发而构造蚁群算法的过程中，人工蚂蚁还具备了真实蚂蚁所不具有一些特性：

- (1) 人工蚂蚁存在于一个离散的空间中，它们的移动是从一个状态到另一个状态的转换；
- (2) 人工蚂蚁具有一个记忆其本身过去行为的内在状态；
- (3) 人工蚂蚁存在于一个与时间无关联的环境之中；
- (4) 人工蚂蚁不是完全盲从的，它还受到问题空间特征的启发。例如有的问题中人工蚂蚁在产生一个解后改变信息量，而有的问题中人工蚂蚁每作出一步选择就更改信息量，但无论哪种方法，信息量的更新并不是随时都可进行的；
- (5) 为了改善算法的优化效率，人工蚂蚁可增加一些性能，如预测未来、局部优化、回退等，这些行为在真实蚂蚁中是不存在的。在很多具体应用中，人工蚂蚁可在局部优化过程中相互交换信息，还有一些改进蚁群算法中的人工蚂蚁可实现简单预测。

## 1.3.2 蚁群算法模型的建立

### 1.3.2.1 对蚂蚁个体的抽象

由于蚁群算法是对自然界中真实蚂蚁觅食行为的一种模拟，是一种机理上的应用，因此首先必须对真实蚂蚁进行抽象，而不可能也没必要对蚂蚁个体进行完全再现<sup>[19]</sup>。抽象的目的就是为了能够更加有效地刻画出真实蚁群中能够为算法所借鉴的机理，同时摒弃与建立算法模型无关的因素。这样抽象出来的人工蚂蚁

可以看做是一个简单的智能体，能够完成所求问题简单解的构造过程，也能通过一种通信手段相互影响。

### 1.3.2.2 问题空间的描述

自然界中的真实蚂蚁存在于一个三维的环境中，而问题空间的求解一般是在平面内进行的，因此需要将蚂蚁觅食的三维空间抽象为一个平面。这一点比较容易理解，因为蚂蚁觅食所走的路径本来就存在于一个二维空间（平面或者曲面）上。另外一个问题的真实蚂蚁是在一个连续的二维平面中行走的，而我们无法用计算机直接来完整的描述一个连续的平面，因为计算机处理的是离散事件，因此必须将连续的平面离散化由一组点组成的离散平面，人工蚂蚁可在抽象出来的点上自由运动。这个抽象过程的可行性在于，尽管蚂蚁是在连续平面行动，但其行动经过的总是离散点，因此抽象过程只是提高了平面点离散分布的粒度，与其觅食行为的本身机理没有任何冲突。

基于上述分析，很容易得到蚁群算法所求解的问题空间可用一个重要的数学工具——图(graph)来描述。在工程实际中的很多问题都可以用图来描述，这便使蚁群算法的广泛应用成为可能。

### 1.3.2.3 寻找路径的抽象

真实蚂蚁在觅食过程中主要按照所处环境中的信息量来决定其前进的方向，而人工蚂蚁是在平面的节点上运动的，因此可把觅食过程抽象成算法中解的构造过程，将信息素抽象为存在于图的边上的轨迹。在每一节点，人工蚂蚁感知连接该节点与相邻节点边上的信息素轨迹浓度，并根据该浓度大小决定走向下一节点的概率。用任意两个节点分别表示蚂蚁的巢穴（初始节点）和食物源（目标节点），人工蚂蚁从初始节点按照一定状态转移概率选择下一节点，依此类推，最终选择行走到目标节点，这样便得到了所求问题的一个可行解。

### 1.3.2.4 信息素挥发的抽象

自然界中的真实蚂蚁总是在所经路径上连续不断地留下信息素，而信息素也会随着时间的推移而连续不断地挥发。由于计算机处理的事件只能是离散事件，所以必须使信息素的挥发离散发生。通常的做法是，当蚂蚁完成从某一节点到下一节点的移动后，即经过一个时间单位之后，进行一次信息素的挥发，而这种在离散时间点进行信息素挥发的方式与蚂蚁觅食过程的机理是完全相符的。

### 1.3.2.5 启发因子的引入

以上几点是对真实蚂蚁觅食行为的抽象，整个过程体现了蚁群算法的自组织性，但是这种自组织系统存在一个缺陷，即系统的演化需要耗费较长的时间。而实际应用时对算法运行时间的要求也是必不可少的，因此在决定蚂蚁行走方向的状态转移概率时，引入了一个随机搜索的过程，即引入了启发因子，根据所求问题空间的具体特征，给蚁群算法一个初始的引导，这个过程极大地增加了算法的时间有效性，从而使蚁群算法的有效应用成为可能。

不难看出，通过上述描述的一些抽象过程，可建立一个蚁群算法的基本模型。其问题空间是用图来描述的，解的获取是构造性的，而且在解的构造过程中人工蚂蚁没有接受任何全局的指导信息，因而求解过程是自组织的。在定义了一些规则之后，人工蚂蚁就可求解那些可用图来描述的问题。

在蚁群算法中，蚂蚁个体是蚁群算法的基本单元。蚂蚁个体所拥有的知识来源于与其他蚂蚁个体的通讯以及对周围环境的感知，因此，蚂蚁个体的知识积累是一个动态的过程。蚂蚁个体通过随机决策机制和相互协调机制可自适应地作出并完成自身评价，蚂蚁个体之间的这种分布性和协作性正是蚁群算法所研究的核心内容。

蚁群算法具有很强的自学习能力，可根据环境的改变和过去的行为结果对自身的知识库或自身的组织结构进行再组织，从而实现算法求解能力的进化，而这种进化是环境变化与算法自学习能力交互作用的产物，同时算法机理的复杂性和环境变化的不确定性进一步增加了蚁群算法的不可预测性。

## 1.4 蚁群算法的研究进展

自 1991 年意大利学者 Dorigo M 等<sup>[1, 2]</sup>提出蚁群算法后的近 5 年时间里，并没有在国际学术界引起广泛关注，自然这段时期在蚁群算法理论及其应用上也没有取得突破性进展。到了 1996 年，Dorigo M 等在《IEEE Transactions on Systems, Man, and Cybernetics-Part B》上发表了“Ant system: optimization by a colony of cooperating agents”一文<sup>[3]</sup>，在这篇文章中，Dorigo M 等不仅更加系统地阐述了蚁群算法的基本原理和数学模型，还将其与遗传算法、禁忌搜索算法、模拟退火算法、爬山法等进行了仿真实验比较，并把单纯地解决对称 TSP 拓展到解决非对称 TSP、指派问题（quadratic assignment problem, QAP）以及车间作业调度问题（job-shop scheduling problem, JSP），且对蚁群算法中初始化参数对其性能的影响做了初步探讨，这是蚁群算法发展史上的又一篇奠基性文章。从目前公开发表的蚁群算法相关论文来看，其中 70% 以上的论文将这篇文

章或 1991 年 Dorigo M 等在 ECAL 上发表的“Distributed optimization by ant colonies”一文列为参考文献。自 1996 年之后的 5 年时间里，蚁群算法逐渐引起了世界许多国家研究者的关注，其应用领域得到了迅速拓宽，这期间也有大量有价值的研究成果陆续发表。

对蚁群算法不断高涨的研究热情导致了 1998 年 10 月 15 日至 10 月 16 日在比利时布鲁塞尔召开了第一届蚁群算法国际研讨会（ANTS'98），会议由创始人 Dorigo M 负责组织。第一届就吸引了来自世界各地的 50 多位蚁群算法研究者，随后每隔两年都要在布鲁塞尔召开一次蚁群算法国际研讨会，历届会议的论文集均由著名的《Lecture Notes in Computer Science》(SCI Index) 结集出版。2000 年，Dorigo M 和 Bonabeau E 等<sup>[20]</sup>在国际顶级学术刊物《Nature》上发表了蚁群算法的研究综述，从而把这一领域的研究推向了国际学术的最前沿。鉴于 Dorigo M 在蚁群算法研究领域的杰出贡献，2003 年 11 月欧盟委员会特别授予他“居里夫人杰出成就奖(Marie Curie Excellence Award)”。

进入 21 世纪后的最近几年，国际著名的顶级学术刊物《Nature》曾多次对蚁群算法的研究成果进行报道<sup>[20~22]</sup>，《Future Generation Computer Systems》(Vol. 16, No. 8) 和《IEEE Transactions on Evolutionary Computation》(Vol. 6, No. 4) 分别于 2000 年和 2002 年出版了蚁群算法特刊。如今，在国内外许多学术期刊和会议上，蚁群算法已经成为一个备受关注的研究热点和前沿性课题。

Gutjahr W J 于 1999 年撰写的技术报告<sup>[23]</sup>和 2000 年发表的学术论文<sup>[24]</sup>在蚁群算法发展史上有着特殊的作用，因为这两篇文章首次对蚁群算法的收敛性进行了证明。Gutjahr W J 将蚁群算法的行为简化为在一幅代表所求问题的有向图上的行走过程，进而从有向图论的角度对一种改进蚁群算法——图搜索蚂蚁系统(graph-based ant system, GBAS) 的收敛性进行了理论分析，证明了在一些合理的假设条件下，他所提出的 GBAS 能以一定的概率收敛到所求问题的最优解。

我国在蚁群算法领域的研究起步较晚，从公开发表的论文（以投稿日期为标准）看，国内最先研究蚁群算法的是东北大学控制仿真研究中心的张纪会博士与徐心和教授（1997 年 10 月）<sup>[25]</sup>。在国内蚁群算法的众多研究者中，值得一提的是当时年仅 17 岁的高二学生陈烨于 2001 年在《计算机工程》(Vol. 27, No. 12) 上发表了“带杂交算子的蚁群算法”一文<sup>[26]</sup>，并基于 Visual Basic 开发了一个功能齐全、界面友好的“蚁群算法实验室”，引起了国内广大蚁群算法研究者的极大关注，正如《计算机工程》的“编者按”所言：“一个高中生能写出如此论文实属不易”。

回顾蚁群算法自创立以来十多年的发展历程，目前人们对蚁群算法的研究已由当初单一的 TSP 领域渗透到了多个应用领域。由解决一维静态优化问题发展到解决多维动态组合优化问题，由离散域范围内研究逐渐拓展到了连续域范围内

研究，而且在蚁群算法的硬件实现上取得了突破性进展，同时在蚁群算法的模型改进及与其他仿生优化算法的融合方面也取得了相当丰富的研究成果，从而使这种新兴的仿生优化算法展现出前所未有的勃勃生机，并已经成为一种完全可与遗传算法相媲美的仿生优化算法。

蚁群算法在数学理论、算法改进、实际应用、硬件实现以及算法融合等方面的研究进展将在后续章节中进行详细介绍，这里不再赘述。表 1.2 列举了部分具有代表性的蚁群算法及其应用情况。

表 1.2 蚁群算法及其应用

问题名称	研究者	算法名称	年代	代表性文献
旅行商问题(traveling salesman problem, TSP)	Dorigo, Maniezzo and Colomi	AS	1991	[1, 2, 3, 27]
	Gambardella and Dorigo	Ant-Q	1995	[28]
	Dorigo and Gambardella	ACS and	1996	[29, 30, 31]
	Stützle and Hoos	ACS-3-opt	1997	[32, 33, 34]
	Bullnheimer, Hartl and Strauss	MMAS	1997	[35, 36]
	Cordón, <i>et al</i>	ASrank	2000	[37]
	Kaji T	BWAS	2001	[38]
指派问题(quadratic assignment problem, QAP)	Maniezzo, Colomi and Dorigo	AS-QAP	1994	[39]
	Gambardella, Taillard and Dorigo	HAS-QAP	1997	[40, 41]
	Stützle and Hoos	MMAS-QAP	1997	[42, 43]
	Maniezzo	ANTS-QAP	1998	[44]
	Maniezzo and Colomi	AS-QAP	1999	[45]
调度问题(scheduling problem)	Colomi, Dorigo and Maniezzo	AS-JSP	1994	[46]
	Stützle	AS-FSP	1997	[47]
	den Besten, Stützle and Dorigo	ACS-SMTWTP	1999	[48]
车辆路径问题(vehicle routing problem, VRP)	Bullnheimer, Hartl and Strauss	AS-VRP	1997	[49, 50]
	Gambardella, Taillard and Agazzi	HAS-VRP	1999	[51]
有向连接网络路由(connection-oriented network routing)	Schoonderwoerd, <i>et al</i>	ABC	1996	[52, 53]
	White, Pagurek and Oppacher	ASGA	1998	[54]
	Di Caro and Dorigo	AntNet-FS	1998	[55]
	Bonabeau, <i>et al</i>	ABC-smart ants	1998	[56]
无连接网络路由(connection-less network routing)	Di Caro and Dorigo	AntNet	1997	[57, 58, 59]
	Subramanian, Druschel and Chen	Regular ants	1997	[60]
	Heusse, <i>et al</i>	CAF	1998	[61]
	van der Put and Rothkrantz	ABC-backward	1998	[62, 63]
序列排序问题(sequential ordering problem, SOP)	Gambardella and Dorigo	HAS-SOP	1997	[64]
图形着色问题(graph coloring problem, GCP)	Costa and Hertz	ANTCOL	1997	[65]
	Ahn S H	ANT_XRLF	2003	[66]

续表

问题名称	研究者	算法名称	年代	代表性文献
最短公共超串问题 (shortest common supersequence)	Michel and Middendorf	AS-SCS	1998	[67, 68]
频率分配问题 (frequency assignment problem, FAP)	Maniezzo and Carbonaro	ANTS-FAP	1998	[69, 70]
广义指派问题 (generalized assignment problem, GAP)	Ramalhinho Lourenco and Serra	MMAS-GAP	1998	[71]
多背包问题 (multiple knapsack problem, MKP)	Leguizamón and Michalewicz Parra-Hernandez R, Dimopoulos N	AS-MKP ACS-MKP	1999 2003	[72] [73]
光纤网络路由 (optical networks routing)	Navarro Varela and Sinclair	ACO-VWP	1999	[74]
冗余分配问题 (redundancy allocation problem, RAP)	Liang and Smith Liang and Smith	AS-RAP ACO-RAP	1999 2004	[75] [76]
单机排序 (machine scheduling)	Bauer, <i>et al</i>	ACS-MTTP	1999	[77]
故障识别 (fault identification)	Chang, <i>et al</i> 孙京皓, 李秋艳等	AS-FI ACA-FI	1999 2004	[78] [79]
约束满足问题 (constraint satisfaction problem, CSP)	Schoofs and Naudts Solnon Fu and Dozier Mertens and Holvoet	AA-CSP ACO-CSP ASoHC CSAA-CSP	2000 2002 2003 2004	[80] [81] [82] [83]
连续函数优化 (continuous function optimization)	Mathur M, <i>et al</i> 魏平, 熊伟清	ACO-CFO ACA-CFO	2000 2001	[84] [85]
全球定位系统 (global positioning system, GPS)	Camara and Loureiro Antonio Saleh	GPSAL ACS-GPS	2000 2002	[86] [87]
集合覆盖问题 (set covering problem, SCP)	de A Silva and Ramalho	AS-SCP	2001	[88]
系统辨识 (system identification)	Wang and Wu Abbaspour, <i>et al</i> 汪镭, 吴启迪 Duan, <i>et al</i>	ASA-SI ACO-SI ASA-SI ACA-SI	2001 2001 2003 2004	[89] [90] [91] [92]
机器人路径规划 (robot path planning)	金飞虎, 洪炳熔等	ACA-RPP	2002	[93]
数据挖掘 (data mining)	Parepinelli, <i>et al</i>	Ant-Miner-DM	2002	[94]
图像处理 (image processing)	Meshoul and Batouche Zheng, <i>et al</i>	ACS-IP HACA-IP	2002 2003	[95] [96]

续表

问题名称	研究者	算法名称	年代	代表性文献
武器目标分配问题 (weapon-target assignment problem, WTAP)	Lee, <i>et al</i>	ACO-WTAP	2002	[97]
二维格模型蛋白质折叠问题 (2D HP protein folding problem)	Shmygelska, <i>et al</i>	ACA-2DHPPFP	2002	[98]
地雷探测问题 (mine detection problem, MDP)	Kumar and Sahin	AC-MDP	2002	[99]
	Kumar and Sahin	AC-MDP	2003	[100]
	Chapman and Sahin	AA-MDP	2004	[101]
岩土工程 (geotechnical engineering)	陈昌富, 谢学斌	HACA-GE	2002	[102]
	高玮, 郑颖人	ACA-GE	2002	[103]
光谱解析 (spectra analyzing problem, SAP)	丁亚平, 苏庆德等	ACA-SAP	2002	[104, 105]
加权最小碰集问题 (weighted minimum hitting set problem, WMHSP)	Cincotti	AA-WMHSP	2003	[106]
最大独立集问题 (maximum independent set problem, MISP)	Li and Xu	ACO-MISP	2003	[107]
网格分割问题 (mesh partitioning problem, MPP)	Korosec, <i>et al</i>	ACA-MPP	2004	[108]
转台控制 (turntable control)	段海滨, 王道波等	IACA-TC	2004	[5, 109]
几何约束 (geometric constraint problem, GCP)	曹春红, 李文辉等	GAAA-GCP	2004	[110, 111]
圆排列问题 (circle permutation problem, CPP)	高尚, 杨静宇等	ACSAA-CPP	2004	[112]
点覆盖问题 (point covering problem, PCP)	Hua, <i>et al</i>	AA-PCP	2004	[113]
粗糙数据约简 (rough data reduction, RDR)	Jensen and Shen	ACO-RDR	2005	[114]

## 1.5 本书的体系结构

本书系统介绍了蚁群算法的理论、方法、应用及其工程实现，其逻辑结构是按照分析、深化、改进、应用、实现、融合的学术研究步骤安排的。全书共分 10 章，其内容（连同附录）基本构成了一个完整的封闭体系，本书的组织结构如图 1.5 所示。

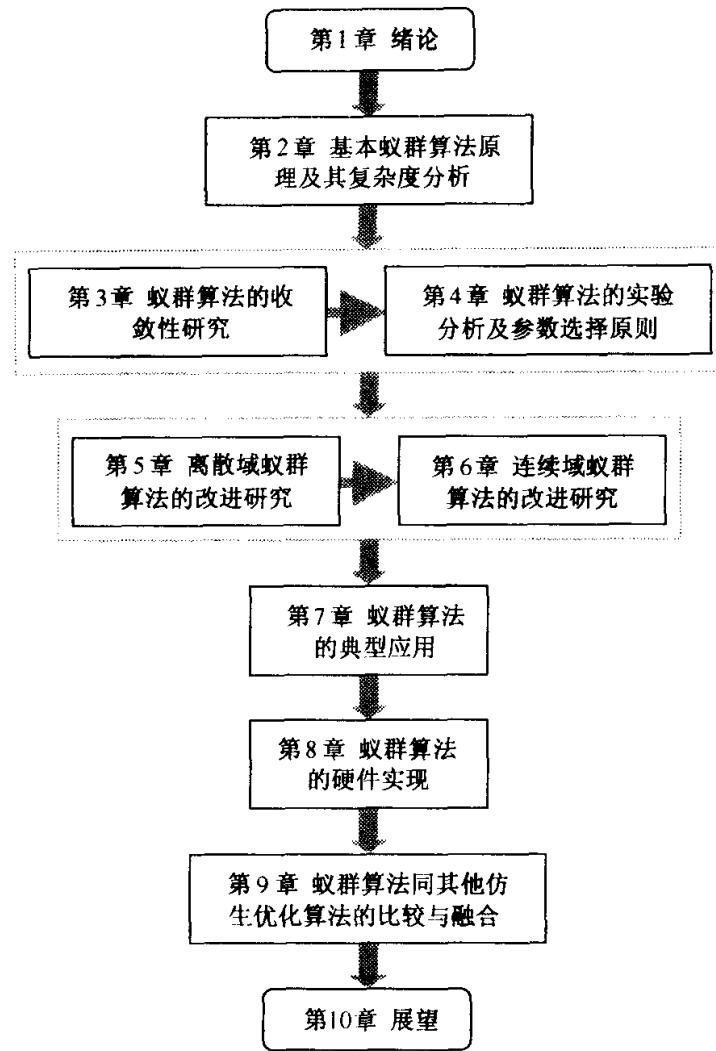


图 1.5 本书的组织结构

具体而言，各章主要包括如下内容。

**第1章 绪论：**在介绍蚂蚁的社会形态、群体行为等生物学特征的基础上，详细阐述了蚁群算法的思想起源，然后评述了蚁群算法的发展现状，同时列举了部分具有代表性的蚁群算法及其应用情况，最后给出了本书的体系结构。

**第2章 基本蚁群算法原理及其复杂度分析：**在从深层意义上进一步分析蚁群算法机制原理的基础上，讨论了基本蚁群算法的系统学特征，随后给出了基本蚁群算法的数学模型、具体实现步骤以及基于不同开发环境的软件实现，最后对基本蚁群算法的空间复杂度和时间复杂度分别进行了分析，并给出了基本蚁群算法的性能评价指标。该章内容是基本蚁群算法的机理分析部分，也是深入理解蚁群算法、利用蚁群算法开展研究工作的基础。

**第3章 蚁群算法的收敛性研究：**从有向图、Markov链、离散鞅、随机过程、分支Wiener过程、不动点理论等角度系统地研究了多种不同形式蚁群算法

的收敛性。该章内容是全书的理论基础部分，可为从数学意义上理解蚁群算法、改进蚁群算法以及应用蚁群算法解决实际问题提供理论依据和指导。

**第 4 章 蚁群算法的实验分析及参数选择原则：**通过大量的数字仿真实验对蚁群算法的参数选择原则问题进行了深入研究，并总结出一种选择蚁群算法最优组合参数的有效方法。该章内容弥补了蚁群算法理论分析方面的某些不足之处，可为后续章节中应用蚁群算法求解实际问题提供参考。

**第 5 章 离散域蚁群算法的改进研究：**以对称 TSP 为应用背景，从不同角度研究了离散域蚁群算法的若干改进策略，其改进目的是竭力避免蚁群算法陷入局部最优解，缩短搜索时间，进而提高蚁群算法的全局收敛性能。该章内容可为进一步改进蚁群算法提供有益的借鉴。

**第 6 章 连续域蚁群算法的改进研究：**在系统分析连续域寻优蚁群算法与离散域寻优蚁群算法的不同之处的基础上，着重研究了基于网格划分策略、信息量分布函数、自适应搜索、交叉变异操作、嵌入确定性搜索、连续交互式等改进策略。该章内容对进一步应用蚁群算法求解连续域优化问题具有重要指导意义。

**第 7 章 蚁群算法的典型应用：**重点对蚁群算法在车间作业调度问题、网络路由问题、车辆路径问题、机器人领域、电力系统、故障诊断、控制参数优化、参数辨识、聚类分析、数据挖掘、图像处理、航迹规划、空战决策、岩土工程、化学工业、生命科学、布局优化等若干领域中的典型应用情况进行详细介绍，同时给出了必要的仿真算例，以验证改进后蚁群算法可行性和有效性。该章内容可为相关专业领域的研究者进一步应用蚁群算法提供借鉴和参考。

**第 8 章 蚁群算法的硬件实现：**在介绍仿生硬件基本概念和原理的基础上，重点研究了基于 FPGA 的蚁群算法硬件实现和基于蚁群-遗传算法的软硬件划分。蚁群算法的硬件实现是蚁群算法发展的高级阶段。

**第 9 章 蚁群算法同其他仿生优化算法的比较与融合：**在阐述遗传算法、人工神经网络、微粒群算法、人工免疫算法、人工鱼群算法等仿生优化算法基本原理的基础上，分析了这些仿生优化算法的异同之处，并给出了蚁群算法与这些仿生优化算法的融合策略，且通过仿真算例对融合策略的可行性和有效性进行了分析。

**第 10 章 展望：**集中讨论了目前蚁群算法研究领域中所存在的主要问题，并着重从算法的模型改进、理论分析、并行实现、应用领域、硬件实现、智能融合等角度对蚁群算法在今后的研究方向进行了讨论。

此外，本书还在附录部分给出了基本蚁群算法的 C 语言、Matlab 语言和 Visual Basic 语言三种版本的程序源代码，同时给出了蚁群算法的相关网站和本书的基本术语（中英文对照）及缩略语，并附上了作者即兴为蚁群算法填写的一首小词——《鹧鸪天·蚁群算法》。

## 1.6 本章小结

本章首先阐述了蚂蚁的社会形态、群体行为等生物学特征，随后在总结真实蚂蚁和人工蚂蚁之间异同点的基础上，介绍了蚁群算法的思想起源，紧接着评述了蚁群算法的发展现状，同时列举了部分具有代表性的蚁群算法及其应用情况，最后给出了本书的体系结构。

## 参 考 文 献

- 1 Colorni A, Dorigo M, Maniezzo V, *et al.* Distributed optimization by ant colonies. Proceedings of the 1st European Conference on Artificial Life, 1991, 134~142
- 2 Dorigo M. Optimization, learning and natural algorithms. Ph. D. Thesis, Department of Electronics, Politecnico diMilano, Italy, 1992
- 3 Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Transaction on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- 4 段海滨, 王道波, 朱家强等. 蚁群算法理论及应用研究的进展. 控制与决策, 2004, 19(12): 1321~1326, 1340
- 5 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 6 吴坚, 王常禄. 中国蚂蚁. 北京: 中国林业出版社, 1995
- 7 John N, Sudd I. The behavioral ecology of ants. New York: Chapman and Hall Press, 1987
- 8 Dumphert K. The social biology of ants. London: Pitman Publishing House, 1981
- 9 Denebourg J L, Pasteels J M, Verhaeghe J C. Probabilistic behavior in ants: a strategy of errors? Journal of Theoretical Biology, 1983, 105: 259~271
- 10 Dorigo M, Bonabeau E, Theraulaz G. Ant algorithms and stigmergy. Future Generation Computer System, 2000, 16(8): 851~871
- 11 Dorigo M, Di Caro G, Gambardella L M. Ant algorithms for discrete optimization. Artificial Life, 1999, 5(2): 137~172
- 12 Dorigo M, Di Caro G. The ant colony optimization meta-heuristic. New Ideas in Optimization, London: McGraw-Hill, 1999
- 13 Goss S, Aron S, Deneubourg J L, *et al.* Self-organized shortcuts in the argentine ant. Naturwissenschaften, 1989, 76: 579~581
- 14 Bonabeau E. Marginally stable swarms are flexible and efficient. Journal De Physique, 6(2): 309~320
- 15 何靖华. 面向仿生制造的蚂蚁算法分析研究及应用. 武汉: 华中科技大学硕士学位论文, 2002
- 16 Chrétien L. Organisation spatiale du matériel provenant de l'excavation du nid chez *Messor Barbarus* et des cadavres d'ouvrières chez *Lasius Niger*, Ph. D. Thesis, Université Libre de Bruxelles, Brussels, 1996
- 17 Deneubourg J L, Goss S, Franks N, *et al.* The dynamics of collective sorting: robot-like ants and ant-like robots. Proceedings of the 1st International Conference on Simulation of Adaptive Behavior: From Animals to Animals, 1991, 356~363
- 18 Bonabeau E, Theraulaz G, Deneubourg J L. Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. Proceedings of the Royal Society the London B, 1996,

- 263:1565~1569
- 19 秦玲. 蚁群算法的改进与应用. 扬州: 扬州大学硕士学位论文, 2004
- 20 Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behavior. *Nature*, 2000, 406(6): 39~42
- 21 Michael J B K, Jean-Bernard B, Laurent K. Ant-like task and recruitment in cooperative robots. *Nature*, 2000, 406(31): 992~995
- 22 Jackson D E, Holcombe M, Ratnieks F L W. Trail geometry gives polarity to ant foraging networks. *Nature*, 2004, 432(7019): 907~909
- 23 Gutjahr W J. A generalized convergence result for the graph based ant system, Technical Report 99-09, Dept. of Statistics and Decision Support Systems, University of Vienna, Austria, 1999
- 24 Gutjahr W J. A graph-based ant system and its convergence. *Future Generation Computer Systems*, 2000, 16(8): 873~888
- 25 张纪会, 徐心和. 一种新的进化算法-蚁群算法. *系统工程理论与实践*, 1999, 19(3): 84~87
- 26 陈烨. 带杂交算子的蚁群算法. *计算机工程*, 2001, 27(12): 74~76, 176
- 27 Dorigo M, Maniezzo V, Colorni A. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991
- 28 Gambardella L M, Dorigo M. Ant-Q: a reinforcement learning approach to the traveling salesman problem. *Proceedings of the 12th International Conference on Machine Learning*, 1995, 252~260
- 29 Dorigo M, Gambardella L M. Ant colonies for the traveling salesman problem. *Bio Systems*, 1997, 43: 73~81
- 30 Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 53~66
- 31 Gambardella L M, Dorigo M. Solving symmetric and asymmetric TSPs by ant colonies. *Proceedings of the IEEE Conference on Evolutionary Computation*, 1996, 622~627
- 32 Stützle T, Hoos H. The MAX-MIN ant system and local search for the traveling salesman problem. *Proceedings of the IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference*, 1997, 309~314
- 33 Stützle T, Hoos H. Improvements on the ant system: introducing MAX-MIN ant system. *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, 1998, 245~249
- 34 Stützle T, Hoos H. MAX-MIN ant system. *Future Generation Computer Systems*, 2000, 16(8): 889~914
- 35 Bullnheimer B, Hartl R F, Strauss C. A new rank-based version of the ant system: a computational study. Technical Report POM-03/97, Institute of Management Science, University of Vienna, Austria, 1997
- 36 Bullnheimer B, Hartl R F, Strauss C. A new rank-based version of the ant system: a computational study. *Central European Journal for Operations Research and Economics*, 1999, 7(1): 25~38
- 37 Cordón, Fernández de Viana I, Herrera F, et al. A new ACO model integrating evolutionary computation concepts: the best-worst ant system. *Proceedings of the 2nd International Workshop on Ant Algorithms/ANTS2000, Lecture Notes in Computer Science*, 2000, 22~29
- 38 Kaji T. Approach by ant tabu agents for traveling salesman problem. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2001, 5: 3429~3434
- 39 Maniezzo V, Colorni A, Dorigo M. The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, IRIDIA, Université Libre de Bruxelles, Belgium, 1994

- 40 Gambardella L M, Taillard È D, Dorigo M. Ant colonies for the QAP. Technical Report IDSIA-4-97, IDSIA, Lugano, Switzerland, 1997
- 41 Gambardella L M, Taillard È D, Dorigo M. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 1999, 50(2): 167~176
- 42 Stützle T, Hoos H. MAX-MIN ant system for the quadratic assignment problem. Technical Report AI-DA-97-4, FG Intellektik, TH Darmstadt, July 1997
- 43 Stützle T, Hoos H. MAX-MIN ant system and local search for combinatorial optimization problems. *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic, Boston, MA, 1999, 313~329
- 44 Maniezzo V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Technical Report CSR 98-1, Scienze dell'Informazione, Università di Bologna, Sede di Cesena, Italy, 1998
- 45 Maniezzo V, Colorni A. The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering*, 1999, 11(5): 769~778
- 46 Colorni A, Dorigo M, Maniezzo V, *et al.* Ant system for job-shop scheduling. *Belgian J. Oper. Res. Statist. Comput. Sci.* 1994, 34: 39~53
- 47 Stützle T. An ant approach to the flow shop problem. Technical Report AIDA-97-07, FG Intellektik, FB Informatik, TH Darmstadt, September 1997
- 48 Besten M den, Stützle T, Dorigo M. Scheduling single machines by ants. Technical Report IRIDIA/99-16, IRIDIA, Université Libre de Bruxelles, Belgium, 1999
- 49 Bullnheimer B, Hartl R F, Strauss C. An improved ant system algorithm for the vehicle routing problem. Technical Report POM-10/97, Institute of Management Science, University of Vienna, Austria, 1997, Ann. Oper. Res. 89, 1999
- 50 Bullnheimer B, Hartl R F, Strauss C. Applying the ant system to the vehicle routing problem. *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, Boston, MA, 1999, 285~296
- 51 Gambardella L M, Taillard È, Agazzi G. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. *New Ideas in Optimization*, McGraw-Hill, London, UK, 1999, 63~76
- 52 Schoonderwoerd R, Holland O, Bruton J. Ant-like agents for load balancing in telecommunications networks. *Proceedings of the 1st International Conference on Autonomous Agents*, 1997, 209~216
- 53 Schoonderwoerd R, Holland O, Bruton J, *et al.* Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 1996, 5(2):169~207
- 54 White T, Pagurek B, Oppacher F. Connection management using adaptive mobile agents. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 1998, 802~809
- 55 Di Caro G, Dorigo M. Extending AntNet for best-effort quality-of-service routing. *Proceedings of the 1st International Workshop on Ant Colony Optimization*, 1998
- 56 Bonabeau E, Henaux F, Guérin S, *et al.* Routting in telecommunication networks with “smart” ant-like agents. *Proceedings of the 2nd International Workshop on Intelligent Agents for Telecommunication Applications*, Lectures Notes in Artificial Intelligence, vol. 1437, 1998
- 57 Di Caro G, Dorigo M. AntNet: a mobile agents approach to adaptive routing Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Belgium, 1997

- 58 Di Caro G, Dorigo M. AntNet: distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 1998, 9(2): 317~365
- 59 Di Caro G, Dorigo M. Two ant colony algorithms for best-effort routing in datagram networks. *Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Systems*, 1998, 541~546
- 60 Subramanian D, Druschel P, Chen J. Ants and reinforcement learning: a case study in routing in dynamic networks. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1997, 832~838
- 61 Heusse M, Guérin S, Snyers D, *et al.* Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1998, 1: 234~257
- 62 van der Put R. Routing in the fax factory using mobile agents. *Technical Report R&D-SV-98-276*, KPN Research, The Netherlands, 1998
- 63 van der Put R. *Routing in packet switched networks using agents*. Master of Science thesis, Delft University of Technology, 1998
- 64 Gambardella L M, Dorigo M. HAS-SOP: a hybrid ant system for the sequential ordering problem. *Technical Report IDSIA-11-97*, IDSIA, Lugano, Switzerland, 1997
- 65 Costa D, Hertz A. Ants can colour graphs. *Journal of the Operational Research Society*, 1997, 48(3): 295~305
- 66 Ahn S H, Lee S G, Chung T C. Modified ant colony system for coloring graphs. *Proceedings of the 2003 Joint Conference of the 4th International Conference on Information, Communications and Signal Processing and the 4th Pacific Rim Conference on Multimedia*, 2003, 3: 1849~1853
- 67 Michel R, Middendorf M. An island model based ant system with look ahead for the shortest supersequence problem. *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, 1998, 692~701
- 68 Michel R, Middendorf M. An ACO algorithm for the shortest supersequence problem. *New Ideas in Optimization*, McGraw-Hill, London, UK, 1999, 51~61
- 69 Maniezzo V, Carbonaro A. An ANTS heuristic for the frequency assignment problem. *Technical Report CSR 98-4*, Scienze dell' Informazione, Università di Bologna, Sede di Cesena, Italy, 1998
- 70 Maniezzo V, Carbonaro A. An ANTS heuristic for the frequency assignment problem. *Future Generation Computer Systems*, 2000, 16(8): 927~935
- 71 Ramalhinho Lourenco H, Serra D. Adaptive approach heuristics for the generalized assignment problem. *Technical Report EWP Series No. 304*, Department of Economics and Management, Universitat Pompeu Fabra, Barcelona, 1998
- 72 Leguizamón G, Michalewicz Z. A new version of ant system for subset problems. *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999, 1459~1464
- 73 Parra-Hernandez R, Dimopoulos N. On the performance of the ant colony system for solving the multi-dimensional knapsack problem. *Proceedings of the 2003 IEEE Pacific Rim Conference on Communications, Computers and signal Processing*, 2003, 1: 338~341
- 74 Navarro Varela G, Sinclair M C. Ant colony optimization for virtual-wavelength-path routing and wavelength allocation. *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999, 1809~1816
- 75 Liang Y C, Smith A E. An ant system approach to redundancy allocation. *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999, 2: 1478~1484
- 76 Liang Y C, Smith A E. An ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Transactions on Reliability*, 2004, 53(3): 417~423

- 77 Bauer A, Bullnheimer B, Hartle R F, *et al.* An ant colony optimization approach for the single machine total tardiness problem. Proceedings of the 1999 Congress on Evolutionary Computation, 1999, 1445~1450
- 78 Chang C S, Tian L, Wen F S. New approach to fault section estimation in power systems using ant system. Electric Power Systems Research, 1999, 49(1): 63~70
- 79 孙京诰, 李秋艳, 杨欣斌等. 基于蚁群算法的故障识别. 华东理工大学学报, 2004, 30(2): 194~198
- 80 Schoofs L, Naudts B. Ant colonies are good at solving constraint satisfaction problems. Proceedings of the 2000 Congress on Evolutionary Computation, 2000, 2: 1190~1195
- 81 Solnon C. Ants can solve constraint satisfaction problems. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 347~357
- 82 Fu S G, Dozier G. Solving distributed constraint satisfaction problems with an ant-like society of hill-climbers. Proceedings of the International Conference on Artificial Intelligence, 2003, 1: 263~269
- 83 Mertens K, Holvoet T. CSAA: a distributed ant algorithm framework for constraint satisfaction. Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference, 2004, 2: 764~769
- 84 Mathur M, Karale S B, Priye S, *et al.* Ant colony approach to continuous function optimization. Industrial and Engineering Chemistry Research, 2000, 39(10): 3814~3822
- 85 魏平, 熊伟清. 用于一般函数优化的蚁群算法. 宁波大学学报, 2001, 14(4): 52~55
- 86 Camara D, Loureiro Antonio A F. A GPS/ant-like routing algorithm for ad hoc networks. Proceedings of the IEEE Wireless Communications and Networking Conference, 2000, 1232~1236
- 87 Saleh H Z. GPS positioning networks design: an application of the ant colony system. Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science, 2002, 2463: 302~303
- 88 De A Silva R M, Ramalho G L. Ant system for the set covering problem. Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics, 2001, 5: 3129~3133
- 89 Wang L, Wu Q D. Linear system parameters identification based on ant system algorithm. Proceedings of the IEEE Conference on Control Applications, 2001, 401~406
- 90 Abbaspour K C, Schulin R, van Genuchten M Th. Estimating unsaturated soil hydraulic parameters using ant colony optimization. Advances in Water Resources, 2001, 24(8): 827~841
- 91 汪镭, 吴启迪. 蚁群算法在系统辨识中的应用. 自动化学报, 2003, 29(1): 102~109
- 92 Duan H B, Wang D B, Zhu J Q, *et al.* Parameter identification of LuGre friction model for flight simulation servo system based on ant colony algorithm. Transactions of Nanjing University of Aeronautics & Astronautics, 2004, 21(3): 179~183
- 93 金飞虎, 洪炳熔, 高庆吉. 基于蚁群算法的自由飞行空间机器人路径规划. 机器人, 2002, 24(6): 526~529
- 94 Parpinelli R S, Lopes H S, Freitas A A. Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 321~332
- 95 Meshoul S, Batouche M. Ant colony system with extremal dynamics for point matching and pose estimation. Proceedings of the 16th International Conference on Pattern Recognition, 2002, 3: 823~826
- 96 Zheng H, Wong A, Nahavandi S. Hybrid ant colony algorithm for texture classification. Proceedings of the 2003 Congress on Evolutionary Computation, 2003, 4: 2648~2652
- 97 Lee Z J, Lee C Y, Su S F. An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem. Applied Soft Computing, 2002, 2(1): 39~47

- 98 Shmygelska A, Aguine-Hemundez R, Hoos H H. An ant colony algorithm for the 2D HP protein folding problem. Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science, 2002, 2463: 40~52
- 99 Kumar V, Sahin F. A swarm intelligence based approach to the mine detection problem. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2002, 3: 65~70
- 100 Kumar V, Sahin F. Foraging in ant colonies applied to the mine detection problem. Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications, 2003, 61~66
- 101 Chapman E, Sahin F. Application of swarm intelligence to the mine detection problem. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2004, 6: 5429~5434
- 102 陈昌富, 谢学斌. 露天采场边坡临界滑动面搜索蚁群算法研究. 湘潭矿业学院学报, 2002, 17(1): 62~64
- 103 高玮, 郑颖人. 蚁群算法及其在硐群施工优化中的应用. 岩石力学与工程学报, 2002, 21(4): 471~474
- 104 丁亚平, 刘平阳, 苏庆德等. 化学蚁群算法及其在光谱解析中的应用. 计算机与应用化学, 2002, 19(3): 326~328
- 105 丁亚平, 苏庆德, 吴庆生. 化学蚁群算法与多组分导数荧光光谱解析. 高等学校化学学报, 2002, 23(9): 1695~1697
- 106 Cincotti A, Cutello V, Pappalardo F. An ant-algorithm for the weighted minimum hitting set problem. Proceedings of the IEEE Swarm Intelligence Symposium, 2003, 1~5
- 107 Li Y M, Xu Z B. An ant colony optimization heuristic for solving maximum independent set problems. Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications, 2003, 206~211
- 108 Korosec P, Silc J, Robic B. Solving the mesh-partitioning problem with an ant-colony algorithm. Parallel Computing, 2004, 30(5-6): 785~801
- 109 段海滨, 王道波, 于秀芬等. 基于蚁群算法的 PID 参数优化. 武汉大学学报, 2004, 37(5): 97~100
- 110 曹春红, 卢奕南, 李文辉. 改进的蚂蚁算法在几何约束求解中的应用. 工程图学学报, 2004, 25(4): 46~50
- 111 曹春红, 李文辉, 张永坚. 遗传蚂蚁算法在几何约束求解中的应用. 仪器仪表学报, 2004, 25(4): 393~396
- 112 高尚, 杨静宇, 吴小俊等. 圆排列问题的蚁群模拟退火算法. 系统工程理论与实践, 2004, (8): 102~106
- 113 Hua Z, Fan H, Li J J, et al. Solving point covering problem by ant algorithm. Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, 2004, 6: 3501~3504
- 114 Jensen R, Shen Q. Fuzzy-rough data reduction with ant colony optimization. Fuzzy Sets and Systems, 2005, 149(1): 5~20

## 第 2 章 基本蚁群算法原理及其复杂度分析

### 2.1 引言

蚁群算法最早成功应用于解决著名的 TSP，该算法采用了分布式并行计算机制，易于与其他方法结合，而且具有较强的鲁棒性<sup>[1~4]</sup>，但搜索时间长、易陷于局部最优解是其最为突出的缺点。

本章首先从深层意义上进一步研究了蚁群算法的行为特征，给出了蚁群寻优的逻辑结构，然后分析了基本蚁群算法在分布式计算、自组织、正反馈等方面系统学特征，随后在阐述 P、NP、NP-C、NP-hard 问题的基础上，给出了这四类问题之间的逻辑关系；紧接着以对称 TSP 为背景，对基本蚁群算法的数学模型进行了深入分析，并给出了该算法的具体实现步骤、程序结构框架以及基于不同开发环境的软件实现；随后在引入算法复杂度若干概念的基础上，对基本蚁群算法的空间复杂度和时间复杂度分别做了分析，进而提出了基本蚁群算法的三个性能评价指标，即最佳性能指标、时间性能指标和鲁棒性能指标，其综合性能指标则为这三个性能评价指标的加权组合。

### 2.2 基本蚁群算法的原理

#### 2.2.1 蚁群行为描述

根据仿生学家的长期研究发现：蚂蚁虽没有视觉，但运动时会通过在路径上释放出一种特殊的分泌物——信息素来寻找路径<sup>[5, 6]</sup>。当它们碰到一个还没有走过的路口时，就随机地挑选一条路径前行，同时释放出与路径长度有关的信息素。蚂蚁走的路径越长，则释放的信息量越小。当后来的蚂蚁再次碰到这个路口的时候，选择信息量较大路径的概率相对较大，这样便形成了一个正反馈机制。最优路径上的信息量越来越大，而其他路径上的信息量却会随着时间的流逝而逐渐消减，最终整个蚁群会找出最优路径。同时蚁群还能够适应环境的变化，当蚁群的运动路径上突然出现障碍物时，蚂蚁也能很快地重新找到最优路径。可见，在整个寻径过程中，虽然单只蚂蚁的选择能力有限，但是通过信息素的作用使整个蚁群行为具有非常高的自组织性，蚂蚁之间交换着路径信息，最终通过蚁群的集体自催化行为找出最优路径。这里用如图 2.1 所示的形象图例来进一步说明蚁群的搜索原理。

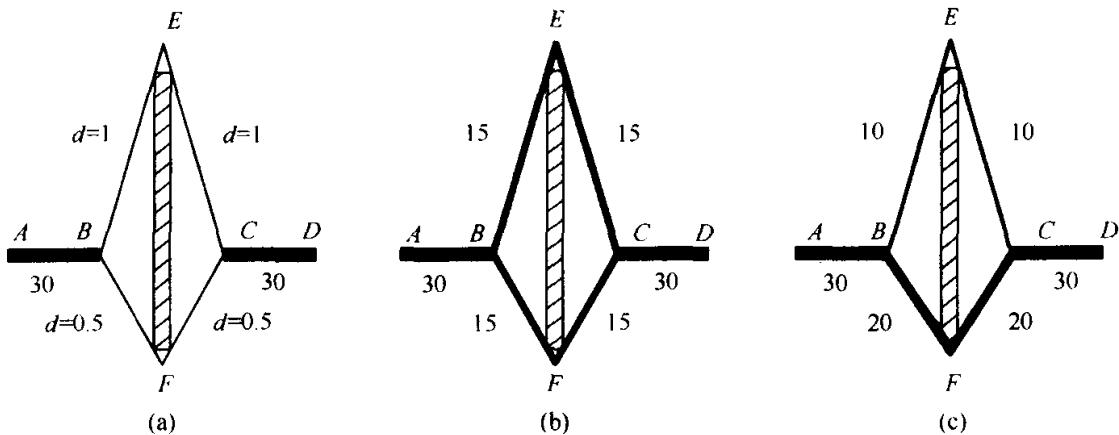


图 2.1 自然界中的蚂蚁觅食模拟

图 2.1 中, 设  $A$  点是蚁巢,  $D$  点是食物源,  $EF$  为一障碍物。由于障碍物的存在, 蚂蚁只能经由  $A$  经  $E$  或  $F$  到达  $D$ , 或由  $D$  到达  $A$ , 各点之间的距离如图 2.1(a)所示。假设每个时间单位有 30 只蚂蚁由  $A$  到达  $D$  点, 有 30 只蚂蚁由  $D$  到达  $A$  点, 蚂蚁过后留下的信息量为 1。为了方便起见, 设该物质停留时间为 1。在初始时刻, 由于路径  $BF$ 、 $FC$ 、 $BE$ 、 $EC$  上均无信息存在, 位于  $A$  和  $D$  的蚂蚁可以随机选择路径, 从统计学的角度可以认为蚂蚁以相同概率选择  $BF$ 、 $FC$ 、 $BE$ 、 $EC$ , 如图 2.1(b)所示。经过一个时间单位后, 在路径  $BFC$  上的信息量是路径  $BEC$  上信息量的 2 倍。又经过一段时间, 将有 20 只蚂蚁由  $B$ 、 $F$  和  $C$  点到达  $D$ , 如图 2.1(c)所示。随着时间的推移, 蚂蚁将会以越来越大的概率选择路径  $BFC$ , 最终将会完全选择路径  $BFC$ , 从而找到由蚁巢到食物源的最短路径。

## 2.2.2 基本蚁群算法的机制原理

模拟蚂蚁群体觅食行为的蚁群算法是作为一种新的计算智能模式引入的, 该算法基于如下基本假设:

- (1) 蚂蚁之间通过信息素和环境进行通信。每只蚂蚁仅根据其周围的局部环境做出反应, 也只对其周围的局部环境产生影响。
- (2) 蚂蚁对环境的反应由其内部模式决定。因为蚂蚁是基因生物, 蚂蚁的行为实际上是其基因的适应性表现, 即蚂蚁是反应型适应性主体。
- (3) 在个体水平上, 每只蚂蚁仅根据环境做出独立选择; 在群体水平上, 单只蚂蚁的行为是随机的, 但蚁群可通过自组织过程形成高度有序的群体行为。

由上述假设和分析可见, 基本蚁群算法的寻优机制包含两个基本阶段: 适应阶段和协作阶段。在适应阶段, 各候选解根据积累的信息不断调整自身结构, 路

径上经过的蚂蚁越多，信息量越大，则该路径越容易被选择；时间越长，信息量会越小<sup>[7~14]</sup>；在协作阶段，候选解之间通过信息交流，以期望产生性能更好的解，类似于学习自动机的学习机制。

蚁群算法实际上是一类智能多主体系统，其自组织机制使得蚁群算法不需要对所求问题的每一方面都有详尽的认识。自组织本质上是蚁群算法机制在没有外界作用下使系统熵增加的动态过程，体现了从无序到有序的动态演化<sup>[15]</sup>，其逻辑结构如图 2.2 所示。

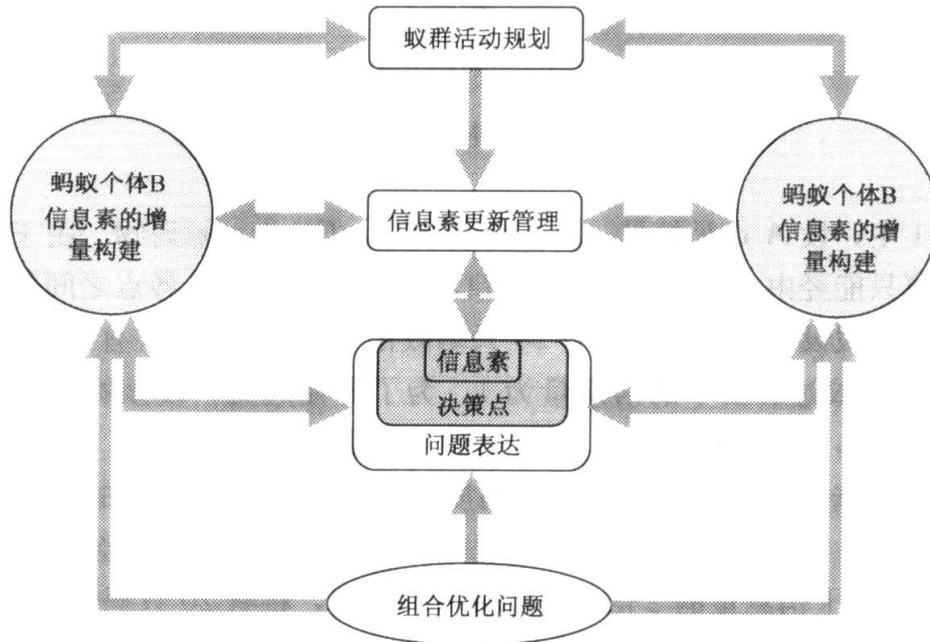


图 2.2 基本蚁群算法的逻辑结构

由图 2.2 可见，先将具体的组合优化问题表述成规范的格式，然后利用蚁群算法在“探索（exploration）”和“利用（exploitation）”之间根据信息素这一反馈载体确定决策点，同时按照相应的信息素更新规则对每只蚂蚁个体的信息素进行增量构建，随后从整体角度规划出蚁群活动的行为方向，周而复始，即可求出组合优化问题的最优解。

## 2.3 基本蚁群算法的系统学特征

### 2.3.1 基本蚁群算法是一个系统

系统学是一门较新的学科，其基本特点是强调整体性。系统学十分庞大，由于不同学科的研究范围和侧重点不同，往往对“系统”给出了不同的定义。在基础科学层次，通常采用系统学创始人 Bertalanffy L V 所给出的定义：“系统可以

确定为处于一定的相互关系中并与环境发生关系的各组成部分（要素）的综合体”。这个定义强调的不是功能，而是系统元素之间的相互作用以及系统对元素的整体作用。该定义可更为精确化的表述为：如果对象  $S$  满足以下两个条件：

- (1)  $S$  中至少包含两个不同的对象。
- (2)  $S$  中的对象按照一定的方式互相联系在一起。

则称  $S$  为一个系统，同时定义  $S$  中的个体对象为系统的元素。

很显然，自然界中的蚁群具备了系统的三个基本特征，即多元性、相关性和整体性，从而构成了一个系统。在这个系统中，蚂蚁的个体行为是系统的元素，蚁群行为的相互影响体现了系统的相关性，而蚁群可以完成个体完成不了的任务则体现了系统的完整性，显示出系统整体大于部分之和的整体突现原理。

作为对蚁群觅食行为抽象的蚁群算法，如果把算法本身看做一个整体，就会发现它具备了系统的特征，这也是仿生优化算法最重要的特征之一<sup>[16]</sup>。对于一个传统的算法而言，比如说一个求无约束值的解析方法——牛顿法，由于算法各个组成部分完成的功能之和就等于算法最终完成的功能（具有加和性），所以不能看做一个系统。然而在基本蚁群算法中，采用多只蚂蚁的求解结果明显好于单只蚂蚁的求解结果，因此不具有加和性，整体大于部分之和，因此基本蚁群算法是一个系统。

### 2.3.2 分布式计算

将基本蚁群算法作为一个系统来研究，它还具备一些更为重要的特征，使它具有了系统学上更加深刻的意义。

生命系统是一个分布式系统，它使得生命体具有强的适应能力。比如说人体的很多细胞相互独立完成同一项工作，当一个细胞停止工作或新陈代谢之后，整体的功能不会因此而受到影响。这就是分布式带来的强适应能力，它依赖于个体行为，但并不单独依赖于每一个体的行为，生命系统适应能力的形象说明如图 2.3 所示。

图 2.3(a) 是一个由非分布式行为形成的链，其中  $A$ 、 $B$ 、 $C$  三个过程出现任何差错都得不到最终的结果；与之相对应的图 2.3(b) 采用了分布式行为形成的链，

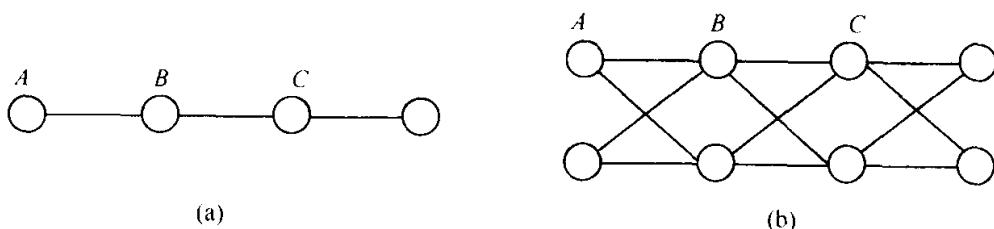


图 2.3 生命系统适应能力的形象说明

当其中的一个连接出现错误中断时，不会影响最终的结果。由图 2.3 可知，分布式系统需要有很多个体来完成同样的过程，即需要个体行为的冗余。

仔细分析可以发现，自然界中的真实蚁群行为也出现了分布式。当蚁群需要完成某项工作时，其中的许多蚂蚁都为共同目的进行着同样的工作，而最终任务的完成不会由于某些个体的缺陷而受到影响。

蚁群算法作为对蚁群觅食行为的抽象，也体现了群体行为的分布式特征。每只人工蚂蚁在问题空间的多个点同时开始相互独立地构造问题解，而整个问题的求解不会因为某只人工蚂蚁无法成功获得解而受到影响。

具体到不同的优化问题而言，蚁群算法体现出的分布式特征就具有了更加现实的意义。因为所有的仿生优化算法均可看做按照一定规则在问题解空间搜索最优解的过程，所以搜索点的初始选取就直接关系到算法求解结果的优劣和算法寻优的效率。当求解许多复杂问题时，从一点出发的搜索受到局部特征的限制，可能得不到所求问题的满意解；而基本蚁群算法则可看做是一个分布式的多智能体系统，它在问题空间的多点同时独立地进行解搜索，不仅使得算法具有较强的全局搜索能力，也增加了算法的可靠性。

### 2.3.3 自组织

基本蚁群算法的另一个重要特征是自组织，这也是遗传算法、人工神经网络、微粒群算法、人工免疫算法、人工鱼群算法等仿生优化算法的共有特征。

自组织的概念是随着系统科学的发展而建立起来的。通常把系统论中的组织行为分为自组织和他组织两大类。其根本区别在于组织力或组织指令是来自于系统内部还是来自于系统外部，前者称为自组织，而后者即为他组织。如果系统在获得时间的、空间的或者功能的结构过程中，没有外界的特定干预，则称该系统是自组织的。

最典型的自组织就是生物体，在生物学上有一个观点，即类似蚂蚁、蜜蜂这样的昆虫，由于个体作用简单，而个体之间的协作作用特别明显，因而可把它们当做一个整体来研究，甚至把它们看做一个独立的生物体。在这样的群落中，生物个体相互作用，协同完成某项群体工作，自然体现出很强的自组织特性。

从抽象意义上讲，自组织就是在没有外界作用下使得系统熵增加的过程（即系统从无序到有序的进化过程）。基本蚁群算法体现了这一过程，以蚁群优化为例，当算法开始的初期，单只人工蚂蚁无序地寻找解，经过一段时间的算法演化，人工蚂蚁越来越趋向于寻找到接近最优解的一些解，这恰恰体现了从无序到有序的自组织进化。

自组织大大增强了算法的鲁棒性，传统的算法都是针对某一具体问题而设计的，这往往建立在对该问题有了全面清晰认识的基础上，通常很难适应其他问

题。而自组织的蚁群算法不需要对待求解问题的所有方面都有所认识，因而较容易应用到一类问题中。

### 2.3.4 正反馈

反馈是控制论中的重要概念，它代表信息输入对输出的反作用。在系统学上认为，反馈就是把系统现在的行为作为影响系统未来行为的原因。反馈分正反馈和负反馈两种，前者指的是以现在的行为去加强未来的行为，而后者则是以现在的行为去削弱未来的行为。

由自然界中真实蚂蚁的觅食行为可见，蚂蚁能够最终找到最短路径，直接依赖于最短路径上信息素的堆积，而信息素的堆积却是一个正反馈的过程。对于基本蚁群算法而言，初始时在环境中存在完全相同的信息量，给系统一个微小的扰动，使得各个边上的信息量大小不同，蚂蚁构造的解就存在了优劣。算法采用的反馈方式是在较优解经过的路径留下更多的信息素，更多的信息素又吸引了更多的蚂蚁，这个正反馈的过程使得初始值不断地扩大，同时又引导整个系统向着最优解的方向进化。

单一的正反馈或者负反馈存在于线性系统之中，是无法实现系统的自我组织的。自组织系统通过正反馈和负反馈的结合，以实现系统的自我创造和自我更新。基本蚁群算法中同样隐含着负反馈机制，它体现于蚁群算法在构造问题解的过程中用到了概率搜索技术，通过该技术增加了生成解的随机性。随机性的影响就在于接受了解在一定程度上的退化，另一方面又使得搜索范围得以在一段时间内保持足够大。这样正反馈缩小搜索范围，保证算法朝着最优解的方向进化；而负反馈保持搜索范围，避免算法过早收敛于不好的结果。恰恰是在正反馈和负反馈共同作用的影响下，基本蚁群算法得以自组织地进化，从而得到问题在一定程度上的满意解。值得一提的是，在已经公开发表的大部分论文中，对于基本蚁群算法所隐含的负反馈机理没有提及，但这并不影响对基本蚁群算法本质特性的理解以及对基本蚁群算法的改进和广泛应用。

以上从系统学的角度进一步分析了蚁群算法的机理，可见蚁群算法体现出许多不同于常规算法的新思想，这也正是基本蚁群算法在系统学上的研究意义所在。

## 2.4 基本蚁群算法的数学模型

### 2.4.1 P、NP、NP-C、NP-hard 问题描述<sup>[17, 18]</sup>

NP-C问题是描述基本蚁群算法的数学模型和对其复杂度进行分析的基础，这里首先引入几个基本概念。

**定义 2.4.1 图灵机** 图灵机概念最早由英国数学家 Turing A 提出，其本质上是一个具有序列存储载体的、按照具体指令可完成左或右移动、放置标记、抹去标记及在计算终止时停机等四种基本操作的、用于描述算法特性的语言。图灵机是对算法进行分析和研究算法复杂度的得力工具，可分为确定性单带图灵机 (deterministic one-tape Turing machine, DTM) 和非确定性单带图灵机 (non-deterministic one-tape Turing machine, NDTM) 两大类。

一个 DTM 是由一个有限状态控制器、一个读写头和一个双向的、具有无限多个带格的线性带所组成，其基本原理如图 2.4 所示。

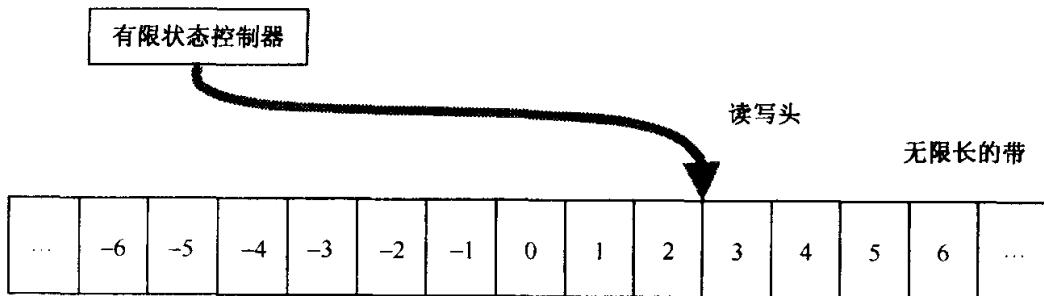


图 2.4 DTM 示意图

一个 DTM 程序应包括以下信息：

- (1) 线性带中所用字符的一个有限集合  $\Gamma_0$ ，它包括输入字符表子集  $\Sigma_0 \subset \Gamma_0$  和一个特别的空白符号  $b \in \Gamma_0 - \Sigma_0$ 。
- (2) 一个有限状态集  $Q'_0$ ，它包括初始状态  $q_0$  和两个特有的停机状态  $q_Y$  或  $q_N$ 。
- (3) 一个转移函数  $\delta': (Q'_0 - \{q_Y, q_N\}) \times \Gamma_0 \rightarrow Q'_0 \times \Gamma_0 \times \{l, r\}$ 。

假设对 DTM 的输入为字符串  $x \in \Sigma_0$ ，则该字符串首先被一格一个字符地顺序存放在带格 1 到带格  $|x|$  中，所有其他带格开始时存放的均为空白符。该程序从初始状态  $q_0$  开始它的运算，并且读写头先位于带格 1。然后算法按照下述规则进行：若当前状态  $q$  为  $q_Y$  或  $q_N$ ，则算法终止；且若  $q = q_Y$ ，就回答“是”，否则回答“非”。若当前状态  $q_0 \in Q'_0 - \{q_Y, q_N\}$ ，且  $s \in \Gamma$  为读写头当前扫描的带格中的字符，而转移函数此时对应的取值为  $\delta(q, s) = (q', s', \Delta)$ ，那么，该程序将执行这样的几个操作：读写头抹去当前带格中的  $s$ ，代之以  $s'$ ；同时若  $\Delta = l$ ，则读写头左移一格，若  $\Delta = r$ ，则右移一格；最后，有限状态控制器将从状态  $q$  变为状态  $q'$ 。这样就完成了程序的一步计算，并为下一步计算做好了准备，除非已处于停机状态。

实际上，许多判定问题都具有多项式时间可验证性，但是多项式的时间可验证性并不意味着多项式的时间可解性。这是由于没有考虑为了找出一个解的猜测所花去的时间，而这常常需要在可能有指数多个猜测的集合中选取一个较合理的

猜测，NDTM 恰恰可以刻画这一过程。

NDTM 完全是一种假想的机器，通常多用猜想模块模型来对其进行描述。猜想模块带有自己只对带写的猜想头，它提供了写下猜想的办法，并仅用于此目的。其机理如图 2.5 所示。

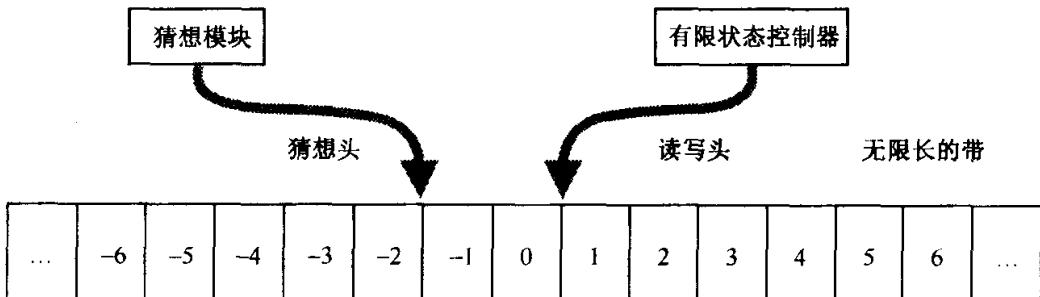


图 2.5 NDTM 示意图

NDTM 和 DTM 的不同之处在于前者把计算分成两个不同的阶段：猜想阶段和检验阶段。在猜想阶段，一开始输入的字符串被写在带中格 1 到格  $|x|$  的带格中，其余带格均为空白字符。读写头在扫描带格 1，而猜想头在扫描带格  $-1$ 。有限状态控制器处于不起作用的状态，猜想模块处于起作用的状态，并一步一步地指示猜想头：要么在正被扫描的带格中写下某一字符并左移一格，要么停止。若停止，则此时猜想模块便不起作用，而有限状态控制器开始起作用。猜想模块是否保持起作用以及若起作用则从带中字符集中选择哪个字符等均由猜想模块以某一完全任意的方式来决定。当有限控制器起作用时，检验阶段就开始了。从此刻起，算法将在该 NDTM 程序的指示下，按照与 DTM 程序完全相同的规则进行，而猜想模块及其猜想头在完成了将所猜字符串写到带上的任务后将不再参与程序的执行。同时，在检验阶段，前面所猜的字符串常被考察。当有限状态控制器进入两个停机状态之一时，计算过程即停止。

**定义 2.4.2 实例** 实例是问题的特殊表现，是确定了描述问题特性的所有参数的问题，其中参数值称为数据，这些数据占用计算机的空间称为数据实例的输入长度。

**定义 2.4.3 P 类问题** 所有可用 DTM 在多项式时间内求解的判定问题 II 的集合，简记为  $O(p(n))$ 。即  $P = \{L : \text{存在一个多项式时间 DTM 程序 } M, \text{ 使得 } L = L_M\}$ ，其中  $L_M$  表示程序  $M$  所识别的语言。若存在一个多项式时间 DTM 程序，它在编码策略  $e$  之下求解判定问题 II，即  $L[\Pi, e] \in P$ ，则称该判定问题 II 属于 P 类问题。P 类问题的每个实例只有“是”或“否”两种回答，并称肯定回答的实例为“是”实例；称否定回答的实例为“否”实例或“非”实例。

**定义 2.4.4 NP (non-deterministic poly-nominal) 类问题** 若存在一个多项式函数  $g(x)$  和一个验证算法  $H$ ，对一类判定问题 A 的任何一个“是”回答，满

足其输入长度  $d(S)$  不超过  $g(d(I))$ , 其中  $d(I)$  为  $I$  的输入长度, 且验证算法中  $S$  为  $I$  的“是”回答的计算时间不超过  $g(d(I))$ , 则称判定问题  $A$  为非多项式确定问题, 简称 NP 类问题。NP 类问题是所有可用 NDTM 在多项式时间内求解的判定问题 II 的集合。

判定问题是否属于 NP 类问题的关键是对“是”的判定实例是否存在满足上述条件的一个字符串和算法, 其中字符串在此可理解为问题的一个解, 而定义中没有强调字符串和算法是如何得到的。能用 DTM 在多项式时间内解决的 P 类问题, 也一定能用 NDTM 在多项式时间内加以解决, 这个关系可表示为:  $P \subseteq NP$ 。

然而, 在当前的计算复杂度理论中, 尚没有解决是否兼属 P 类或 NP 类的问题, 即 P 类与 NP 类交集是否为空的问题。目前证明  $P \neq NP$  的难度和证明  $P = NP$  的难度同样大。

归纳和转换是描述问题特性的常用方法, 若能将几类问题归结为一个问题, 这样一旦解决了一类归结后的问题, 其他几类问题也就迎刃而解了。

**定义 2.4.5** 给定问题  $A_1$  和  $A_2$ , 若存在一个多项式函数  $g(x)$  和一个字符串满足:

- (1) 对于  $A_1$  的任何一个实例  $I_1$ , 在其输入长度的多项式时间  $g(d(I_1))$  内构造  $A_2$  的一个实例  $I_2$ , 使其长度不超过  $g(d(I_1))$ ;
- (2) 由此构造使得实例  $I_1$  和  $I_2$  的解一一对应, 且  $d_1$  为  $I_1$  的“是”回答的充要条件为  $d_1$  对应的解是  $I_2$  的一个“是”回答。

则称  $A_1$  可以转化为  $A_2$ 。

**定义 2.4.6** 给定判定问题  $A_1$  和  $A_2$ , 若存在多项式函数  $g_1(x)$  和  $g_2(x)$ , 使得对  $A_1$  的任何一个实例  $I$ , 在多项式时间内构造  $A_2$  的一个实例, 其输入长度不超过  $g_1(d(I))$ , 并对  $A_2$  的任何一个算法  $H_2$ , 都存在问题  $A_1$  的一个算法  $H_1$ , 使得  $H_1$  调用  $H_2$  且计算时间满足

$$f_{H_1}(d(I)) \leq g_2(f_{H_1}(g_1(d(I)))) \quad (2.4.1)$$

则称  $A_1$  可多项式归纳为  $A_2$ 。

由此可见, 若问题  $A_2$  存在多项式时间算法, 则问题  $A_1$  一定存在多项式时间算法。若问题  $A_1$  可多项式转换为问题  $A_2$ , 对  $A_2$  的一个算法  $H_2$ , 可以按照如下步骤构造  $A_1$  算法:

- (1) 对问题  $A_1$  的任何一个实例  $I_1$ , 先用多项式时间  $g_1(d(I_1))$  构造问题  $A_2$  的一个实例  $I_2$ ;
- (2) 调用算法  $A_2$  求解  $I_2$ , 此步计算为  $f_{H_2}(g_1(d(I_1)))$ 。

如此构造的算法对问题  $A_1$  的任何一个实例  $I_1$  的计算时间不超过  $g_1(d(I_1)) + f_{H_2}(g_1(d(I_1)))$ 。

**定义 2.4.7 NP-C (NP-complete) 类问题** 它是 NP 类中最困难的一类问题。所有的 NP-C 类问题是同等困难的, 每一个 NP 类问题都可以用多项式算法

转换至 NP-C。因此，如果 NP-C 类问题中有一个问题能用多项式确定性算法解决，则其他所有的 NP-C 类问题都能用多项式确定性算法来解决。

NP-C 类问题具有重要的实际意义和工程背景，许多问题被证明为 NP-C 类问题，如 TSP、QAP、VRP、JCP、PCP 等。此外，NP-C 类问题也是检验仿生优化算法有效性和可靠性的有效平台。

**定义 2.4.8** 若  $A \subseteq NP$ ，且 NP 类问题中的任何一个问题可多项式归约为问题  $A$ ，称判定问题  $A \subseteq NP-C$ ；只要上述第二个条件成立，称问题  $A$  为 NP-hard 类问题。

由此可见， $NP-C \subseteq NP-hard$ ，而且两者之间的区别仅在于 NP-C 必须判断问题属于 NP 类。同时，若已知一个问题为 NP-C 或 NP-hard，当遇到一个新问题时，若已知问题多项式归约为新问题，则新问题为 NP-hard 类问题，进而若可验证新问题属于 NP 类，则新问题为 NP-C 类问题。

由此，可得到 P、NP、NP-C 及 NP-hard 类问题之间的逻辑关系如图 2.6 所示。

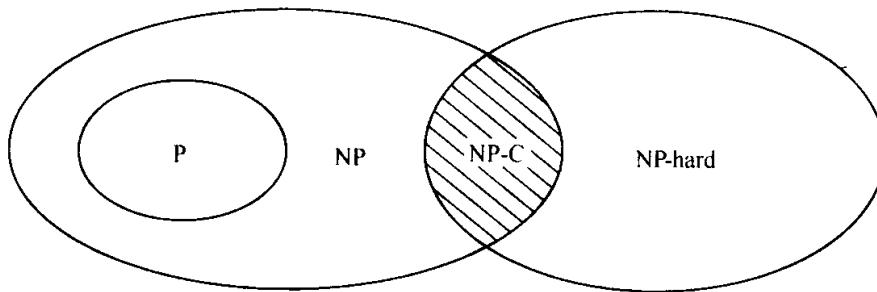


图 2.6 P、NP、NP-C 及 NP-hard 类问题之间的逻辑关系

## 2.4.2 基本蚁群算法的模型特征

### 2.4.2.1 TSP 描述

**定义 2.4.9 有向图** 给定一个有向图  $D$  的三元组为  $(V, E, f)$ ，其中  $V$  是一个非空集合，其元素称为有向图的结点； $E$  是一个集合，其元素称为有向图的弧段（边）； $f$  是从  $E$  到  $V \times V$  上的一个映射（函数）。

由定义 2.4.9 可知， $E$  中的元素总是和  $V$  中的序偶有对应关系，因此，可用  $V$  中的序偶代替  $E$  中的元素。一个有向图  $D$ ，可简记为  $(V, E)$ 。

**定义 2.4.10 TSP** 设  $C = \{c_1, c_2, \dots, c_n\}$  是  $n$  个城市的集合， $L = \{l_{ij} \mid c_i, c_j \in C\}$  是集合  $C$  中元素（城市）两两连接的集合， $d_{ij}$  ( $i, j = 1, 2, \dots, n$ ) 是  $l_{ij}$  的 Euclidean 距离，即

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2.4.2)$$

$G=(C, L)$  是一个有向图，TSP 的目的是从有向图  $G$  中寻出长度最短的 Hamilton 圈，此即一条对  $C=\{c_1, c_2, \dots, c_n\}$  中  $n$  个元素（城市）访问且只访问一次的最短封闭曲线。

TSP 的简单形象描述是：给定  $n$  个城市，有一个旅行商从某一城市出发，访问各城市一次且仅有一次后再回到原出发城市，要求找出一条最短的巡回路径。

TSP 可分为对称 TSP (symmetric traveling salesman problem) 和非对称 TSP (asymmetric traveling salesman problem) 两大类，若两城市往返的距离相同，则为对称 TSP，否则为非对称 TSP。本书若不特别说明，均指对称 TSP。

**定理 2.4.1** TSP 是 NP-C 问题<sup>[18]</sup>。

对于 TSP 解的任意一个猜想，若要检验它是否为最优，则需要将其与其他所有的可行遍历进行比较，而这些比较有指数多个，故根本不可能在多项式时间内对任何猜想进行检验。因此从本质上说，TSP 是一类被证明了的 NP-C 计算复杂度的组合优化难题，如果这一问题得到解决，则同一类型中的多个问题都可以迎刃而解。

TSP 的已知数据包括一个有限完全图中各条边的权重，其目标是寻找一个具有最小总权重的 Hamilton 圈。对于  $n$  个城市规模的 TSP，则存在  $\frac{(n-1)!}{2}$  条不同的闭合路径。求解该问题最完美的方法应该是全局搜索，但当  $n$  较大的时候，用全局搜索法精确地求出其最优解几乎不可能，这一点将在本章第 2.6 小节中进行详细分析。而 TSP 又具有广泛的代表意义和应用前景，许多现实问题均可抽象为 TSP 的求解。

#### 2.4.2.2 基本蚁群算法的数学模型<sup>[1~3, 19, 20]</sup>

设  $b_i(t)$  表示  $t$  时刻位于元素  $i$  的蚂蚁数目， $\tau_{ij}(t)$  为  $t$  时刻路径  $(i, j)$  上的信息量， $n$  表示 TSP 规模， $m$  为蚁群中蚂蚁的总数目，则  $m = \sum_{i=1}^n b_i(t)$ ； $\Gamma = \{\tau_{ij}(t) | c_i, c_j \subset C\}$  是  $t$  时刻集合  $C$  中元素（城市）两两连接  $l_{ij}$  上残留信息量的集合。在初始时刻各条路径上信息量相等，并设  $\tau_{ij}(0) = \text{const}$ ，基本蚁群算法的寻优是通过有向图  $g=(C, L, \Gamma)$  实现的。

蚂蚁  $k$  ( $k=1, 2, \dots, m$ ) 在运动过程中，根据各条路径上的信息量决定其转移方向。这里用禁忌表  $\text{tabu}_k$  ( $k=1, 2, \dots, m$ ) 来记录蚂蚁  $k$  当前所走过的城市，集合随着  $\text{tabu}_k$  进化过程作动态调整。在搜索过程中，蚂蚁根据各条路径上的信息量及路径的启发信息来计算状态转移概率。 $p_{ij}^k(t)$  表示在  $t$  时刻蚂蚁  $k$  由元素（城市） $i$  转移到元素（城市） $j$  的状态转移概率

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ik}(t)]^\beta}{\sum_{s \in \text{allowed}_k} [\tau_{is}(t)]^\alpha \cdot [\eta_{is}(t)]^\beta}, & \text{若 } j \in \text{allowed}_k \\ 0, & \text{否则} \end{cases} \quad (2.4.3)$$

式中,  $\text{allowed}_k = \{C - \text{tabu}_k\}$  表示蚂蚁  $k$  下一步允许选择的城市;  $\alpha$  为信息启发式因子, 表示轨迹的相对重要性, 反映了蚂蚁在运动过程中所积累的信息在蚂蚁运动时所起的作用, 其值越大, 则该蚂蚁越倾向于选择其他蚂蚁经过的路径, 蚂蚁之间协作性越强;  $\beta$  为期望启发式因子, 表示能见度的相对重要性, 反映了蚂蚁在运动过程中启发信息在蚂蚁选择路径中的受重视程度, 其值越大, 则该状态转移概率越接近于贪心规则;  $\eta_{ij}(t)$  为启发函数, 其表达式如下

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \quad (2.4.4)$$

式中,  $d_{ij}$  表示相邻两个城市之间的距离。对蚂蚁  $k$  而言,  $d_{ij}$  越小, 则  $\eta_{ij}(t)$  越大,  $p_{ij}^k(t)$  也就越大。显然, 该启发函数表示蚂蚁从元素(城市)  $i$  转移到元素(城市)  $j$  的期望程度。

为了避免残留信息素过多引起残留信息淹没启发信息, 在每只蚂蚁走完一步或者完成对所有  $n$  个城市的遍历(也即一个循环结束)后, 要对残留信息进行更新处理。这种更新策略模仿了人类大脑记忆的特点, 在新信息不断存入大脑的同时, 存储在大脑中的旧信息随着时间的推移逐渐淡化, 甚至忘记。由此,  $t+n$  时刻在路径  $(i, j)$  上的信息量可按如下规则进行调整

$$\tau_{ij}(t+n) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (2.4.5)$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij}^k(t) \quad (2.4.6)$$

式中,  $\rho$  表示信息素挥发系数, 则  $1-\rho$  表示信息素残留因子, 为了防止信息的无限积累,  $\rho$  的取值范围为:  $\rho \in [0, 1]$ ;  $\Delta\tau_{ij}(t)$  表示本次循环中路径  $(i, j)$  上的信息素增量, 初始时刻  $\Delta\tau_{ij}(0)=0$ ,  $\Delta\tau_{ij}^k(t)$  表示第  $k$  只蚂蚁在本次循环中留在路径  $(i, j)$  上的信息量。

根据信息素更新策略的不同, Dorigo M 提出了三种不同的基本蚁群算法模型, 分别称之为 Ant-Cycle 模型、Ant-Quantity 模型及 Ant-Density 模型, 其差别在于  $\Delta\tau_{ij}^k(t)$  求法的不同。

在 Ant-Cycle 模型中

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L_k}, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (2.4.7)$$

式中,  $Q$  表示信息素强度, 它在一定程度上影响算法的收敛速度;  $L_k$  表示第  $k$  只蚂蚁在本次循环中所走路径的总长度。

在 Ant-Quantity 模型中

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{d_{ij}}, & \text{若第 } k \text{ 只蚂蚁在 } t \text{ 和 } t+1 \text{ 之间经过 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (2.4.8)$$

在 Ant-Density 模型中

$$\Delta\tau_{ij}^k(t) = \begin{cases} Q, & \text{若第 } k \text{ 只蚂蚁在 } t \text{ 和 } t+1 \text{ 之间经过 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (2.4.9)$$

区别：式 (2.4.8) 和式 (2.4.9) 中利用的是局部信息，即蚂蚁完成一步后更新路径上的信息素；而式 (2.4.7) 中利用的是整体信息，即蚂蚁完成一个循环后更新所有路径上的信息素，在求解 TSP 时性能较好，因此通常采用式 (2.4.7) 作为蚁群算法的基本模型。

## 2.5 基本蚁群算法的具体实现

### 2.5.1 基本蚁群算法的实现步骤

以 TSP 为例，基本蚁群算法的具体实现步骤如下：

- (1) 参数初始化。令时间  $t=0$  和循环次数  $N_c=0$ ，设置最大循环次数  $N_{c_{\max}}$ ，将  $m$  蚂蚁置于  $n$  个元素（城市）上，令有向图上每条边  $(i, j)$  的初始化信息量  $\tau_{ij}(t)=\text{const}$ ，其中 const 表示常数，且初始时刻  $\Delta\tau_{ij}(0)=0$ 。
- (2) 循环次数  $N_c \leftarrow N_c + 1$ 。
- (3) 蚂蚁的禁忌表索引号  $k=1$ 。
- (4) 蚂蚁数目  $k \leftarrow k+1$ 。
- (5) 蚂蚁个体根据状态转移概率公式 (2.4.3) 计算的概率选择元素（城市） $j$  并前进， $j \in \{C - \text{tabu}_k\}$ 。
- (6) 修改禁忌表指针，即选择好之后将蚂蚁移动到新的元素（城市），并把该元素（城市）移动到该蚂蚁个体的禁忌表中。
- (7) 若集合  $C$  中元素（城市）未遍历完，即  $k < m$ ，则跳转到第 (4) 步，否则执行第 (8) 步。
- (8) 根据公式 (2.4.5) 和式 (2.4.6) 更新每条路径上的信息量。
- (9) 若满足结束条件，即如果循环次数  $N_c \geq N_{c_{\max}}$ ，则循环结束并输出程序计算结果，否则清空禁忌表并跳转到第 (2) 步。

### 2.5.2 基本蚁群算法的程序结构流程

以 TSP 为例，基本蚁群算法的程序结构流程如图 2.7 所示。

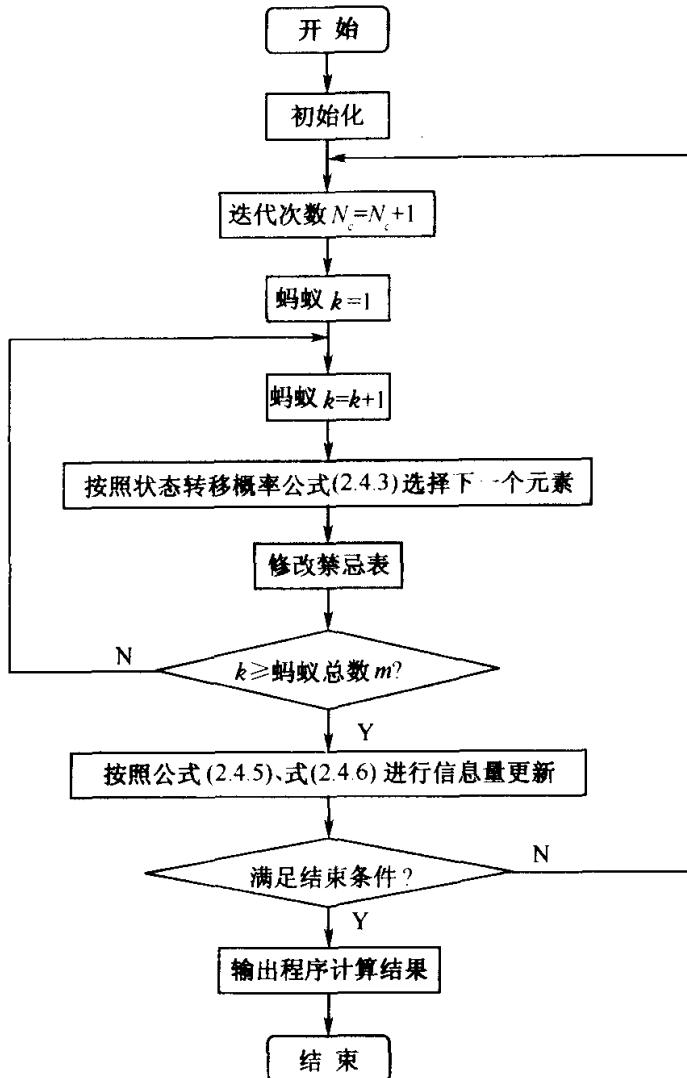


图 2.7 基本蚁群算法的程序结构流程

### 2.5.3 基本蚁群算法的软件实现

以 TSP 为例，基于不同开发环境的基本蚁群算法软件如图 2.8 所示，其中图 2.8(a)是 Eyckelhof C J 基于 Delphi 开发的“简易蚁群算法仿真软件”<sup>[21]</sup>；图 2.8(b)是段海滨基于 Matlab 开发的“蚁群算法仿真平台”<sup>[15]</sup>；图 2.8(c)是陈烨基于 Visual Basic 开发的“蚁群算法实验室”。

图 2.8(a)所示的软件界面简洁，所实现的功能也比较简单，只能实时显示城市的路径状态。图 2.8(b)所示的软件界面在图 2.8(a)的基础上做了许多改进，在图 2.8(b)所示的软件界面上，用户只要输入要修改的参数即可完成 TSP 的求解，该软件还可以实时显示程序的当前运行状态，即城市的路径状态、正在运行的迭代周期、到目前所发现最优路径的迭代周期、最优路径上城市排列的序号以

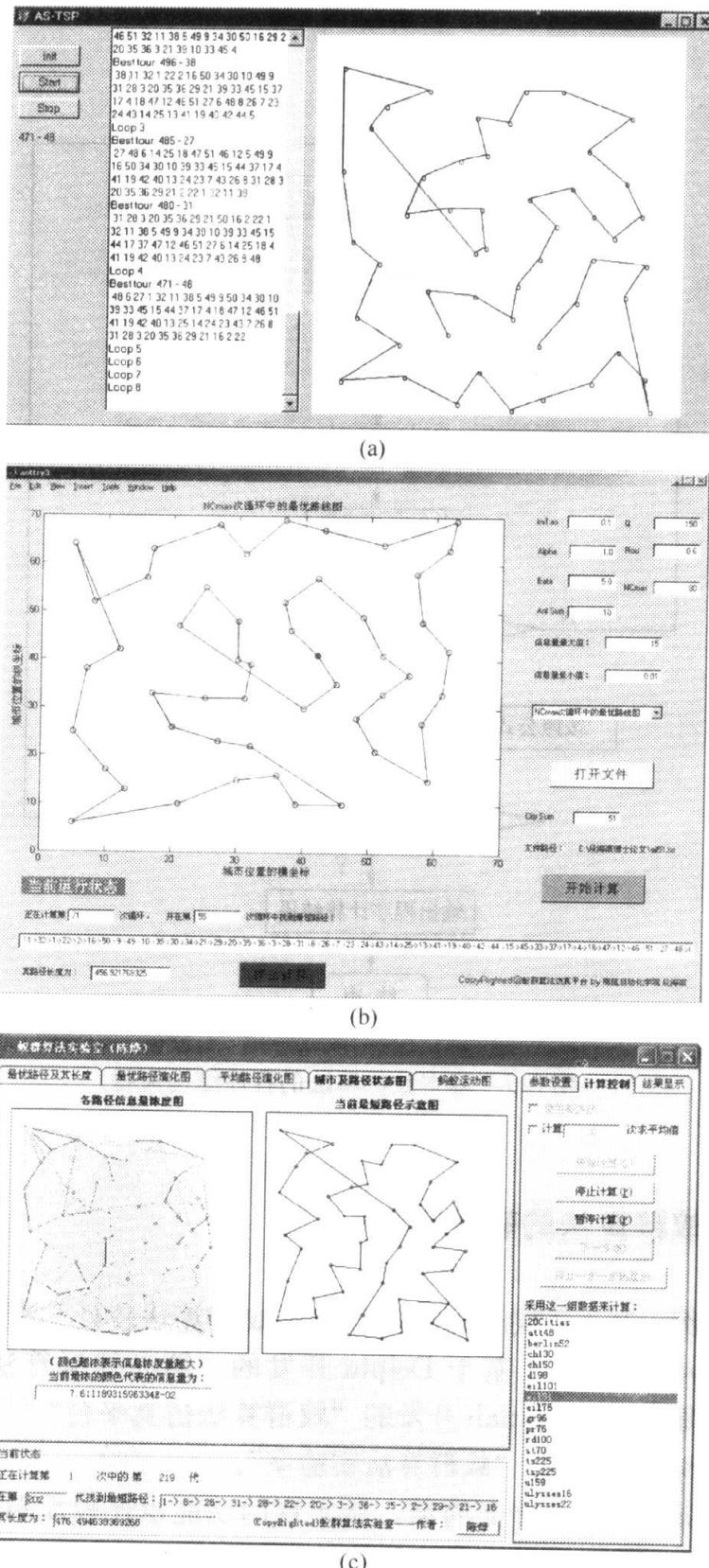


图 2.8 基于不同开发环境的基本蚁群算法软件

及当前最短路径的总长度等，用户根据需要可在城市状态图和最优路径演化图之间进行切换，并可在程序运行过程中随时停止运行，此外该软件还具有良好、开放的可扩展性，用户在应用中可根据具体问题改写相应的文件。图 2.8(c)不仅具有图 2.8(b)的全部功能，还增加了平均路径演化图，并在程序中固化了 TSPLIB 中 18 个具有代表性的 TSP，同时还可实时显示当前信息量，并可根据需要进行单步演示。

## 2.6 基本蚁群算法的复杂度分析

### 2.6.1 复杂度的判定标准和基本概念

如前所述，对于 TSP，所有的可行路径共有  $\frac{(n-1)!}{2}$  条。若以路径比较为基本操作，则需要  $\frac{(n-1)!}{2} - 1$  次基本操作才能保证得到绝对最优解。对于每秒可执行 100 万次浮点运算的计算机来说，当  $n=10$  时需要 0.19 秒能找到最优解；当  $n=20$  时就需要 1929 年才能找到最优解；而当  $n=30$  时就需要约  $1.4 \times 10^{17}$  年（精确为 140185216795720794.6 年）才能找到最优解。因此，对较大规模问题进行穷举是不切实际的，也就是说，许多优化问题所需要的计算时间和存储空间是难以承受的，所以算法可解的问题在实践中并不一定可解。鉴于许多问题的求解复杂度远远大于 TSP，因此只有了解所研究问题和算法的复杂度，才能有针对性地设计和改进算法，进而提高全局优化效率。

基本蚁群算法的复杂度理论主要研究蚁群算法在求解问题时所需各种资源的量，其主要考虑的是设计可以用于估计、定界用蚁群算法求解某类问题时所需要的和仅需的计算资源量的技术或方法。

根据一般算法理论<sup>[22]</sup>，判定一个算法的好坏主要有如下标准：

- (1) 算法的正确性：要求算法能够正确地执行所规定的性能要求。
- (2) 算法的可使用性：要求算法能够方便地使用。
- (3) 算法的可读性：算法必须逻辑清晰、结构简单，以便于测试和修改。
- (4) 算法的执行效率：这主要指算法执行时时间的消耗，包括运行时间开销和存储时间开销两个方面，前者称为算法的时间代价，后者称为算法的空间代价。
- (5) 算法的健壮性：这主要体现在两个方面，其一体现在算法代码的健壮性，即代码本身应具有对输入数据的检查、对运行错误报告的容错机制和意外处理机制；其二体现在利用算法模型求解问题的宽广程度。

很显然，这五个标准中的前三项都涉及到了算法代码的编制问题，而不涉及

到算法模型本身。由于本书主要研究蚁群算法的基本理论及其应用，因而不考虑这三个标准。算法的执行效率涉及到了算法建模和算法编制两个方面。然而，同样编写的程序在不同的计算机硬件、软件环境下的时间效率和空间效率都有所差异，这种差异大多在算法完成后进行事后测试估计，很难从理论上进行深入探讨。通常，对算法效率在理论上的探讨又称为算法的事前估计，可分为算法的时间复杂度分析和空间复杂度分析，其定义如下<sup>[92]</sup>。

**定义 2.6.1 算法的时间复杂度** 算法的时间复杂度是指求解该问题的所有算法中时间复杂性最小的算法时间复杂度。

在算法的设计和分析中，沿用实用性的复杂度概念，即把求解问题的关键操作（如加、减、乘、除、比较等运算）指定为基本操作，通常把算法执行基本操作的次数定义为算法的时间复杂度。

**定义 2.6.2 算法的空间复杂度** 算法的空间复杂度是指求解该问题的所有算法中空间复杂性最小的算法空间复杂度。

在实际应用中，通常把算法执行时间内所占用的存储单元定义为算法的空间复杂度。

**定义 2.6.3** 给定自然数  $n$  的两个函数  $F(n)$  与  $G(n)$ ，当且仅当存在一个正常数  $K$  和一个  $n_0$ ，使得当  $n \geq n_0$  时，有  $F(n) \leq KG(n)$ ，则称函数  $F(n)$  以函数  $G(n)$  为界，记作  $F(n) = O(G(n))$ ，或称  $F(n)$  是  $O(G(n))$ 。此处的“ $O$ ”表示数量级的概念。

在分析算法复杂度时，可以求出复杂度函数  $p(n)$ ，也可以用复杂度函数主要项的阶  $O(p(n))$  来表示。若算法  $A$  的时间复杂度为  $T_A(n) = O(p(n))$ ，且  $p(n)$  为  $n$  的多项式函数，则称算法  $A$  为多项式算法，而把时间复杂度大于多项式时间的算法统称为指数时间算法。

基本蚁群算法的复杂度可表示为问题规模  $n$  (TSP 中的城市数目) 的函数，其中把时间复杂度记为  $T(n)$ ，而把空间复杂度记为  $S(n)$ 。

## 2.6.2 基本蚁群算法的时间复杂度分析

准确计算一个算法的时间复杂度，不仅需要对算法程序进行一步一步地分析，还要考虑系统的编译时间，这在理论上意义不是很大，因为算法时间复杂度分析的目的在于说明实现相同功能的不同算法的计算效率高低<sup>[16]</sup>。因此，实际中通常采用时间复杂度的渐进表示法，说明程序执行步数的数量级，从而估算算法执行效率的高低。

设  $n$  为 TSP 的规模， $m$  为基本蚁群算法所采用的蚂蚁数目，循环变量为  $N_c$ ，最大循环次数为  $N_{c_{\max}}$ ，则针对前面对基本蚁群算法程序结构流程的描述，可逐步分析出其时间复杂度，如表 2.1 所示。

表 2.1 基本蚁群算法的时间复杂度分析

步 骤	内 容	时间复杂度
(1)	初始化参数	$O(n^2 + m)$
(2)	设置蚂蚁禁忌表	$O(m)$
(3)	每只蚂蚁单独构造解	$O(n^2 m)$
(4)	解的评价和轨迹更新量的计算	$O(n^2 m)$
(5)	信息素轨迹浓度的更新	$O(n^2)$
(6)	判断是否达到算法的终止条件（达到设定的最大循环次数 $N_{c_{\max}}$ ），若没有，则转到第(2)步	$O(nm)$
(7)	输出计算结果	$O(1)$

当  $n$  足够大时，低次幂的影响可忽略不计，则由表 2.1 可知，基本蚁群算法中  $m$  只蚂蚁要遍历  $n$  个元素（城市），经过  $N_c$  次循环，则整个计算过程的时间复杂度为

$$T(n) = O(N_c \cdot n^2 \cdot m) \quad (2.6.1)$$

### 2.6.3 基本蚁群算法的空间复杂度分析

基本蚁群算法在实现中所要用到的数据对其空间复杂度影响很大，此处的空间指的是蚁群算法执行过程中相应 NDTM 的读写头所访问过的不同带格的总数目。基本蚁群算法中的数据主要实现两方面功能：一是问题的描述，二是实现蚁群算法功能的辅助数据。

还是以 TSP 为例，由于基本蚁群算法的求解通常采用有向图来描述，这里假设所求解问题的规模为  $n$ ，则需要一个  $n$  阶二维距离矩阵描述问题本身的特点；为了表示有向图上的信息量，需要用另外一个  $n$  阶二维距离来表示图上的信息素轨迹浓度；同时，在基本蚁群算法的求解过程中，为了保证 TSP 中的城市不重复，需要为每只蚂蚁设定一个  $n$  阶一维数组的禁忌表；为了保存蚂蚁寻到的解，需要为每只蚂蚁设立一个数组；为了便于更新轨迹，需要设置每条边上的信息素更新量，其为一个二维数组；为了评价解的优劣，需要定义中间变量；等等。这些都是在基本蚁群算法中所用到的一些基本存储变量。渐进空间复杂度的定义与渐进时间复杂度的定义类似，通过对基本蚁群算法各步骤的综合分析，可得其整个计算过程的空间复杂度为

$$S(n) = O(n^2) + O(n \cdot m) \quad (2.6.2)$$

由上式可见，在数据存储上基本蚁群算法的空间复杂度是非常简单的，从而算法易于程序编制。

## 2.7 基本蚁群算法的性能评价指标

算法的性能评价主要是指对算法的时间复杂度、空间复杂度在量化意义上的计算性能进行分析<sup>[23]</sup>。为了比较全面地衡量基本蚁群算法性能的优劣程度，这里引入了评价基本蚁群算法性能的三个基本指标。

### (1) 最佳性能指标。

定义相对误差  $E_O$  为最佳优化性能指标，其公式如下

$$E_O = \frac{c_b - c^*}{c^*} \times 100\% \quad (2.7.1)$$

式中， $c_b$  表示算法多次运行所得到的最佳优化值； $c^*$  表示所求问题的理论最优值，当理论最优值未知时，可用已知最佳优化值来代替。最佳性能指标用以衡量基本蚁群算法对问题的最佳优化度，其值越小意味着蚁群算法的优化性能越好。

### (2) 时间性能指标。

定义基本蚁群算法的时间性能指标  $E_T$  如下

$$E_T = \frac{I_a T_0}{I_{\max}} \times 100\% \quad (2.7.2)$$

式中， $I_a$  表示算法多次运行后，满足终止条件时的迭代次数平均值； $I_{\max}$  表示给定的最大迭代次数； $T_0$  表示算法一次迭代的平均计算时间。时间性能指标用以衡量基本蚁群算法对问题解的搜索快慢程度，在  $I_{\max}$  固定的前提下， $E_T$  越小说明蚁群算法的收敛速度越快。

### (3) 鲁棒性能指标。

定义基本蚁群算法的鲁棒性能指标  $E_R$  如下

$$E_R = \frac{c_a - c^*}{c^*} \times 100\% \quad (2.7.3)$$

式中， $c_a$  表示算法多次运行所得到的平均值； $c^*$  表示所求问题的理论最优值。鲁棒性能指标用以衡量基本蚁群算法对随机初值和操作的依赖程度。

由此，基本蚁群算法的综合性能指标  $E$  可表示为上述三个性能指标的加权组合

$$E = \alpha_O E_O + \alpha_T E_T + \alpha_R E_R \quad (2.7.4)$$

式中， $\alpha_O$ 、 $\alpha_T$  和  $\alpha_R$  分别表示最佳性能指标、时间性能指标和鲁棒性能指标的加权系数，且满足

$$\alpha_O + \alpha_T + \alpha_R = 1 \quad (2.7.5)$$

$E$  值越小，则基本蚁群算法的综合性能越好。

## 2.8 本章小结

本章首先从深层意义上对基本蚁群算法的机制原理进行了探讨，然后从系统  
• 42 •

学的角度进一步分析了基本蚁群算法在分布式计算、自组织、正反馈等方面的系统学特征，随后在阐述 P、NP、NP-C、NP-hard 类问题的基础上，从 TSP 的角度对基本蚁群算法的数学模型进行了深入分析，并给出了其具体实现步骤、程序结构框架以及基于不同开发环境的软件实现，然后在引入算法复杂度若干概念的基础上，对基本蚁群算法的空间复杂度和时间复杂度进行了分析，最后提出了基本蚁群算法的性能评价指标。

本章内容是基本蚁群算法的机理分析部分，也是深入理解蚁群算法、利用蚁群算法开展研究工作的基础。

## 参 考 文 献

- 1 Colomi A, Dorigo M, Maniezzo V, *et al.* Distributed optimization by ant colonies. Proceedings of the 1st European Conference on Artificial Life, 1991, 134~142
- 2 Dorigo M, Maniezzo V, Colomi A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- 3 Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66
- 4 段海滨, 王道波, 朱家强等. 蚁群算法理论及应用研究的进展. 控制与决策, 2004, 19(12): 1321~1326, 1340
- 5 Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behavior. Nature, 2000, 406(6): 39~42
- 6 Jackson D E, Holcombe M, Ratnieks F L W. Trail geometry gives polarity to ant foraging networks. Nature, 2004, 432(7019): 907~909
- 7 Gambardella L M, Dorigo M. Solving symmetric and asymmetric TSPs by ant colonies. Proceedings of the IEEE International Conference on Evolutionary Computation, 1996, 622~627
- 8 Katja V, Ann N. Colonies of learning automata. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 2002, 32(6): 772~780
- 9 Dorigo M, Di Caro G. Ant colony optimization: a new meta-heuristic. Proceedings of the 1999 Congress on Evolutionary Computation, 1999, 2: 1470~1477
- 10 Dorigo M, Gambardella L M. Ant colonies for the traveling salesman problem. Bio Systems, 1997, 43: 73~81
- 11 Dorigo M, Bonabeau E, Theraulaz G. Ant algorithms and stigmergy. Future Generation Computer Systems, 2000, 16(8): 851~871
- 12 Chu S C, Roddick J F, Pan J S. Ant colony system with communication strategies. Information Sciences, 2004, 167(1-4): 63~76
- 13 Sun R Y, Tatsumi S, Zhao G. Multiagent reinforcement learning method with an improved ant colony system. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2001, 3: 1612~1617
- 14 Chu S C, Roddick J F, Pan J S, *et al.* Parallel ant colony systems. Lecture Notes in Artificial Intelligence, 2003, 2871: 279~284
- 15 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005

- 16 何靖华. 面向仿生制造的蚂蚁算法分析研究及应用. 武汉: 华中科技大学硕士学位论文, 2002
- 17 王凌. 车间调度及其遗传算法. 北京: 清华大学出版社, 2003
- 18 陈志平, 徐宗本. 计算机数学. 北京: 科学出版社, 2001
- 19 Dorigo M, Caro G D, Gambardella L M. Ant algorithms for discrete optimization. *Artificial Life*, 1999, 5(2): 137~172
- 20 Merkle D, Middendorf M. Modeling the dynamics of ant colony optimization. *Evolutionary Computation*, 2002, 10(3): 235~262
- 21 Eyckelhof C J. Ant systems for dynamic problems. Ph. D. Thesis, University of Twente, Netherlands, 2001
- 22 殷人昆, 陶永雷, 谢若阳等. 数据结构-用面向对象方法与 C++ 描述. 北京: 清华大学出版社, 1999
- 23 王凌. 智能优化算法及其应用. 北京: 清华大学出版社, 2001

## 第3章 蚁群算法的收敛性研究

### 3.1 引言

研究算法的收敛性不仅对深入理解算法机理具有重要的理论意义,而且对改进算法、编写算法程序具有非常重要的现实指导意义。因此,所有的仿生优化算法都要考虑它们的收敛性问题。

虽然蚁群算法创立至今已经十余年,但是对其收敛性的研究是最近几年才刚刚开始的。Gutjahr W J 最先从有向图论的角度对一种改进蚁群算法——GBAS 的收敛性进行了证明<sup>[1, 2]</sup>; Stüezle T 和 Dorigo M 针对具有组合优化性质的极小化问题提出了一类改进蚁群算法—— $ACO_{gb, \tau_{\min}}$  算法<sup>[3]</sup>, 并对其收敛性进行了理论分析, 证明了当计算时间趋于无穷时, 蚁群算法总能搜索到全局最优解, 且与最优解所对应路径上的信息量大于其他任意路径上的信息量; 针对上述两种改进蚁群算法中的一些缺陷, Gutjahr W J 又提出了两种新的 GBAS, 即 GBAS/tdev 和 GBAS/tdlb<sup>[4]</sup>, 并证明可通过选择合适的参数来保证蚁群算法的动态随机过程使其收敛到最优解; 段海滨等<sup>[5~8]</sup>在对基本蚁群算法的 Markov 链分析过程中, 运用离散鞅(discrete martingale)作为研究工具, 把最优解集序列转变为下鞅序列来考察信息素轨迹向量的收敛性, 并对蚁群算法的几乎处处(almost surely, A. S.)收敛问题和停时问题进行了研究, 提出了蚁群算法首达时间的定义, 同时还对蚁群算法首次到达时间的期望值作了初步分析; Yoo J H 等<sup>[9, 10]</sup>对一类分布式蚂蚁路由算法的收敛性进行了深入的理论研究; Badr A 等<sup>[11]</sup>将蚁群算法模型转化为分支随机过程, 从分支随机路径和分支 Wiener 过程的角度推导了蚂蚁路径存亡的比率, 并证明了该过程为稳态分布; 孙焘等<sup>[12]</sup>对一类简单蚁群算法的收敛性及有关参数问题做了初步研究; 丁建立等<sup>[13]</sup>对一种遗传-蚁群算法的收敛性进行了 Markov 理论分析, 并证明其优化解满意值序列单调不增且收敛; Hou Y H 等<sup>[14, 15]</sup>基于不动点(fixed-point)理论对一类广义蚁群算法(generalized ant colony algorithm, GACA)的收敛性进行了初步分析。本章内容可为改进蚁群算法提供理论依据和指导。

### 3.2 图搜索蚂蚁系统(GBAS)的收敛性研究

Gutjahr W J 最先对 GBAS 的收敛性进行了理论证明<sup>[1, 2]</sup>, GBAS 对基本蚁群算法做了进一步规范和抽象, 规定了更加通用的路径转移机制。

### 3.2.1 GBAS的思想起源

用于求解 TSP 的 GBAS 与自然界中的真实蚁群系统类似：首先，蚂蚁智能体被安置在有向图中的某个节点上；随后，每个蚂蚁智能体都按照状态转移概率在临近节点间随机移动<sup>[16]</sup>。只要有一个智能体遍历完有向图中的所有节点，则计算该智能体所遍历的路径长度，且其所经路径上的信息量会根据其遍历质量成比例地增加，这一过程持续多次。然而，与信息素积累过程相反，信息素挥发机制会按照给定的信息素挥发系数周期性地减少信息量。蚂蚁智能体在有向图中某一特定弧段上的状态转移概率按照该弧段上的残留信息量和弧段长度这两个参数来进行计算，信息量越大且弧段越短，则蚂蚁智能体沿着该弧段移动的可能性越大。

Gutjahr W J 提出的 GBAS 是对基本蚁群算法的一个扩展，其基本思想是把所求问题的可行解转化为有向图上的可行路径。

### 3.2.2 GBAS 的数学模型

**定义 3.2.1** 给定一个组合优化问题的实例，可通过该实例的构造图给出有向图  $C = (V, A)$  和映射  $\Phi$  的如下性质：

- (1) 有向图  $C$  中，起始节点仅有一个。
- (2) 令  $W$  为有向图  $C$  中满足如下条件(有向)路径  $w$  的集合：
  - ① 路径  $w$  开始于有向图  $C$  中的起始节点；
  - ② 路径  $w$  最多包含一次有向图  $C$  中的有向节点；
  - ③ 路径  $w$  的最后一个节点在有向图  $C$  中没有后继节点。

映射  $\Phi$  将集合  $W$  映射到给定实例的可行解集合上。换言之，对于满足条件①～③的每一条(有向)路径  $w$ ，都按照映射  $\Phi$  对应一个可行解；反之，对于每一个可行解，至少有一条(有向)路径  $w$  按照逆映射  $\Phi^{-1}$  满足条件①～③。

由定义 3.2.1 可见，构造图  $(C, \Phi)$  把所求组合优化问题的可行解定义为“路径”，而路径的长度对应于所求问题的目标函数值。一般而言，对于某一特定问题可设计出多种构造图，而构造图的选择对算法的性能有很大的影响。

基于蚁群算法<sup>[17~20]</sup>的 GBAS 主要包含如下几个组成部分：

- (1) 由定义 3.2.1 所给出的构造图  $(C, \Phi)$ 。
- (2) 由蚂蚁组成的智能体集合，即  $\{A_1, \dots, A_s\}$ 。每个蚂蚁智能体可根据构造图中所选择的状态转移概率完成一次随机移动。在多处理器系统中，每个蚂蚁智能体的移动可由单独的处理器并行计算；而在单处理器系统中，每个蚂蚁智能体的移动只能在单一的处理器上进行串行计算。将蚂蚁智能体在构造图上完成一次搜索的时间称为搜索周期。GBAS 包含着若干个搜索周期，假设包含  $1, \dots, M$  个搜

索周期,搜索周期的数量  $M$  可事先确定,也可在 GBAS 运行过程中确定。

(3) 每个搜索周期中蚂蚁智能体随机移动的状态转移概率。令  $u = \{u_0, \dots, u_{t-1}\}$  表示在某一固定的  $m$  次搜索周期中,第  $t$  个搜索周期前某一蚂蚁智能体所遍历的部分路径,其中  $u_0, \dots, u_{t-1}$  表示构造图中的节点指针( $u_0$  为起始点指针)。若节点  $l$  包含在部分路径  $u$  中,则记为  $l \in u$ ;否则,记为  $l \notin u$ 。进一步,令  $A$  为构造图中路径弧段的集合,则在当前搜索周期  $m$  内,已遍历部分路径  $u = \{u_0, \dots, u_{t-2}, u_{t-1} = k\}$  的蚂蚁智能体从节点  $k$ (当前位置)到节点  $l$  的状态转移概率为

$$p_{kl}(m, u) = \begin{cases} \frac{[\tau_{kl}(m)]^\alpha [\eta_{kl}(u)]^\beta}{\sum_{r \notin u, (k, r) \in A} [\tau_{kr}(m)]^\alpha [\eta_{kr}(u)]^\beta}, & \text{若 } l \notin u \text{ 且 } (k, l) \in A \\ 0, & \text{否则} \end{cases} \quad (3.2.1)$$

式中,  $\tau_{kl}(m)$  表示信息量;  $\eta_{kl}(u)$  表示期望值(启发函数)。

(4) 构造图中弧段  $(k, l)$  上的信息量  $\tau_{kl}$ 。信息量随着搜索周期的不同而变化,把信息量对搜索周期指针  $m$  的依赖性记为  $\tau_{kl}(m)$ 。在初始的第一个搜索周期内,把每一弧段  $(k, l)$  上的信息量设为  $\tau_{kl} = 1/(弧段数目)$ 。当每个搜索周期结束时,都要按照如下规则对信息素进行更新。

首先,对于任意蚂蚁智能体  $A_s$  和任意路径弧段  $(k, l)$ ,  $\Delta\tau_{kl}^{(s)}$  是由当前搜索周期内蚂蚁智能体  $A_s$  所对应的解函数决定的。假定该解的代价值(目标函数值)为  $f_s$ ,则对于路径弧段  $(k, l)$ ,有

$$\Delta\tau_{kl}^{(s)} = \begin{cases} \varphi(f_s), & \text{若蚂蚁智能体 } A_s \text{ 已穿越弧段 } (k, l) \\ 0, & \text{否则} \end{cases} \quad (3.2.2)$$

式中,  $\varphi$  是非增且以  $f_s$  为变量的非负函数,它在一定程度上依赖于前  $m-1$  次搜索周期中蚂蚁智能体所遍历的路径。令

$$C = \sum_{(k, l) \in A} \sum_{s=1}^S \Delta\tau_{kl}^{(s)} \quad (3.2.3)$$

若  $C=0$ ,则对于所有的路径弧段  $(k, l)$ ,有

$$\tau_{kl}(n+1) = \tau_{kl}(n) \quad (3.2.4)$$

即第  $n+1$  个搜索周期的  $\tau_{kl}$  值与第  $n$  个搜索周期的  $\tau_{kl}$  值相等;若  $C>0$ ,则有

$$\tau_{kl}(n+1) = (1 - \rho)\tau_{kl}(n) + \rho\Delta\tau_{kl} \quad (3.2.5)$$

$$\Delta\tau_{kl} = \frac{1}{C} \sum_{s=1}^S \Delta\tau_{kl}^{(s)} \quad (3.2.6)$$

由式(3.2.3)~式(3.2.6),易证信息素总量  $\sum_{(k, l) \in A} \tau_{kl}(m)$  恒等于 1。

上述信息素更新规则可进一步描述如下:若没有路径被强化,则信息量保持为常值;否则,由于挥发因子的存在,在  $n$  个搜索周期后,构造图上只有  $1-\rho$  倍的信息量存留下来,其余  $\rho$  倍的信息量根据其各自的目标函数值作为“奖赏”补偿蚂蚁

智能体在  $n$  个搜索周期中所遍历的路径。每个蚂蚁智能体  $A_s$  都可认为其在  $n$  个搜索周期中所得的“奖赏”为  $\Delta\tau_{kl}^{(s)}$ , 而实际的信息素增量是以  $\rho$  倍“奖赏”的形式按比例分配给  $\Delta\tau_{kl}^{(s)}$  的。

由此可见, 构造图中最优路径上的信息量会增加, 从而使其将来更容易被其他蚂蚁智能体穿越。若令  $\rho=0$ , 则代价函数对蚂蚁智能体所经路径的影响消失; 若令  $\beta=0$ , 则期望值  $\eta_{kl}(u)$  的作用消失, 这时算法将陷入纯粹的随机搜索状态。

(5) 构造图中路径弧段  $(k, l)$  上的期望值  $\eta_{kl}$ 。期望值  $\eta_{kl}$  依赖于蚂蚁智能体的整个遍历历史, 即蚂蚁智能体已穿越的部分路径  $u=(u_0, \dots, u_{t-2}, u_{t-1}=k)$ , 这样可将期望值记为  $\eta_{kl}=\eta_{kl}(u)$ 。典型地,  $\eta_{kl}(u)$  可由所求组合优化问题的贪心算法求得, 由此, 可将  $\eta_{kl}(u)$  理解为特定贪心式搜索的函数值<sup>[21]</sup>, 这样能逐步构造出所求问题的优良解(但通常不是最优解)。在 GBAS 体系中, 这一过程可通过构造图中的遍历路径来实现。该贪心式搜索策略为与节点  $k$  相连的所有可行路径弧段  $(k, l)$  定义了“权重”, 并根据路径弧段  $(k, l)$  上“权重”值最大这一贪心规则来决定路径的下一节点  $l$ 。此处可考虑以路径弧段  $(k, l)$  的“权重”值作为蚂蚁智能体  $A_s$  从节点  $k$  到节点  $l$  的期望值, 即  $\eta_{kl}(u)=\text{权重}(k, l)$ ; 也可选择另一种定义“权重”值的方法, 即在节点  $k$  的所有后续节点中, 若路径弧段  $(k, l)$  的“权重”最大, 则令  $\eta_{kl}(u)=1$ , 否则, 令  $\eta_{kl}(u)=0$ 。此外,  $\eta_{kl}(u)$  也可用来防止对应于非可行解的路径被遍历。

在上述分析中, GBAS 是通过贪心规则的自然随机过程产生的。若将参数  $\alpha$  设置为 0, 选择上述两种定义期望值方法中的任意一种, 则蚂蚁智能体的行为将按照贪心规则被控制。

GBAS 适用于有限解空间内的所有组合优化问题。对于实际最优解不唯一的问题而言, 不能保证 GBAS 最优路径上的弧段信息量大于 0。如图 3.1 所示, 假定在构造图中有两条最优路径, 即  $w_1^* = (0, 1, 3, 4, 6, 7, 9, 10, 12)$  和  $w_2^* = (0, 2, 3, 5, 6, 8, 9, 11, 12)$ , 而其他路径是次优的(即具有同最优路径几乎同样多的信息量)。假设对于所有的路径弧段均有  $\eta_{kl}(u)=1$ , 如果蚂蚁智能体穿越路径  $w_1^*$  和  $w_2^*$  的相对频率均为  $1/2$ , 则这两条路径上会得到几乎相等的信息量, 而其他次优路径弧段被正处于最优路径  $w_1^*$  和  $w_2^*$  上的蚂蚁智能体穿越的概率也相等。换言之, 尽

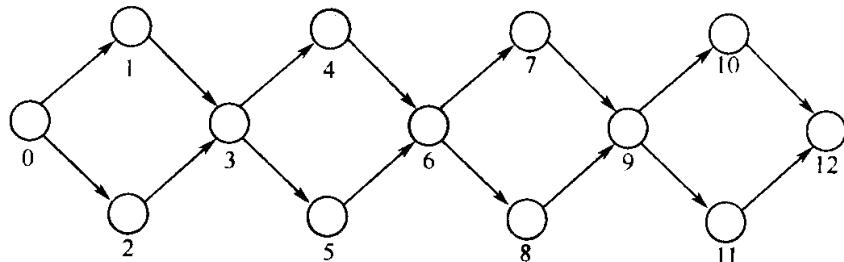


图 3.1 构造图示例

管最优路径上的信息素得到强化,GBAS 仍以很高的概率输出次优解。

下面要证明的定理 3.2.1 并不把上述特殊情况考虑在内,该定理仅考虑最优路径  $w_1^*$  和  $w_2^*$  中一条路径上的信息量得到增强,而其他路径上的信息量会逐渐减少,直至消逝。

### 3.2.3 GBAS 的收敛性研究

在满足某些条件的前提下,GBAS(所对应  $m$  次循环过程中的蚂蚁智能体  $A_1, A_2, \dots, A_s$  所遍历路径的可行解)能够以接近于 1 的概率收敛于最优解。其要满足的基本条件如下:

- (1) 状态转移概率公式(3.2.1)中的参数  $\alpha=1$ 。
- (2) 在  $W$  上仅有一条最优路径,即最优解唯一,且最优解仅通过  $W$  上的一条路径进行编码。
- (3) 对于最优路径  $w^*$  上的所有路径弧段  $(k, l)$  和与其对应的部分路径  $u$ ,其期望值满足  $\eta_{kl}(u) > 0$ 。
- (4) 令  $f^* = f^*(m)$  为  $1, 2, \dots, m-1$  个搜索周期中所花费的最小代价,即最小目标函数值; $f_s$  对应于  $m-1$  个搜索周期中某个蚂蚁智能体  $A_s$  所走过的路径,当  $m=1$  时,令  $f^* = \infty$ ;将函数  $\varphi$  定义为在第  $m+1$  个搜索周期开始时的  $\Delta\tau_{kl}^{(s)}$  值,且函数  $\varphi$  具有如下性质:

- ① 当  $f_s \leq f^*$  时,有  $\varphi(f_s) > 0$ ;
- ② 当  $f_s > f^*$  时,有  $\varphi(f_s) = 0$ 。

换言之,只有当路径至少和到目前为止所发现的最优路径一样好时,才能赋给  $\Delta\tau_{kl}^{(s)}$  一个正的增量。对于正的代价函数, $\varphi(f_s)$  则按照“当  $f_s \leq f^*$  时, $\varphi(f_s) = \frac{1}{f_s}$ ;而当  $f_s > f^*$  时, $\varphi(f_s) = 0$ ”这一法则进行计算。

这里对条件(1)~(4)做进一步说明:

条件(1)仅仅是为了证明方便而提出的,它容易处理公式(3.2.1)右端分母的“归一化因子”。但这并不意味着条件(1)非常重要,因为参数  $\alpha$  和  $\beta$  并不用来影响 GBAS 的信息量和期望值,而是用以控制信息量和期望值之间的相互作用。

条件(2)具有较多的限制,但是已经有结论证明可以放弃该条件。

由于条件(3)需要最优路径的知识,因此可能显得有些繁琐。实际上,该条件并不是一个非常棘手的问题,很容易通过对期望值进行小的调整而满足这一条件,即对于  $\eta_{kl}(u) = 0$  的部分路径  $u$  的每一个后续可行路径弧段  $(k, l)$  而言,用  $\eta_{kl}(u) = \delta$  ( $\delta$  很小且  $\delta > 0$ ) 来替代  $\eta_{kl}(u) = 0$ 。虽然条件(3)不需要有较多的限制,但是该条件非常重要,因为如果期望值选择不当,就可能使最优路径上的某一特定路径弧段不能被蚂蚁智能体穿越。

条件(4)是蚁群算法中“精灵策略”的一个版本,即只有最优路径上的信息素才能得到增强,而其他已被穿越路径上的信息量不再增加。Gambardella L M 和 Dorigo M 在其所改进的 Ant-Q<sup>[22, 23]</sup>和蚁群系统(ant colony system, ACS)<sup>[24, 25]</sup>中首先提出了“只有目前最优路径上的信息素得以增强”这一信息素更新策略,并将其命名为“全局最优”信息素更新策略。

在上述条件下,可以给出如下定理。

**定理 3.2.1** 在满足条件(1)~(4)的情况下,令  $P_m$  表示某个蚂蚁智能体(比如  $A_1$ )在第  $m$  个搜索周期内穿越最优路径  $w^*$  的概率,则有下述两个结论成立:

(1) 对于任意  $\epsilon > 0$  和固定的参数  $\rho$  与  $\beta$ ,且对于所有的  $m \geq m_0$ (整数  $m_0$  依赖于  $\epsilon$ ),则当蚂蚁智能体数目  $S$  充分大时,有  $P_m \geq 1 - \epsilon$ 。

(2) 对于任意  $\epsilon > 0$  和固定的参数  $S$  与  $\beta$ ,且对于所有的  $m \geq m_0$ (整数  $m_0$  依赖于  $\epsilon$ ),则当信息素挥发系数  $\rho$  充分接近于 0 时,有  $P_m \geq 1 - \epsilon$ 。

在给出该定理的证明之前,首先简要介绍一下该定理的证明思路:

首先,证明了 GBAS 可被理解为一个 Markov 过程(见命题 3.2.1),即该随机过程中搜索周期  $m$  状态的概率分布仅依赖于  $m-1$  状态的概率分布。一般而言,概率算法可用随机过程的形式对其进行不同角度的有效描述(见评注 3.2.2);这里通过建立 Markov 过程对证明过程中所引用的概念构建了一个数学框架,并从概率论的角度阐述了其明确意义。

其次,推导了某一固定搜索周期  $m$  内至少有一个蚂蚁智能体穿越最优路径的概率下界(见引理 3.2.1)。通过深入研究可发现,尽管所得最优路径的概率下界依赖于  $m$ ,但其并不依赖于以前搜索周期中的事件“历史”(见推论 3.2.1)。

再次,证明了只要在某一时刻最优路径至少被某一蚂蚁智能体穿越一次,则该最优路径上所得的信息量会越来越接近于  $1/(最优路径长度)$ ,而其他路径弧段的信息量将趋于 0(见引理 3.2.2)。

继而,证明了在最优路径至少被某一蚂蚁智能体穿越一次的条件下,最后所提及的信息素使得决定穿越最优路径弧段的状态转移概率值越来越趋于 1(见引理 3.2.3 和推论 3.2.2)。

随后,证明了在前述四个基本条件下,“某一蚂蚁智能体穿越最优路径”这一事件发生的概率会越来越趋于 1(见引理 3.2.4)。

最后,通过理论分析和对“没有蚂蚁智能体穿越最优路径”这一事件的概率估计证明了定理 3.2.1,即通过利用引理 3.2.1 所推导的至少有一个蚂蚁智能体穿越最优路径的概率下界,证明了可通过增加蚂蚁智能体数目或减小信息素挥发系数而使得这一概率任意小。

这里非常有必要将条件(1)~(4)引入到 GBAS 的收敛性证明中:条件(1)仅仅是引理 3.2.1 证明中所采用信息素更新规则的一个简化;通过利用条件(2)将该证明弱化为对唯一最优路径上所发生事件的研究;条件(3)给出了引理 3.2.1 的概

率下界;条件(4)可避免 GBAS 收敛到次优解这一早熟事件的发生。

需要指出的是,这里的证明在  $\beta=0$  这一特殊情况(即  $\eta_{kl}(u)$  根本不起作用的情况)下也是有效的(见评注 3.2.3)。另一方面,对于引理 3.2.1 而言,必须排除  $\rho=0$  这一情况,从而设定  $\rho>0$ 。因此,定理 3.2.1 所证得的结论不包括随机搜索。

如前所述,这里可以把基于 GBAS 的迭代求解过程理解为离散时间的 Markov 过程,而该 Markov 过程的状态即为一个三元组:

$$(\underline{\tau}(m), \underline{w}(m), f^*(m)) \quad (m = 1, 2, \dots) \quad (3.2.7)$$

式中,  $\underline{\tau}(m)$  表示第  $m$  个搜索周期中所有路径弧段  $(k, l)$  上信息量  $\tau_{kl}(m)$  的向量;  $\underline{w}(m)$  表示第  $m$  个搜索周期中蚂蚁智能体  $A_1, A_2, \dots, A_n$  的路径  $w^s(m)$  ( $s = 1, \dots, S$ ) 所组成的向量;  $f^*(m)$  表示任意蚂蚁智能体在搜索周期  $1, 2, \dots, m-1$  所搜索到的与路径相关的最好代价,由条件(4)可见,  $f^*$  控制着信息量的值。当  $m=1$  时,令  $f^*(1)=\infty$ 。

**命题 3.2.1** 状态变量  $(\underline{\tau}(m), \underline{w}(m), f^*(m))$  ( $m = 1, 2, \dots$ ) 构成一个 Markov 过程。

**证明** 因为状态转移概率是按照下述规则给出的:

- (1)  $\underline{\tau}(m)$  根据信息素更新规则由  $\underline{\tau}(m-1), \underline{w}(m-1)$  和  $f^*(m-1)$  共同决定。
- (2)  $\underline{w}(m)$  的概率分布只依赖于  $\underline{\tau}(m)$ , 进而由  $(\underline{\tau}(m-1), \underline{w}(m-1), f^*(m-1))$  决定。
- (3)  $f^*(m)$  由  $\underline{w}(m-1)$  和  $f^*(m-1)$  共同决定。

由此,搜索周期  $m$  的状态分布  $(\underline{\tau}(m), \underline{w}(m), f^*(m))$  仅依赖于搜索周期  $m-1$  的状态分布  $(\underline{\tau}(m-1), \underline{w}(m-1), f^*(m-1))$ , 所以满足 Markov 的过程特性。

[证毕]

**评注 3.2.1** 值得注意的是,在给定的 Markov 过程解释中,公式(3.2.1)所定义的  $p_{kl}(m, u)$  是  $\underline{\tau}(m)$  的函数,从而也是搜索周期  $m-1$  的状态函数。特别地,  $p_{kl}(m, u)$  是随机变量,因此,将其解释成概率是一种间接的表达形式,该概率通过 Markov 过程的状态转移规则决定在搜索周期  $m$  中路径向量  $\underline{w}(m)$  的分布。

**评注 3.2.2** 此处也可通过非 Markov 过程来描述蚁群搜索过程,即用三元组  $(\underline{\tau}(m), \underline{w}(m), f_{opt}(m))$  替代  $(\underline{\tau}(m), \underline{w}(m), f^*(m))$  来跟踪当前状态,其中  $f_{opt}(m)$  表示搜索周期  $m$  中所得到的最优代价值。例如,在证明引理 3.2.4 的过程中,只考虑搜索周期  $m'-1$  的状态,而非该过程的整个历史状态。

下面给出证明过程中所用主要符号的具体含义:

- $w^*$  表示(唯一的)最优路径;
- $L$  表示最优路径  $w^*$  的长度;
- $Pr$  表示上述所定义 Markov 过程的概率度量;
- $E_m^{(s)}$  表示在第  $m$  个搜索周期中,蚂蚁智能体  $A_s$  穿越最优路径的概率事件,即  $w^{(s)}(m)=w^*$ ;

- $B_m$  是  $\neg E_m^{(1)} \wedge \cdots \wedge \neg E_m^{(s)}$  的缩写, 即对于  $\forall s=1, 2, \dots, S$ , 有  $w^{(s)}(m) \notin w^*$  (在  $m$  个搜索周期中, 没有蚂蚁智能体穿越最优路径的概率事件);
- $F_m$  是  $B_1 \wedge \cdots \wedge B_{m-1} \wedge \neg B_m$  的缩写, 即在第  $m$  个搜索周期中, 最优路径被某个蚂蚁智能体穿越, 而在搜索周期  $1, 2, \dots, m-1$  中没有被蚂蚁智能体穿越的事件。显然, 事件  $F_1, F_2, \dots$  是相互独立的;
- $A$  是  $F_1 \vee F_2 \vee \cdots$  的缩写, 表示在  $m$  个搜索周期中, 最优路径被某个蚂蚁智能体  $A_s$  穿越的事件, 即  $\exists n, s$ , 使得  $w^{(s)}(n) \in w^*$ 。

$(k, l) \in w$  表示弧段  $(k, l)$  位于路径  $w$  上。根据条件(3), 且路径弧段  $(k, l)$  和可行路径  $u$  有限, 则有

$$\gamma = \min\{\lceil \eta_{kl}(u) \rceil^\beta \mid (k, l) \in w^*, u \text{ 为 } w^* \text{ 上的部分路径}\} > 0 \quad (3.2.8)$$

且

$$\Gamma = \max\{\lceil \eta_{kl}(u) \rceil^\beta \mid (k, l)\} < \infty \quad (3.2.9)$$

若有无穷多条路径弧段或可行路径, 则无法定义其上界和下界, 但其上确界和下确界分别为  $\infty$  和 0。如果将所有的期望值  $\eta_{kl}(u)$  都定义为一个固定常数, 并不会改变公式(3.2.1)所示的状态转移概率。因此, 不失一般性,  $\eta_{kl}(u)$  可按照  $\Gamma=1$  的形式进行归一化, 对于所有的路径弧段  $(k, l)$  及所有的部分路径  $u$ , 有

$$\lceil \eta_{kl}(u) \rceil^\beta \leq 1 \quad (3.2.10)$$

**引理 3.2.1** 至少有一个蚂蚁智能体在搜索周期  $m$  中穿越最优路径的概率 (即事件  $\neg B_m$  发生的概率)  $\Pr(\neg B_m)$  大于或等于  $1 - (1 - c^{m-1} p)^S$ , 其中,  $m \geq 1$ ,  $c = (1 - \rho)^L$ ,  $p = \gamma^L \prod_{(k, l) \in w^*} \tau_{kl}(1)$ ,  $\gamma$  的定义如公式(3.2.8)所示。

**证明** 由于  $\Delta \tau_{kl} \geq 0$ ,  $\rho > 0$ , 则当  $C > 0$  时, 由信息素更新公式(3.2.5)和式(3.2.6)可得

$$\tau_{kl}(m+1) \geq (1 - \rho) \tau_{kl}(m) \quad (3.2.11)$$

由于  $\rho > 0$ , 则当  $C=0$  时也可推出上式。对公式(3.2.11)不断迭代, 有

$$\tau_{kl}(m) \geq (1 - \rho)^{m-1} \tau_{kl}(1) \quad (3.2.12)$$

由公式(3.2.10)及  $\sum_{(k, l)} \tau_{kl}(m) = 1$ , 可得

$$\sum_{r \notin u, (k, r) \in A} \tau_{kr}(m) [\eta_{kr}(u)]^\beta \leq \sum_{r \notin u, (k, r) \in A} \tau_{kr}(m) \leq 1$$

从而, 对于节点  $l$  ( $l \notin u$ ), 状态转移概率  $p_{kl}(m, u)$  满足如下不等式

$$p_{kl}(m, u) = \frac{\tau_{kl}(m) [\eta_{kl}(u)]^\beta}{\sum_{r \notin u, (k, r) \in A} \tau_{kr}(m) [\eta_{kr}(u)]^\beta} \geq \tau_{kl}(m) [\eta_{kl}(u)]^\beta \quad (3.2.13)$$

令  $w^* = (v_0, \dots, v_L)$ , 则由式(3.2.8)、式(3.2.11)和式(3.2.13)可得

$$\Pr(E_m^{(s)}) = \prod_{i=0}^{L-1} p_{v_i, v_{i+1}}(m, (v_0, \dots, v_i)) \geq \prod_{i=0}^{L-1} \tau_{v_i, v_{i+1}}(m) [\eta_{v_i, v_{i+1}}]^\beta$$

$$\begin{aligned} &\geq \gamma^L \prod_{i=0}^{L-1} \tau_{v_i, v_{i+1}}(m) \geq \gamma^L \prod_{i=0}^{L-1} (1-\rho)^{m-1} \tau_{v_i, v_{i+1}}(1) \\ &= \gamma^L (1-\rho)^{L(m-1)} \prod_{(k,l) \in w^*} \tau_{kl}(1) = c^{m-1} p \end{aligned}$$

由于蚂蚁智能体  $S$  的遍历路径是相互独立的,故有

$$\Pr(B_m) \leq (1 - c^{m-1} p)^s$$

[证毕]

**推论 3.2.1** 假定在搜索周期  $m$  前没有蚂蚁智能体穿越最优路径,则在搜索周期  $m$  内至少有一个蚂蚁智能体穿越最优路径的条件概率  $\Pr(\neg B_m | B_1 \wedge B_2 \wedge \dots \wedge B_{m-1})$  大于或等于  $1 - (1 - c^{m-1} p)^s$ 。

**证明** 该推论的证明是引理 3.2.1 证明过程的重复。不等式(3.2.11)和式(3.2.12)在任何情况下,只要独立于搜索周期  $0, 1, \dots, m-1$ ,也就独立于条件  $B_1 \wedge B_2 \wedge \dots \wedge B_{m-1}$ 。

[证毕]

接下来的引理将给出事件  $F_m$  的条件概率。按照规范,这些条件概率通常记为  $\Pr\{\dots | F_m\}$ 。

**引理 3.2.2** 对于  $\forall \epsilon > 0$ ,  $\forall m \in N$ , 则  $\exists d(\epsilon, m) \in N$ , 对于所有的  $m' \geq m + d(\epsilon, m)$ , 有

$$\Pr\left\{ \left| \tau_{kl}(m') - \frac{1}{L} \right| < \epsilon, \text{对所有的 } (k, l) \in w^* | F_m \right\} \geq 1 - \epsilon \quad (3.2.14)$$

及

$$\Pr\{\tau_{kl}(m') < L_\epsilon, \text{对所有的 } (k, l) \notin w^* | F_m\} \geq 1 - \epsilon \quad (3.2.15)$$

**证明** 设第  $m$  个搜索周期内首次穿越最优路径  $w^*$  的事件为  $F_m$ 。考虑到搜索周期  $m' > m$  有如下两种可能情况:

情况 1: 若搜索周期  $m'$  结束时,  $C=0$ ; 这时, 对于所有的路径弧段有  $\tau_{kl}(m'+1) = \tau_{kl}(m')$ ;

情况 2: 若搜索周期  $m'$  结束时,  $C>0$ ; 这种情况只可能发生在某个蚂蚁智能体  $A_s$  在搜索周期  $m'$  中的信息素增量  $\Delta\tau_{kl}^{(s)} > 0$  时。根据公式(3.2.2)和条件(4), 这意味着蚂蚁智能体  $A_s$  已穿越了与最小代价  $f^*(m')$  相对应的路径。因为  $m' > m$ ,  $f^*(m')$  已经是  $f^*$  的最小代价(其代价对应于最优路径  $w^*$ )。所以情况(2)意味着至少有一个蚂蚁智能体  $A_s$  在搜索周期  $m'$  中穿越了最优路径  $w^*$ 。不失一般性, 假定恰是蚂蚁智能体  $A_s$  ( $s=1, 2, \dots, S' | 1 \leq s \leq S$ ) 在搜索周期  $m'$  穿越了  $w^*$ , 则对于某一路径弧段  $(k, l) \notin w^*$ , 有  $\Delta\tau_{kl}^{(s)} = 0$ 。因此, 在所有搜索周期结束时令  $\Delta\tau_{kl} = 0$ , 则对于某一路径弧段  $(k, l) \in w^*$ , 有

$$\Delta\tau_{kl}^{(s)} = \begin{cases} \varphi(f^*), & \text{若 } s \leq S' \\ 0, & \text{否则} \end{cases}$$

由此,可得

$$C = \sum_{(k',l') \in A} \sum_{s=1}^S \Delta \tau_{k'l'}^{(s)} = \sum_{(k',l') \in w^*} S' \varphi(f^*) = LS'(f^*)$$

从而,对于  $\forall (k,l) \in w^*$ ,有

$$\Delta \tau_{kl} = \sum_{s=1}^S \Delta \tau_{kl}^{(s)} = \frac{S' \varphi(f^*)}{LS' \varphi(f^*)} = \frac{1}{L} \quad (3.2.16)$$

因此,有

$$\tau_{kl}(m'+1) = (1-\rho)\tau_{kl}(m') + \frac{\rho}{L} \quad (3.2.17)$$

或

$$\tau_{kl}(m'+1) - \tau_{kl}(m') = \rho \left( \frac{1}{L} - \tau_{kl}(m') \right) \quad (3.2.18)$$

特殊地,当  $\tau_{kl}(m') < \frac{1}{L}$  时,有  $\tau_{kl}(m'+1) > \tau_{kl}(m')$ ; 而当  $\tau_{kl}(m') \geq \frac{1}{L}$  时,由公式(3.2.17),则有  $\tau_{kl}(m'+1) \geq \frac{1}{L}$ 。换言之,在情况(2)条件下,对于  $(k,l) \in w^*$ ,  $\tau_{kl}(m')$  要么增加,要么存在  $\tau_{kl}(m') \geq \frac{1}{L}$ 。

综合情况(1)和情况(2)可知,当  $(k,l) \in w^*$  且  $m' > m$  时,有

$$\tau_{kl}(m') \geq \min \left( \tau_{kl}(m+1), \frac{1}{L} \right) \geq \alpha_{kl}(m) > 0$$

式中,  $\alpha_{kl}(m) = (1-\rho)^m \tau_{kl}(1)$ 。令  $(k,l) \in w^*$ ,  $u = (u_0, u_1, \dots, u_{t-1} = k)$  为最优路径  $w^*$  上从起始节点到节点  $k$  的部分路径,则由式(3.2.8)和式(3.2.13)可得

$$p_{kl}(m', u) \geq \gamma \tau_{kl}(m') \geq \gamma \alpha_{kl}(m)$$

由此,对于固定的蚂蚁智能体  $A_S$ ,当  $m' > m$  时,有

$$\Pr(E_m^{(s)} | F_m) \geq \prod_{(k,l) \in w^*} \gamma \alpha_{kl}(m) \geq \gamma^L \prod_{(k,l) \in w^*} \alpha_{kl}(m) = a(m) > 0$$

式中,  $a(m)$  不依赖于  $m'$ 。

类似于推论 3.2.1 和引理 3.2.1 的证明,可推知在不考虑搜索周期  $m+1, m+2, \dots, m'-1$  搜索情况的条件下,上述估计成立,即当所考虑的概率对任意搜索周期内的任意可能事件有附加条件时,该结论亦成立。很显然,搜索周期  $m$  后的  $g$  个固定的后继搜索周期内,没有蚂蚁智能体穿越最优路径  $w^*$  的概率小于或等于  $(1-a(m))^{S_g}$ 。通过选择充分大的  $g$ ,可使该概率任意小。由此可知,情况(2)中事件  $F_m$  在搜索周期  $m$  和搜索周期  $m+d$  之间发生的条件概率要低于  $h$  次,且通过选择充分大的  $d$ ,可使该概率任意小。选择  $0 < \delta < \frac{1}{2}$ ,使得  $g(\delta)$  满足

$$\Pr\{m \text{ 个搜索周期后的 } g(\delta) \text{ 个搜索周期之内没有蚂蚁智能体穿越 } w^* | F_m\} \leq \delta$$

对于任意的  $0 < \epsilon < 1$ , 设  $\delta = \epsilon/2h < \frac{1}{2}$ , 则情况(2)中事件  $F_m$  在每个搜索周期  $h$  (由  $g(\delta)$  个后继搜索周期组成) 至少发生一次的条件概率大于  $(1 - \delta)^h \geq 1 - 2\delta h = 1 - \epsilon$ 。因此在  $d = g(\delta)h$  个搜索周期内, 情况(2)中事件  $F_m$  至少发生  $h$  次的条件概率至少为  $1 - \epsilon$ 。

通过  $(1 - \rho)^h < \epsilon$  来选择  $h = h(\epsilon)$ , 而通过下式来选择  $d = d(\epsilon, m)$

$$\Pr\{\text{在搜索周期 } m \text{ 和 } m + d \text{ 之间, 情况(2)发生少于 } h \text{ 次} | F_m\} \leq \epsilon$$

针对情况(2)中的指针  $m'$ , 当  $(k, l) \in w^*$  时, 由公式(3.2.17)可得

$$\tau_{kl}(m' + 1) - \frac{1}{L} = (1 - \rho)\left(\tau_{kl}(m') - \frac{1}{L}\right)$$

即对于某一路径弧段而言, 情况(2)每发生一次,  $\tau_{kl}(m')$  和  $1/L$  之差以  $1 - \rho$  的倍数递减一次; 而情况(1)每发生一次,  $\tau_{kl}(m')$  和  $1/L$  之间的距离保持不变。若重复迭代情况(2)下的这一过程, 则事件  $F_m$  发生的条件概率大于或等于  $1 - \epsilon$ , 进而有

$$\left| \tau_{kl}(m + d) - \frac{1}{L} \right| \leq (1 - \rho)^h \left| \tau_{kl}(m) - \frac{1}{L} \right| \leq (1 - \rho)^h < \epsilon$$

因此, 对于所有的  $(k, l) \in w^*$  和任意  $m' \geq m + d$ , 有

$$\left| \tau_{kl}(m') - \frac{1}{L} \right| \leq \left| \tau_{kl}(m + d) - \frac{1}{L} \right| < \epsilon$$

由此, 该引理的第一部分得到了证明。

该引理第二部分的推导可由下式得出, 即

$$\sum_{(k', l') \in A} \tau_{k'l'}(m') = 1$$

根据该引理的第一部分可知, 在事件  $F_m$  发生的条件概率大于或等于  $1 - \epsilon$  条件下, 对于  $m' \geq m + d$ , 最优路径  $w^*$  上的信息素总量大于

$$L\left(\frac{1}{L} - \epsilon\right) = 1 - L\epsilon$$

因此, 不在最优路径  $w^*$  上的路径弧段信息量不会超过  $L\epsilon$ 。

[证毕]

**引理 3.2.3** 令  $u^*(k)$  为最优路径  $w^*$  上与节点  $k (k \in w^*)$  相连的部分路径, 则对于每一个  $\epsilon > 0$  且  $m \in \mathbb{N}$ , 存在一个整数  $d'(\epsilon, m) \in \mathbb{N}$ , 当  $m' \geq m + d'(\epsilon, m)$  时, 有

$$\Pr\{p_{kl}(m', u^*(k)) \geq 1 - \epsilon, \text{对所有的 } (k, l) \in w^* | F_m\} \geq 1 - \epsilon \quad (3.2.19)$$

**证明** 由引理 3.2.2, 当  $m' \geq m + d(\epsilon, m)$ , 且  $(k, l) \in w^*$  时, 有

$$\left| \tau_{kl}(m') - \frac{1}{L} \right| \leq \epsilon \quad (3.2.20)$$

当  $m' \geq m + d(\epsilon, m)$ , 且  $(k, l) \notin w^*$  时, 有

$$\tau_{kr}(m') \leq L\tilde{\epsilon} \quad (3.2.21)$$

令  $(k, l) \in w^*$ , 且  $u = u^*(k)$ , 则可得

$$p_{kl}(m', u) = \frac{\tau_{kl}(m')[\eta_{kl}(u)]^\beta}{\sum_{r \notin u, r \neq l, r \in A} \tau_{kr}(m')[\eta_{kr}(u)]^\beta + \tau_{kl}(m')[\eta_{kl}(u)]^\beta}$$

令  $\eta = [\eta_{kl}(u)]^\beta > \gamma$  (由公式(3.2.10)可得  $\eta \leq 1$ ), 且  $v$  表示集合  $C$  中一个节点的最大输出端数, 则由公式(3.2.20)、式(3.2.21)可得

$$p_{kl}(m', u) \geq \frac{(1/L - \tilde{\epsilon})\eta}{vL\epsilon + (1/L + \tilde{\epsilon})\eta} = \frac{1 - L\tilde{\epsilon}}{1 + \tilde{\epsilon}(vL^2\eta + L)}$$

根据当  $x \geq 0$  时, 有  $\frac{1}{1+x} \geq 1+x$ , 从而

$$\begin{aligned} p_{kl}(m', u) &\geq (1 - L\tilde{\epsilon}) \left( 1 - \tilde{\epsilon} \left( \frac{vL^2}{\eta} + L \right) \right) \geq 1 - \left( 2L + \frac{vL^2}{\eta} \right) \tilde{\epsilon} \\ &\geq 1 - \left( 2L + \frac{vL^2}{\gamma} \right) \tilde{\epsilon} \end{aligned}$$

令  $\tilde{\epsilon} = \frac{\epsilon}{2L + vL^2/\gamma} < \epsilon$ , 并将其代入上式, 原命题即可得证。

[证毕]

**推论 3.2.2** 由引理 3.2.3 中的一些概念, 令

$$Y_{m'} = \prod_{(k, l) \in w^*} p_{kl}(m', u^*(k)) \quad (3.2.22)$$

则对于  $\forall \epsilon > 0$  且  $m \in \mathbb{N}$ , 都存在一个整数  $d''(\epsilon, m) \in \mathbb{N}$ , 使得当  $m' \geq m + d''(\epsilon, m)$  时, 有

$$\Pr\{Y_{m'} \geq 1 - \epsilon | F_m\} \geq 1 - \epsilon \quad (3.2.23)$$

**证明** 由引理 3.2.3, 用  $\epsilon/2L$  替代  $\epsilon$ , 由此可知, 除了事件  $F_m$  发生的条件概率最多为  $\epsilon/2L < \epsilon$  这一情况之外, 公式(3.2.22)中的每个因子均大于或等于  $1 - \epsilon/2L$  (不失一般性,  $\epsilon \leq 1$ ), 由于  $\epsilon/2L \leq \frac{1}{2}$ , 从而

$$\left(1 - \frac{\epsilon}{2L}\right)^L \geq 1 - 2\left(\frac{\epsilon}{2L}\right)L = 1 - \epsilon$$

[证毕]

**引理 3.2.4** 对于  $\forall \epsilon > 0$ , 都存在一个整数  $d'''(\epsilon, m) \in \mathbb{N}$ , 则当  $m' \geq m + d'''(\epsilon, m)$  时, 对于固定的  $S$ , 有

$$\Pr(E_{m'}^{(s)} | F_m) \geq 1 - \epsilon \quad (3.2.24)$$

**证明** 定义事件  $w^{(s)}(m') = w^*$  为  $E_{m'}^{(s)}$ , 对于 Markov 过程中搜索周期  $m' - 1$  内的固定状态而言, 该事件的概率是公式(3.2.22)所定义的  $Y_{m'}$  值, 而  $Y_{m'}$  是具有特定分布的随机变量。因此, 为了在不确定先前状态的前提下得到(条件)概率  $E_{m'}^{(s)}$ , 这里可取与  $Y_{m'}$  分布相关的期望值。特别地, 当  $m' \geq m + d''(\epsilon, m)$  时, 由推论

3.2.2 及引理 3.2.3 可得

$$\Pr\{Y'_m \geq 1 - \tilde{\epsilon} | F_m\} \geq 1 - \tilde{\epsilon}$$

因此,对于上述的  $m'$ ,可得

$$\begin{aligned}\Pr(E_{m'}^{(s)} | F_m) &\geq \Pr\{E_{m'}^{(s)} \wedge (Y_{m'} \geq 1 - \tilde{\epsilon}) | F_m\} \\ &= \Pr\{E_{m'}^{(s)} | (Y_{m'} \geq 1 - \tilde{\epsilon}) \wedge F_m\} \Pr\{Y_{m'} \geq 1 - \tilde{\epsilon} | F_m\} \\ &\geq (1 - \tilde{\epsilon})(1 - \tilde{\epsilon}) \geq 1 - 2\tilde{\epsilon}\end{aligned}$$

令  $\tilde{\epsilon} = \epsilon/2$ ,并将其代入上式,即可证得该引理。

[证毕]

在上述研究的基础上,可证明定理 3.2.1。

**定理 3.2.1 的证明** 根据  $P_m = \Pr(E_m^{(1)}) = \Pr(E_m^{(2)}) = \dots = \Pr(E_m^{(s)})$ ,可得

$$\Pr(B_1 \wedge \dots \wedge B_m) = \Pr(B_1)\Pr(B_2 | B_1)\dots\Pr(B_m | B_1 \wedge \dots \wedge B_{m-1})$$

进而,由引理 3.2.1 及推论 3.2.1,可得

$$\begin{aligned}\Pr(B_1 \wedge B_2 \wedge \dots \wedge B_m) &\leq (1-p)^s(1-cp)^s(1-c^2p)^s\dots(1-c^{m-1}p)^s \\ &= \left[ \prod_{i=1}^m (1-c^{i-1}p) \right]^s \quad (3.2.25)\end{aligned}$$

设  $w(p, c, S) = \left[ \prod_{i=1}^{\infty} (1-c^{i-1}p) \right]^s$ , 事件  $A$  可表示为  $A = \neg(B_1 \wedge B_2 \dots)$ ,

从而

$$\begin{aligned}\Pr(A) &= 1 - \lim_{m \rightarrow \infty} \Pr(B_1 \wedge B_2 \wedge \dots \wedge B_m) \\ &\geq 1 - \lim_{m \rightarrow \infty} \left[ \prod_{i=1}^m (1-c^{i-1}p) \right]^s = 1 - w(p, c, S)\end{aligned}$$

另一方面,根据当  $0 < c < 1$  时,有  $\lg x \leq x - 1$ ,因此,可通过选择充分大的  $S$  或充分小的  $p$  使  $w(p, c, S)$  任意小。

$$\lg w(p, c, S) = S \sum_{i=1}^{\infty} \lg(1 - c^{i-1}p) \leq -S \sum_{i=1}^{\infty} c^{i-1}p = -Sp \sum_{i=0}^{\infty} c^i = -\frac{Sp}{1-c}$$

即

$$w(p, c, S) \leq \exp\left(-\frac{Sp}{1-c}\right)$$

由于  $0 < c < 1$ ,所以,当  $c$  固定且  $S \rightarrow \infty$  时,或当  $S$  固定且  $c \rightarrow 1$ (即  $p \rightarrow 0$ )时,上述表达式趋于 0。

综上可见,通过选择适当的  $S$  或  $p$ ,可使  $w(p, c, S) \leq \epsilon/4$ ,从而有

$$\Pr(A) \geq 1 - \epsilon/4$$

由于

$$\Pr(A) = \Pr(F_1 \vee F_2 \vee \dots) = \sum_{m=1}^{\infty} \Pr(F_m) < 1$$

即当  $K \rightarrow \infty$  时, 上述级数的部分和  $\sum_{m=1}^K \Pr(F_m)$  收敛。从而, 存在一整数  $K = K(\epsilon)$ , 使得

$$\sum_{m=K+1}^{\infty} \Pr(F_m) < \frac{\epsilon}{4}$$

故有

$$\Pr(F_1 \vee F_2 \vee \cdots \vee F_K) = \sum_{m=1}^K \Pr(F_m) \geq \Pr(A) - \frac{\epsilon}{4} \geq 1 - \frac{\epsilon}{2}$$

令  $d(\epsilon) = \max\left(d''\left(\frac{\epsilon}{2}, 1\right), \dots, d''\left(\frac{\epsilon}{2}, K\right)\right)$ , 且  $m_0 = m_0(\epsilon) = K + d(\epsilon)$ , 则由引理 3.2.4, 当  $m \leq K$  且  $m' \geq m_0$  时, 对于所有的  $m' \geq m + d''(\epsilon/2, m)$ , 有

$$\Pr(E_{m'}^{(1)} | F_m) \geq 1 - \frac{\epsilon}{2} \quad (3.2.26)$$

因此, 对于所有的  $m' \geq m_0$ , 有

$$\begin{aligned} P_{m'} &= \Pr(E_{m'}^{(1)}) \\ &= \Pr(E_{m'}^{(1)} | F_1) \Pr(F_1) + \Pr(E_{m'}^{(1)} | F_2) \Pr(F_2) + \cdots + \Pr(E_{m'}^{(1)} | F_K) \Pr(F_K) \\ &\quad + \Pr(E_{m'}^{(1)} | \neg(F_1 \vee F_2 \vee \cdots \vee F_K)) \Pr(\neg(F_1 \vee F_2 \vee \cdots \vee F_K)) \\ &\geq \Pr(E_{m'}^{(1)} | F_1) \Pr(F_1) + \Pr(E_{m'}^{(1)} | F_2) \Pr(F_2) + \cdots + \Pr(E_{m'}^{(1)} | F_K) \Pr(F_K) \\ &\geq \left(1 - \frac{\epsilon}{2}\right) (\Pr(F_1) + \Pr(F_2) + \cdots + \Pr(F_K)) \\ &\geq \left(1 - \frac{\epsilon}{2}\right) \left(1 - \frac{\epsilon}{2}\right) \\ &\geq 1 - 2\left(\frac{\epsilon}{2}\right) = 1 - \epsilon \end{aligned}$$

[证毕]

**评注 3.2.3** 上述证明结论所蕴涵的理论界限并不代表着实际应用中能得到“蚂蚁智能体数目”和“信息素挥发因子”的合理配置。例如, 要通过如下不等式选择足够大数目  $S$  的蚂蚁智能体使其满足条件  $\Pr(A) \geq 1 - \epsilon$ :

$$\Pr(A) \geq 1 - w(p, c, S) \geq 1 - \exp\left(-\frac{Sp}{1-c}\right) \quad (3.2.27)$$

此即意味着  $S \geq \frac{(1-c)(-\lg \epsilon)}{p}$  往往是一个很大的数。这是由于如引理 3.2.1 所示,  $p$  很小(例如, 对于某一由 20 个弧段组成、最优路径为 10 且  $\gamma=1$  的构造图而言,  $p=0.05^{10}$ )。同样地, 如果  $S$  固定, 一个大的  $p$  则需要一个极小的信息素挥发系数  $\rho$ 。

值得注意的是,  $p$  的极小值源于对蚂蚁智能体所发现最优路径概率的粗略估计。在实际应用中, 信息启发值往往取较大值, 从而使至少有一个蚂蚁智能体在某

一个搜索周期中发现最优路径的几率得到充分改善。基于这些分析,GBAS 以较高概率收敛到全局最优解可通过选择适当数目的蚂蚁智能体来实现。

算法分析中,区分“能优化”启发式和“不能优化”启发式这两个概念显得非常重要。“能优化”启发式意味着只要有更高性能的计算机,就不会存在根本性的困难来阻止对全局最优解的求解;而“不能优化”启发式是指不管有性能多么优越的计算机硬件,所得解的质量距全局最优解总会有一定的差距。本节所研究的 GBAS 属于“能优化”启发式算法。

### 3.3 一类改进蚁群算法的收敛性证明

针对具有组合优化性质的极小化问题,Stüezle T 和 Dorigo M 提出了一类改进蚁群算法—— $\text{ACO}_{gb, \tau_{\min}}$  算法,并对其收敛性进行了理论证明<sup>[3, 26]</sup>,该算法对信息量设置了下界  $\tau_{\min}$ ,可有效地改善蚁群算法的全局收敛性能。

#### 3.3.1 极小化问题的表示与 $\text{ACO}_{gb, \tau_{\min}}$ 算法的描述

考虑极小化问题  $(S, f, \Omega)$ ,其中  $S$  表示(候选)解集,  $f$  表示目标函数,而  $f(s)$  表示候选解  $s \in S$  的目标函数(代价)值,  $\Omega$  表示可行候选解约束条件的集合<sup>[28, 29]</sup>。极小化问题的目的是寻找一个最优解  $s^*$ ,即寻找一个具有最小代价值的可行候选解。

可将具有组合优化性质的极小化问题  $(S, f, \Omega)$  映射为具有如下特征的问题:

- (1) 一个包含该问题元素的有限集合  $C = \{c_1, c_2, \dots, c_{N_C}\}$ 。
- (2) 一个包含该问题状态的有限集合  $X$ ,其元素由集合  $C$  中所有可能的序列  $x = \langle c_i, c_j, \dots, c_k, \dots \rangle$  组成。序列的长度(即序列中所包含元素的数目)用  $|x|$  表示,而序列的最大长度用一个  $n < \infty$  的正常数界定。
- (3) (候选)解集  $S$  是有限集合  $X$  的一个子集,即  $S \subseteq X$ 。
- (4) 一个可行状态集合  $\tilde{X}$ ,且  $\tilde{X} \subseteq X$ , $\tilde{X}$  依赖于具体问题,且已经证明完全可由序列  $x \in \tilde{X}$  计算出满足约束条件  $\Omega$  的解。
- (5) 一个非空的最优解集合  $S^*$ ,且满足  $S^* \subseteq \tilde{X}$  和  $S^* \subseteq S$ 。

在上述表述的基础上,蚂蚁通过在完全连接且带权重的构造图  $G = (C, L, T)$  上随机移动而产生候选解,图  $G = (C, L, T)$  中的节点为集合  $C$  所包含的元素,集合  $L$  是连接集合  $C$  中所有元素的集合,而集合  $T$  为聚集着信息素轨迹的向量。

每只蚂蚁被安置在构造图中随机选择的节点上,然后这些蚂蚁会根据目前所在路径弧段上的信息量从构造图中的某一节点移动到另一节点。在蚂蚁移动的过程中,约束条件  $\Omega$  用来避免蚂蚁生成非可行解。一旦蚂蚁完成路径搜索,信息素

即进行更新。这里，蚂蚁构造解的过程可描述如下：

```

While ( $x_k \in \tilde{X}$  且  $x_k \notin S$ ) do
{
    对于构建序列  $x_k = \langle c_1, c_2, \dots, c_k \rangle$  后的每一步  $k$ ，按照下式随机选择后继元素  $c_{k+1}$ 

    
$$P(c_{k+1} = c | T, x_k) = \begin{cases} \frac{\tau(c_k, c)^a}{\sum_{y \in C, (c_k, y) \in J_{c_k}} \tau(c_k, y)^a}, & \text{若 } (c_k, c) \in J_{c_k} \\ 0, & \text{否则} \end{cases}$$

}

```

其中， $a \subset (0, +\infty)$ ， $(c_k, y) \in J_{c_k}$ ，序列  $x_{k+1} = \langle c_1, c_2, \dots, c_k, y \rangle$ ，且  $x_{k+1} \in \tilde{X}$ 。若在构造图的某一点上集合  $J_{c_k}$  为空，则终止解的构造过程。

一旦所有蚂蚁都完成上述解的构造之后，则立即对信息素进行更新。令  $\hat{s}$  为算法到目前为止所发现的最好可行解，而  $s_t$  为算法在第  $t$  次迭代所发现的最好可行解， $f(\hat{s})$  和  $f(s_t)$  分别表示与其相对应的目标函数。集合  $L$  上所有路径弧段的信息量将以  $\rho$  的倍数挥发一部分，而算法到目前为止所发现的最好可行解  $\hat{s}$  上的信息素得以增强。信息素的更新过程可描述如下：

```

 $\forall (i, j) : \tau(i, j) \leftarrow (1 - \rho) \cdot \tau(i, j)$ 
If ( $x_k \in \tilde{x}$  且  $x_k \notin S$ ) then  $\hat{s} \leftarrow s_t$ 
 $\forall (i, j) \in \hat{s} : \tau(i, j) \leftarrow \tau(i, j) + g(\hat{s})$ 
 $\forall (i, j) : \tau(i, j) \leftarrow \max\{\tau_{\min}, \tau(i, j)\}$ 

```

其中， $\rho \subset (0, 1)$ ， $\tau_{\min} > 0$ ， $0 < g(s) < +\infty$  且  $g: SR^+ \rightarrow \mathbb{R}$ ， $f(s) < f(s') \Rightarrow g(x) \geq g(s')$ 。  
ACO <sub>$g$ ,  $\tau_{\min}$</sub>  算法的初始化过程如下：

```

产生可行解  $s'$ ，并令  $s = s'$ 
对于  $\forall (i, j)$ ，令  $\tau(i, j) = \tau_0$ 
For 每只蚂蚁
    根据某一具体问题选择起始节点  $c_1$ 
    令  $k = 1$  且  $x_k = \langle c_1 \rangle$ 

```

其中， $\tau_{\min} \leq \tau_0 < +\infty$ 。一旦初始化完成之后，算法即按照“构造解”和“信息素更新”这两个过程不断迭代，直到满足终止条件为止，这里将上述算法称之为 ACO <sub>$g$ ,  $\tau_{\min}$</sub>  算法。在该算法中，可通过设置  $\tau_0 < g(s')/2$  使得  $\tau_{\min} < g(s^*)$ ，其中  $s'$  为该算法的初始解。

### 3.3.2 ACO <sub>$g$ , $\tau_{\min}$</sub> 算法的收敛性证明

本小节首先证明了如果 ACO <sub>$g$ ,  $\tau_{\min}$</sub>  算法运行时间足够长，则可保证该算法以无

限接近于1的概率收敛到最优解;其次证明了经过一定的迭代次数 $t_0$ 之后,若已经搜索到最优解,则与该最优解所对应路径弧段上的信息量大于其他任意路径弧段上的信息量;最后对该结论进行了拓展,证明了构建最优解的概率大于 $1 - \hat{\epsilon}(\tau_{\min}, \tau_{\max})$ 。

为了证明上述结论,首先引入了如下两个引理:

**引理 3.3.1** 设信息量 $\tau_{ij}(t)$ 的全局最大值为 $\tau_{\max}$ ,则对于 $\forall \tau_{ij}$ 有下式成立

$$\lim_{t \rightarrow \infty} \tau_{ij}(t) \leq \tau_{\max} = \frac{1}{\rho} \cdot g(s^*) \quad (3.3.1)$$

**证明** 假定每次迭代后,任意路径弧段 $(i, j)$ 上信息量不会超过 $g(s^*)$ 。显然,在第一次迭代后,最大可能的信息量为 $(1 - \rho) \cdot \tau_0 + g(s^*)$ ;第2次迭代后,则为 $(1 - \rho)^2 \cdot \tau_0 + (1 - \rho) \cdot g(s^*) + g(s^*)$ ;其余依此类推。因此,由于信息素的挥发,在第 $t$ 次迭代后,信息量的上限值为

$$\tau_{ij}^{\max}(t) = (1 - \rho)^t \cdot \tau_0 + \sum_{i=1}^t (1 - \rho)^{t-i} \cdot g(s^*)$$

由此,当 $\rho \subset (0, 1)$ 时,其和将最终收敛到

$$\tau_{\max} = \frac{1}{\rho} \cdot g(s^*)$$

[证毕]

**引理 3.3.2** 设信息量 $\tau_{ij}(t)$ 的全局最小值为 $\tau_{\min}$ ,若采用全局最优信息素更新规则,对于 $\forall (i, j) \subset s^*$ 且 $\tau_{ij}^*(t) \geq \tau_{\min}$ ,则在搜索到最优解 $\tau_{ij}^*(t)$ 后,该最优解 $\tau_{ij}^*(t)$ 会单调增加,且有

$$\lim_{t \rightarrow \infty} \tau_{ij}^*(t) = \tau_{\max} = \frac{1}{\rho} \cdot g(s^*) \quad (3.3.2)$$

引理 3.3.2 的证明是引理 3.3.1 证明过程的重复,这里不再赘述,只不过是将引理 3.3.1 中的 $\tau_0$ 用 $\tau_{ij}^*(t^*)$ 代替。其中, $t^*$ 是首次发现最优解时的迭代次数。

**定理 3.3.1** 设 $P^*(t)$ 为 $t$ 次迭代内算法首次发现最优解 $s^*$ 的概率,则对于任意小的 $\epsilon > 0$ 和充分大的迭代次数 $t$ ,有

$$P^*(t) \geq 1 - \epsilon \quad (3.3.3)$$

且

$$\lim_{t \rightarrow \infty} P^*(t) = 1 \quad (3.3.4)$$

**证明** 由于信息量 $\tau_{ij}(t)$ 的值被限制在 $\tau_{\min}$ 和 $\tau_{\max}$ 之间,因此在蚂蚁构造解的过程中,状态转移概率的可行性选择为 $p_{\min} > 0$ ,且有

$$p_{\min} \geq \hat{p}_{\min} = \frac{\tau_{\min}^a}{(N_c - 1) \cdot \tau_{\max}^a + \tau_{\min}^a} \quad (3.3.5)$$

式中, $N_c$ 表示集合 $C$ 中的元素个数。从而,对于任意解 $s'$ (包括任意最优解 $s^* \in S^*$ )均可以 $\hat{p} \geq \hat{p}_{\min} > 0$ 的概率产生,其中 $n$ 表示序列的最大长度。假定只要能有一只蚂蚁搜索到最优解即可满足要求,由此, $P^*(t)$ 的一个下界为

$$P^*(t) = 1 - (1 - \hat{p})^t$$

对于任意小的  $\epsilon > 0$ , 当  $t$  足够大时, 有

$$P^*(t) > 1 - \epsilon$$

从而, 当  $t \rightarrow \infty$  时, 有

$$\lim_{t \rightarrow \infty} P^*(t) = 1$$

[证毕]

**定理 3.3.2** 设  $t^*$  为首次发现最优解  $s^*$  时的迭代次数, 对于  $\forall (i, j) \in s^*$ ,  $\forall (k, l) \in L \wedge (k, l) \notin s^*$  且  $\forall t > t^* + t_0 = t^* + [(1 - \rho)/\rho]$ , 则存在一个  $t_0$ , 使得

$$\tau_{ij}(t) > \tau_{kl}(t) \quad (3.3.6)$$

**证明** 首次发现最优解后的  $t_0$  次迭代之后(即  $t > t^* + t_0$ ), 最优路径弧段上的信息量大于其他可行路径弧段上的信息量。事实上, 由于该算法采用了全局信息素更新规则, 这意味着只有属于最优路径  $s^*$  上的信息量得到增强, 而其他路径弧段上的信息量在每次迭代之后以  $\rho$  的倍数减少, 直至减少到下界值  $\tau_{\min}$ 。

这里以最恶劣的情况为例, 令第  $t^*$  次迭代后, 与最优解  $s^*$  所对应路径弧段  $(i, j)$  的信息量为  $\tau_{ij}^*(t^*) = \tau_{\min}$ , 而令不与最优解  $s^*$  所对应路径弧段  $(k, l)$  的信息量为  $\tau_{kl}^*(t^*) = \tau_{\max}$ , 则在  $t^* + t'$  次迭代时,  $\tau_{ij}^*(t)$  变为

$$\begin{aligned} \tau_{ij}^*(t^* + t') &= (1 - \rho)^{t'} \cdot \tau_{\min} + \sum_{i=0}^{t'-1} (1 - \rho)^i \cdot g(s^*) \\ &> t' \cdot (1 - \rho)^{t'-1} \cdot g(s^*) \end{aligned} \quad (3.3.7)$$

此外, 在  $t^* + t'$  次迭代时,  $\tau_{kl}(t)$  变为

$$\tau_{kl}(t^* + t') = \max\{\tau_{\min}, (1 - \rho)^{t'} \cdot \tau_{\max}\} \quad (3.3.8)$$

由此, 欲使  $\tau_{ij}^*(t^* + t') > \tau_{kl}(t^* + t')$ , 只需

$$t' \cdot (1 - \rho)^{t'-1} \cdot g(s^*) > (1 - \rho)^{t'} \cdot \tau_{\max}$$

即当

$$t' > \frac{\tau_{\max} \cdot (1 - \rho)}{g(s^*)} = \frac{1 - \rho}{\rho} \equiv t_0$$

时, 该定理得证。

[证毕]

由定理 3.3.2 显而易见, 当  $t > t^* + t_0$  时, 任意蚂蚁都能以确定的概率构造出最优解  $s^*$ 。

**定理 3.3.3** 一旦搜索到最优路径, 对于满足条件  $(i, j) \notin S^*$  的任意  $\tau_{ij}(t)$ , 则有

$$\lim_{t \rightarrow \infty} \tau_{ij}(t) = \tau_{\min} \quad (3.3.9)$$

**证明** 由于一旦搜索到最优路径之后, 与非最优解相对应的路径弧段不再接受新的信息素, 且其上的信息量不断减少。令  $t^*$  为首次发现最优解  $s^*$  时的迭代次

数,则算法经过一次迭代后,有

$$\tau_{ij}(t^* + 1) = \max\{\tau_{\min}, (1 - \rho)\tau_{\max}\}$$

类似地,经过  $t'$  次迭代后,有

$$\tau_{ij}(t^* + t') = \max\{\tau_{\min}, (1 - \rho)^{t'}\tau_{\max}\}$$

很明显,当  $t \rightarrow \infty$  时,有  $\tau_{ij}(t) \rightarrow \tau_{\min}$ 。

[证毕]

**定理 3.3.4** 令  $t_0 = \frac{\ln \tau_{\min} - \ln \tau_{\max}}{\ln(1 - \rho)}$ ,  $t^*$  表示首次发现最优解  $s^*$  时的迭代次数,对于  $\forall (i, j) \notin s^*$ , 则从迭代次数  $t'(t' \geq t^* + t_0)$  开始, 有

$$\tau_{ij}(t) = \tau_{\min} \quad (3.3.10)$$

**证明** 这里可假定在  $t^*$  次迭代中至少有一个路径弧段  $(i, j) \notin s^*$ , 同时给迭代次数  $t_0$  定义一个界限, 令  $\tau_{ij}(t^*) = \tau_{\max}$ 。由定理 3.3.3 的推导过程可知, 经过  $t'$  次迭代后, 有

$$\tau_{ij}(t^* + t') = \max\{\tau_{\min}, (1 - \rho)^{t'}\tau_{\max}\}$$

设  $t_0$  为首次满足条件  $(1 - \rho)^{t_0} \cdot \tau_{\max} \leq \tau_{\min}$  时的迭代次数, 则当  $t_0 = \frac{\ln \tau_{\min} - \ln \tau_{\max}}{\ln(1 - \rho)}$  时, 该定理得证。

[证毕]

**推论 3.3.1** 设  $t^*$  表示首次发现最优解  $s^*$  时的迭代次数,  $P(s^*, t, k)$  表示任意蚂蚁  $k$  在  $t$  次迭代内构建最优解  $s^*$  的概率, 当  $t > t^*$  时, 则有

$$\lim_{t \rightarrow \infty} P(s^*, t, k) \geq 1 - \hat{\epsilon}(\tau_{\min}, \tau_{\max}) \quad (3.3.11)$$

**证明** 令蚂蚁  $k$  位于构造图中的节点  $i$ , 且  $(i, j)$  为与最优解  $s^*$  相对应的路径弧段。由于蚂蚁是从集合  $J_i$  中选择下一节点  $j$ , 则蚂蚁  $k$  “正确选择”路径弧段  $(i, j)$  的概率下界为

$$\hat{p}_{ij}^*(t) = \frac{(\tau_{ij}^*(t))^a}{(\tau_{ij}^*(t))^a + \sum_{(i,k) \notin S^*} (\tau_{ik}^*(t))^a}$$

由引理 3.3.2 和定理 3.3.3 可得

$$\hat{p}_{ij}^* = \lim_{t \rightarrow \infty} \hat{p}_{ij}^*(t) = \frac{\lim_{t \rightarrow \infty} (\tau_{ij}^*(t))^a}{\lim_{t \rightarrow \infty} \left[ (\tau_{ij}^*(t))^a + \sum_{(i,k) \notin S^*} (\tau_{ik}^*(t))^a \right]} = \frac{\tau_{\max}^a}{\tau_{\max}^a + (N_C - 1) \cdot \tau_{\min}^a}$$

因此,  $P(s^*, t, k)$  的下界极限值为  $\hat{p}_k^* = (\hat{p}_{ij}^*)^n$ 。令  $\hat{\epsilon} = 1 - \hat{p}_k^*$  即可证得该推论。

[证毕]

### 3.3.3 讨论

本小节将对 ACO <sub>$\phi, \tau_{\min}$</sub>  算法收敛性定理(重点针对定理 3.3.1 和定理 3.3.2)

及推论的深层含义做进一步说明和讨论,同时对  $\text{ACO}_{gb,\tau_{\min}}$  算法收敛性证明与 GBAS 收敛性证明之间的关系进行深入分析。

### 3.3.3.1 $\text{ACO}_{gb,\tau_{\min}}$ 算法的几个定理和推论的深层含义

由前述可见,定理 3.3.1 主要证明了所提出的  $\text{ACO}_{gb,\tau_{\min}}$  算法不能排除最终搜索到最优解的可能性;而定理 3.3.2 主要证明了最优解一旦被搜索到,则与最优解相对应路径弧段上的信息量大于其他任意路径弧段上的信息量。在这两个定理的基础上,推论 3.3.1 给出了构建最优解的概率界。但是,这些证明并没有给出搜索到最优解所花费的时间,而这个时间往往非常长。

这里还需要强调的一点是  $\text{ACO}_{gb,\tau_{\min}}$  算法信息素更新过程中的严格不等式。假如用“ $\leq$ ”替代信息素更新过程中的“ $<$ ”,则对于具有多个不同全局最优解的优化问题而言,会导致多个不同全局最优解的交替出现。尽管这样不会影响定理 3.3.1,但是在这种假设下定理 3.3.2 将不再成立。有趣的是,所有采用信息素全局更新规则的蚁群算法都采用了这种严格不等式,即只有当更好的解被搜索到之后,方可进行信息素全局更新。

$\tau_{\min}$  和  $\tau_{\max}$  在定理 3.3.1 的证明中扮演着非常重要的角色,即  $\tau_{\max}/\tau_{\min}$  的比值越大,则证明中所引用  $\hat{p}_{\min}$  的下界越大。若  $\hat{p}_{\min}$  越大,则在恶劣情况下对用来保证以大于  $1-\epsilon$  的概率搜索到最优解所需迭代次数  $t$  的估计值就越小。实际上,当所有的信息量都相等时,将得到最窄的界限范围。这种在某种程度上违反直觉的结论是由于本节的收敛性证明是建立在对最恶劣情况分析基础上的,而这里有必要考虑在构造解的过程中可能会产生次优解这一情况的发生,即不得不在最恶劣的情况下,假定蚂蚁在构建最优解过程中与  $\tau_{\max}$  相对应的信息量下界为  $\tau_{\min}$  时的情况。实际上,本节收敛性定理主要证明了如果  $\text{ACO}_{gb,\tau_{\min}}$  算法的运行时间足够长,则可保证该算法以较高的概率收敛到全局最优解。

### 3.3.3.2 $\text{ACO}_{gb,\tau_{\min}}$ 算法收敛性证明与 GBAS 收敛性证明的关系

由前述分析可知,就算法本身而言,GBAS 除了“ $\tau_{\min}=0$ ”和“只有目前最优路径上的信息素得到增强”这两个方面与  $\text{ACO}_{gb,\tau_{\min}}$  算法不同之外,其他基本类同于  $\text{ACO}_{gb,\tau_{\min}}$  算法。而就对算法收敛性的证明思路而言,GBAS 和  $\text{ACO}_{gb,\tau_{\min}}$  算法存在着许多不同之处。实际上,在对  $\text{ACO}_{gb,\tau_{\min}}$  算法收敛性定理 3.3.1 的证明过程中采用了数值法的证明思路(即证明了该算法终将会收敛到最优解),GBAS 的收敛性证明则采用了算法解的证明思路(即证明了该算法将回归到一种产生最优解的状态); $\text{ACO}_{gb,\tau_{\min}}$  算法收敛性定理的结论适用于任何满足  $\tau_{\min}>0$  和  $\tau_{\max}<+\infty$  条件

的蚁群算法,而 GBAS 的收敛性证明仅适用于特定的 GBAS。

综合分析,可归纳出这两种算法的证明主要存在着如下五点不同:

(1)  $\text{ACO}_{gb, \tau_{\min}}$  算法中收敛性定理 3.3.1 的成立不依赖于信息素更新方式(确切而言,只要信息素更新规则中的  $\rho \in (0, 1)$ ,且信息素增量为有界值,定理 3.3.1 即成立);而 GBAS 的收敛性定理的证明是建立在 GBAS 中信息素更新规则基础上的。

(2)  $\text{ACO}_{gb, \tau_{\min}}$  算法的收敛性定理和推论必须在信息量  $\tau_{\min} > 0$  时才成立;而 GBAS 的收敛性定理和推论允许信息量为 0。

(3) GBAS 的收敛性定理证明了当蚂蚁智能体数目趋于无穷时,算法每次迭代产生最优解的概率趋于 1;而  $\text{ACO}_{gb, \tau_{\min}}$  算法的收敛性分析中,只证明了由于  $\tau_{\min}$  的存在,算法每次迭代产生最优解的概率趋于  $1 - \hat{\epsilon}$ 。

(4)  $\text{ACO}_{gb, \tau_{\min}}$  算法的收敛性定理和推论不依赖于最优解集  $S^*$  中的元素数目;而 GBAS 的收敛性定理成立的基本条件之一就是假定所求问题的最优解唯一;

(5) 在  $\text{ACO}_{gb, \tau_{\min}}$  算法的收敛性定理 3.3.1 中,收敛性是  $\tau_{\min}$  和  $\rho$  的函数;而在 GBAS 的收敛性定理中,收敛性是蚂蚁智能体数目和  $\rho$  的函数。

### 3.4 GBAS/tdev 和 GBAS/tdlb 的确定性收敛证明

在文献[1, 2]中,Gutjahr W J 针对 GBAS 的收敛性进行了深入分析,其主要结论是对于给定的优化问题,GBAS 能以大于或等于  $1 - \epsilon$  的概率收敛到最优解,且证明了可通过选择蚂蚁数目  $S$  或信息素挥发因子  $\rho$  使  $\epsilon$  任意小,但该证明没有对  $S$  或  $\rho$  进行定界,因此这种收敛性在某种意义上是不可控的,即在不断调整  $S$  或  $\rho$  的条件下也很难保证使算法按照事先设定的最小概率收敛;文献[3, 26]中,Stüezle T 和 Dorigo M 对  $\text{ACO}_{gb, \tau_{\min}}$  算法的收敛性问题进行了深入研究,证明了当计算时间趋于无穷时, $\text{ACO}_{gb, \tau_{\min}}$  算法总能寻到最优解,且最优路径上的信息量大于其他任意路径上的信息量,但所给出的证明过程比较弱化。

根据所采用信息素更新规则的不同,Gutjahr W J 进而又提出了两种新的 GBAS<sup>[4]</sup>,即时间相关挥发系数的 GBAS/tdev(GBAS with time-dependent evaporation factor)和时间相关信息量下界的 GBAS/tdlb(GBAS with time-dependent lower pheromone bound),GBAS/tdev 和 GBAS/tdlb 均不依赖于启发函数  $\eta_{kl}$ 。针对 GBAS 和  $\text{ACO}_{gb, \tau_{\min}}$  算法的收敛性证明缺陷,本节首先证明了蚁群算法能以数值 1 的概率收敛到最优解,其次证明了可通过选择合适的参数使蚁群算法的动态随机过程充分保证其收敛到全局最优解。

#### 3.4.1 算法描述

根据定义 3.2.1,映射  $\Phi$  将集合  $W$  中的一个子集  $\bar{W}$  映射到给定问题的可行

解空间,即对于  $\bar{W}$  中的每一条路径,都可按映射  $\Phi$  对应一个可行解;反之,通过逆映射  $\Phi^{-1}$  由问题的一个可行解至少可求得集合  $\bar{W}$  中的一条路径。

基于定义 3.2.1,GBAS/tdev 和 GBAS/tdlb 的伪代码描述如下:

```
初始化构造图 C 中弧段( $k, l$ )上的信息素轨迹  $\tau_{kl}$ 
```

```
For 迭代次数  $n = 1, 2, \dots$ 
```

```
{
```

```
    For 蚂蚁  $s = 1, \dots, S$ 
```

```
{
```

```
        令蚂蚁的当前位置  $k$  等于  $C$  中的起始节点
```

```
        While (蚂蚁处在路径  $u$  上的可行弧段( $k, l$ ))
```

```
{
```

```
            按照概率  $p_{kl}$  选择后继节点  $l$ 
```

```
            If 弧段( $k, l$ )不可行, then  $p_{kl} = 0$ 
```

```
            otherwise  $p_{kl} = \tau_{kl} / \sum_{(k,r)} \tau_{kr}$ 
```

```
            通过弧段( $k, l$ )将目前蚂蚁的路径  $u$  延续到  $l$ 
```

```
}
```

```
}
```

```
    信息素更新
```

```
}
```

**评注 3.4.1** 忽略“能见度”增加了问题表述的透明度,但这并不对算法构成本质的限制,这些证明可拓展到文献[1, 2]中能见度为非常数值时的情况。该算法借鉴了 Stützle T 和 Dorigo M 在文献[26]中所提出的“算法总是增强首次发现而不是最后所发现的相等信息量的路径”这一思想,且该思想并不受具体问题的约束。

**评注 3.4.2** 定义 3.2.1 是对文献[1]中构造图定义的一个总结,该总结注重子集  $\bar{W}$  中可行路径的约束。

**评注 3.4.3** 文献[1]表明任何静态组合优化问题都可视为在构造图上寻找最优可行路径的问题。最简单的实例就是规模为  $n$  的 TSP,其构造图是一个  $n$  阶封闭图,且对于每一条可行路径,都直接对应着 TSP 的一次遍历,而对于其他问题可采用其他类别的图对其进行描述。对于某些特定问题,需预先确定一个起始节点和终止节点,以满足定义 3.2.1 中性质(2)的条件③。

在 GBAS/tdev 和 GBAS/tdlb 这两个算法中,信息素初始化是通过定义图  $C$  中所有路径弧段( $k, l$ )上的  $\tau_{kl} = \tau_{kl}(1) = 1/|A|$  而实现的,其中  $|A|$  表示图  $C$  中路径弧段的数目,其信息素更新规则是所有路径弧段上的信息量以  $\rho_n$  为系数进行挥发,随后以“奖赏”的形式增强所发现最优路径弧段上的信息量。

### 3.4.1.1 GBAS/tdev 的信息素更新规则

$$\tau_{kl}(n+1) = \begin{cases} (1 - \rho_n)\tau_{kl}(n) + \frac{\rho_n}{L(\hat{w}(n))}, & \text{若 } (k, l) \in \hat{w}(n) \\ (1 - \rho_n)\tau_{kl}(n), & \text{否则} \end{cases} \quad (3.4.1)$$

式中,  $0 < \rho_n < 1$  ( $n = 1, 2, \dots$ );  $\hat{w}(n)$  表示  $n$  次迭代后所发现的最优路径, 该路径存储在一个特定的变量表中, 每当有蚂蚁发现具有较小代价(目标函数)值的路径时, 即以该路径替代变量表中的最优路径;  $L(\hat{w}(n))$  表示路径  $w$  的长度。

GBAS/tdev 中变量的主要特点是所选择的信息素挥发因子  $\rho_n$  是时间相关的, 其值是迭代次数  $n$  的函数。

### 3.4.1.2 GBAS/tdlb 的信息素更新规则

$$\tau_{kl}(n+1) = \begin{cases} \max\left(\left((1 - \rho)\tau_{kl}(n) + \frac{\rho}{L(\hat{w}(n))}\right), \tau_{\min}(n)\right), & \text{若 } (k, l) \in \hat{w}(n) \\ \max((1 - \rho)\tau_{kl}(n), \tau_{\min}(n)), & \text{否则} \end{cases} \quad (3.4.2)$$

式中, 信息素挥发因子  $\rho$  是时间独立的常数, 且  $0 < \rho < 1$ ;  $\tau_{\min}(n)$  是一个实数序列; 而信息量  $\tau_{kl}$  有界且时间相关, 即其值是迭代次数  $n$  的函数。与 GBAS/tdev 中所有的信息量之和恒为 1 不同, GBAS/tdlb 中所有的信息量之和是时变的。

## 3.4.2 算法的随机过程描述

为了证明算法的随机收敛性, 有必要把上述算法的变量归并到一个明确的概率框架中对其进行研究。这一点可通过考虑与算法运行相关的某些现有状态变量来实现, 此处着重考虑如下两个状态变量:

- (1) 向量  $\underline{\tau}(n)$  表示第  $n$  次迭代开始时图  $C$  中所有弧段上的信息量。
- (2) 向量  $\hat{w}(n-1)$  表示第  $n-1$  次迭代结束时算法所找到的最优路径, 其中  $\hat{w}(0)$  可定义为任意可行路径。

定义  $\underline{\tau}(n)$  是一个由  $|A|$  个元素组成的向量,  $\hat{w}(n-1)$  表示最多由  $|V|$  个元素组成的向量, 其中  $|V|$  为图  $C$  的节点数目。由于路径集合中的元素个数有限, 故可仅用一个整数来标识  $\hat{w}(n-1)$ , 因此,  $(\underline{\tau}(n), \hat{w}(n-1))$  是  $\mathbb{R}^{|A|} \times \mathbb{N}$  中的一个元素。

**引理 3.4.1** 在 GBAS/tdev 和 GBAS/tdlb 中, 由状态

$$X_n = (\underline{\tau}(n), \hat{w}(n-1)) \quad (n = 1, 2, \dots) \quad (3.4.3)$$

所构成的随机过程是时间离散空间上的非齐次 Markov 过程。

**证明** 根据 Markov 过程的性质可知,只要证明  $X_n$  的分布仅依赖于  $X_{n-1}$ ,则该定理即可得证。很明显,蚂蚁  $s(s=1,2,\dots,S)$  在  $n-1$  次迭代中所生成路径  $w^{(s)}(n-1)$  的分布仅仅依赖于  $\underline{\tau}(n-1)$ ,且路径  $w^{(s)}(n-1)$  和  $\hat{w}(n-2)$  共同决定着  $\hat{w}(n-1)$ ,而它们又与  $\hat{w}(n-1)$  一起通过信息素更新规则决定着  $\underline{\tau}(n)$ 。所以,  $(\underline{\tau}(n),\hat{w}(n-1))$  的分布完全由  $(\underline{\tau}(n-1),\hat{w}(n-2))$  决定。又因为信息素更新依赖于  $n$ ,所以该 Markov 过程是非齐次的。

[证毕]

### 3.4.3 确定性收敛性定理

**定理 3.4.1** 在 GBAS/tdev 算法中,令

$$\rho_n \leqslant 1 - \frac{\lg(n)}{\lg(n+1)} \quad (n \geqslant N) \quad (3.4.4)$$

对于某些  $N \geqslant 1$ ,且

$$\sum_{n=1}^{\infty} \rho_n = \infty \quad (3.4.5)$$

则当  $n \rightarrow \infty$  时,Markov 过程  $X_n = (\underline{\tau}(n),\hat{w}(n-1))$  收敛到状态  $(\underline{\tau}[w^*],w^*)$  的概率为 1,其中  $w^*$  为最优路径,而  $\underline{\tau}[w^*]$  可通过如下定义获取

$$\tau_{kl}[w^*] = \begin{cases} \frac{1}{L(w^*)}, & \text{若 } (k,l) \in w^* \\ 0, & \text{否则} \end{cases} \quad (3.4.6)$$

特别地,在满足公式(3.4.4)和公式(3.4.5)的情况下,假定某只蚂蚁  $s$  在第  $n$  次迭代时经过最优路径  $w^*$  的概率为  $p_{w^*}(n)$ ,则当  $n \rightarrow \infty$  时,  $p_{w^*}(n) \rightarrow 1$ 。

**证明** (1) 定义事件  $F_n$  为第  $n$  次迭代时某只蚂蚁首次穿越的最优路径。考虑到某一固定的最优路径  $w^*$ ,有

$$\neg F_1 \wedge \neg F_2 \wedge \neg F_3 \wedge \dots \Rightarrow \text{没有任何蚂蚁穿越最优路径 } w^*$$

从而有

$$\begin{aligned} P(\neg F_1 \wedge \neg F_2 \wedge \neg F_3 \wedge \dots) &\leqslant P(\text{没有任何蚂蚁穿越最优路径 } w^*) \\ &= \sum_{n=1}^{\infty} P(\text{第 } n \text{ 次迭代中没穿越最优路径 } w^* \mid \text{前 } n-1 \text{ 次迭代中没穿越 } w^*) \end{aligned} \quad (3.4.7)$$

由此,可得第  $n$  次迭代结束后每条固定路径弧段  $(k,l)$  上的信息量下界。以最恶劣情况为例,路径弧段  $(k,l)$  上的信息素只挥发而不增强。不失一般性,令  $N \geqslant 2$ ,则对于  $n \geqslant N$ ,由公式(3.4.4)可得

$$\tau_{kl}(n) = \left[ \prod_{i=1}^{n-1} (1 - \rho_i) \right] \tau_{kl}(1) \geqslant \left[ \prod_{i=1}^{N-1} (1 - \rho_i) \right] \left[ \prod_{i=N}^{n-1} \frac{\lg i}{\lg(i+1)} \right] \tau_{kl}(1)$$

$$= \left[ \prod_{i=1}^{N-1} (1 - \rho_i) \right] \tau_{kl}(1) \cdot \frac{\lg N}{\lg n} = \frac{\text{const}}{\lg n} \quad (3.4.8)$$

由于图中某一固定节点的可行后继节点数少于  $|V|$  个,且所有的信息素轨迹小于或等于 1,从而

$$p_{kl}(n) \geq \frac{\text{const}}{|V| \cdot \lg n} \quad (n \geq N)$$

故某只蚂蚁  $s$  在  $n(n \geq N)$  次迭代中穿越最优路径  $w^*$  的概率为

$$\prod_{(k,l) \in w^*} p_{kl}(n) \geq \left( \frac{\text{const}}{|V| \cdot \lg n} \right)^{L(w^*)}$$

很明显,不等式右边项的下界与第  $n$  次迭代之前的事件是独立的,对于第 1, 2, …,  $n-1$  次迭代而言情况类似。因此,公式(3.4.7)右边项的上界为

$$1 \cdot \prod_{n=N}^{\infty} \left[ 1 - \left( \frac{\text{const}}{|V| \cdot \lg n} \right)^{L(w^*)} \right] \quad (3.4.9)$$

对上式取对数,可得

$$\sum_{n=N}^{\infty} \lg \left( 1 - \left( \frac{\text{const}}{|V| \cdot \lg n} \right)^{L(w^*)} \right) \leq - \sum_{n=N}^{\infty} \left( \frac{\text{const}}{|V| \cdot \lg n} \right)^{L(w^*)} = -\infty$$

由于当  $n$  足够大时,  $(\lg n)^L \leq \text{const} \cdot n$ , 其中  $L$  为正整数。因此对于每一个  $L$  而言,  $\sum_n (\lg n)^{-L}$  是一个发散级数,从而公式(3.4.7)和公式(3.4.9)的右边项为 0,这样即证明了某次迭代中最优路径被穿越事件  $F_1 \vee F_2 \vee \dots$  发生的概率为 1。

当最优路径被首次穿越后,将该路径上的  $\hat{w}(n)$  设为均等值。由此,随机过程  $X_n = (\underline{\tau}(n), \hat{w}(n-1))$  在某次迭代并入集合  $S_{|A|} \times \{w^*\}$  的概率为 1,且有

$$S_p = \{(x_1, x_2, \dots, x_p) \mid x_i \geq 0, \sum_{i=1}^p x_i = 1\}$$

(2) 下面将证明对于最优路径集中的每条最优路径  $w^*$  而言,状态  $(\underline{\tau}[w^*], w^*)$  是 Markov 过程  $(X_n)$  吸引域  $S_{|A|} \times \{w^*\}$  上的一个确定吸引子,其中  $\underline{\tau}[w^*]$  由公式(3.4.6)确定,即若 Markov 过程的实现  $X_1(\omega), X_2(\omega), \dots$  对某个  $n$  满足  $X_n(\omega) \in S_{|A|} \times \{w^*\}$ ,则有  $\lim_{n \rightarrow \infty} X_n(\omega) = (\underline{\tau}[w^*], w^*)$ 。与本证明的第(1)部分一起,该定理的第一个结论即可得证。

令  $m$  为最优路径  $w^*$  被首次穿越时的迭代次数,则过程  $(\underline{\tau}(n), \hat{w}(n-1))$  在第  $m+1$  次迭代时并入集合  $S_{|A|} \times \{w^*\}$ 。从而,在  $n > m$  的所有迭代中,只有最优路径  $w^*$  上的信息素得到强化,且  $\hat{w}(n) = w^*$ 。根据信息素更新公式(3.4.1)中的上一行,对于  $(k,l) \in w^*$  且  $r=1, 2, 3, \dots$ ,有

$$\tau_{kl}(m+r) = \left[ \prod_{n=m}^{m+r-1} (1 - \rho_n) \right] \tau_{kl}(m) + \frac{1}{L(w^*)} \sum_{i=0}^{r-1} \rho_{m+i} \prod_{j=i+1}^{r-1} (1 - \rho_{m+j}) \quad (3.4.10)$$

由公式(3.4.5)可知,级数  $\sum \rho_n$  发散,即

$$\prod_{n=1}^{\infty} (1 - \rho_n) = 0$$

从而,由公式(3.4.10)可得

$$\lim_{r \rightarrow \infty} \left[ \prod_{n=m}^{m+r-1} (1 - \rho_n) \right] \tau_{kl}(m) = \tau_{kl}(m) \prod_{n=m}^{\infty} (1 - \rho_n) = 0 \quad (3.4.11)$$

对于路径弧段 $(k, l) \notin w^*$ 而言,经过第 $m$ 次迭代后,该路径弧段上的信息素没有得到强化,从而当 $r \rightarrow \infty$ 时,有

$$\tau_{kl}(m+r) = \left[ \prod_{n=m}^{m+r-1} (1 - \rho_n) \right] \tau_{kl}(m) \rightarrow 0 \quad (3.4.12)$$

因此,不属于最优路径 $w^*$ 上的信息素总量趋于0,而属于最优路径 $w^*$ 上的信息素之和趋于1。由公式(3.4.10)和公式(3.4.11)可得,对于任意 $(k, l) \in w^*$ , $\limsup_{r \rightarrow \infty} \tau_{kl}(m+r)$ 的值均相等,且其值之和为1(公式(3.4.10)右边第二项不依赖于 $(k, l)$ );对于 $\liminf_{r \rightarrow \infty} \tau_{kl}(m+r)$ 亦如此。因此,对于任意 $(k, l) \in w^*$ ,有

$$\limsup_{r \rightarrow \infty} \tau_{kl}(m+r) = \liminf_{r \rightarrow \infty} \tau_{kl}(m+r) = \lim_{r \rightarrow \infty} \tau_{kl}(m+r) = \frac{1}{L(w^*)} \quad (3.4.13)$$

进一步,由于 $X_n$ 的第二项满足 $\lim_{n \rightarrow \infty} w^* = w^*$ ,故有下式成立

$$\lim_{n \rightarrow \infty} X_n = (\underline{\tau}[w^*], w^*)$$

(3) 特别地,若上述过程并入集合 $S_{|A|} \times \{w^*\}$ ,则由公式(3.4.12)、公式(3.4.13)及状态转移概率 $p_{kl}$ 的定义可知,对于每一个 $(k, l) \in w^*$ ,有

$$p_{kl}(n) \rightarrow 1$$

由此,对所有的 $(k, l) \in w^*$ 做乘积运算所产生的 $p_{w^*}(n)$ 收敛到1。由于满足上述条件的最优路径 $w^*$ 存在的概率为1,因此定理3.4.1的第二个结论成立。

[证毕]

**评注3.4.4** 由定理3.4.1的第二个结论可知,穿越最优路径 $w^*$ 的概率 $p_{w^*}(n)(n=1, 2, \dots)$ 是信息量 $\tau_{kl}(n)$ 的函数。在此处所涉及的Markov过程体系中,信息量是状态、随机变量及 $p_{w^*}(n)$ 的组元,而 $p_{w^*}(n)(n=1, 2, \dots)$ 是可能收敛也可能不收敛的实数序列。定理3.4.1说明了这些序列中的某一个能够以数值为1的概率收敛。

**推论3.4.1** 在GBAS/tdev算法中,令

$$\rho_n = \frac{c_n}{n \lg(n+1)} \quad (n \geq 1) \quad (3.4.14)$$

且

$$0 \leq \lim_{n \rightarrow \infty} c_n < 1$$

则定理3.4.1的结论成立。

**证明** 显然,只要证得该推论条件与公式(3.4.4)和公式(3.4.5)等价即可。

当  $n \rightarrow \infty$  时,有

$$1 - \frac{\lg n}{\lg(n+1)} = \frac{\lg\left(1 + \frac{1}{n}\right)}{\lg(n+1)} \sim \frac{1}{n \cdot \lg(n+1)}$$

当  $c_n \rightarrow c \in [0, 1]$  且  $n$  充分大时,有

$$\frac{c_n}{n \lg(n+1)} < 1 - \frac{\lg n}{\lg(n+1)}$$

从而证明了公式(3.4.4)。

另一方面,由于

$$\sum_{n=1}^{\infty} \frac{1}{n \lg(n+1)} \geq \int_1^{\infty} \frac{dx}{x \lg(x+1)} = \infty$$

从而,当  $N$  充分大时,有

$$\sum_{n=1}^{\infty} \frac{c_n}{n \lg(n+1)} \geq \sum_{n=N}^{\infty} \frac{\frac{c}{2}}{n \lg(n+1)}$$

发散,从而证明了公式(3.4.5)。

[证毕]

**定理 3.4.2** 在 GBAS/tldb 算法中,令

$$\tau_{\min}(n) = \frac{c_n}{\lg(n+1)} \quad (n \geq 1) \quad (3.4.15)$$

式中,  $\lim_{n \rightarrow \infty} c_n > 0$ , 则定理 3.4.1 的结论成立。

**证明** 该定理的主体证明部分类似于定理 3.4.1, 这里只强调与定理 3.4.1 证明过程中的一些不同之处。

在定理 3.4.1 证明的第(1)部分中,对最恶劣情况(即选择了信息素从未得到强化的路径弧段( $k, l$ ))进行了研究。在 GBAS/tldb 算法证明中,引入了信息量下界  $\tau_{\min}(n)$  来证明,有

$$\tau_{kl}(n) \geq \tau_{\min}(n) = \frac{c_n}{n \lg(n+1)} \geq \frac{c}{2 \lg(n+1)}$$

当  $n$  充分大时,有  $\lim_{n \rightarrow \infty} c_n = c$ , 这与公式(3.4.8)中的下界本质上相同,随后的证明过程类似。

在第(2)部分中,值得注意的是,由于信息量不再归一化,从而  $\underline{\tau}(n)$  不再单一地受限于  $S_{|A|}$ 。特别地,信息素在路径弧段( $k, l$ )  $\notin w^*$  上挥发不再意味着信息素在路径弧段( $k, l$ )  $\in w^*$  上的和趋于 1。将这一论断扩展,容易发现过程( $X_n$ )可能并入集合  $\mathbb{R}^{|A|} \times \{w^*\}$ , 且  $\lim_{n \rightarrow \infty} X_n(\omega) = (\underline{\tau}[w^*], w^*)$ 。这一点可证明如下:

令  $m$  表示最优路径  $w^*$  被首次穿越的迭代次数,则在忽略  $\tau_{kl}(n+1)$  可能被设置为  $\tau_{\min}(n)$  的前提下,得到路径弧段( $k, l$ )  $\in w^*$  上信息量  $\tau_{kl}(n)$  的下界。令公式

(3.4.12)中的 $\rho_n \equiv \rho$ ,可得

$$\tau_{kl}(m+r) \geq (1-\rho)^r \tau_{kl}(m) + \frac{\rho}{L} \sum_{i=0}^{r-1} (1-\rho)^{r-i-1} \rightarrow \frac{\rho}{L} \sum_{j=0}^{\infty} (1-\rho)^j = \frac{1}{L} (r \rightarrow \infty) \quad (3.4.16)$$

进一步,当 $r$ 充分大时,有

$$\tau_{kl}(m+r) > \frac{1}{2L}$$

由于当 $n \rightarrow \infty$ 时,有 $\tau_{\min}(n) \rightarrow 0$ ,则当 $r$ 充分大时,有

$$\tau_{\min}(m+r) < \frac{1}{2L}$$

从而,当 $r$ 充分大时,令 $\tau_{kl}(n+1)$ 等价于 $\tau_{\min}(n)$ 这一信息素更新规则不再适用,这时,公式(3.4.16)中的“ $\geq$ ”将被“ $>$ ”替代。这意味着对于任意 $(k,l) \in w^*$ ,有

$$\lim_{r \rightarrow \infty} \tau_{kl}(m+r) = \frac{1}{L}$$

又由于当 $n \rightarrow \infty$ 时, $\tau_{\min}(n) \rightarrow 0$ ,故对于任意 $(k,l) \notin w^*$ ,有

$$\lim_{r \rightarrow \infty} \tau_{kl}(m+r) = 0$$

其余部分(包括第(3)部分)的证明过程与定理3.4.1类似,这里不再赘述。

[证毕]

**评注3.4.5** 到目前为止,还不能给出定理3.4.1和定理3.4.2中以最优方式加速算法收敛性的参数选择框架。在满足给定条件的范围内,算法在拓展搜索空间和寻找最优解(即“探索”和“利用”)这两个矛盾过程之间保持着一种合理的动态平衡。

### 3.5 基本蚁群算法的A.S.收敛性研究

本节在不依赖Gutjahr W J所提四个基本条件的前提下,对基本蚁群算法的收敛性进行了理论证明。在对蚁群算法的Markov链分析过程中,运用离散鞅作为研究工具,把最优解集序列转变为下鞅序列来考察信息素轨迹向量的收敛性,对蚁群算法的A.S.收敛问题和停时问题进行了研究,提出了蚁群算法首达时间的定义,并对蚁群算法首次到达时间的期望值进行了理论分析。

#### 3.5.1 基本定义<sup>[30~32]</sup>

**定义3.5.1** 设 $\Omega$ 是一个基本集合, $F$ 是 $\Omega$ 上部分子集构成的 $\sigma$ 代数,即满足

- (1)  $\emptyset \in F$
- (2) 当 $A \in F$ 时, $A^c \in F$

(3) 若  $A_k \in F (\forall k=1,2,\dots)$ , 则  $\bigcup_{k=1}^{\infty} A_k \in F$

若  $P$  是  $F$  上的概率测度, 则  $(\Omega, F, P)$  构成概率空间,  $(\Omega, F)$  称为可测空间。若  $f$  为可测空间  $(\Omega_1, F_1)$  到可测空间  $(\Omega_2, F_2)$  的映射, 且对于  $\forall x \in F_2$ , 有

$$f^{-1}(A) = \{x, f(x) \in A\} \in F_1 \quad (3.5.1)$$

则称  $f$  为可测映射。若  $f$  为实值可测映射, 则称之为随机变量。

**定义 3.5.2 离散型随机变量的数学期望** 设  $\zeta$  为离散型随机变量, 其分布列为

$$P\{\zeta = x_i\} = p_i, i = 0, 1, 2, \dots, \text{且 } p_i > 0, \sum_{i=0}^{\infty} p_i = 1 \quad (3.5.2)$$

记  $E(\zeta) = \sum_{i=0}^{\infty} x_i p_i$ , 若  $E(\zeta) < \infty$ , 则称  $E(\zeta)$  为  $\zeta$  的数学期望。

**定义 3.5.3 离散型随机变量的条件期望** 设  $(\zeta, \eta)$  为离散型随机变量, 其概率函数为

$$p(i, k) = P\{\zeta = x_i, \eta = y_k\}, i, k = 1, 2, \dots \quad (3.5.3)$$

条件概率函数为

$$p(k | i) = P\{\eta = y_k, \zeta = x_i\}, p(i | k) = P\{\zeta = x_i, \eta = y_k\}, i, k = 1, 2, \dots \quad (3.5.4)$$

则条件期望为

$$\begin{cases} E(\eta | x_i) = \sum_k y_k p(k | i) = \sum_k y_k \frac{p(i | k)}{p(i, \cdot)}, i = 1, 2, 3, \dots \\ E(\zeta | y_k) = \sum_i x_i p(i | k) = \sum_i x_i \frac{p(i | k)}{p(\cdot, k)}, k = 1, 2, 3, \dots \end{cases} \quad (3.5.5)$$

设  $(\Omega, F, P)$  是一个概率空间,  $F_1 \subset F$  也是一个  $\sigma$  函数,  $\zeta$  是  $\Omega$  上可测与可积的随机变量,  $\zeta$  关于  $F_1$  的条件期望  $E(\zeta | F_1)$  是可测可积的, 且对于  $\forall A \in F_1$ , 有

$$\int_A E(\zeta | F_1) dP = \int_A \zeta dP \quad (3.5.6)$$

很明显, 条件期望具有如下性质:

$$(1) E(E(\zeta | F_1)) = E(\zeta) \text{ (A. S.)} \quad (3.5.7)$$

(2) 若  $\alpha', \beta'$  是常数, 则

$$E(\alpha' \zeta + \beta' \varsigma | F_1) = \alpha' E(\zeta | F_1) + \beta' E(\varsigma | F_1) \text{ (A. S.)} \quad (3.5.8)$$

**定义 3.5.4 鞍、下鞍和上鞍** 设  $F_n$  是单调的  $\sigma$  代数列,  $f_n$  是  $\Omega$  上的实值函数列, 则称  $(f_n, F_n)$  为鞍(离散鞍)。若  $f_n$  关于  $F_n$  是可测可积, 则对于  $\forall m < n$ , 有

$$E(f_n | F_m) = f_m \text{ (A. S.)} \quad (3.5.9)$$

若  $E(f_n | F_m) \geq f_m$  (A. S.), 称  $(f_n, F_n)$  为下鞍; 否则, 称  $(f_n, F_n)$  为上鞍。

若  $\{f_n\}$  是有界的, 则对于鞍、下鞍和上鞍而言,  $\lim_{n \rightarrow \infty} f_n$  几乎处处存在且有界。

### 3.5.2 基于下鞅序列的蚁群算法 A. S. 收敛性证明

由于蚁群算法寻找最优解的途径是根据路径向量  $\bar{w}(t)$  上的信息素轨迹向量  $\bar{\tau}(t)$  来求解问题。通常希望蚁群算法在每次迭代之后, 最优路径上的残留信息量趋于最大。因此, 在引入离散鞅的概念后, 可将最优解集序列转变为下鞅序列来考察  $\{\bar{\tau}(t)\}$  的收敛性。

**定理 3.5.1** 当第  $t$  个搜索周期中蚂蚁的路径向量  $\bar{w}(t)$  几乎处处收敛到最优解集序列  $w^* = (w_0, w_1, \dots, w_L)$ , 则当  $t \rightarrow \infty$  时, 其进化过程是时间离散的非齐次 Markov 过程, 且信息素轨迹向量  $\bar{\tau}(t)$  A. S. 收敛到最优值, 即

$$\bar{w}(t) \rightarrow w^* \text{ (A. S.)} \Leftrightarrow \bar{\tau}(t) \rightarrow \tau^* \text{ (A. S.)} \quad (3.5.10)$$

**证明** 该定理第一部分证明类同于引理 3.4.1 的证明过程; 至于第二部分, 由于路径向量  $\bar{w}(t)$  中的元素有限, 则最优解集序列  $w^* = (w_0, w_1, \dots, w_L)$  为有限集。所以, 当  $t \rightarrow \infty$  时,  $\bar{w}(t) \rightarrow w^* \text{ (A. S.)}$  等价于  $\bar{\tau}(t) \rightarrow \tau^* \text{ (A. S.)}$ 。

[证毕]

**定理 3.5.2** 设  $F_t$  表示在  $t$  个搜索周期中至少有一只蚂蚁穿越最优路径的事件,  $\bar{f}(\bar{\tau}(t))$  表示在  $t$  个搜索周期中蚁群所搜索到的解向量, 且该解向量单调递增,  $f^*$  表示解向量  $\bar{f}(\bar{\tau}(t))$  中的最优解, 则  $(\bar{f}(\bar{\tau}(t)), F_t)$  是正的有界下鞅, 即

$$E(\bar{f}(\bar{\tau}(t+1)) | F_t) \geq \bar{f}(\bar{\tau}(t)) \quad (t \geq 1) \quad (3.5.11)$$

且

$$\lim_{t \rightarrow \infty} \bar{f}(\bar{\tau}(t)) = f^* \text{ (A. S.)} \quad (3.5.12)$$

**证明** 显然,  $F_t$  由  $\{\bar{w}(t), \bar{\tau}(t)\}$  随机向量组合中的随机序列产生, 且其为  $\sigma$  域上的随机向量。对于时间离散非齐次 Markov 过程中  $t$  个搜索周期的给定状态而言, 蚂蚁选定子路径  $l$  的概率为

$$P_l = \prod_{l \in w^*} P\{\bar{\tau}_l(t, w^*)\}$$

从而, 当  $t \geq 1$  时, 有

$$\begin{aligned} E(\bar{f}(\bar{\tau}(t+1)) | F_t) &= \bar{f}(\bar{\tau}(t)) \\ &= \sum_{l=1}^L \bar{f}(\bar{\tau}(t+1)) \{P_{l+1}\{\bar{\tau}_l(t+1, w^*)\} | \{P_l\{\bar{\tau}_l(t, w^*)\}\}} \\ &\geq 0 \end{aligned}$$

所以,  $(\bar{f}(\bar{\tau}(t)), F_t)$  是正的有界下鞅。

令  $G_t = \{\bar{\tau}(t) \cap \bar{\tau}_l(t, w^*) \neq \emptyset\} = \{\bar{f}(\bar{\tau}(t)) \neq f^*\}$ , 对于下鞅  $\bar{f}(\bar{\tau}(t))$ , 由条件期望的性质可得

$$f^* P\{G_t\} \geq \int_{G_t} \bar{f}(\bar{\tau}(t+1)) dP \geq \int_{G_t} \bar{f}(\bar{\tau}(t)) dP = f^* P\{G_t\}$$

即

$$\int_{G_t} \bar{f}(\bar{\tau}(t+1)) dP = f^* P\{G_t\}$$

又由于解向量  $\bar{f}(\bar{\tau}(t))$  单调递增, 且存在

$$0 \leq \bar{f}(\bar{\tau}(t)) \leq f^*$$

故

$$\lim_{t \rightarrow \infty} \bar{f}(\bar{\tau}(t)) = f^* \text{ (A. S.)}$$

[证毕]

### 3.5.3 蚁群算法的停时问题研究

如果期望知道蚁群算法首次搜索到全局最优解的时间, 此即停时问题。由于某只蚂蚁在任意时刻搜索到全局最优解的概率是随机的, 因此首次到达全局最优解的时间也是随机的。这里所要研究的重点是蚁群算法首次到达时间的期望值。

**定义 3.5.5 蚁群算法的首达时间** 设  $\bar{w}(t)$  表示第  $t$  个搜索周期中, 蚂蚁  $A_1, A_2, \dots, A_t$  的路径向量,  $w^*(t)$  表示第  $t$  个搜索周期中某只蚂蚁在  $1, 2, \dots, t-1$  所搜索到的全局最优解, 则称  $T = \min\{t, \bar{w}(t) \cap w^*(t) \neq \emptyset\}$  为蚁群算法的首达时间。

**定理 3.5.3** 设  $E(T)$  表示首达时间  $T = \min\{t, \bar{w}(t) \cap w^*(t) \neq \emptyset\}$  的期望值,  $c$  表示期望搜索次数,  $T_0$  表示算法一次迭代的平均计算时间。当  $E(T) \leq T_0$  时, 独立地使用蚁群算法搜索全局最优解  $c$  次, 则至少有一次在  $cT_0$  时刻之前找到全局最优解的概率不小于  $1 - \frac{1}{c}$ 。

**证明** 由已知条件, 可得

$$P\{T > cT_0\} \leq \frac{E(T)}{cT_0}$$

当  $E(T) \leq T_0$  时, 有

$$P\{T > cT_0\} \leq \frac{T_0}{cT_0} = \frac{1}{c}$$

从而

$$P\{T < cT_0\} \geq 1 - \frac{1}{c} \quad (3.5.13)$$

上式表示独立地使用蚁群算法搜索全局最优解  $c$  次, 则至少有一次在  $cT_0$  时刻之前找到全局最优解的概率不小于  $1 - \frac{1}{c}$ 。

[证毕]

## 3.6 一类分布式蚂蚁路由算法的收敛性研究

本节将对一类分布式蚂蚁路由算法的收敛性作深入分析,该算法是一种基于“后向”学习且根据蚁群优化思想改进的随机算法<sup>[9, 10]</sup>。

### 3.6.1 分布式蚂蚁路由算法的基本思想

假设在一个分布式网络中,主机由  $R$  个路由器组成的网络进行链接,若任意两个路由之间存在双向的点到点链接,则这两个路由器称为临近链接路由。令  $N_r$  表示临近路由  $r(r=1, 2, \dots, R)$  的集合,对于每一个目标主机  $d$ ,路由  $r$  与分离向量  $(d, (i, p_i), i \in N_r)$  构成一个概率路由表。对于  $N_r$  中的每一个临近路由  $i$ ,定义  $p_i$  为路由  $r$  利用链接  $(r, i)$  传送数据包到目标主机  $d$  的概率<sup>[33]</sup>。同时定义  $c_n$  为链接  $(r, i)$  的代价,且该代价对于给定的路由  $r$  是对称的,即  $c_n = c_{ri}$ 。

本节定义蚂蚁为用来控制  $(d, s \parallel c)$  的信息,其中  $s$  表示源主机,  $c$  表示到达目标主机  $d$  的代价估计值。对于随机选择的主机  $s \neq d$  且  $c$  初始化为 0 而言,目标主机  $d$  会周期性地产生一只蚂蚁  $(d, s \parallel c)$ ,而这只蚂蚁会被网络上的内联路由器传送到源主机  $s$ ,并通过中继路由器更新路由表。

当蚂蚁  $(d, s \parallel c)$  从路由器  $i$  通过链接  $(i, r)$  到达中继路由器  $r$  时,路由器  $i$  到  $d$  的代价估计值  $c$  则通过链接  $(r, i)$  按照下式进行递增

$$c \leftarrow c + c_n \quad (3.6.1)$$

进而,目标主机  $d$  路由表中的引入向量可按下式进行更新

$$p_i \leftarrow \frac{p_i + \Delta}{1 + \Delta}, \quad p_j \leftarrow \frac{p_j}{1 + \Delta} \quad (j \in N_r \setminus \{i\}) \quad (3.6.2)$$

式中,  $\Delta = f(c)$ , 其中  $f(c)$  是增量  $c$  的减函数。在进行上述更新的过程中,路由  $r$  将更新后的蚂蚁  $(d, s \parallel c)$  传送到  $N_r$  中与其临近的一个路由  $i$ , 蚂蚁  $(d, s \parallel c)$  最终能利用点到点的代价估计值  $c$  从  $s$  到达  $d$ 。实际上,  $c$  的最终值为蚂蚁穿越网络中从  $s$  到  $d$  路径的代价值。

### 3.6.2 两点模型

这里研究的网络是由两个路由(或节点)构成的,具体表示为节点  $i=0$  和节点  $i=1$ ,这两个节点通过并行双向的节点集合  $L$  进行链接。所有的主机都被定义为  $i=0$  或  $i=1$ ,每个链接  $l(l=1, 2, \dots, L)$  在正反方向上都有一个传递代价  $c_l$ ,且  $c_l > 0$ 。不失一般性,假定这些链接为递增代价,即

$$c_1 \leq c_2 \leq \dots \leq c_L \quad (3.6.3)$$

选择节点  $i$  ( $i=0, 1$ ) 作为研究对象, 对于每一个  $n=1, 2, \dots$ , 目标主机在  $\{t_n^i, n=1, 2, \dots\}$  且  $t_n^i < t_{n+1}^i$  时刻产生蚂蚁, 而将第  $n$  只蚂蚁传送到第  $1-i$  个节点则需选择链接节点  $i$  和  $1-i$  的路径  $l_{1-i}(n)$ 。随后, 第  $n$  只蚂蚁在  $a_n^i$  (且  $t_n^i \leq a_n^i$ ) 时刻通过路径  $l_{1-i}(n)$  传送到节点  $1-i$ 。这里, 假设下述条件成立

$$a_n^i < a_{n+1}^i, \quad n = 1, 2, \dots \quad (3.6.4)$$

定义节点  $i$  的概率表为

$$p^i(n) = (p_1^i(n), \dots, p_L^i(n)) \quad (3.6.5)$$

式中,  $0 \leq p_l^i(n) \leq 1$  ( $l=1, \dots, L$ ), 且满足  $\sum_{l=1}^L p_l^i(n) = 1$ , 其中  $p_l^i(n)$  为间隔  $[a_n^{1-i}, a_{n+1}^{1-i}]$  内的概率。

在  $a_{n+1}^i$  时刻, 节点  $1-i$  通过链接  $l_{i-1}(n+1)=l$  接收到第  $n+1$  只蚂蚁后, 立即更新下述状态转移概率公式

$$p_l^{1-i}(n+1) = \frac{p_l^{1-i}(n) + C_l(1-i)}{1 + C_l(1-i)} \quad (3.6.6)$$

且对于  $C_l(1-i) > 0$ , 有

$$p_k^{1-i}(n+1) = \frac{p_k^{1-i}(n)}{1 + C_l(1-i)}, \quad k \neq l, k = 1, \dots, L \quad (3.6.7)$$

对于每一个节点  $i=0, 1$  和每一个链接  $l=1, \dots, L$ , 由公式 (3.6.6) 和公式 (3.6.7) 中的  $C_l(i) > 0$ , 可根据下式定义映射  $\phi_l^i: [0, 1]^L \rightarrow [0, 1]^L$

$$\phi_{l,k}^i(p) = \begin{cases} \frac{p_l + C_l(i)}{1 + C_l(i)} & k = l \\ \frac{p_k}{1 + C_l(i)} & k \neq l \end{cases}, \quad p \in [0, 1]^L \quad (3.6.8)$$

由此, 若  $l_i(n+1)=l$ ,  $n=0, 1, \dots$ , 则有

$$p^i(n+1) = \phi_l^i(p^i(n)) \quad (3.6.9)$$

**定理 3.6.1** 若对于严格递减的函数  $f: \mathbb{R} \rightarrow (0, \infty)$ , 存在

$$C_l(i) = f(c_l) = \Delta_l, \quad i = 0, 1, l = 1, \dots, L \quad (3.6.10)$$

则

(1) 对于每一个节点  $i=0, 1$  和  $[0, 1]^L$  中的概率  $p$ , 有

$$p_*^i = \lim_{n \rightarrow \infty} (\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n)}^i)(p) \quad (3.6.11)$$

(2) 在不考虑初始条件  $(p^0(0), p^1(0))$  的前提下, 存在一对类似  $p_*^0$  和  $p_*^1$  分布的无约束随机向量  $p^0$  和  $p^1$ , 即

$$(p^0(n), p^1(n)) \Rightarrow_n (p^0, p^1) \quad (3.6.12)$$

式中,  $\Rightarrow_n$  表示当  $n$  趋于无穷时的分布收敛性。

**证明** 由于概率表中的元素相互独立, 对于每一个  $i=0, 1$ , 只需证明  $p^i(n) \Rightarrow_n p_*^i$  即可。

因此,设  $i=0, 1$ ,且  $n=0, 1, \dots$ ,不断迭代,可得到如下关系式

$$p^i(n+1) = (\phi_{l_i(n+1)}^i \circ \phi_{l_i(n)}^i \circ \dots \circ \phi_{l_i(2)}^i \circ \phi_{l_i(1)}^i)(p^i(0)) \quad (3.6.13)$$

上式也可表示为

$$p^i(n+1) = {}_{st}(\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n)}^i \circ \phi_{l_i(n+1)}^i)(p^0(0)) \quad (3.6.14)$$

这样,要证明  $p^i(n) \Rightarrow_n p^i$ ,就得证明下式成立

$$\lim_{n \rightarrow \infty} (\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n)}^i \circ \phi_{l_i(n+1)}^i)(p^0(0)) = p^i \quad (3.6.15)$$

对于  $\mathbb{R}^L$  中的任意向量  $x$ ,定义有如下范数

$$\|x\| = \sum_{l=1}^L |x_l|$$

该范数等价于一般意义的欧式范数。设  $l=1, \dots, L$ ,对于  $[0, 1]^L$  上随机的  $x$  和  $y$ ,有

$$\begin{aligned} \|\phi_l^i(x) - \phi_l^i(y)\| &= \sum_{k=1}^L |\phi_{l,k}^i(x) - \phi_{l,k}^i(y)| \\ &= \sum_{k \neq l} \left| \frac{x_k - y_k}{1 + c_l(i)} \right| + \frac{|x_l + c_l(i) - (y_l + c_l(i))|}{1 + c_l(i)} \\ &= \frac{\|x - y\|}{1 + c_l(i)} \leq K_i \|x - y\| \end{aligned}$$

同时,有

$$K_i = \max_{l=1, \dots, L} (1 + c_l(i))^{-1} < 1$$

由此,可得

$$\max_{l=1, \dots, L} \|\phi_l^i(x) - \phi_l^i(y)\| \leq K_i \|x - y\| \quad (3.6.16)$$

由公式(3.6.16),对于  $[0, 1]^L$  上随机的  $p$  和  $q$ ,有

$$\begin{aligned} &\|(\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n-1)}^i \circ \phi_{l_i(n)}^i)(p) - (\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n-1)}^i \circ \phi_{l_i(n)}^i)(q)\| \\ &\leq K_i^n \|p - q\| \end{aligned}$$

进一步,对于每一个  $m=1, 2, \dots$ ,有

$$\begin{aligned} &\|(\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n+m)}^i)(p) - (\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n)}^i)(p)\| \\ &\leq K_i^n \|(\phi_{l_i(n+1)}^i \circ \phi_{l_i(n+2)}^i \circ \dots \circ \phi_{l_i(n+m)}^i)(p) - p\| \\ &\leq K_i^n L \end{aligned}$$

由于  $K_i < 1$ ,则对于每一个  $p$ ,序列  $\{(\phi_{l_i(1)}^i \circ \phi_{l_i(2)}^i \circ \dots \circ \phi_{l_i(n)}^i)(p), n=1, 2, \dots\}$  形成一个 Cauchy 序列,因此是收敛的。

[证毕]

对于每一个  $n=0, 1, 2, \dots$ ,这里引入由如下随机向量产生的  $\sigma$ -域  $F_n$

$$\{p^0(0), p^1(0), p^0(k), p^1(k), l_0(k), l_1(k), k=1, 2, \dots, n\}$$

对于每一个  $l=1, \dots, L$  和  $i=0, 1$ ,有

$$P[l_i(n+1) = l | F_n] = p_l^{1-i}(n) \quad (3.6.17)$$

由公式(3.6.9)和公式(3.6.17)易见,路由表 $\{p(n), n=0, 1, \dots\}$ 中的序列形成了一个不可数状态空间 $[0, 1]^L \times [0, 1]^L$ 上时间均匀的Markov链。这样很自然会产生一个问题:该Markov链能否收敛到与最小代价相对应的理想点 $((1, 0, \dots, 0), (1, 0, \dots, 0))$ ,由此引出了定理3.6.2:

**定理3.6.2** 假定 $f$ 为严格递减函数,且对于所有的 $l=2, \dots, L$ ,有 $c_1 < c_l$ 。在初始条件 $p_1^0(0) + p_1^1(0) > 0$ 下,对于每一个 $i=0, 1$ ,则路由表 $\{p^i(n), n=0, 1, \dots\}$ 几乎处处收敛,即

$$\lim_{n \rightarrow \infty} p^i(n) = (1, 0, \dots, 0) \quad \text{A. S.} \quad (3.6.18)$$

**证明** 这里需要证明序列 $\{p^0(n), n=0, 1, \dots\}$ 和序列 $\{p^1(n), n=0, 1, \dots\}$ 均几乎处处收敛到向量 $(1, 0, \dots, 0)$ 。由于对于所有的 $n=0, 1, \dots$ ,有 $\sum_{l=1}^L p_l^i(n) = 1$ 。对于每一个 $i=0, 1$ ,这等价于

$$\lim_{n \rightarrow \infty} p_1^i(n) = 1 \quad \text{A. S.} \quad (3.6.19)$$

为了证明上式成立,这里引入了一个随机向量 $\{Z(n), n=0, 1, \dots\}$ ,且有

$$Z(n) = p_1^0(n) + p_1^1(n) \quad n = 0, 1, \dots \quad (3.6.20)$$

要证明公式(3.6.19)对于 $i=0$ 和 $i=1$ 均成立,则只要证明下式成立即可

$$\lim_{n \rightarrow \infty} Z(n) = 2 \quad \text{A. S.} \quad (3.6.21)$$

对于任意 $l=2, \dots, L$ ,有 $f(c_1) > f(c_l)$ ,则选取固定的 $i=0, 1$ 和 $n=0, 1, \dots$ ,对于每一个 $k=1, \dots, L$ ,由公式(3.6.9)和公式(3.6.17)可得

$$E[p_k^i(n+1) | F_n] = \sum_{l=2}^L p_l^{1-i}(n) \phi_{l,k}^i(p^i(n)) \quad \text{A. S.} \quad (3.6.22)$$

则由公式(3.6.10)、公式(3.6.20)及公式(3.6.22)可得

$$\begin{aligned} E[Z(n+1) | F_n] - Z(n) \\ = \sum_{l=2}^L \frac{f(c_1) - f(c_l)}{(1+f(c_1))(1+f(c_l))} (p_1^0(n)p_l^1(n) - p_1^1(n)p_l^0(n)) \end{aligned} \quad (3.6.23)$$

从而

$$E[Z(n+1) | F_n] \geq Z(n) \quad \text{A. S.} \quad (3.6.24)$$

由上式可见,对于所有的 $n=0, 1, \dots$ ,随机向量 $\{Z(n), n=0, 1, \dots\}$ 形成一个有界 $F_n$ 下鞅,且 $0 \leq Z(n) \leq 2$ 。由鞅收敛定理可知<sup>[34]</sup>,下鞅 $\{Z(n), n=0, 1, 2, \dots\}$ 几乎处处收敛到随机向量 $Z$ ,且有 $0 \leq Z \leq 2$ 。

由公式(3.6.23),可知公式(3.6.24)几乎处处不等,且其仅在下述条件下才能相等

$$p_1^0(n)p_l^1(n) = p_1^1(n)p_l^0(n), \quad l = 2, \dots, L$$

即

$$p_1^0(n)(1-p_1^1(n)) = p_1^1(n)(1-p_1^0(n)) = 0$$

换言之,只有当  $p_1^0(n)=p_1^1(n)=0$  或  $p_1^0(n)=p_1^1(n)=1$  时,等式才能成立。

由此,若  $p_1^0(n)=p_1^1(n)=0$ ,则对于所有的  $n=0,1,\dots$ ,有  $Z(n)=0$  且  $Z=0$ ;若  $p_1^0(n)=p_1^1(n)=1$ ,则对于所有的  $n=0,1,\dots$ ,有  $Z(n)=2$  且  $Z=2$ 。另一方面,若在满足  $0 \leq Z(0) \leq 2$  的前提下,对于所有的  $n=1,2,\dots$ ,则几乎处处存在  $0 \leq Z(n) \leq 2$ 。

为了证明结论,这里需要证明几乎处处存在  $Z=2$ 。对于  $P[Z=2] < 1$ ,由于几乎处处存在  $Z>0$ ,则对于某一  $\epsilon>0$ ,有如下条件成立

$$\delta = P[\epsilon < Z < 2 - \epsilon] > 0 \quad (3.6.25)$$

则由  $\{Z(n), n=0,1,2,\dots\}$  对  $Z$  的几乎处处收敛性可得

$$P\left[\frac{\epsilon}{2} < Z(n) < 2 - \frac{\epsilon}{2}\right] \geq \frac{\delta}{2}, \quad n \geq n^* \quad (3.6.26)$$

上式中的  $n^*$  由  $\epsilon$  和  $\delta$  共同决定。

记  $K = \frac{f(c_1) - f(c_2)}{(1 + f(c_1))^2}$ , 则对于所有的  $n=1,2,\dots$ ,由公式(3.6.23)可得

$$\begin{aligned} E[Z(n+1)] - E[Z(n)] &= E[E[Z(n+1)|F_n] - Z(n)] \\ &= \sum_{l=2}^L \frac{f(c_1) - f(c_l)}{(1 + f(c_1))(1 + f(c_l))} \cdot E[\dots] \\ &\geq K \cdot E[p_1^0(n)p_1^1(n) - p_1^1(n)p_1^0(n)] \\ &= K \cdot E[p_1^0(n)(1 - p_1^1(n)) + p_1^1(n)(1 - p_1^0(n))] \\ &\geq K \cdot E\left[(\dots)1\left[\frac{\epsilon}{2} < Z(n) < 2 - \frac{\epsilon}{2}\right]\right] \end{aligned}$$

由于对于  $K_\epsilon = \left\{ (x,y) \in [0,1]^2 : \frac{\epsilon}{2} < x+y < 2 - \frac{\epsilon}{2} \right\}$ , 有

$$K(\epsilon) = \inf\{x+y-2xy : (x,y) \in K_\epsilon\} > 0$$

故对于  $n \geq n^*$ , 有

$$\begin{aligned} E[Z(n+1)] - E[Z(n)] &\geq KI(\epsilon) \cdot P\left[\frac{\epsilon}{2} < Z(n) < 2 - \frac{\epsilon}{2}\right] \\ &\geq KI(\epsilon) \frac{\delta}{2} \end{aligned}$$

由此,  $\liminf_{n \rightarrow \infty} (E[Z(n+1)] - E[Z(n)]) > 0$ , 而这与由  $\{Z(n), n=0,1,2,\dots\}$  的几乎处处收敛定理和有界收敛定理导出的  $\lim_{n \rightarrow \infty} E[Z(n+1)] = E[Z(n)]$  相矛盾, 这样就完成了公式(3.6.21)的证明。

[证毕]

## 3.7 基于分支路由和 Wiener 过程的蚁群算法收敛性证明

本节将蚁群算法模型转化为分支随机过程模型<sup>[11]</sup>, 并从分支随机路径和分支 Wiener 过程的角度推导了蚂蚁路径存亡的比率, 进而证明了这一过程为稳态分布, 这意味着蚁群算法能以数值为 1 的概率收敛。

### 3.7.1 分支过程

#### 3.7.1.1 分支随机路径模型<sup>[35, 36]</sup>

假定某只蚂蚁从  $t=0$  时刻开始寻优, 在  $t=1$  时刻, 这只蚂蚁会通过信息素利用状态转移概率  $p_k$  通知其他  $k(k=0,1,2,\dots)$  只蚂蚁。令

$$P\{Y(i,j) = k\} = p_k \quad (i = 1, 2, \dots; \quad j = 1, 2, \dots; \quad k = 0, 1, 2, \dots)$$

式中,  $Y(i,j)$  表示在  $t=i$  时刻接受第  $j$  只蚂蚁所传递信息的蚂蚁数目,  $p_k \geq 0$ , 且

$$\sum_{k=0}^{\infty} p_k = 1.$$

定义序列  $\{B(n), n=0,1,2,\dots\}$  为

$$\left\{ \begin{array}{l} B(0) = 1 \\ B(1) = Y(1,1) \\ B(2) = Y(2,1) + Y(2,2) + \dots + Y(2,B(1)) \\ \vdots \\ B(n) = Y(n,1) + Y(n,2) + \dots + Y(n,B(n-1)) \quad (n = 3,4,\dots) \end{array} \right. \quad (3.7.1)$$

令  $E(Y(i,j)) = \sum_{k=0}^{\infty} kp_k = m < \infty$ , 且  $0 < \text{Var}(Y(i,j)) = \sum_{k=0}^{\infty} (k - m)^2 p_k = \sigma^2 < \infty$ , 同时假设  $\{S_t, t=0,1,2,\dots\}$  为  $Z^d$  上最近的对称随机路径, 且满足  $S_0 = 0$ ,  $P(u \rightarrow v, t) = P(S_{S+t} = v | S_s = u)$ ,  $\lambda(x,t) = m^{T-t} \sum_{y \in Z^d} \lambda(y,t) p(y \rightarrow x, T-t)$ ,  $0 \leq t \leq T$ 。其中,  $\lambda(y,t)$  表示  $t$  时刻节点  $y$  上得到信息的蚂蚁数目。

#### 3.7.1.2 分支 Wiener 过程模型<sup>[36, 37]</sup>

用 Wiener 过程替代前述的最近对称随机路径, 有

(1) 蚂蚁从节点  $0 \in \mathbb{R}^d$  开始执行 Wiener 过程  $W(t) \in \mathbb{R}^d$ 。

(2) 在  $t=1$  时刻, 蚂蚁到达新的节点  $W(1)$ 。

(3) 某只蚂蚁利用信息素以概率  $P\{Y=l\} = p_l$  ( $l=0, 1, 2, \dots$ ) 通知其他 Y 只蚂蚁。

(4) 每只蚂蚁均执行 Wiener 过程，并按照上述步骤进行迭代。

显然，有  $\sum_{k=0}^{\infty} kp_k = m > 1$ ，且  $0 < \sum_{k=0}^{\infty} (k - m)^2 p_k = \sigma^2 < \infty$ 。

### 3.7.2 蚂蚁路径序列

若  $x_{s+i}$  和  $x_{s+i+1}$  ( $i=0, 1, 2, \dots, t-s-1$ ) 是  $\mathbb{Z}^d$  上的临近路径，则令  $\{x_s, x_{s+1}, \dots, x_t\}$  ( $x_i \in \mathbb{Z}^d, 0 \leq s \leq i \leq t$ ) 为最近临近路径序列，即  $|x_{s+i+1} - x_{s+i}| = 1$ 。若考虑分支随机过程的所有  $(0, T)$  路径，则这些路径的总数为  $B(T)$ ，其收敛性能很大程度上依赖于  $m$ 。令  $d$  为待优化问题的维数（相当于 TSP 的城市规模），这里需要考虑两种情况：

- (1)  $m > 2d$  时的情况：用分支随机路径进行收敛性证明；
- (2)  $m > 1$  时的情况：用分支 Wiener 过程进行收敛性证明。

#### 3.7.2.1 当 $m > 2d$ 时的收敛性证明

令  $\{x_N=0, x_{N+1}, \dots, x_{N+T}\}$  ( $T=1, 2, \dots$ ) 为分支随机路径上的路径 ( $N, N+T$ )，且令  $\tilde{\lambda}(s, t) = \tilde{\lambda}\{x_s, x_{s+1}, \dots, x_t\}$  为  $(s, t)$  路径的数目，则对于任意与  $(s, t+1)$  最临近的序列  $\{x_s, x_{s+1}, \dots, x_{t+1}\}$ ，当  $0 < \epsilon < 1$  时，有

$$P\left\{\tilde{\lambda}(s, t+1) < (1-\epsilon) \frac{m}{2d} \tilde{\lambda}(s, t) \mid \tilde{\lambda}(s, t)\right\} \leq \exp(-O(1)\epsilon^2 \tilde{\lambda}(s, t))$$

进而，有

$$P\left\{\tilde{\lambda}(s, t+1) < (1-\epsilon) \frac{m}{2d} K \mid \tilde{\lambda}(s, t) \geq K\right\} \leq \exp(-O(\epsilon^2 K)) \quad (3.7.2)$$

同理，若  $x_{t+2}$  是  $x_{t+1}$  的近邻，则有

$$\begin{aligned} P\left\{\tilde{\lambda}(s, t+2) \geq \left((1-\epsilon) \frac{m}{2d}\right)^2 K \mid \tilde{\lambda}(s, t) \geq (1-\epsilon) \frac{m}{2d} K\right\} \\ \geq 1 - \exp\left(-O\left((1-\epsilon)\epsilon^2 \frac{m}{2d} K\right)\right) \end{aligned} \quad (3.7.3)$$

由此，可归纳为

$$\begin{aligned} P\left\{\prod_{i=1}^n \left\{\tilde{\lambda}(s, t+i) \geq \left((1-\epsilon) \frac{m}{2d}\right)^i K\right\} \mid \tilde{\lambda}(s, t) \geq K\right\} \\ \geq \prod_{i=1}^n \left(1 - \exp\left(-O(\epsilon^2 K \left((1-\epsilon) \frac{m}{2d}\right)^{i-1})\right)\right) \\ \geq 1 - \exp(-O(\epsilon^2 K)) \end{aligned} \quad (3.7.4)$$

设  $\lambda_i$  ( $i=1, 2, \dots, d$ ) 表示基于信息量和路径能见度的路径再生率, 且  $\lambda_i \propto \exp(-O(\epsilon^2 K))$ , 其中  $K$  表示还未搜索路径的序号。同时, 令  $\mu_i$  ( $i=1, 2, \dots, d$ ) 表示基于信息素挥发速度的路径消逝率, 且  $\mu_i \propto \exp(-O(\epsilon^2 K))$ , 令

$$\begin{aligned}\lambda_i &= C \exp(-O(\epsilon^2 K)) \\ \mu_i &= 1 - C \exp(-O(\epsilon^2 K))\end{aligned}$$

则根据存亡过程公式<sup>[38]</sup>, 有

$$P_d = \prod_{i=0}^d \frac{\lambda_i}{\mu_i + 1} P_0 \quad (3.7.5)$$

式中,  $P_0$  表示初始分布,  $P_d$  表示稳态(最终)分布。取  $0 < \epsilon < 1$  且  $P_1 + P_2 + \dots + P_d = 1$ , 可得

$$\left\{ \begin{aligned} &\frac{C \exp(-\epsilon^2 \cdot d)}{1 - C \exp(-\epsilon^2 \cdot d)} + \frac{C \exp(-\epsilon^2 \cdot d) \cdot C \exp(-\epsilon^2 \cdot (d-1))}{(1 - C \exp(-\epsilon^2 \cdot d)) \cdot (1 - C \exp(-\epsilon^2 \cdot (d-1)))} + \dots \\ &+ \frac{C \exp(-\epsilon^2 \cdot d) \cdot C \exp(-\epsilon^2 \cdot (d-1)) \cdot \dots \cdot C \exp(-\epsilon^2 \cdot 1)}{(1 - C \exp(-\epsilon^2 \cdot d)) \cdot (1 - C \exp(-\epsilon^2 \cdot (d-1))) \cdot \dots \cdot (1 - C \exp(-\epsilon^2 \cdot 1))} \end{aligned} \right\} P_0 = 1 \quad (3.7.6)$$

但由于  $P_d \gg \sum_{j=1}^{d-1} P_j \Rightarrow P_d \cong 1$ , 上式表明该分布是概率为 1 的稳态分布, 从而证明了蚁群算法能以数值为 1 的概率收敛。

[证毕]

### 3.7.2.2 当 $m > 1$ 时的收敛性证明

令  $W_1(t), W_2(t), \dots, W_{B(T)}(t)$ , ( $0 \leq t \leq T$ ) 为  $\mathbb{R}^d$  上  $(0, T)$  区间内分支 Wiener 过程的分支, 其中  $W_i(\cdot)$  表示非独立性 Wiener 过程; 令  $w_T(i, x) = T^{-1}W_i(xT)$ ,  $i=1, 2, \dots, B(T)$ , 且  $0 \leq x \leq 1$ ;  $w_T = \{w_T(1, x), w_T(2, x), w_T(B(T), x), \dots\}$ ,  $0 \leq x \leq 1$ ; 令  $j_{\alpha\beta} = 0, 1, 2, \dots$ , 其中  $\alpha = 1, 2, \dots, r$ ,  $\beta = 1, 2, \dots, d$ , 则

$$\begin{cases} j^{(1)} = (j_{11}, j_{12}, \dots, j_{1d}) \\ j^{(2)} = (j_{21}, j_{22}, \dots, j_{2d}) \\ \vdots \\ j^{(r)} = (j_{r1}, j_{r2}, \dots, j_{rd}) \end{cases} \quad (3.7.7)$$

$$J(\mu) = \begin{bmatrix} j^{(1)} \\ j^{(2)} \\ \vdots \\ j^{(\mu)} \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} & \cdots & j_{1d} \\ j_{21} & j_{22} & \cdots & j_{2d} \\ \vdots & \vdots & & \vdots \\ j_{\mu 1} & j_{\mu 2} & \cdots & j_{\mu d} \end{bmatrix}, \quad \mu = 2, 3, \dots, r \quad (3.7.8)$$

令  $W_1^\mu(t), W_2^\mu(t), \dots, W_{B(\mu T)}^\mu(t)$ , ( $0 \leq t \leq \mu T$ ,  $\mu = 2, 3, \dots, r$ ) 为  $(0, \mu T)$  区间内

Wiener 过程的分支,且令  $V_\mu = V_\mu(j_2, j_3, \dots, j_\mu) = \{i : T^{-1}(W_i^{(\mu)}(lT) - W_i^{(\mu)}((l-1)T))\} \in B_l, l=2,3,\dots,\mu$ , 式中  $\mu=2,3,\dots,r$ , 则对于任意  $l=1,2,\dots,\mu$ , 有

$$B_l = B_l(j_l) = \left\{(y_1, y_2, \dots, y_d) : \frac{j_\beta}{M} \leq y_\beta \leq \frac{j_\beta + 1}{M}, \beta = 1, 2, \dots, d\right\}$$

而对于任意  $0 < \epsilon < 1$ , 有

$$P\left\{V_2(j_2) < \frac{1-\epsilon}{(2\pi)^{d/2}} \frac{m^{2T}}{M^d} B \exp\left(-\frac{x_2^2}{2} T\right)\right\} \leq e^{-CT}$$

式中,  $C$  为足够小的正常数。由此, 可得

$$P\left\{V_\mu(j_1, j_2, \dots, j_\mu) < \left(\frac{1-\epsilon}{(2\pi)^{d/2}} \frac{1}{M^\mu}\right)^{\mu-1} m^{\mu T} B \exp\left(-\frac{T}{2} \sum_{i=2}^\mu x_i^2\right)\right\} \leq e^{-CT} \quad (3.7.9)$$

在某个点上根据信息量、信息素的挥发度和能见度来定义路径的存亡率, 即令  $\lambda_t$  为  $t$  时刻的路径生存率, 有

$$\lambda_t \propto e^{-C(T-t)}, 1 \leq t \leq T \Rightarrow \lambda_t = \rho e^{-C(T-t)}, \rho \text{ 为常数} \Rightarrow \mu_t = t \text{ 时刻路径的消逝率} \quad (3.7.10)$$

从而, 有

$$\mu_t \propto 1 - e^{-C(T-t)} \Rightarrow \mu_t = 1 - \rho e^{-C(T-t)}, 1 \leq t \leq T \quad (3.7.11)$$

根据存亡过程, 则有

$$P_T = \prod_{t=0}^T \frac{\lambda_t}{\mu_t + 1} P_0 \quad (3.7.12)$$

式中,  $P_0$  表示初始分布,  $P_T$  表示  $T$  时刻的稳态(最终)分布, 且有

$$P_1 + P_2 + \dots + P_T = 1 \quad (3.7.13)$$

从而

$$\left\{ \frac{\rho e^{-C(T-1)}}{1 - \rho e^{-C(T-1)}} + \frac{\rho e^{-C(T-1)} \cdot \rho e^{-C(T-2)}}{(1 - \rho e^{-C(T-1)}) \cdot (1 - \rho e^{-C(T-2)})} + \dots + \frac{\rho e^{-C(T-1)} \cdot \rho e^{-C(T-2)} \cdot \dots \cdot \rho e^{-C(T-T+1)} \cdot \rho e^{-C(T-T)}}{(1 - \rho e^{-C(T-1)}) \cdot (1 - \rho e^{-C(T-2)}) \cdot \dots \cdot (1 - \rho e^{-C(T-T+1)}) \cdot (1 - \rho e^{-C(T-T)})} \right\} P_0 = 1 \quad (3.7.14)$$

由此, 可得

$$P_T \cong 1 \quad (3.7.15)$$

公式(3.7.15)表明该分支 Wiener 过程是概率为 1 的稳态分布, 从而证明了蚁群算法以数值为 1 的概率收敛。

[证毕]

### 3.8 一种简单蚁群算法及其收敛性分析

本节对一种可用于函数优化的简单蚁群算法的收敛性及相关参数做了初步分

析<sup>[12]</sup>。首先在给定近似精度的基础上,基于 Markov 过程分析了简单蚁群算法的全局收敛性,随后探讨了简单蚁群算法中衰减度、变异率等参数对其性能的影响。

### 3.8.1 简单蚁群算法描述

考虑目标函数

$$f_{\min} = \min\{f(x) | x \in B^L\}, B^L = \{0,1\}^L, 0 < f(x) < +\infty \quad (3.8.1)$$

设  $t$  时刻蚂蚁  $A(t) = \{a_0(t), a_1(t), \dots, a_k(t), \dots, a_N(t)\}$ , 其中  $a_k(t) \in B^L$ ,  $N$  表示蚂蚁数目, 定义  $X_k(A(t)) = a_k(t)$ 。对于  $x \in B^L$ , 定义  $r_j(x)$  表示  $x$  的第  $j$  位 ( $j = 0, 1, \dots, L-1$ ), 取值范围为  $\{0, 1\}$ 。设信息量集合  $W(t) = \{w_{00}(t), w_{10}(t), \dots, w_{ij}(t), \dots, w_{0,L-1}(t), w_{1,L-1}(t)\}$ , 其中  $i=0, 1; j=0, 1, \dots, L-1$ 。简化的蚁群算法步骤如下:

(1) 置  $t=0, w_{ij}(0)=w_0$  ( $w_0 > 0$  为常数)。

(2) For  $k=1$  to  $N, j=0$  to  $L-1$  do  $r_j(a_k(t))$ , 按概率

$$\Pr_{ij}(t) = (1 - p_{\text{mut}}) \frac{(w_{ij}(t))^{\alpha} \cdot (E_{ij})^{\beta}}{(w_{0j}(t))^{\alpha} \cdot (E_{0j})^{\beta} + (w_{1j}(t))^{\alpha} \cdot (E_{1j})^{\beta}} + \frac{p_{\text{mut}}}{2} \quad (3.8.2)$$

取为 0, 1。其中,  $0 < p_{\text{mut}} < 1$ , 而  $E_{0j}$  和  $E_{1j}$  分别表示在第  $j$  位取为 0 和 1 的静态启发值。

(3) 置  $a_0(t) = a_b(t)$  (其中  $b \in \{0, 1, \dots, N\}$ , 且  $f(a_b(t)) = \min\{f(a_k(t)) | k=0, 1, \dots, N\}$ )。

For  $i=0$  to  $1, j=0$  to  $L-1$  do

置  $w_{ij}(t) = w_{ij}(t)(1-\eta)$  (其中  $0 < \eta < 1$ ,  $\eta$  为衰减度系数)。

(4) For  $k=1$  to  $N, j=0$  to  $L-1$  do

置  $w_{r_j(a_k(t)),j}(t+1) = w_{r_j(a_k(t)),j}(t) + \delta/f(a_k(t))$ , 其中  $\delta$  为常数。

(5) 令  $t=t+1$ , 如果  $t$  满足事先给定的最大迭代次数或  $f(a_0(t))$  优化趋势不明显时, 输出当前最优解  $a_0(t)$ ; 否则转入第(2)步。

上述算法本质上已经具备了基本蚁群算法的特征, 这里对其做了如下两点改进:

(1) 类似遗传算法的最优保存策略<sup>[39]</sup>, 在第(3)步中引入  $a_0(t)$  用来记录当前的最优解;

(2) 在公式(3.8.2)中, 这里引入  $P_{\text{mut}}$ , 使蚂蚁能以较小概率选择非最优解, 这一点类似于遗传算法的变异操作<sup>[40]</sup>。这里将第(2)步中的选择概率公式取为

$$\Pr_{ij}(t) = \frac{(w_{ij}(t))^{\alpha} \cdot (E_{ij})^{\beta}}{(w_{0j}(t))^{\alpha} \cdot (E_{0j})^{\beta} + (w_{1j}(t))^{\alpha} \cdot (E_{1j})^{\beta}}, i = 0, 1 \quad (3.8.3)$$

而文献[41]将选择概率公式定义为

$$\Pr_{ij}(t) = \frac{(w_{ij}(t))^{\alpha} + (E_{ij})^{\beta}}{(w_{0j}(t))^{\alpha} \cdot (E_{0j})^{\beta} + (w_{1j}(t))^{\alpha} \cdot (E_{1j})^{\beta}}, i = 0, 1 \quad (3.8.4)$$

**引理 3.8.1** (1)  $w_0(1-\eta)^t \leq w_{ij}(t) \leq w_0(1-\eta)^t + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\min}}$  (3.8.5)

(2)  $2w_0(1-\eta)^t + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\max}} \leq w_{0j}(t) + w_{1j}(t)$   
 $\leq 2w_0(1-\eta)^t + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\min}}$  (3.8.6)

**证明** 只证  $w_{ij}(t) \leq w_0(1-\eta)^t + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\min}}, t=0$  时命题显然成立。

由算法规则  $w_{ij}(t_0+1) \leq w_{ij}(t_0)(1-\eta) + \frac{N\delta}{\eta \cdot f_{\min}}$ , 再用数学归纳法易证。

[证毕]

### 3.8.2 收敛性分析

很明显,  $\{A(t) | t = 0, 1, 2, \dots\}$  是一个随机过程,  $t$  时刻  $A(t)$  的出现概率取决于信息量集合  $W(t)$ , 而  $W(t)$  又取决于初始信息量以及  $A(t-1), A(t-2), \dots, A(0)$ , 因此,  $t$  时刻状态  $A(t)$  的出现概率是与前面所有状态相关的。但在给定近似精度的前提下, 当  $M$  为足够大的常数时, 可认为  $A(t)$  的出现概率只与  $A(t-1), A(t-2), \dots, A(\max(0, t-M))$  有关。

下面说明上述“近似”的合理性, 对任意时刻  $t$ , 设  $t_0 = \max(0, t-M)$ , 同时可设  $w_{ij}(t) = w_{ij}(t_0)(1-\eta)^{t-t_0} + g_{ij}(t_0, t-t_0)$ , 其中函数  $g_{ij}(t_0, x)$  ( $t_0, x$  取为非负整数) 递归定义为:

$$(1) g_{ij}(t_0, 0) = 0;$$

$$(2) g_{ij}(t_0, x+1) = g_{ij}(t_0, x)(1-\eta) + \sum_{k=1}^N \frac{\delta(r_j(a_k(t_0+x)) \circ i)}{f(a_k(t_0+x))} \quad (\circ \text{为同或运算})$$

由引理 3.8.1 和  $0 < 1-\eta < 1$ ,  $w_{ij}(t_0) \leq w_0(1-\eta)^{t_0} + \frac{(1-(1-\eta)^{t_0}) \cdot N\delta}{\eta \cdot f_{\min}}$ , 可知必存在正数  $C$ , 使  $w_{ij}(t_0) \leq C$  成立(若有上界则必有上确界)。对于任意小的正数  $\epsilon$ , 当  $M = \lceil \frac{\ln \epsilon - \ln C}{\ln(1-\eta)} \rceil + 1$  时, 有

$$|w_{ij}(t) - g_{ij}(t_0, t-t_0)| \leq \epsilon \quad (3.8.7)$$

由于  $g_{ij}(t_0, t-t_0)$  只与  $A(t_0), A(t_0+1), A(t_0+2), \dots, A(t-1)$  有关, 而与  $A(t_0-1), A(t_0-2), \dots, A(0)$  无关, 故原来的“近似”也是合理的。

由于蚁群算法一般要迭代较长时间, 在下述分析中不妨只讨论  $t \geq M-1$  时刻的情形。定义状态向量  $S(t) = [s_1(t), s_2(t), \dots, s_m(t), \dots, s_M(t)]^T$ , 其中  $s_m(t) = A(t-m+1)$ , 并定义  $P_m(S(t)) = s_m(t)$ 。设在这种情形下由  $S(t)$  所有可能取值构成的空间为  $I$ , 易知  $I$  中所含状态向量的个数不超过  $2^{(N+1)L^M}$ 。显然, 状态  $S(t)$  的出现概率只取决于状态  $S(t-1)$ , 故  $\{S(t) | t = 0, 1, 2, \dots\}$  是一个有限的 Markov 过程。由于当  $t \geq M-1$  时, 残留在路径上的初始信息素几乎耗尽, 故此时蚁群的 Markov 过程体现出时齐性。

定义等价关系  $R$  为  $S_1 R S_2, RS_2 \cong f(X_0(\pi_1(S_1))) = f(X_0(\pi_1(S_2)))$ , 其中  $S_1, S_2 \in I$ 。

先将  $I$  按关系  $R$  划分为若干等价类, 设  $I = I_1 \cup I_2 \cup \dots \cup I_{L_f}$ , 可知  $L_f$  为  $f(x)$  所有不同取值的总个数, 且  $L_f \leq L$ 。不妨规定当  $S_1 \in I_u, S_2 \in I_v$  时, 若  $u < v$ , 必有

$$f(X_0(\pi_1(S_1))) < f(X_0(\pi_1(S_2))) \quad (3.8.8)$$

因此, 若  $S \in I_1$ , 则有

$$f(X_0(\pi_1(S_1))) = f_{\min} \quad (3.8.9)$$

设当  $f(X_0(A(t))) = f_{\min}$  时, 由  $A(t)$  的所有可能取值构成空间为  $J_1$ , 其中共有  $2^{NL}$  个元素。令  $H_{uv}$  表示  $I_u$  中元素向  $I_v$  中元素的一步迁移矩阵。

- 引理 3.8.2** (1) 矩阵  $H_{11}$  为概率矩阵(即每行之和为 1);  
(2) 当  $u > v$  时,  $H_{uv}$  为全零矩阵(由简单蚁群算法保存当前最优解易证)。

这样, 所有状态向量的一步状态迁移概率矩阵可表示为

$$P = (p_{uv}) = \begin{vmatrix} H_{11} & 0 & \cdots & 0 \\ H_{11} & H_{11} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ H_{11} & H_{11} & \cdots & H_{11} \end{vmatrix} = \begin{vmatrix} H_{11} & 0 \\ H & Q \end{vmatrix} \quad (3.8.10)$$

**引理 3.8.3** 在公式(3.8.10)中, 状态迁移概率矩阵  $H$  中的每行至少有一个元素值大于零。

**证明** 易知矩阵  $H$  是  $I - I_1$  中元素向  $I_1$  中元素的一步迁移概率矩阵。设  $S(t) \in I - I_1 (t \geq M-1), f(b) = f_{\min}$ 。显然,  $S(t+1) \in I_1$  的概率不小于  $A(t+1)$  中至少一个分量为  $b$  的概率, 而后者的表达式为

$$1 - \prod_{k=1}^N \left( 1 - \prod_{i=0}^{L-1} \left( (1 - P_{\text{mut}}) \frac{(w_{r_j(b),j}(t))^{\alpha} \cdot (E_{r_j(b),j})^{\beta}}{(w_{0j}(t))^{\alpha} \cdot (E_{0j})^{\beta} + (w_{1j}(t))^{\alpha} \cdot (E_{1j})^{\beta} + \frac{P_{\text{mut}}}{2}} \right) \right) > 0 \quad (3.8.11)$$

故引理 3.8.3 得证。

[证毕]

由引理 3.8.3 易知:

**引理 3.8.4** 公式(3.8.10)的  $Q$  中的每行之和小于 1。

**引理 3.8.5** 概率矩阵  $H_{11}$  非周期且不可约。

**证明** 只需证  $H_{11}^n > 0 (n \geq M)$  严格成立即可。

设  $S_u, S_v \in I_1$ ,  $S(t_0) = S_u$ ; 当  $t \geq t_0 + M$  时, 设  $t' = t - M (t \geq M - 1)$ ,  $S(t')$  为  $S(t_0)$  经  $t' - t_0$  转移后的状态向量, 显然  $p_{S_u, S(t')}^{(t'-t_0)} > 0$ 。公式(3.8.2)所示的概率公式表明, 解的每位取 0 或 1 的概率均大于零, 由此可知, 对于所有的  $m = 1, \dots, M$  而言,  $\Pr(A(t'+m)) = \pi_{M-m+1}(S_v) > 0$  均成立。由定义, 对于  $m = 1, \dots, M$  而言,  $\pi_{M-m+1}(S(t)) = \pi_1(S(t'+m)) = A(t'+m)$  成立。因此  $P_r(S(t) = S_v | S(t') \in I_1) > 0$ , 即  $p_{S(t'), S_v}^{(M)} > 0$ , 并由  $p_{S_u, S(t')}^{(t'-t_0)} \geq p_{S_u, S(t')}^{(t'-t_0)} \cdot p_{S(t'), S_v}^{(M)} > 0$  可知, 原命题成立。

**定理 3.8.1** 简单蚁群算法以概率 1 收敛于全局最优解。

**证明** 考虑极限

$$\lim_{t \rightarrow +\infty} P^t = \begin{vmatrix} \lim_{t \rightarrow +\infty} H_{11}^t & 0 \\ \lim_{t \rightarrow +\infty} \sum_{m=1}^t Q^{m-1} \cdot H \cdot H_{11}^{t-m} & \lim_{t \rightarrow +\infty} Q^t \end{vmatrix} \quad (3.8.12)$$

根据引理 3.8.4 可得

$$\lim_{t \rightarrow +\infty} Q^t = 0 \quad (3.8.13)$$

由引理 3.8.5 和 Markov 过程的极限分布定理可知:

(1) 存在唯一概率向量  $\Psi^T$ , 使  $\Psi^T H_{11} = \Psi^T$  成立。

(2) 对于任意初始概率向量  $\lambda^T$ , 有

$$\lim_{t \rightarrow +\infty} \lambda^T H_{11}^t = \Psi^T \quad (3.8.14)$$

(3)  $\bar{H}_{11} = \lim_{t \rightarrow +\infty} H_{11}^t$  存在, 且每行为  $\Psi^T$ 。设  $\xi_t = \sum_{m=1}^t Q^{m-1} \cdot H \cdot H_{11}^{t-m}$ , 可得

$$\xi_t = \xi_{t-1} \cdot H_{11} + Q^t \cdot H \quad (3.8.15)$$

因为  $\lim_{t \rightarrow +\infty} \xi_t = \lim_{t \rightarrow +\infty} \xi_{t-1} \cdot H_{11} + \lim_{t \rightarrow +\infty} Q^t \cdot H$ , 由此可得

$$\lim_{t \rightarrow +\infty} \xi_t = \lim_{t \rightarrow +\infty} \xi_t \cdot T_{11} \quad (3.8.16)$$

又因为  $\Psi^T$  唯一, 所以  $\lim_{t \rightarrow +\infty} \xi_t$  的每行向量为  $\Psi^T$ 。注意到  $\Psi^T$  为概率向量, 各分量之和为 1, 因此不论  $t = M - 1$  时蚁群处于何种状态, 当  $t \rightarrow +\infty$  时, 必以数值为 1 的概率收敛于全局最优解。

[证毕]

### 3.8.3 公式与参数讨论

这里将定性讨论在其他参数给定的条件下, 关于选择概率公式和衰减度  $\eta$  的取值问题。当不考虑变异因素时, 有如下定理:

**定理 3.8.2** 若简单蚁群算法中第(2)步的选择概率公式取为公式(3.8.3), 则算法不以概率 1 遍历解空间。

**证明** 考虑迭代次数  $t$  从 0 到  $\infty$  时, 设解的第  $j$  位每次都选择 1 的概率为  $w_{1j}$ , 此时信息量只在  $w_{1j}(t)$  上增加, 则由数学归纳法可得

$$w_{1j}(t) \leq w_0(1-\eta)^t + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\max}} \quad (3.8.17)$$

及

$$w_{0j}(t) \leq w_0(1-\eta)^t \quad (3.8.18)$$

因此, 有

$$\lambda_{1j}(t) \geq \prod_{t=0}^{\infty} \left( 1 - \frac{(w_0(1-\eta)^t)^{\alpha} \cdot (E_{0j})^{\beta}}{\left( (w_0(1-\eta)^t)^{\alpha} \cdot (E_{0j})^{\beta} + (w_0(1-\eta)^t)^{\alpha} + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\max}} \right)^{\alpha} \cdot (E_{1j})^{\beta}} \right)^N \quad (3.8.19)$$

易知必存在正常数  $C$ , 使得

$$\left( (w_0(1-\eta)^t)^{\alpha} \cdot (E_{0j})^{\beta} + (w_0(1-\eta)^t)^{\alpha} + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\max}} \right)^{\alpha} \cdot (E_{1j})^{\beta} \geq C \geq 0$$

成立(有下界必有下确界), 由此可得

$$\lambda_{1j}(t) > \prod_{t=0}^{\infty} \left( 1 - \frac{(w_0(1-\eta)^t)^{\alpha} \cdot (E_{0j})^{\beta}}{C} \right)^N$$

即

$$(\lambda_{1j}(t))^{\frac{1}{N}} \geq \prod_{t=0}^{\infty} \left( 1 - \frac{(w_0(1-\eta)^t)^{\alpha} \cdot (E_{0j})^{\beta}}{C} \right) \quad (3.8.20)$$

又因为  $0 < 1-\eta < 1$ , 所以  $\sum_{t=0}^{\infty} \left( 1 - \frac{(w_0(1-\eta)^t)^{\alpha} \cdot (E_{0j})^{\beta}}{C} \right) > 0$  收敛。

再根据文献[42]中定理 2 所给出的无穷乘积收敛判别定理, 有

$$\prod_{t=0}^{\infty} \left( 1 - \frac{(w_0(1-\eta)^t)^{\alpha} \cdot (E_{0j})^{\beta}}{C} \right) > 0 \quad (3.8.21)$$

由此可得  $\lambda_{1j} > 0$ 。而在迭代次数  $t$  从 0 到  $\infty$  过程中, 第  $j$  位至少一次选择 0 的概率  $\lambda'_{0j} = 1 - \lambda_{1j} < 1$ 。这说明算法并不保证以数值为 1 的概率遍历解空间内的每个状态, 因而不保证以数值为 1 的概率搜索到最优解。

根据定理 3.8.1, 部分文献采用公式(3.8.3)作为概率公式在理论上存在着一定的缺陷。但在其他参数给定的条件下, 当  $\eta$  较小时,  $\lambda'_{0j}$  较大; 而当  $\eta \rightarrow 1$  时,  $\lambda'_{0j}$  最小, 也最不利于对解空间的遍历。弥补该问题可将选择概率公式取为公式(3.8.2), 即引入变异机制; 或将选择概率公式取为公式(3.8.4)。用类似于文献[42]中定理 2 的证明方法, 可知在两种条件下均能保证对解空间的遍历。但后者随着迭代次数的增加, 启发因素的作用会越来越小, 故很少有文献采用该形式。

针对当  $\eta \rightarrow 0$  时简单蚁群算法性能是否最佳这一问题的探讨,这里仍采用公式(3.8.3)作为选择概率公式,当  $t$  充分大时,不妨设  $\Pr_{0j}(t+1) \geq \Pr_{0j}(t)$ (否则必有  $\Pr_{1j}(t+1) \geq \Pr_{1j}(t)$ )。由引理 3.8.1 易证,对于任意  $j$ ,有

$$\begin{aligned} w_j(t) &= w_{0j}(t) + w_{1j}(t) \geq 2w_0(1-\eta)^t \\ &\quad + \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\max}} > \frac{(1-(1-\eta)^t) \cdot N\delta}{\eta \cdot f_{\max}} \end{aligned} \quad (3.8.22)$$

$$w_{0j}(t+1) \leq w_{0j}(t)(1-\eta) + \frac{N\delta}{f_{\min}} \quad (3.8.23)$$

$$w_{1j}(t+1) \leq w_{1j}(t)(1-\eta) + \frac{N\delta}{f_{\min}} \quad (3.8.24)$$

由不等式(3.8.23)、式(3.8.24),可得

$$\Pr_{0j}(t+1) < \frac{\left(w_{0j}(t)(1-\eta) + \frac{N\delta}{f_{\min}}\right)^\alpha \cdot (E_{0j})^\beta}{\left(w_{0j}(t)(1-\eta) + \frac{N\delta}{f_{\min}}\right)^\alpha \cdot (E_{0j})^\beta + \left(w_{1j}(t)(1-\eta) - \frac{N\delta}{f_{\min}}\right)^\alpha \cdot (E_{1j})^\beta} \quad (3.8.25)$$

设函数  $\mu(x) = \frac{(x)^\alpha \cdot (E_{0j})^\beta}{(x)^\alpha \cdot (E_{0j})^\beta + (1-x)^\alpha \cdot (E_{1j})^\beta}$ , 则可证

$$\Pr_{0j}(t) = \mu\left(\frac{w_{0j}(t)}{w_j(t)}\right)$$

整理不等式(3.8.25),可得

$$\Pr_{0j}(t+1) < \mu\left(\frac{w_{0j}(t)}{w_j(t)} + \frac{N\delta}{w_j(t)f_{\min}}\right) \quad (3.8.26)$$

由于  $\mu(x)$  为区间  $[0, 1]$  的连续单增函数,则

$$0 < \Pr_{0j}(t+1) - \Pr_{0j}(t) < \mu\left(\frac{w_{0j}(t)}{w_j(t)} + \frac{\eta \cdot f_{\max}}{(1-(1-\eta)^t) \cdot f_{\min}}\right) - \mu\left(\frac{w_{0j}(t)}{w_j(t)}\right) \quad (3.8.27)$$

根据  $\lim_{t \rightarrow \infty, \eta \rightarrow 0} \frac{\eta \cdot f_{\max}}{(1-(1-\eta)^t) \cdot f_{\min}} = 0$  和  $\mu(x)$  的连续性,当  $t$  充分大时,若  $\eta \rightarrow 0$

将使  $|\Pr_{0j}(t+1) - \Pr_{0j}(t)|$  很小,这不利于简单蚁群算法中自组织行为的持续进行。

[证毕]

### 3.9 遗传-蚁群算法的 Markov 收敛性分析

遗传算法(genetic algorithm, GA)具有大范围的快速全局搜索能力,但当求解到一定程度时,往往作大量的冗余迭代,对于系统中的反馈信息利用不够,求解  
• 90 •

效率降低。遗传算法与蚁群算法的融合正是基于遗传算法的快速全局搜索能力和蚁群算法的正反馈收敛机制,初期采用遗传算法过程生成信息素分布,后期利用蚁群算法正反馈求精确解,优势互补。基于上述思想,本节在介绍一种遗传-蚁群算法融合模型的基础上,基于 Markov 理论对该模型的收敛性作了简要分析<sup>[13]</sup>,并证明其优化解满意值序列单调不增且收敛。

### 3.9.1 遗传-蚁群算法的基本原理

通过遗传算法得到一定的路径信息素,因此可把信息素初值设为  $\tau_S = \tau_C + \tau_G$ ,其中  $\tau_C$  是根据所求解问题规模而给定的一个信息量常数,  $\tau_G$  是遗传算法求解结果所转换的信息量。

设  $S^N = \{x = (x_1, x_2, \dots, x_N), x_i \in S (i \leq N)\}$  为种群(优化解)空间;  $S^2 = \{(x_1, x_2), x_1, x_2 \in S\}$  为母体空间,其中  $S$  表示个体空间,  $x_i$  表示个体,其长度  $L$  称为链长;  $N$  为种群(优化解)规模;  $n$  为优胜选择遗传算法的迭代次数;  $m$  为 Ant-Cycle 模型的移动周期;  $P$  为  $S^N$  上的概率分布。这里所研究的遗传-蚁群算法由 4 个过程构成,即选择算子  $T_s$ 、杂交算子  $T_c$ 、变异算子  $T_m$  和信息素算子  $T_g$ 。

### 3.9.2 基本定义<sup>[32]</sup>

**定义 3.9.1** 适应值函数  $f: S \rightarrow R^+$ ,即个体空间到正实数空间的映射,则全局最优解集为  $M = \{x; \forall y \in S, f(x) \leq f(y)\}$ ,满意解集为  $B = \{x \in M; \forall y \notin B, f(x) \leq f(y)\}$ 。

**定义 3.9.2** 选择算子  $T_s: S^N \rightarrow S$ ,此即从种群中选择个体的随机映射,则母体选择算子的概率为

$$P\{T_s(x) = (x_i, x_j)\} = \frac{f(x_i)}{\sum_{k=1}^N f(x_k)} \cdot \frac{f(x_j)}{\sum_{l=1}^N f(x_l)} \quad (3.9.1)$$

**定义 3.9.3** 杂交算子  $T_c: S^2 \rightarrow S$ ,此即从母体空间到个体空间的映射,令  $k$  表示可以生成  $y$  的基因位置的个数,则单点杂交和单点随机杂交的概率分别为

$$p_c = P\{T_c(x_1, x_2) = y\} = \frac{k}{l} \quad (3.9.2)$$

$$P\{T_c(x_1, x_2) = y\} = \begin{cases} \frac{kp_c}{l}, & y \neq x_1 \\ (1 - p_c) + \frac{kp_c}{l}, & y = x_1 \end{cases} \quad (3.9.3)$$

**定义 3.9.4** 变异算子  $T_m: S \rightarrow S$ ,此即从个体空间到个体空间的随机映射。

$$P\{T_m(x) = y\} = p_m^{d(x,y)} (1 - p_m)^{L-d(x,y)} \quad (3.9.4)$$

式中,  $d(x, y) = \sum_{i=1}^l |a_i - b_i|$  表示  $X$  与  $Y$  之间的 Hamming 距离,  $p_m$  表示变异概率。

**定义 3.9.5** 信息素算子  $T_g: S^N \rightarrow S$ , 此即从 Ant-Cycle 序列中选择最优个体的信息素更新映射, 信息素算子概率为

$$P\{T_g(x) = x_i\} = \frac{f(x_i)}{\sum_{k=1}^N f(x_k)} \quad (3.9.5)$$

### 3.9.3 遗传-蚁群算法的 Markov 收敛性分析

**引理 3.9.1**<sup>[32]</sup> 优胜选择遗传算法种群序列  $\{x(n); n \geq 0\}$  是有限齐次 Markov 链。

**定理 3.9.1** 遗传-蚁群算法的优化解序列  $\{x(n), n \geq 0\}$  是有限齐次 Markov 链。

**证明** 由引理 3.9.1 知优胜选择遗传算法  $x(n+1)$  仅与  $x(n)$  有关, 与迭代次数  $n$  无关; 由引理 3.2.1 易知, 蚁群算法  $\{\tau(m+1), x(m+1), f^*(m+1)\}$  仅与  $\{\tau(m), x(m), f^*(m)\}$  有关, 而与搜索周期  $m$  无关。同时, 由 Ant-Cycle 模型可知, 每次迭代仅更新 Ant-Cycle 模型中的最优信息素, 因此, 这里取

$$\begin{aligned} x(n+1) &= (x_i(n+1)) = T_g^i \circ T_m^i \circ T_c^i \circ T_s^i(x(n)), \\ &(i \leq N-1); x_N(n+1) = x_{i_0}(n) \end{aligned} \quad (3.9.6)$$

式中,  $i_0 = \arg \min_j \{f(x_j(n))\}$  表示  $f(x_j(n))$  取最小值的个体为  $x_j(n)$ , 且转移概率矩阵为

$$\begin{aligned} P\{x, y\} &= P\left\{\frac{x(n+1) = y}{x(n) = x}\right\} \\ &= \begin{cases} \prod_{k=1}^N P\{T(x(n))_k = y_k\}, & \exists i_0 \in M(x), \text{使 } y_N = x_{i_0} \\ 0, & \text{否则} \end{cases} \quad (3.9.7) \end{aligned}$$

上式表明,  $x(n+1)$  仅与  $x(n)$  有关, 而与  $n$  无关。遗传-蚁群算法的优化解序列  $\{x(n); n \geq 0\}$  是有限齐次 Markov 链。

[证毕]

**定理 3.9.2** 遗传-蚁群算法 Markov 链序列的优化解满意值序列是单调不增的, 即对于任意的  $n \geq 0$ , 有

$$F(x(n)) \geq F(x(n+1)) \quad (3.9.8)$$

**证明** 首先, 令  $i_0 = \arg \min_j \{f(x_j(n))\}$ , 由于遗传-蚁群算法中采用的是优胜选择遗传算法, 则

$$x_N(n+1) = x_{i_0}(n), F(x(n)) \geq F(x_N(n+1)) \geq F(x(n+1))$$

其次,遗传-蚁群算法中采用了 Ant-Cycle 模型,则

$$x_N(m+1) = x_{i_0}(m), F(x(m)) \geq F(x_N(m+1)) \geq F(x(m+1))$$

(3.9.9)

而 Ant-Cycle 模型是以优胜选择遗传算法结果为初始分布,具有优值继承性,因而对于任意  $n \geq 0$ ,有  $F(x(n)) \geq F(x(n+1))$ ,即遗传-蚁群算法的 Markov 链序列的优化解满意值序列是单调不增的。

[证毕]

**引理 3.9.2**<sup>[32]</sup> 优胜选择遗传算法种群 Markov 链序列  $\{x(n), n \geq 0\}$  以数值为 1 的概率收敛到满意种群集  $M^*$  的子集  $M_0^*$ , 定义  $M_0^* = \{y = (y_1, \dots, y_N); y_N \in M\}$ , 即有

$$\lim_{n \rightarrow \infty} P\left\{\frac{x(n) \in M_0^*}{x(0) = x_0}\right\} = 1 \quad (3.9.10)$$

**定理 3.9.3** 遗传-蚁群算法的优化解 Markov 序列以数值为 1 的概率收敛到满意解集  $B$  的子集  $B_0^*$ , 这里定义  $B_0^* = \{y = (y_1, \dots, y_N); y_N \in M\}$ , 即有

$$\lim_{n \rightarrow \infty} P\left\{\frac{x(n) \in B_0^*}{x(0) = x_0}\right\} = 1 \quad (3.9.11)$$

**证明** 设  $x'$  是  $f(x)$  的唯一最小值解,由定理 3.9.1 和引理 3.9.2 知  $P\{x, y\}$  有如下性质:

- (1) 当  $x, y \in B_0^*$  时,有  $P\{x, y\} > 0, P\{y, x\} > 0$ , 即  $x \leftrightarrow y$ ;
- (2) 当  $x \in B_0^*, y \notin B_0^*$ , 时,有  $P\{y, x\} = 0$ , 即  $x \rightarrow y$ 。

因此,  $B_0^*$  为正常返的非周期不可约闭集,  $S^N \setminus B_0^*$  为非常返的状态集。

$$\lim_{n \rightarrow \infty} P\left\{\frac{x(n) = y}{x(0) = x_0}\right\} = \begin{cases} \pi(y), & y \in B_0^* \\ 0, & y \notin B_0^* \end{cases} \quad (3.9.12)$$

故有

$$\lim_{n \rightarrow \infty} P\left\{\frac{x(n) \in B_0^*}{x(0) = x_0}\right\} = 1$$

[证毕]

**推论 3.9.1** 对于遗传-蚁群算法,如果  $\{x(n), n \geq 0\}$  对于任意满意解集是收敛的,则必收敛到全局最优解集。

因为对于  $\forall \epsilon > 0$ , 有满意解集  $B(\epsilon) = \{y; f(y) \leq f(x^*) + \epsilon\}$ , 其中  $x^*$  表示最优解,即  $x^* \in M$ ,亦即全局最优解集  $M$  即为所有满意解集之交集,  $M$  为最小满意解集。一般而言,实际问题中只要求收敛到满意解集  $B(\epsilon)$  即可。

### 3.10 一类广义蚁群算法(GACA)的收敛性分析

本节研究了一类可用于求解一般形式的非凸、非线性约束优化问题的GACA,

并基于不动点理论对这类 GACA 的收敛性问题作了初步分析<sup>[14, 15]</sup>。

### 3.10.1 GACA 的基本原理

对于一般定义在紧集合上的约束优化问题,可用外点法构造辅助函数<sup>[43]</sup>。这里将不便于计入可行域的约束以罚函数的形式计人到目标函数中,其表达式为

$$\begin{aligned} \min F(X) = & \min \left\{ f(x) + \sigma_1 \sum_{i=1}^{l_0} (h_i(X))^2 \right. \\ & \left. + \sigma_2 \sum_{j=1}^{u_0} [\max\{0, -g_j(X)\}]^2 \right\} \\ \text{s. t. } & \begin{cases} h_i(X) = 0, & i = 1, 2, \dots, l - l_0 \\ g_j(X) \geq 0, & j = 1, 2, \dots, u - u_0 \end{cases} \end{aligned} \quad (3.10.1)$$

式中,  $X = (x_1, x_2, \dots, x_n)^T$  表示待优化向量,  $l, u$  分别表示原问题等式约束和不等式约束的个数。

公式(3.10.1)的可行域记为

$$S = \{X | h_i(X) = 0, i = 1, 2, \dots, l - l_0, g_j(X) \geq 0, j = 1, 2, \dots, u - u_0\} \quad (3.10.2)$$

为了加速迭代过程,该算法中  $\sigma_i$  与当前迭代次数  $K$  有关,且有

$$\sigma_i(K) = \left( \frac{2}{1 + e^{-\frac{\alpha K}{T}}} - 1 \right) \sigma_i^0, \quad i = 1, 2 \quad (3.10.3)$$

式中,  $\alpha$  表示正系数,用以调节  $\sigma_i$  的变化速度;  $\sigma_i^0$  表示  $\sigma_i(K)$  上限;  $T$  表示迭代次数上限; 公式(3.10.3)使  $\sigma_i(K)$  由 0 逐步趋近于  $\sigma_i$ 。开始时罚系数较小,有助于大范围搜索,随后会逐步变大。针对公式(3.10.1)的 GACA 基本步骤如下。

(1) 初始化。

当前迭代次数  $K=0$ ,在  $S$  中取  $N$  组服从均匀分布的随机数,给定  $N$  只蚂蚁的初始位置,形成初始蚁群  $C_0$ ,且  $C_0 = (X_1, X_2, \dots, X_N)^T$ ,其中  $X_i \in S$ 。

初始化信息量矩阵  $\tau_{N \times N}, \tau_{ij} = \tau_0, \tau_0$  为一小正数。

设  $b(i)$  为位置  $i$  的蚂蚁数目,初始时  $b(i)=1, i=1, 2, \dots, N$ 。定义初始化蚂蚁的可见域  $D(K)$ ,当

$$\|X_i - X_j\| \leq \tau_{ij} D(K) \quad (3.10.4)$$

时,蚂蚁  $i, j$  可互相移动至对方位置。 $\|\cdot\|$  为某种范数。为减小计算量,这里采用了  $R^n$  上的 2 范数,记为  $\|\cdot\|_2$ 。 $D(K)$  与迭代次数  $K$  有关,此处采用

$$D(K) = 2 \left( 1 - \frac{1}{1 + e^{-\frac{\alpha K}{T}}} \right) D_0 \quad (3.10.5)$$

式中,  $D_0$  表示可见域上限,其余同公式(3.10.3)。公式(3.10.5)使每只蚂蚁开始

能大范围搜索,后来逐步精细化。

(2) 搜索策略。

对于蚂蚁位置  $i(i=1, 2, \dots, N)$ , 当  $b(i) \geq 1$  时, 形成集合  $A_i$ , 且

$$A_i = \{X_j \mid \|X_i - X_j\| \leq \tau_{ij} D(K)\} \quad (3.10.6)$$

当  $A_i \neq \emptyset$  时, 跳转到第(3)步; 当  $A_i = \emptyset$  时, 则跳转到第(4)步, 其中  $\emptyset$  表示空集。

(3) 计算。

设  $A_i$  中元素个数为  $m$ , 计算

$$\eta_{ij} = F(X_i) - F(X_j), \quad X_j \in A_i \quad (3.10.7)$$

$$l = \frac{1}{m} \left( \frac{2}{1 + e^{-\frac{\eta_{ij}}{T}}} - 1 \right) \sum_{X_j \in A_i} \eta_{ij} \quad (3.10.8)$$

令  $\eta_{ij}^{\min} = |\min(\eta_{ij})|$ ,  $G = (1+\epsilon)\eta_{ij}^{\min}$ ,  $\epsilon$  为一小正数。定义状态转移概率

$$P_0 = \frac{(l+G)^{\gamma_1} \left( \frac{1}{m} \sum_{X_j \in A_i} \tau_{ij} \right)^{\gamma_2}}{\sum_{X_j \in A_i} (\eta_{ij} + G)^{\gamma_1} (\tau_{ij})^{\gamma_2} + (l+G)^{\gamma_1} \left( \frac{1}{m} \sum_{X_j \in A_i} \tau_{ij} \right)^{\gamma_2}} \quad (3.10.9)$$

$$P_{ij} = \frac{(\eta_{ij} + G)^{\gamma_1} (\tau_{ij})^{\gamma_2}}{\sum_{X_j \in A_i} (\eta_{ij} + G)^{\gamma_1} (\tau_{ij})^{\gamma_2} + (l+G)^{\gamma_1} \left( \frac{1}{m} \sum_{X_j \in A_i} \tau_{ij} \right)^{\gamma_2}} \quad (3.10.10)$$

式中,  $\gamma_1$  和  $\gamma_2$  均为常数, 这里取  $\gamma_1 = \gamma_2 = 1$ 。

由公式(3.10.9)和公式(3.10.10)可得

$$P_0 + \sum_{X_j \in A_i} P_{ij} = 1 \quad (3.10.11)$$

随着  $F(X_j)$  的减小,  $\tau_{ij}$  增大,  $P_{ij}$  增大。 $P_0$  为进行邻域搜索的概率, 与可见域中蚂蚁对应目标函数的平均值及当前迭代次数有关。

对  $P_0$ 、 $P_{ij}$  分别进行赌轮选择(roulette wheel selection), 以选择不同的修正规则: 若选中某一  $P_{ij}$ , 则执行如下修正规则 1。

**修正规则 1** 在  $X_i$  处蚂蚁转向  $X_j$  处,  $\Delta\tau_{ij} = P_{ij}$ ,  $b_i = b_i - 1$ ,  $b_j = b_j + 1$ ,  $X_i \leftarrow X_j$ 。跳转到第(5)步。

若选中  $P_0$ , 则执行如下修正规则 2。

**修正规则 2** 在  $X_i$  邻域中进行搜索, 邻域定义为

$$S_{X_i} = \{Y \mid [\|X_i - Y\| \leq \beta D(K)] \cap S\} \quad (3.10.12)$$

设搜索结果为  $Y$ , 则  $X_i \leftarrow Y$ , 且

$$\Delta\tau_{ij} = \frac{[F(X_i) - F(Y) + G]^{\gamma_1} \left( \frac{1}{m} \sum_{X_j \in A_i} \tau_{ij} \right)^{\gamma_2}}{\sum_{X_j \in A_i} (\eta_{ij} + G)^{\gamma_1} (\tau_{ij})^{\gamma_2}} \quad (3.10.13)$$

跳转到第(5)步。

(4) 直接邻域搜索。

搜索域同公式(3.10.12)。设搜索结果为  $Y$ , 则执行如下修正规则 3。

**修正规则 3**  $X_i \leftarrow Y, \Delta\tau_{ij} = r, X_j \in A_i$ 。其中,  $r$  为常数。

(5) 修正信息量矩阵。

$$\tau_{ij}(K+1) = (1 - \rho)\tau_{ij}(K) + \Delta\tau_{ij} \quad (3.10.14)$$

这里需要注意以下两点:

① 当  $\Delta\tau_{ij}$  由修正规则 1 得到时, 仅修正  $i$  至  $j$  的连接; 当  $\Delta\tau_{ij}$  由修正规则 2 得到时, 对于所有  $X_j \in A_i$ , 且  $X_i \neq X_j$  的  $\tau_{ij}$  都修正; 当  $\Delta\tau_{ij}$  由修正规则 3 得到时, 则所有  $X_i \in C^K$ , 且  $X_i \neq X_j$  的  $\tau_{ij}$  都修正。其中  $C^K$  为第  $K$  次迭代形成的蚁群。

② 仅修正  $\tau_{ij}, \tau_{ji}$  不修正。

(6) 对所有  $X_i \in C^K$ , 都完成一次移动, 统计结果为:

① 若不满足接受条件, 取消本次迭代第(2)~(4)步的结果, 跳转到第(2)步。

② 若连续经过多次迭代后, 若蚁群统计结果不变, 则对蚁群加扰动。此处采取扰动可扩大邻域搜索范围。若多次扰动无效, 则输出结果。

③ 当  $K < T, K = K+1$  时, 跳转到步骤(2); 否则, 输出结果。

### 3.10.2 GACA 的收敛性分析

这里将基于压缩映象的不动点理论对 GACA 的收敛性作简单分析, 并给出 GACA 收敛的充分条件。

**定义 3.10.1** GACA 的空间  $\Omega$  为

$$\Omega = \{C^0, C^1, \dots, C^K, \dots\}$$

式中,  $C^K = (X_1^K, X_2^K, \dots, X_N^K)$  表示第  $K$  次迭代形成的蚁群, 其中  $N$  为蚂蚁数目,  $S$  为可行域,  $X_i^K \in S, i=1, 2, \dots, N$ 。

由前述可知: 对于  $\forall X \in S$  都可找到  $C^K$ , 使  $X \in C^K$ , 且  $C^K \in \Omega$ 。

**定义 3.10.2**  $\Omega$  中  $C^i, C^j$  距离  $d(C^i, C^j)$  为

$$d(C^i, C^j) = \begin{cases} |M + F'(C^i)| + |M + F'(C^j)|, & \text{若 } C^i \neq C^j \\ 0, & \text{否则} \end{cases} \quad (3.10.15)$$

式中,  $F'(C) = \frac{1}{N} \sum_{t=1}^N F(X_t')$ ,  $L$  为  $F'(C)$  的一个下界, 由于待求为最小化问题, 则  $L$  必存在;  $C^i \in \Omega$ ;  $\epsilon$  为小正数。

**定理 3.10.1**  $(\Omega, d)$  构成完备的度量空间。

**证明** 首先,  $d(C^i, C^j)$  作为  $\Omega$  上的一个度量具有如下性质:

(1)  $d(C^i, C^j) = d(C^j, C^i)$ ;

(2)  $d(C^i, C^k) + d(C^k, C^j) = |M + F'(C^i)| + |M + F'(C^k)| + |M + F'(C^j)|$

$$\geq |M+F'(C^i)| + |M+F'(C^j)| = d(C^i, C^j);$$

(3)  $d(C^i, C^j) \geq 0$ ; 当且仅当  $C_i = C_j$  时,  $d(C^i, C^j) = 0$ 。

其次,  $\Omega$  是完备的: 设  $\{C^i\}$  是  $\Omega$  中任一 Cauchy 列, 则对于  $\forall \epsilon > 0$ , 存在  $N$ , 当  $i, j > N$  时, 有

$$d(C^i, C^j) < \epsilon \quad (3.10.16)$$

式中, 令  $j \rightarrow \infty$ ,  $C^i \rightarrow C$  ( $i \rightarrow \infty$ )。由定义可得  $C \in \Omega$ 。因此,  $(\Omega, d)$  是完备的度量空间。

[证毕]

由 GACA 的描述可将其看成  $\Omega \rightarrow \Omega$  的映射(记为  $\varphi$ ), 则在完备的度量空间  $(\Omega, d)$  上,  $\varphi$  为  $\Omega$  的自映象, 当  $\varphi$  满足以下压缩映象条件之一时,  $\varphi$  在  $\Omega$  中存在唯一不动点<sup>[23]</sup>。对于  $\forall C^i, C^j \in \Omega$ , 有

$$d(\varphi C^i, \varphi C^j) \leq h d(C^i, C^j), h \in (0, 1) \quad (3.10.17)$$

$$d(\varphi C^i, \varphi C^j) \leq h \{d(C^i, \varphi C^i) + d(C^j, \varphi C^j)\}, h \in (0, 1/2) \quad (3.10.18)$$

$$d(\varphi C^i, \varphi C^j) < \max\{d(C^i, \varphi C^i), d(C^j, \varphi C^j), d(C^i, C^j)\}, C^i \neq C^j \quad (3.10.19)$$

$$d(\varphi C^i, \varphi C^j) \leq h \max\{d(C^i, C^j), d(C^i, \varphi C^i), d(C^j, \varphi C^j), d(C^i, C^j), d(C^j, C^i)\}, h \in (0, 1) \quad (3.10.20)$$

满足公式(3.10.17)~(3.10.20)的充分条件是

$$F'(\varphi C^K) < F'(C^K) \quad (3.10.21)$$

因此步骤(6)的统计结果①中的一个接受条件为

$$F'(C^{K+1}) < F'(C^K) \quad (3.10.22)$$

由此可得如下推论 3.10.1。

**推论 3.10.1** 在完备度量空间  $(\Omega, d)$  中, 若 GACA 构成的映射  $\varphi: \Omega \rightarrow \Omega$  满足式(3.10.21), 则  $\varphi$  存在唯一的不动点  $C^*$ , 对任意  $C^0 \in \Omega$ , GACA 形成的迭代序列为

$$\varphi^K C^0 \rightarrow C^* \quad (3.10.23)$$

### 3.11 本章小结

对蚁群算法收敛性的研究是最近几年内才刚刚开始的。目前已经取得了相当丰富的研究成果, 但是还有许多问题需要进一步解决, 比如初始化参数选择问题、信息素分配问题、收敛速度问题等, 这些均带有经验和直觉性, 至今没有经过严格的数学论证; 同时, 连续域蚁群算法的收敛性证明仍然存在许多研究空白。今后蚁群算法的收敛性研究仍然是一个非常重要的研究内容。

本章内容是全书的理论分析部分, 对更加深入地理解蚁群算法、改进蚁群算法和应用蚁群算法解决实际问题具有指导意义。

## 参 考 文 献

- 1 Gutjahr W J. A generalized convergence result for the graph based ant system. Technical Report 99-09, Dept. of Statistics and Decision Support Systems, University of Vienna, Austria, 1999
- 2 Gutjahr W J. A graph-based ant system and its convergence. Future Generation Computer Systems, 2000, 16(8): 873~888
- 3 Stützle T, Dorigo M. A short convergence proof for a class of ant colony optimization algorithms. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 358~365
- 4 Gutjahr W J. ACO algorithms with guaranteed convergence to the optimal solution. Information Processing Letters, 2002, 82(3): 145~153
- 5 段海滨, 王道波. 蚁群算法的全局收敛性研究及改进. 系统工程与电子技术, 2004, 26(10): 1506~1509
- 6 段海滨, 王道波, 于秀芬. 基本蚁群算法的 A. S. 收敛性研究. 应用基础与工程科学学报, 待发表
- 7 Duan H B, Wang D B, Yu X F. A novel approach to the convergence of ant colony algorithm and its Matlab GUI-based realization. International Journal of Plant Engineering and Management, to appear
- 8 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 9 Yoo J H, La R J, Makowski A M. Convergence results for ant routing. Technical Report CSHCN 2003-46, Institute for Systems Research, University of Maryland, College Park(MD), 2003
- 10 Yoo J H, La R J, Makowski A M. Convergence of ant routing algorithms--results for simple parallel network and perspectives. Technical Report CSHCN 2003-44, Institute for Systems Research, University of Maryland, College Park(MD), 2003
- 11 Badr A, Fahmy A. A proof of convergence for ant algorithms. International Journal of Intelligent Computing and Information, 2003, 3(1): 22~32
- 12 孙焘, 王秀坤, 刘业欣等. 一种简单蚂蚁算法及其收敛性分析. 小型微型计算机系统, 2003, 21(8): 1524~1526
- 13 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法融合的马尔可夫收敛性分析. 自动化学报, 2004, 30(4): 659~664
- 14 Hou Y H, Wu Y W, Lu L J, et al. Generalized ant colony optimization for economic dispatch of power systems. Proceedings of the 2002 International Conference on Power System Technology, 2002, 1: 225~229
- 15 侯云鹤, 熊信良, 吴耀武等. 基于广义蚁群算法的电力系统经济负荷分配. 中国电机工程学报, 2003, 23(3): 59~64
- 16 Bullnheimer B, Hartl R F, Strauss C. A new rank-based version of the ant system: a computational study. Central European Journal for Operations Research and Economics, 1999, 7(1): 25~38
- 17 Dorigo M. Optimization, learning and natural algorithms. Ph. D. Thesis, Department of Electronics, Politecnico di Milano, Italy, 1992
- 18 Dorigo M, Di Caro G, Gambardella L M. Ant algorithms for distributed discrete optimization. Artificial Life, 1999, 5(2): 137~172
- 19 Dorigo M, Maniezzo V, Colorni A. The ant system: an autocatalytic optimization process. Technical Report 91-016, Department of Electronics, Politecnico di Milano, Italy, 1991
- 20 Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE

- Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1):29~41
- 21 Feo T A, Resende M G C. Greedy randomized adaptive search procedures. Journal of Global Optimization, 1995, 6: 109~133
  - 22 Gambardella L M, Dorigo M. Ant-Q: a reinforcement learning approach to the traveling salesman problem, Proceedings of the 12th International Conference on Machine Learning, 1995, 252~260
  - 23 Gambardella L M, Dorigo M. Solving symmetric and asymmetric TSPs by ant colonies. Proceedings of the IEEE Conference on Evolutionary Computing, 1996, 622~627
  - 24 Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. Technical Report IRIDIA/96-05, IRIDIA, Université Libre de Bruxelles, Belgium, 1996
  - 25 Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66
  - 26 Stützle T, Dorigo M. A short convergence proof for a class of ACO algorithms. Technical Report IRIDIA/2000-35, Universite Libre de Bruxelles, Belgium, 2000
  - 27 Stützle T. MAX-MIN ant system for the quadratic assignment problem. Technical Report AIDA-97-4, FG Intellektik, TU Darmstadt, Germany, 1997
  - 28 Stützle T, Hoos H H. The MAX-MIN ant system and local search for the travelling salesman problem. Proceedings of the 1997 International Conference on Evolutionary Computation, 1997, 309~314
  - 29 Stützle T, Hoos H H. MAX-MIN ant system. Future Generation Computer Systems, 2000, 16(8): 889~914
  - 30 李裕奇. 随机过程. 北京: 国防工业出版社, 2003
  - 31 史及民. 离散鞅及其应用. 北京: 科学出版社, 1999
  - 32 张文修, 梁怡. 遗传算法的数学基础. 西安: 西安交通大学出版社, 2003
  - 33 Subramanian D, Druschel P, Chen J. Ants and reinforcement learning: a case study in routing in dynamic networks. Technical Report TR 96-259, Department of Computer Science, Rice University, Houston, 1998
  - 34 Chung k L. A course in probability theory. Second Edition, New York: Academic Press, 1974
  - 35 Alexander K S, Watkins J C, Ney P. Progress in probability, spatial stochastic processes. Boston: Birkhauser Publishing House, 1991
  - 36 Revesz P. Path properties of an infinitely many particles. World Scientific Press, 1993
  - 37 Revesz P. Path properties of an infinite system of Wiener processes. Journal of Theoretical Probability, 1993, 6: 353~383
  - 38 Taylor H, Karlin S. An introduction to stochastic modelling. New York: Academic Press, 1998
  - 39 Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Networks, 1994, 5(1): 96~101
  - 40 吴庆洪, 张纪会, 徐心和. 具有变异特征的蚁群算法. 计算机研究与发展, 1999, 36(10): 1240~1245
  - 41 Vittorio M. An ANTS heuristic for the frequency assignment problem. Future Generation Computer Systems, 2000, 16(8): 927~935
  - 42 陈传璋, 金福临. 数学分析. 北京: 高等教育出版社, 1983
  - 43 张石生. 不动点理论及应用. 重庆: 重庆出版社, 1984

## 第4章 蚁群算法的实验分析及参数选择原则

### 4.1 引言

在“探索”和“利用”之间建立一个平衡点是蚁群算法研究的关键问题之一,也就是说既要使得蚁群算法的搜索空间尽可能大,以寻找那些可能存在最优解的解区间;同时,又要充分利用蚂蚁群体内当前所具有的有效信息,使得蚁群算法搜索的侧重点放在那些可能具有较高适应值的个体所在的区间内,从而以较大的概率收敛到全局最优解<sup>[1, 2]</sup>。蚁群算法的信息交互主要通过信息素来完成,其收敛到最优解的过程即为信息正反馈的动态过程。正反馈机制旨在强化性能较好的解,却使蚁群算法在求解问题时容易出现停滞现象。

在蚁群算法中,信息素和启发函数、信息量-启发函数乘积 $[\tau_{kl}(m)]^\alpha \cdot [\eta_{kl}(u)]^\beta$ 、蚂蚁之间的合作行为会严重影响到算法的收敛性,同时,蚁群算法的参数也是影响其求解性能和效率的关键因素,信息素残留因子 $1 - \rho$ 、信息启发式因子 $\alpha$ 、期望启发式因子 $\beta$ 、信息素强度 $Q$ 、蚂蚁数目 $m$ 等都是非常重要的参数,其选取方法和选取原则直接影响到蚁群算法的全局收敛性和求解效率。但由于蚁群算法参数空间的庞大性和各参数之间的关联性,如何确定最优组合参数使蚁群算法求解性能最佳一直是一个极其复杂的优化问题,目前尚没有完善的理论依据,大多情况下都是根据经验而定。一般而言,蚁群算法的参数可通过反复试凑得到,这显然会对蚁群算法的计算效率和收敛性产生不利影响。

基于此,Dorigo M 等<sup>[3]</sup>最早对 $\alpha, \beta, \rho, m$ 等参数的选择进行了初步研究;Botee H M 等<sup>[4]</sup>用遗传算法求得 $\alpha, \beta, \rho, m$ 等参数的最优组合,但是这种用遗传算法求解组合参数的方法比较麻烦;随后,段海滨等<sup>[5, 6]</sup>、郝晋等<sup>[7]</sup>、詹士昌等<sup>[8]</sup>、叶志伟等<sup>[9, 10]</sup>在蚁群算法的参数分析和优化组合方面进行了大量卓有成效的研究工作。本章将通过大量的数字仿真实验对蚁群算法的参数选择原则进行了深入研究,并总结出一种“三步走”选择蚁群算法最优组合参数的有效方法。

### 4.2 蚁群行为和参数对算法性能影响的实验分析

蚁群算法中的参数设定尚无严格的理论依据,至今还没有确定最优参数的一般方法。对于蚁群算法中的 $\alpha, \beta, \rho, m, Q$ 等主要参数,解析法难以确定其最佳组合,本节在大量数字仿真的基础上,研究了蚁群算法中主要参数的优化设置问题。目前已经公布的蚁群算法参数设置成果都是针对利用不同蚁群算法模型所解决的

特定问题而言的。以应用最多的 Ant-Cycle 模型为例,其最好的经验结果为:  
 $0 \leq \alpha \leq 5$ ;  $0 \leq \beta \leq 5$ ;  $0.1 \leq \rho \leq 0.99$ ;  $10 \leq Q \leq 10000$ 。

#### 4.2.1 信息素和启发函数对蚁群算法性能的影响

信息素是表征过去信息的载体,而启发函数则是表征未来信息的载体,它们直接影响到蚁群算法的全局收敛性和求解效率<sup>[11, 12]</sup>。大量的实验结果已经表明启发函数  $\eta_{ij}$  对保证蚁群算法在合理的时间内搜索到全局最优解非常重要。

实验采用了 TSPLIB 中的 Oliver30TSP 作为仿真对象,蚂蚁数目  $m$  为变量,定义迭代次数为  $10000/m$ 。基本蚁群算法、无启发信息蚁群算法及无信息素蚁群算法的性能比较如图 4.1 所示。

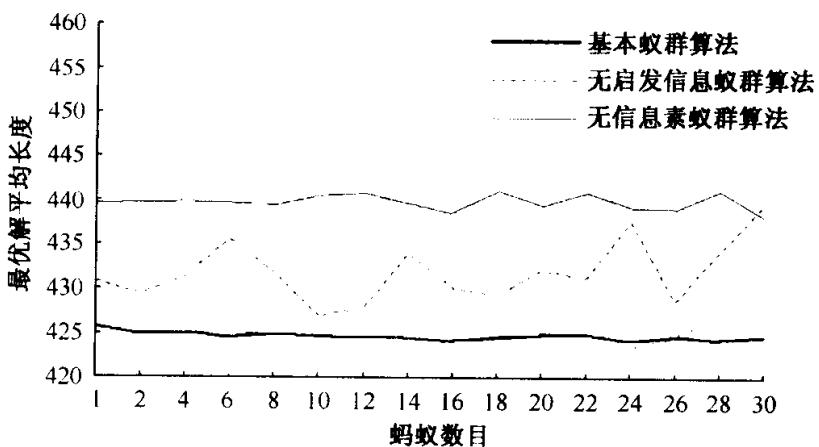


图 4.1 基本蚁群算法、无启发信息蚁群算法及无信息素蚁群算法的性能比较

由图 4.1 可知,不仅没有启发信息的蚁群算法性能会严重恶化,而且无信息素的蚁群算法性能也非常差。由此,当  $\beta=0$  时,蚁群算法的全局收敛性能会严重恶化。

#### 4.2.2 信息量-启发函数乘积对蚁群算法性能的影响

为了深入理解蚁群算法采用何种机制来指导其搜索最优解,就需要深入研究信息量-启发函数乘积  $[\tau_{kl}(m)]^\alpha [\eta_{kl}(u)]^\beta$  随迭代次数的变化关系<sup>[13, 14]</sup>。图 4.2 给出了蚂蚁在构建最优解时信息量-启发函数乘积  $[\tau_{kl}(m)]^\alpha [\eta_{kl}(u)]^\beta$  随迭代次数的变化关系。

图 4.2 将路径弧段分为 3 类:

- (1) 最好弧段(best edge, BE): 属于最好路径的路径弧段。
- (2) 测试弧段(testable edge, TE): 不属于最好路径,但会在前两次迭代中至

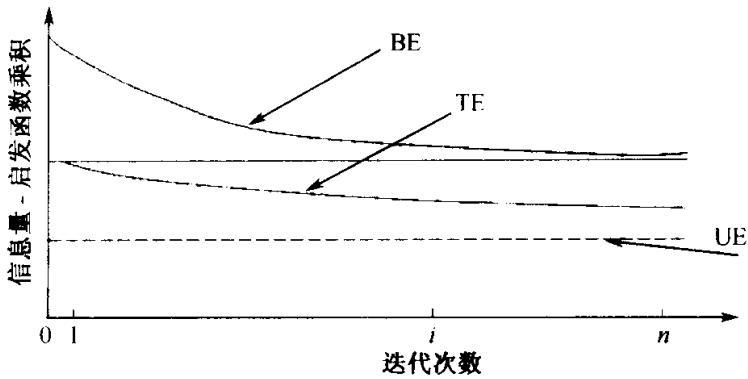


图 4.2 路径弧段分类及  $[\tau_{kl}(m)]^\alpha [\eta_{kl}(u)]^\beta$  与迭代次数的对应关系

少被蚂蚁穿越一次的路径弧段。

(3) 不感兴趣弧段(uninteresting edge, UE):既不属于最好路径,又没有在前两次迭代中被蚂蚁穿越的路径弧段。

由图 4.2 可见,蚁群算法倾向于利用 BE 类路径弧段上的信息和挖掘 TE 类路径弧段上的信息。

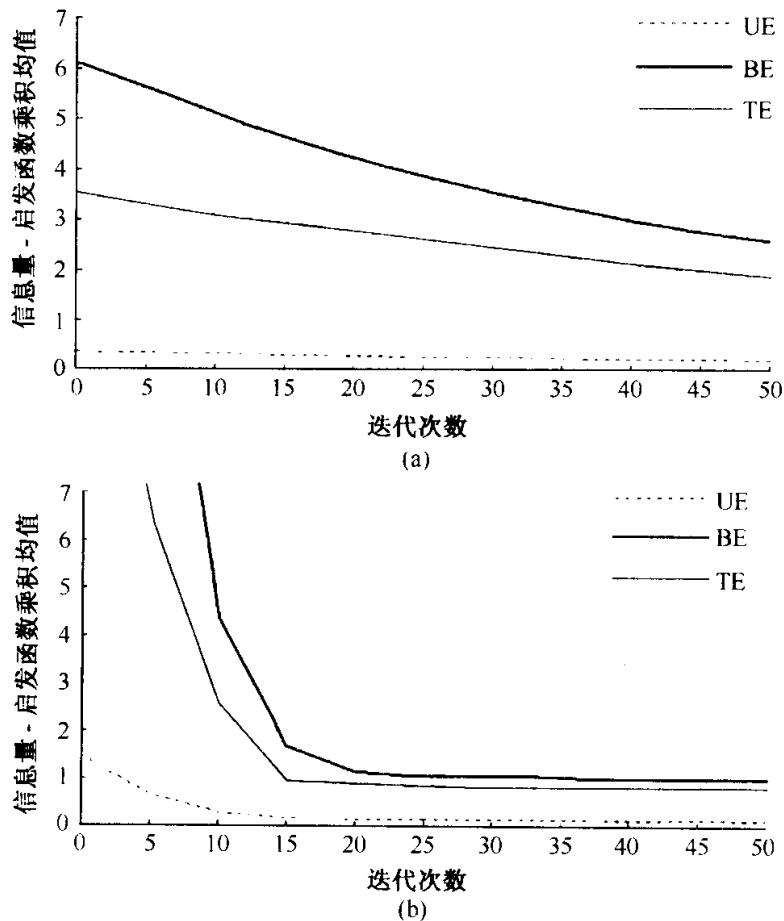


图 4.3 不同种类路径弧段对蚁群算法求解性能的影响

蚂蚁在穿越路径弧段时,信息素更新规则会使路径弧段上的信息量逐渐消减,从而使其吸引力越来越小,这时蚂蚁择路时会倾向于未曾穿越的路径弧段。局部信息的更新会逐渐减少已穿越路径弧段上的信息量,从而使其他后继蚂蚁选中该路径弧段的概率变小,因此导致蚂蚁很难最终收敛到全局最优路径。

实验研究还发现,当蚁群算法达到较好的求解性能时,只要没有再找到新的最短路径,则属于 BE 路径弧段的信息量-启发函数乘积会随着迭代次数的增加而逐渐下降至 TE,而属于 TE 路径弧段的信息量-启发函数乘积会随着迭代次数的增加而逐渐下降至 UE。

在较好和较差情况下信息量-启发函数乘积对蚁群算法求解性能的影响分别如图 4.3(a)和图 4.3(b)所示,此处所采用的仿真算例是 TSPLIP 中的 Eil51TSP。

在图 4.3(a)中,设  $\alpha=\beta=0.1$ ,则最终所寻到的最优解为 426,该情况下蚁群算法的求解性能较好;而在图 4.3(b)中,设  $\alpha=\beta=0.5$ ,则最终所寻到的最优解为 465,该情况下蚁群算法的求解性能较差。

#### 4.2.3 蚂蚁之间的合作行为对蚁群算法性能的影响

蚁群算法作为一种多智能体的仿生优化算法,其各智能体之间的合作会对蚁群算法的收敛性能和求解效率产生极大的影响<sup>[15~17]</sup>。为了研究蚁群算法如何有效利用信息素这一媒介来寻找最优解,本小节分别对一群通过信息素进行合作的蚂蚁和另一群不通过信息素进行合作的蚂蚁进行了仿真实验。采用算法执行所需的 CPU 时间来评价其性能指标,以此来描述由于信息素更新而导致的蚁群合作行为的高复杂性。

在第一个实验中,把首次完成时间定义为发现第一个最优解时所经历的时间,并利用这一概念来比较合作机制与非合作机制对蚁群算法求解性能的影响。算法的迭代次数是 10000,这里可采用有向图的形式来表示搜索到最优解时 CPU 时间的概率分布(概率密度),例如,假设经过 100 次尝试,发现在第 220 次迭代时算法找到了最优解,则对于这 220 次迭代而言,有  $P(220)=100/10000$ 。该实验所采用的仿真对象为 CCAO<sup>[18]</sup>,设蚂蚁数目  $m=4$ 。合作与非合作蚁群算法的概率密度随 CPU 时间的变化关系如图 4.4 所示。

由图 4.4 可见,蚂蚁之间的合作行为可以有效地增大蚁群迅速发现最优解的概率。

在第二个实验中,对于采用合作机制和采用非合作机制的蚂蚁而言,所发现的最优解为时间(ms)的函数。该实验仍采用 CCAO 作为仿真对象,设蚂蚁数目  $m=4$ 。合作与非合作蚁群算法的路径长度随 CPU 时间的变化关系如图 4.5 所示。

由图 4.5 可见,对于采用合作机制的蚁群算法而言,300ms 后蚁群算法总能找到最优解;而对于没有采用合作机制的蚁群算法不能在 300ms 内找到最优解。在

最初的 150ms(图 4.5 中两条曲线的交叉点)之前,非合作蚂蚁的求解性能略优于合作蚂蚁的求解性能。

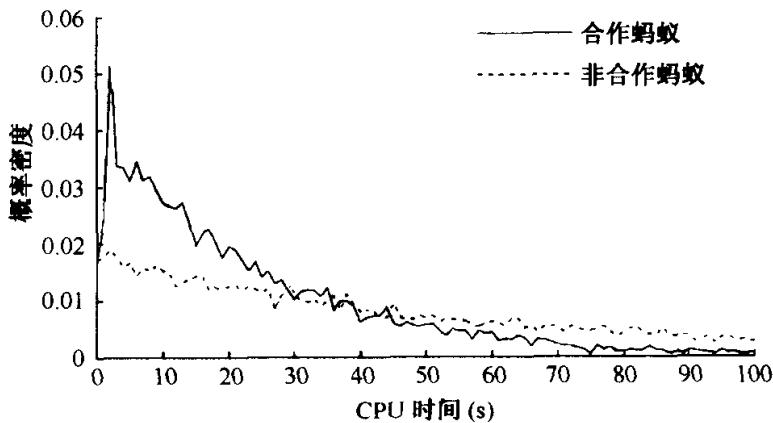


图 4.4 合作与非合作蚁群算法的概率密度随 CPU 时间的变化关系

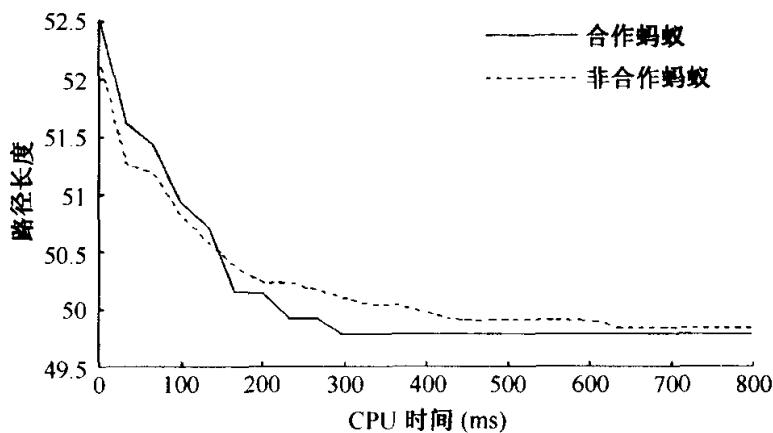


图 4.5 合作与非合作蚁群算法的路径长度随 CPU 时间的变化关系

#### 4.2.4 信息素残留因子对蚁群算法性能的影响

蚁群算法中的人工蚂蚁具有人类记忆功能,随着时间的推移,以前留下的信息会逐渐消失。如前所述,在蚁群算法模型中用参数  $\rho$  表示信息素挥发因子,则  $1-\rho$  就是信息素残留因子。信息素挥发因子  $\rho$  的大小直接关系到蚁群算法的全局搜索能力及其收敛速度;而信息素残留因子  $1-\rho$  反映了蚂蚁之间个体相互影响的强弱。这是由于  $\rho$  的存在,当要处理的问题规模比较大时,会使那些从来未被搜索到的路径上的信息量减小到接近于 0,因而降低了算法的全局搜索能力,而且当  $\rho$  过大时,以前搜索过的路径被再次选择的可能性过大,也会影响到算法的随机性能和全局搜索能力;反之,通过减小  $\rho$  虽然可以提高算法的随机性能和全局搜索能力,但又会使算法的收敛速度降低。

基于上述分析,本小节在对 TSPLIB 中 Eil51TSP 大量仿真实验的基础上对蚁群算法中信息素残留因子  $1-\rho$  的选择问题作了进一步探讨,Eil51TSP 的实际最优解为 426.0000。本仿真实验采用了蚁群算法中的全局搜索机制,即采用了 Ant-Cycle 模型,这样可以充分利用蚂蚁群体的整体信息。

实验时,设置  $m=10$ ,  $Q=100$ ,  $\alpha=1$ ,  $\beta=5$ ,停止条件为相邻两次循环搜索中最优解的差别小于 0.001。该实验所得的  $1-\rho$  与迭代次数  $N_c$  及  $1-\rho$  与最优路径长度  $L$  之间的对应关系分别如图 4.6 和图 4.7 所示。

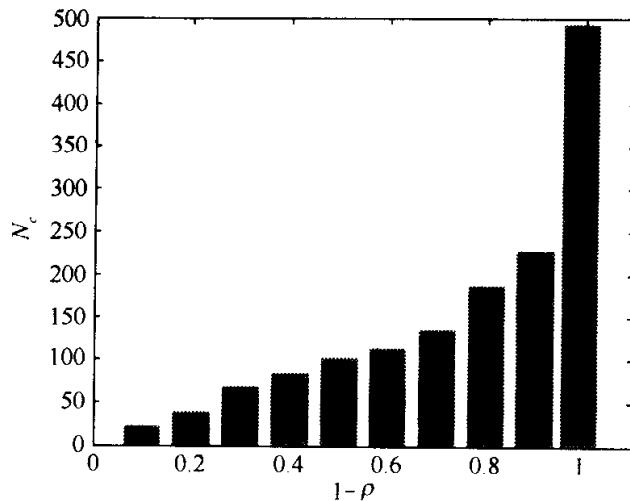


图 4.6  $1-\rho$  与迭代次数  $N_c$  的关系图

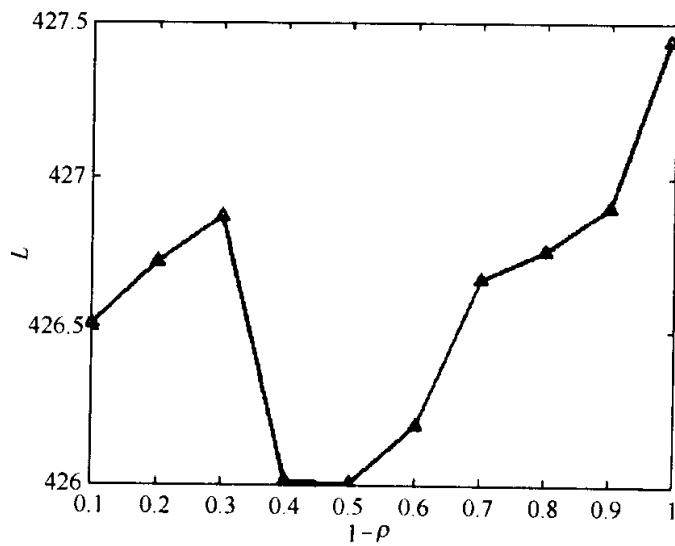


图 4.7  $1-\rho$  与最优路径长度  $L$  的关系

由图 4.6 和图 4.7 可知,在其他初始化参数相同的情况下,信息素残留因子  $1-\rho$  的大小对蚁群算法的收敛性能影响非常大。 $1-\rho$  与迭代次数  $N_c$  近似成正比

关系,在 0.1~0.99 范围内, $1-\rho$  越大( $\rho$  越小),迭代次数  $N_c$  越大;当  $1-\rho \approx 0.9$  时,迭代次数  $N_c$  会急剧增大。这是由于,当  $1-\rho$  很大时,由于路径上的残留信息占主导地位,信息正反馈的作用相对较弱,搜索的随机性增强,而且存在于环境中的信息素保持时间较长,解空间对问题空间的反馈信息较弱,因而蚁群算法收敛速度很慢。在算法收敛的情况下, $1-\rho$  对最优解终值的影响不是很大,但是在  $1-\rho$  较小时,虽然收敛速度加快,但是计算结果易于陷入局部最优状态,这是由于当  $1-\rho$  比较小的时候,信息的正反馈作用占主导地位,搜索的随机性减弱。

由图 4.6 和图 4.7 可知,当  $1-\rho \approx 0.5$ (即  $\rho \approx 0.5$ )时,蚁群算法的全局收敛性和收敛速度都比较好,计算性能也比较稳定。

通过对 EIL51TSP 的多次仿真还发现,当蚂蚁数目很少时,比如  $m=3$ ,信息素残留因子  $1-\rho$  必须选得较大(即  $\rho$  较小)才能保证算法的全局收敛性。其原因很明显:若蚂蚁数目过少,同一条路径上两次有蚂蚁经过的时间间隔增大,若  $1-\rho$  太小,则该路径上的信息素会很快挥发掉,从而使蚂蚁丧失了选择这条路径的机会。

#### 4.2.5 蚂蚁数目对蚁群算法性能的影响

蚁群算法是一种并行随机搜索算法,与其他仿生优化算法一样,它是通过多个候选解组成的群体进化过程来搜索最优解,该过程既需要蚂蚁个体的自适应能力,又需要蚂蚁群体内部的相互协作。蚁群在搜索过程中之所以表现出复杂而有序的行为,个体之间的信息交流与相互协作起着至关重要的作用。

对于 TSP,单只蚂蚁在一次循环中所经过的路径,表现为问题可行解集中的一个解; $m$  只蚂蚁在一次循环中所经过的路径,则表现为问题可行解集中的一个子集。由上一章的理论分析可知,子集大(即蚂蚁数目多),可以提高蚁群算法的全局搜索能力及算法的稳定性,但在实际应用中,当蚂蚁数目过多时,也会使大量的曾被搜索过的解(路径)上的信息量的变化趋于平均,信息正反馈作用减弱,虽然这时全局搜索的随机性得到了加强,但收敛速度减慢。反之,子集较小(即蚂蚁数目少),特别是当要处理的问题规模比较大时,会使那些从来未被搜索到的解(路径)上的信息量减小到接近于 0,全局搜索的随机性减弱,虽然这时收敛速度加快,但会使算法的稳定性变差,且容易出现过早停滞现象。

至于蚂蚁最佳数目的选择问题,Dorigo M 等<sup>[14]</sup>曾提出了如下计算思路:令  $\varphi_2\tau_0$  为全局更新后属于 BE 路径弧段的平均信息量,  $\varphi_1\tau_0$  为全局更新前属于 BE 路径弧段的平均信息量。局部更新规则为  $T_z = T_{z-1}(1-\rho) + \tau_0\rho$  的一阶线性循环关系,继而有

$$T_z = T_{z-1}(1-\rho)^z - \tau_0(1-\rho)^z + \tau_0 \quad (4.2.1)$$

由于在全局更新信息量  $T_0 = \varphi_2\tau_0$  之前(此即对应于图 4.8 中 BE 曲线的起始

点),且所有蚂蚁都建立了各自解还没有进行全局更新信息量  $T_z = \varphi_1 \tau_0$  之前(此即对应于图 4.8 中 BE 曲线的终止点),有

$$\varphi_1 = \varphi_2 (1 - \rho)^z - (1 - \rho)^z + 1 \quad (4.2.2)$$

考虑到属于 BE 的路径弧段被每只蚂蚁以大于  $q_0$  的概率选择,则对位于属于 BE 的路径弧段上蚂蚁数目的可信估计为

$$z = m \cdot q_0 \quad (4.2.3)$$

由此,蚂蚁数目的最优估计值为

$$m = \frac{\lg(\varphi_1 - 1) - \lg(\varphi_2 - 1)}{q_0 \cdot \lg(1 - \rho)} \quad (4.2.4)$$

上式给出了蚂蚁数目的最优估计值  $m$  与  $\varphi_1$  和  $\varphi_2$  的关系。通过大量的实验分析,当  $(\varphi_1 - 1)/(\varphi_2 - 1) \approx 0.4$  时,蚁群算法能获得较好的求解性能。属于 BE 类路径弧段上的平均信息量与迭代次数的对应关系如图 4.8 所示。

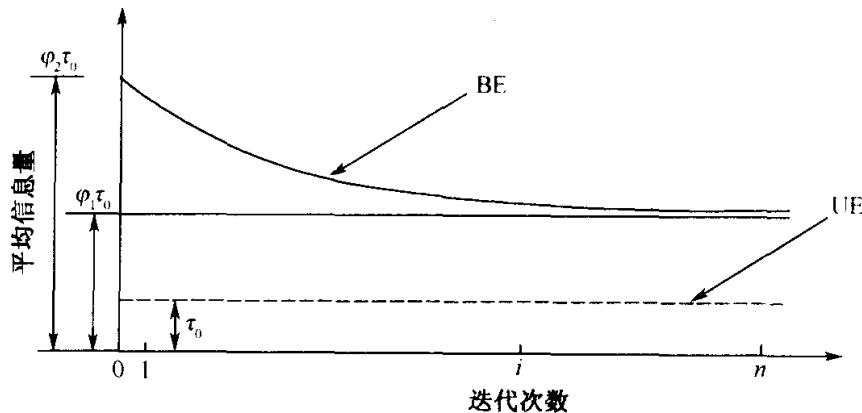


图 4.8 属于 BE 类路径弧段上的平均信息量与迭代次数的对应关系

由图 4.8 可见,在初始阶段,BE 类路径弧段上的平均信息量为  $\varphi_2 \tau_0$ ,其值随着蚂蚁遍历 BE 类路径弧段的持续进行而不断递减,经过一个循环周期后,属于 BE 的每个路径弧段平均被访问  $m q_0$  次,且其信息量的最终值为  $\varphi_1 \tau_0$ 。

上述蚂蚁数目的计算思路比较繁琐,且对于很多优化问题并不实用。这里通过对 TSPLIB 中 Eil51TSP 的大量仿真实验探讨了蚂蚁数目  $m$  的选择问题。

本仿真实验仍采用蚁群算法中的 Ant-Cycle 模型。实验时,设置  $Q = 50$ ,  $\alpha = 1$ ,  $\beta = 4$ ,  $\rho = 0.5$ ,停止条件为相邻两次循环搜索中最优解的差别小于 0.001。

实验所得的该 Eil51TSP 算例中,  $m$  与迭代次数  $N_c$  及  $m$  与最优路径长度  $L$  之间的对应关系分别如图 4.9 和图 4.10 所示。

由图 4.9 和图 4.10 可见,蚂蚁数目  $m$  对蚁群算法循环次数的影响大致呈线性规律变化,但是在  $m \approx 30$  时,搜索次数和最优解发生突变,当蚂蚁数目继续增大,虽然搜索的稳定性和全局性能得到进一步提高,但蚁群算法循环次数也随之增大,当蚂蚁数目远远大于问题规模时,继续增加蚂蚁数目对算法性能虽有改善,但

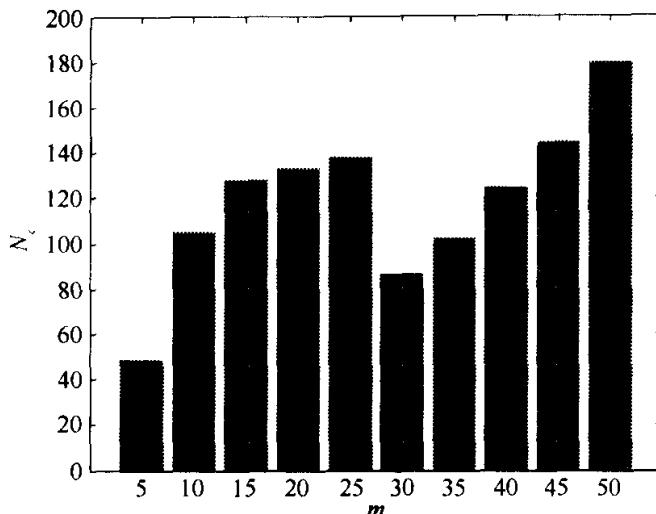


图 4.9  $m$  与迭代次数  $N_c$  的关系

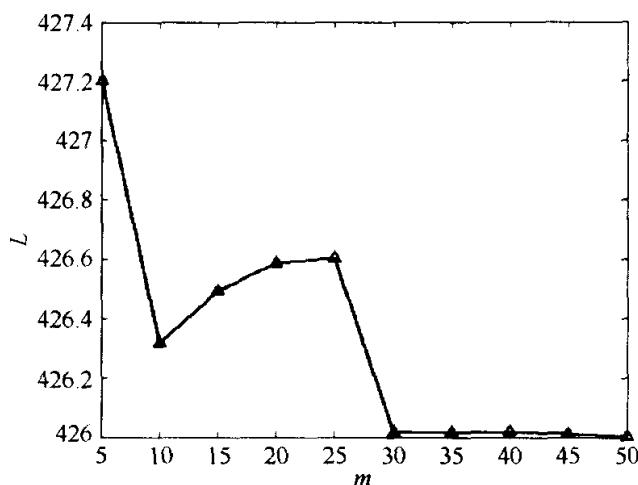


图 4.10  $m$  与最优路径长度  $L$  的关系

不是特别明显。这是由于蚁群算法的并行性, 蚂蚁数目多可以提高蚁群算法的全局搜索能力及算法的稳定性, 但当蚂蚁数目增大到一定程度后, 会使大量曾被搜索过路径上的信息量变化趋于平均, 信息正反馈作用不明显, 搜索的随机性虽然得到了加强, 但收敛速度减慢; 反之, 若蚁群数目太少, 搜索的随机性减弱, 虽然收敛速度加快, 但会使算法的全局收敛性能降低, 容易出现过早停滞现象。

由本算例的仿真实验结果可知, 当城市规模大致是蚂蚁数目的 1.5 倍时, 蚁群算法的全局收敛性和收敛速度都比较好。

#### 4.2.6 启发式因子对蚁群算法性能的影响

启发式因子  $\alpha$  反映蚂蚁在运动过程中所积累的信息量在指导蚁群搜索中的相对重要程度, 其值越大, 蚂蚁选择以前走过路径的可能性就越大, 搜索的随机性减

弱;而当启发式因子  $\alpha$  值过小时,则易使蚁群的搜索过早陷于局部最优。

关于蚁群算法中  $\alpha$  对算法性能的影响及其在实际应用中的选择,也可通过仿真实验来分析和确定。以 EIL51TSP 为研究对象,本实验仍采用了蚁群算法中的 Ant-Cycle 模型。实验时,设置  $m=30$ ,  $Q=150$ ,  $\beta=4$ ,  $\rho=0.5$ ,停止条件为相邻两次循环搜索中最优解的差别小于 0.001。

多次仿真实验所测得的该 Eil51TSP 算例中,  $\alpha$  与迭代次数  $N_c$  及  $\alpha$  与最优路径长度  $L$  之间的对应关系分别如图 4.11 和图 4.12 所示。

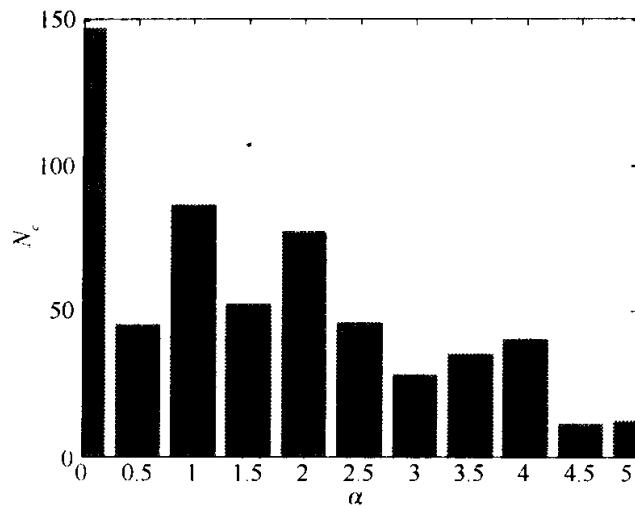


图 4.11  $\alpha$  与迭代次数  $N_c$  的关系

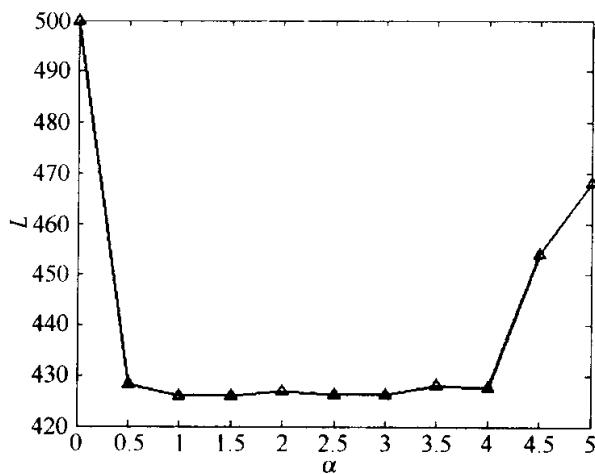


图 4.12  $\alpha$  与最优路径长度  $L$  的关系

由图 4.11 及图 4.12 所示的实验结果不难看出,蚁群算法中启发式因子  $\alpha$  对算法性能有极大的影响,  $\alpha$  过小,不仅收敛速度慢,而且算法易陷入局部最优解;  $\alpha$  过大,相当于给予信息素在蚂蚁搜索过程中的重要性以充分的重视,则将导致局部最优路径上的正反馈作用很强,算法也会出现过早收敛现象。本算例中,当  $\alpha \in [1.0, 4.0]$  时,蚁群算法的综合求解性能较好。

#### 4.2.7 期望启发式因子对蚁群算法性能的影响

期望启发式因子  $\beta$  反映了启发式信息在指导蚁群搜索过程中的相对重要程度, 其大小反映了蚁群寻优过程中先验性、确定性因素的作用强度。其值越大, 则蚂蚁在某个局部点上选择局部最短路径的可能性越大, 虽然这时算法的收敛速度得以加快, 但蚁群搜索最优路径的随机性减弱, 易于陷入局部最优。

关于蚁群算法中期望启发式因子  $\beta$  对算法性能的影响及其在实际应用中的选择, 也可通过大量的仿真实验来分析和确定。以 Eil51TSP 为研究对象, 本实验仍采用了蚁群算法中的 Ant-Cycle 模型。实验时, 设置  $m=30$ ,  $Q=150$ ,  $\alpha=1$ ,  $\rho=0.5$ , 停止条件为相邻两次循环搜索中最优解的差别小于 0.001。

多次实验所得的该 Eil51TSP 算例中  $\beta$  与迭代次数  $N_c$  及  $\beta$  与最优路径长度  $L$  之间的对应关系分别如图 4.13 和图 4.14 所示。

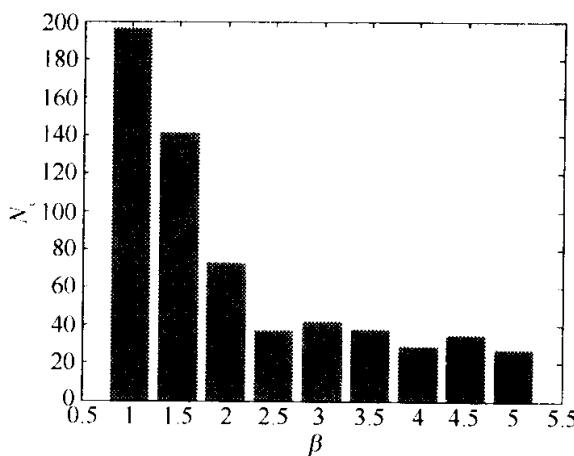


图 4.13  $\beta$  与迭代次数  $N_c$  的关系

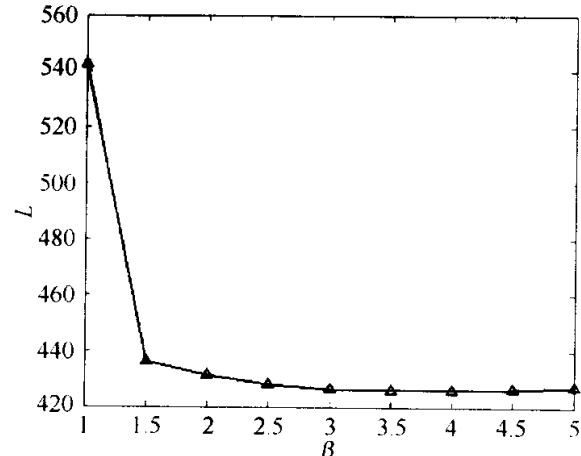


图 4.14  $\beta$  与最优路径长度  $L$  的关系

由图 4.13 和图 4.14 可见, 蚁群算法中期望启发式因子  $\beta$  对算法性能影响较大。 $\beta$  过小, 将导致蚂蚁群体陷入纯粹的随机搜索, 在这种情况下一般很难找到最优解;  $\beta$  过大时, 算法的收敛速度增快, 但其收敛性能有变差的趋势。在本算例中, 当  $\beta \in [3.0, 4.5]$  时, 蚁群算法的综合求解性能较好。

实际上, 期望启发式因子  $\beta$  是和启发式因子  $\alpha$  关联性很强的参数, 蚁群算法的全局寻优性能, 首先要求蚁群的搜索过程必须有较强的随机性, 而蚁群算法的快速收敛性能又要求蚁群的搜索过程必须具有较高的确定性。因此,  $\beta$  和  $\alpha$  对蚁群算法性能的影响和作用是相互配合、密切相关的。只有正确选定它们的搭配关系, 才能避免在搜索过程中出现过早停滞或陷入局部最优等情况的发生。

#### 4.2.8 信息素强度对蚁群算法性能的影响

在 Ant-Cycle 模型中,信息素强度  $Q$  为蚂蚁循环一周时释放在所经路径上的信息素总量,其作用是为了充分利用有向图上的全局信息反馈量,使得算法在正反馈机制作用下以合理的演化速度搜索到所求问题的全局最优解。 $Q$  越大,则在蚂蚁已遍历路径上信息素的累积加快,可以加强蚁群搜索时的正反馈性能,有助于算法的快速收敛。

这里采用了 TSPLIB 中 Eil51TSP 为例来进行仿真实验,本实验仍采用了蚁群算法中的 Ant-Cycle 模型。实验时,设置  $m=30$ ,  $\rho=0.5$ ,  $\alpha=1$ ,  $\beta=5$ ,停止条件为相邻两次循环搜索中最优解的差别小于 0.001。

多次实验所得的该 Eil51TSP 算例中  $Q$  与迭代次数  $N_c$  及  $Q$  与最优路径长度  $L$  之间的对应关系分别如图 4.15~图 4.18 所示。

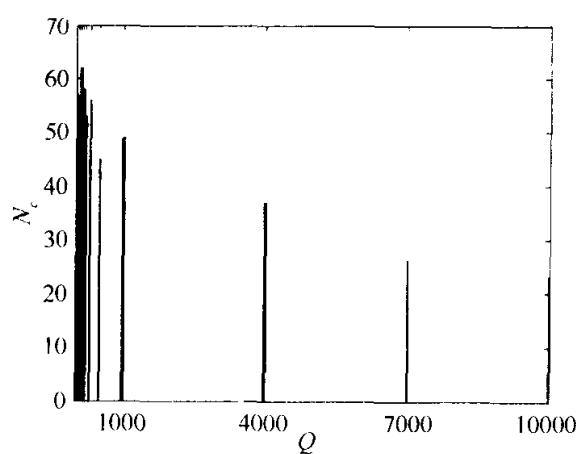


图 4.15  $Q$  与迭代次数  $N_c$  关系的全局图

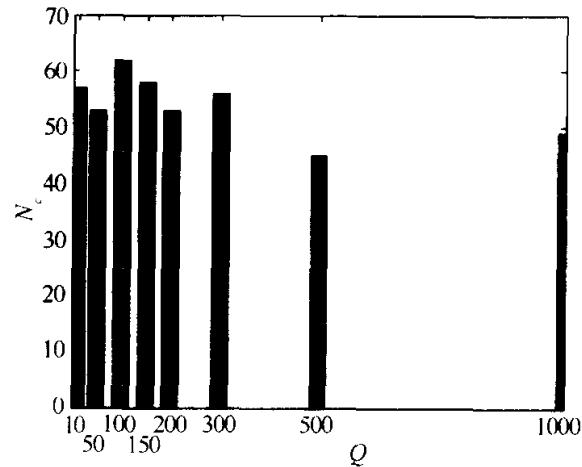


图 4.16  $Q$  与迭代次数  $N_c$  关系的局部放大图

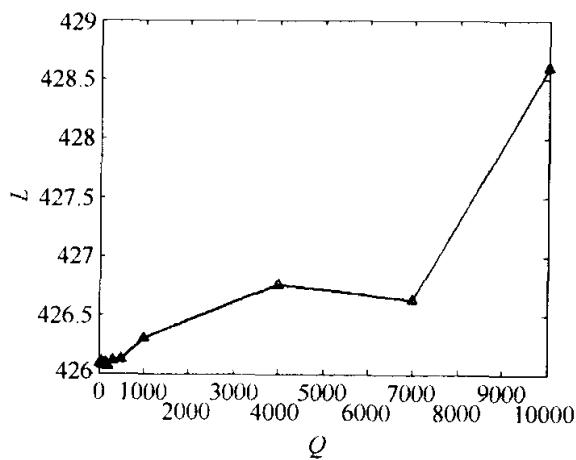


图 4.17  $Q$  与最优路径长度  $L$  关系的全局图

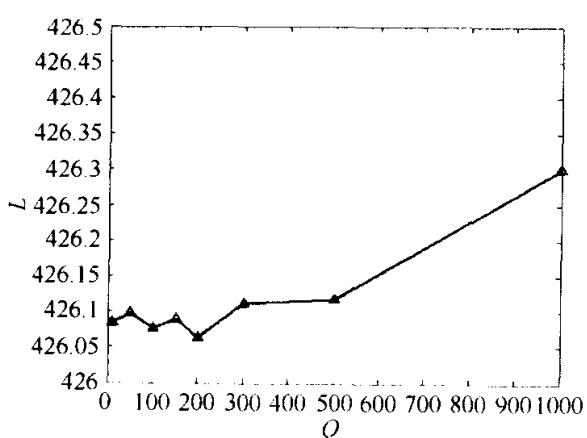


图 4.18  $Q$  与最优路径长度  $L$  关系的局部放大图

由图 4.15~图 4.18 可知,信息素强度  $Q$  越大,蚁群算法的收敛速度越快。当  $Q < 1000$  时,这种规律虽然存在,但对蚁群算法的总体求解性能影响并不大;当  $Q$  特别大(本算例中  $Q > 7000$ )时,虽然蚁群算法的收敛速度很快,但此时算法的全局搜索能力变差,极易陷于局部最优解,计算性能也变得很不稳定。

#### 4.2.9 $\alpha$ 、 $\beta$ 、 $\rho$ 组合配置对蚁群算法性能的影响

实际上,蚁群算法中各参数的作用是紧密耦合的,其中对算法性能起着最关键作用的应该是信息启发式因子  $\alpha$ 、期望启发式因子  $\beta$  和信息素挥发因子  $\rho$  等三个参数。信息素强度  $Q$  对算法性能的影响则有赖于上述 3 个参数的配置以及对算法模型(Ant-Cycle、Ant-Density 和 Ant-Quantity)的选取, $Q$  对算法性能的影响情况显然有较大的差异<sup>[9,10]</sup>。如果  $\alpha$ 、 $\beta$ 、 $\rho$  的组合参数配置不当,会导致求解速度很慢且所得解的质量特别差,因此,在研究蚁群算法中主要参数对其性能影响的基础上,研究其关键参数  $\alpha$ 、 $\beta$ 、 $\rho$  的最佳组合配置策略有着非常重要的意义。

这里用 1000 次循环作为算法终止条件,实验仍采用改变一个参数、其他参数不变的策略来探讨参数设置对蚁群算法性能的影响。默认参数设置为  $\alpha = 1$ ,  $\beta = 1$ ,  $\rho = 0.3$ ,  $Q = 100$ , 在得到每条备选路径概率的情况下,蚂蚁运用随机选择策略确定下一步要到达的城市。每组数据实验 10 次取平均作比较,实验中所用的 TSP 数据来源于 TSPLIB 中的 Oliver30TSP。

Ant-Cycle、Ant-Density 及 Ant-Quantity 模型的实验结果分别如表 4.1、表 4.2 及表 4.3 所示。

表 4.1 Ant-Cycle 模型实验结果

参数名称	参数值	平均值	最优解	最差解	差 值
$\alpha$	0	684.28	660.22	714.91	54.59
	0.5	538.60	502.82	561.73	58.91
	1	431.05	425.26	436.40	11.14
	2	449.76	434.89	469.22	34.33
$\beta$	0	872.32	813.04	895.22	82.02
	0.5	472.23	446.39	482.33	35.94
	1	431.05	425.26	436.40	11.14
	2	425.44	423.90	427.17	3.27
	5	425.07	423.90	426.53	2.63
	10	427.26	425.98	432.39	6.41
$\rho$	0.1	431.05	428.63	436.01	7.38
	0.3	430.65	424.94	435.57	10.63
	0.5	428.53	424.69	431.31	6.62
	0.7	430.92	426.53	434.26	7.73

表 4.2 Ant-Density 模型实验结果

参数名称	参数值	平均值	最优解	最差解	差 值
$\alpha$	0	688.54	660.22	714.91	54.69
	0.5	479.44	451.78	503.95	52.17
	1	463.08	443.45	479.78	36.33
	2	511.43	489.56	530.70	41.14
	5	667.06	660.22	714.91	54.69
$\beta$	0	929.06	887.56	957.37	69.81
	1	463.07	443.45	479.78	36.33
	2	429.23	423.90	437.25	13.35
	5	428.65	423.90	433.02	9.12
	10	427.54	424.69	430.54	5.85
	20	431.70	426.60	438.96	12.36
$\rho$	0.001	434.04	429.91	440.93	11.02
	0.1	433.27	426.60	440.67	14.07
	0.3	463.08	443.45	479.78	36.33
	0.5	604.07	519.83	566.32	46.49
	0.7	633.53	612.92	663.78	50.86

表 4.3 Ant-Quantity 模型实验结果

参数名称	参数值	平均值	最优解	最差解	差 值
$\alpha$	0	687.62	660.22	714.91	54.69
	0.5	439.19	428.63	452.91	24.48
	1	428.92	423.90	434.93	11.03
	2	457.16	435.12	492.03	56.91
	5	486.12	441.67	576.49	134.82
$\beta$	0	465.07	442.22	478.61	36.39
	1	428.92	423.90	434.93	11.03
	2	428.42	423.90	434.23	10.33
	5	427.20	423.90	429.91	6.01
	10	429.49	426.60	435.88	9.28
	20	433.10	429.42	434.64	5.22
	30	437.76	431.33	442.83	11.49
$\rho$	0.001	425.25	423.90	426.69	2.79
	0.1	426.61	423.90	430.61	6.71
	0.3	428.92	423.90	434.93	11.03
	0.5	455.04	429.91	470.46	40.55
	0.7	502.40	483.02	524.81	36.39

在表 4.1~表 4.3 中, 平均值表示将 10 次运行中每次得到的最短路径长的平均值; 最优解表示 10 次运行中得到的 10 条最短路径中的最小值; 最差解表示 10 次运行中每次得到的最短路径中的最大值; 差值表示实验中得到的最优解和最差

解之间的差值。分析表 4.1~表 4.3 所示的实验结果,可以得到如下结论。

(1) Ant-Cycle 模型中最佳参数配置为: $\alpha=1$ ,  $\beta=5$ ,  $\rho=0.5$ ; Ant-Density 模型中最佳参数配置为: $\alpha=1$ ,  $\beta=10$ ,  $\rho=0.1$ ; Ant-Quantity 模型中最佳参数配置为: $\alpha=1$ ,  $\beta=5$ ,  $\rho=0.001$ 。

(2) 3 种蚁群算法模型中,若  $\beta$  和  $\rho$  取默认值, $\alpha=1$  时所得最优值和平均值比  $\alpha$  取其他值时更好,此时其最优值和最差值之差也最小。这说明解的质量和稳定性都是最好的,所以 3 种模型中  $\alpha$  值的最佳设置应为 1。对于  $\beta$ ,其值在 1~5 之间逐渐增大, $\alpha$  和  $\rho$  取默认值时,3 种模型所得解的质量越来越高。Ant-Cycle 和 Ant-Quantity 模型中,当  $\beta$  的取值超过 5 时,所得解的质量开始下降,所以其  $\beta$  值的最佳设置为 5;而 Ant-Density 模型中, $\beta$  值的最佳设置应为 10。对于信息素挥发因子  $\rho$ ,在 Ant-Density 和 Ant-Quantity 模型中表现出相似的设置规律。随着  $\rho$  在 0~0.7 范围内取值的逐渐增大,所得的解越来越好,但  $\rho$  值应该大于 1;而在 Ant-Cycle 模型中, $\rho$  值在 0.1~0.7 之间变化,解变化不大,当其取值为 0.5 时,解的质量最优。

上述分析是在给定 1000 次迭代的条件下得出的。这里还增加了不是最佳参数配置时算法模型的最大运行次数实验,结果表明,对于不是最佳参数配置的算法模型,即使再增加 1000 次或者 2000 次运行,其所得解的质量与 1000 次时相比没有明显改善;而采用最佳参数配置的模型,可以很快地搜索到质量很高的全局优化解。同时,这些实验的结果也没有像理论上分析的那样,由于正反馈作用使所有的蚂蚁均在已知的最短路径(路径总长为 423.73)上移动。然而,如果参数配置恰当,至少可有一只蚂蚁能搜索到较优解 423.90。

在保证获得解的前提下,为了提高计算速度,对基本蚁群算法中的路径选择策略进行了调整,并通过 3 个实验对调整策略做了仿真验证。下面分别介绍这 3 个实验。

**实验 A** 本实验将蚂蚁利用备选路径的概率作为选择路径的方法改变为以  $(\tau_{ij})^\alpha (\eta_{ij})^\beta$  作为选择路径的方法,其具体做法如下:

- (1) 根据 tabu( $k$ ) 表列出备选城市;
- (2) 根据备选城市计算出各备选路径的  $(\tau_{ij})^\alpha (\eta_{ij})^\beta$ ;
- (3) 运用备选路径的  $(\tau_{ij})^\alpha (\eta_{ij})^\beta$  乘积作比较。哪条备选路径的乘积大,就选哪条,参数设置情况和基本蚁群算法相同。

**实验 B** 本实验中,蚂蚁选择路径方法类似实验 A,但参数设置情况与实验 A 不同。这里把参数作动态调整,实验中在达到预定最大循环次数的 1/4 时,动态调整算法参数。算法运行的初始阶段(即最初开始循环阶段,此处指最大循环次数的前 1/4 循环数),3 个参数设置较小( $\alpha=0.5$ ,  $\beta=1$ ,  $\rho=0.3$ ),随着循环次数增加到最大循环次数的 1/4 时,参数调整为  $\alpha=1$ ,  $\beta=5$ ,  $\rho=0.1$ 。

**实验 C** 本实验中的参数设置类似实验 B 作动态调整,但是蚂蚁用随机方法

选择备选路径。

蚂蚁选路策略调整实验结果说明,实验 A、实验 B 中蚂蚁选择路径时没有随机性,故其实验只运行一次。实验 C 中蚂蚁运用随机性选择路径,所以运行了 5 次。实验 A 的最大循环数设定为 200;实验 B 和实验 C 设定 3 个最大循环数,即 80, 100, 200。

3 种方法实验结果如表 4.4~表 4.6 所示。表中平均值表示实验中获得最短路径那一循环中,所有蚂蚁的总的平均路径长;最优解表示实验中获得的最短路径长度;循环表示获得的运行结果中最早出现最短路径的循环次数;最早循环表示 5 次运行中最先得到 423.73 的循环数;最迟循环表示 5 次运行中最迟得到 423.73 的循环数。

表 4.4 实验 A 的结果

最大循环	$\alpha$	$\beta$	$\rho$	平均值	最优解	循环
200	0.5	1	0.3	477.98	443.50	5
200	1	0.5	0.1	516.45	473.29	1

表 4.5 实验 B 的结果

最大循环	$\alpha$	$\beta$	$\rho$	平均值	最优解	循环
80	0.5	1	0.3	492.21	423.73	37
100	0.5	1	0.3	492.21	423.73	43
200	0.5	1	0.3	492.21	423.73	73

表 4.6 实验 C 的结果

最大圈数	$\alpha$	$\beta$	$\rho$	最优解	平均值	平均圈数	最早循环	最迟循环
80	0.5	1	0.3	423.73	492.21	35	32	36
100	0.5	1	0.3	423.73	492.21	40	38	42
200	0.5	1	0.3	423.73	492.21	65	62	71

对表 4.4~表 4.6 所示实验结果的分析如下:

(1) 在 3 种实验方法中,实验 A 的结果较差(表 4.4)。这是因为实验 A 在这个计算过程中始终使用一种参数配置,其中给予较小值配置的结果显得好一些,但也并不是最优解。另一组给予较大值的参数配置,所得结果更差。实验 A 中,参数初始值设置较小时,那些较长的“较差”路径从备选路径中排除的较少;较长的路径上也可能有蚂蚁经过并留下信息素,这样解空间就较大。这种参数设置导致那些较优路径上的信息素随着迭代次数的增加并没有依照“正反馈”原理得到加强,还有可能随着迭代次数的增加,较优路径上的信息素变得越来越少,不能吸引更多

的蚂蚁选择它,其结果是所得解的质量不高。同样,如果开始将参数设置过大,则会把那些较长的“较差”路径从备选路径中排除的过多,将解空间压缩的过小,形成“伪正反馈”,导致蚂蚁寻优的空间过小,极易陷入局部最优解。因为有些路径虽然比较长,但是和它相邻的路径可能会比较优,如果将这种路径排除出了备选路径,则会导致局部最优。

(2) 当计算运行到一定的循环次数(实验中设定为 1/4 最大循环次数)时,可采用增大参数值的措施来提高解的质量。由表 4.5、表 4.6 可见,实验 B、实验 C 所得到的最优解均达到了预计数值 423.73。而从循环次数上考虑,第 3 种方法略显优势。实验 B、实验 C 的参数设置弥补了实验 A 的缺陷,在初始阶段将参数设置较小,得到一个较大的解空间。在后期阶段增大参数减少解的空间,使蚂蚁逐渐向较优的路径上靠拢形成正反馈。这种策略可以防止那些较优路径的信息素强度在循环过程中被削弱,从而使得较优的路径更易在蚂蚁寻路时被选中。虽然解的整体情况不理想,得到最优解的那次循环的平均路径长为 492.21,但是能在较少的循环次数条件下就能求得已知的最优解 423.73。

(3) 从获得一个最优解的角度考虑,增加循环次数并没有什么作用。表 4.5 和表 4.6 中采用了 3 组最大循环次数(即 80、100、200),它们的最优解均相同,并没有因为循环次数的增加而带来更优的解,更不需要像前面那样计算 1000 次。与基本蚁群算法比较,这种新的选择路径策略在减少计算量、快速搜索所求问题的最优解方面有了较大的提高。

### 4.3 蚁群算法参数最优组合的“三步走”方法

在对蚁群算法的参数选择规律实验分析的基础上,本节总结出一种“三步走”选择蚁群算法最优组合参数的有效方法,其具体步骤如下:

(1) 确定蚂蚁数目,即可参照本章第 4.2 节所提出的  $\frac{\text{城市规模}}{\text{蚂蚁数目}} \approx 1.5$  的选择策略来确定蚂蚁的总数目。

(2) 参数粗调,即调整取值范围较大的信息启发式因子  $\alpha$ 、期望启发式因子  $\beta$  以及信息素强度  $Q$  等参数,以得到较理想的解。

(3) 参数微调,即调整取值范围较小的信息素挥发因子  $\rho$ 。

上述步骤反复进行,直到最终确定出一组较为理想的组合参数为止。“三步走”方法对解决不同规模的 TSP 具有一定的参考价值,对用蚁群算法解决其他领域的优化问题也具有一定的指导意义。

## 4.4 本章小结

本章对信息素、启发函数、信息量-启发函数乘积 $[\tau_{kl}(m)]^\alpha [\eta_{kl}(u)]^\beta$ 、蚂蚁之间的合作行为会对蚁群算法收敛性能的影响作了深入分析，随后对蚁群算法中信息素残留因子 $1-\rho$ 、信息启发式因子 $\alpha$ 、期望启发式因子 $\beta$ 、信息素强度 $Q$ 、蚂蚁数目 $m$ 以及 $\alpha$ 、 $\beta$ 、 $\rho$ 组合配置对蚁群算法性能的影响进行了实验研究，并讨论了蚁群算法中相关参数的最佳设置原则。最后总结出一种“三步走”选择蚁群算法最优组合参数的有效方法，该方法对用蚁群算法解决其他领域的优化问题也具有一定的指导意义，这非常有利于蚁群算法在其他优化领域中的进一步推广和应用。

## 参 考 文 献

- 1 Chang H S. An ant system based exploration-exploitation for reinforcement learning. Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, 2004, 4:3805~3810
- 2 段海滨, 王道波. 一种快速全局优化的改进蚁群算法及仿真. 信息与控制, 2004, 33(2): 241~244
- 3 Dorigo M, Maniezzo V, Colomi A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- 4 Botee H M, Bonabeau E. Evolving ant colony optimization. Advances in Complex Systems, 1998, 1(2): 149~159
- 5 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 6 Duan H B, Wang D B, Yu X F. Research on the optimum configuration strategy for the adjustable parameters in ant colony algorithm. Journal of Communication and Computer, 2005, 2(9): 32~35
- 7 郝晋, 石立宝, 周家启. 求解复杂 TSP 问题的随机扰动蚁群算法. 系统工程理论与实践, 2002, 22(9): 88~91
- 8 詹士昌, 徐婕, 吴俊. 蚁群算法中有关算法参数的最优选择. 科技通报, 2003, 19(5): 381~386
- 9 叶志伟, 郑肇葆. 蚁群算法中参数 $\alpha$ 、 $\beta$ 、 $\rho$ 设置的研究—以 TSP 为例. 武汉大学学报, 2004, 29(7): 597~601
- 10 Ye Z W, Zheng Z B. Research on the configuration of parameter  $\alpha$ ,  $\beta$ ,  $\rho$  in ant algorithm exemplified by TSP. Proceedings of the International Conference on Machine Learning and Cybernetics, 2003, 4:2106~2111
- 11 Daniel M, Martin M. Ant colony optimization with global pheromone evaluation for scheduling a single machine. Applied Intelligence, 2003, 18: 105~111
- 12 Merkle D, Middendorf M, Schmeck H. Pheromone evaluation in ant colony optimization. Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society, 2000, 4:2726~2731
- 13 Coloni A, Dorigo M, Maniezzo V. Ant system: optimization by a colony of cooperating agent. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- 14 Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66
- 15 Gambardella L M, Dorigo M. Ant-Q: a reinforcement learning approach to the traveling salesman problem. Proceedings of the 12th International Conference on Machine Learning, 1995, 252~260

- 16 Dorigo M, Gambardella L M. A study of some properties of Ant-Q. Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, 1996, 656~665
- 17 Hadeli, Paul V, Martin K, *et al.* Multi-agent coordination and control using stigmergy. Computers in Industry, 2004, 53(1): 75~96
- 18 Golden B, Stewart W. Empiric analysis of heuristics. The Traveling Salesman Problem, Lawler E L, Lenstra J K, Rinnooy-Kan A H G , *et al*(Eds. ), New York: Wiley and Sons, 1985

## 第 5 章 离散域蚁群算法的改进研究

### 5.1 引言

任何事物都具有两面性，蚁群算法作为一种新兴的仿生优化算法也不例外。蚁群算法固然具有采用分布式并行计算机制、易于与其他方法结合、具有较强的鲁棒性等优点，但搜索时间长、易陷于局部最优解是其最为突出的缺点<sup>[1~5]</sup>。针对这些缺陷，近些年来众多国内外学者在蚁群算法的改进方面做了大量的研究工作，这些改进有一个共同的目的，那就是在合理时间复杂度的限制条件下，尽可能提高蚁群算法在一定空间复杂度下的寻优能力，从而改善蚁群算法的全局收敛性，并拓宽蚁群算法的应用领域。

Dorigo M 等<sup>[6, 7]</sup>在基本蚁群算法的基础上提出了一种称之为 Ant-Q System 的蚁群算法，该算法仅让每次循环中最短路径上的信息量作更新，且仅让信息量最大的路径以较大的概率被选中，以充分利用学习机制，强化最优信息的反馈；德国学者 Stützle T 和 Hoos H 提出了另一种改进的蚁群算法——“最大最小蚂蚁系统”（MAX-MIN ant system, MMAS）<sup>[8~10]</sup>，MMAS 限定了信息量允许值的上下限，并在算法中采用了轨迹平滑机制（trail smoothing mechanism）。初始时 MMAS 将所有路径弧段上的信息量设为最大值  $\tau_{\max}$ ，每次迭代后，按挥发系数  $\rho$  减小信息量，只有最佳路径上的弧段才允许增加其信息量；同时为了避免发生早熟现象，该算法将各条路径可能的信息量限制在区间  $[\tau_{\min}, \tau_{\max}]$  之内，这样可以有效地避免某条路径上的信息量远大于其他路径，使得所有的蚂蚁都集中到同一条路径上，从而使算法不再扩散。直到今天，MMAS 仍然是解决 TSP、QAP 等离散域优化问题的最好蚁群算法模型之一，很多对蚁群算法的改进策略都渗透着 MMAS 的思想。吴庆洪等<sup>[11]</sup>从遗传算法中变异算子的作用得到启发，在蚁群算法中采用了逆转变异机制，进而提出了一种具有变异特征的蚁群算法，这是国内学者对蚁群算法所做的最早改进（1997 年 11 月）。

本章将从不同角度研究离散域蚁群算法的若干改进策略，其仿真算例都是 TSP。

### 5.2 自适应蚁群算法

蚁群算法在构造解的过程中，随机选择策略使得算法的进化速度变慢，正反馈原理旨在强化性能较好的解，却容易出现停滞现象。对蚁群算法的状态转移概

率、信息素挥发因子、信息量等因素采取自适应调节策略是一个基本的改进思路<sup>[12]</sup>，可在一定程度上有效地克服了基本蚁群算法的一些不足之处。

本节首先介绍了 Ant-Q 算法的自适应改进思路，然后详细讨论了信息素本身以及信息素挥发因子的自适应改进策略，并对改进后的蚁群算法做了仿真验证。

### 5.2.1 基于 Q-学习的自适应蚁群算法

蚁群算法的状态转移概率反映了蚁群算法与 Q-学习算法之间的内在联系。其中， $\tau_{ij}$  相当于 Q-学习算法中的 Q 值，表示学习所得到的经验，而  $\eta_{ij}$  由某种启发式算法确定，如何将二者有效结合是提高蚁群算法收敛性能的最基本的关键性问题。

Dorigo M 等<sup>[1]</sup>在提出基本蚁群算法后不久，又提出一种新的蚁群算法，并将其命名为 Ant-Q System<sup>[6,7]</sup>。为了避免出现停滞现象，Dorigo M 等在该算法中采用了确定性选择和随机性选择相结合的选择策略，并在搜索过程中动态调整状态转移概率<sup>[13,14]</sup>。具体而言，Ant-Q System 使用了自适应伪随机比率选择规则，即对位于城市  $r$  的蚂蚁  $k$  按照下式选择下一城市  $r$

$$s = \begin{cases} \arg \max_{u \in \text{allowed}_r} \{ [AQ(r, u)]^\alpha [HE(r, u)]^\beta \}, & \text{若 } q \leq q_0 \\ \text{公式(5.2.2),} & \text{否则} \end{cases} \quad (5.2.1)$$

$$p_k(r, s) = \begin{cases} \frac{[AQ(r, s)]^\alpha \cdot [HE(r, s)]^\beta}{\sum_{u \in J_k(r)} [AQ(r, u)]^\alpha \cdot [HE(r, u)]^\beta}, & \text{若 } s \in J_k(r) \\ 0, & \text{否则} \end{cases} \quad (5.2.2)$$

式中， $q_0$  是区间  $[0, 1]$  内的一个随机数； $J_k(r)$  表示待选择的城市集合； $HE(r, u)$  表示启发式信息，信息量 AQ 值按照如下规则进行更新

$$AQ(r, s) \leftarrow (1 - \delta) \cdot AQ(r, s) + \delta \cdot (\Delta AQ(r, s) + \gamma \cdot \max_{u \in \text{allowed}_r} AQ(s, u)) \quad (5.2.3)$$

公式 (5.2.1) ~ (5.2.3) 进一步揭示了 Ant-Q System 与 Q-学习算法之间的内在联系。其中，信息素增量  $\Delta AQ(r, s)$  的求法有两种，即全局最优 (global-best) 法和本次迭代最优 (iteration-best) 法。下面分别介绍如下。

(1) 全局最优法的公式为

$$\Delta AQ(r, s) = \begin{cases} W / L_{k_{gb}}, & \text{若 } (r, s) \text{ 为蚂蚁 } k_{gb} \text{ 所经过的路径} \\ 0, & \text{否则} \end{cases} \quad (5.2.4)$$

式中， $W$  为常数，一般设  $W=10$ ； $k_{gb}$  表示寻到与全局最优解相对应路径的蚂蚁； $L_{k_{gb}}$  表示该蚂蚁所经过的路径长度。公式 (5.2.4) 表明只有与全局最优解相对

应的 AQ 值才可得到强化。

(2) 本次迭代最优法的公式为

$$\Delta AQ(r,s) = \begin{cases} W/L_{k_b}, & \text{若 } (r,s) \text{ 为蚂蚁 } k_b \text{ 所经过的路径} \\ 0, & \text{否则} \end{cases} \quad (5.2.5)$$

式中,  $k_b$  表示本次迭代中寻到最优路径的蚂蚁,  $L_{k_b}$  表示该蚂蚁所经过的路径长度。公式 (5.2.4) 表明只有与全局最优解相对应的 AQ 值方可得到强化。

Dorigo M 等通过大量的仿真实验表明, 全局最优法和本次迭代最优法的求解效果非常接近, 但在实际应用中更倾向于使用本次迭代最优法。其原因有两个: 一是在同样的求解效果下, 本次迭代最优法比全局最优法速度快; 二是本次迭代最优法对因子  $\gamma$  的选择不敏感。

## 5.2.2 自适应调整信息素挥发因子的蚁群算法

由上一章的实验分析可知, 蚁群算法中信息素挥发因子  $\rho$  的大小直接关系到蚁群算法的全局搜索能力及其收敛速度。当要处理的问题规模比较大时, 这种影响更为突出。基于此, 这里研究了一种自适应调整信息素挥发因子的蚁群算法改进策略<sup>[15,16]</sup>, 旨在通过自适应的改变信息素挥发因子  $\rho$  来提高算法的全局性。

### 5.2.2.1 算法改进

假设信息素挥发因子  $\rho$  的初始值  $\rho(t_0)=1$ , 则当蚁群算法求得的最优值在  $N$  次循环内没有明显改进时,  $\rho$  按照下式作自适应调整

$$\rho(t) = \begin{cases} 0.95\rho(t-1), & \text{若 } 0.95\rho(t-1) \geq \rho_{\min} \\ \rho_{\min}, & \text{否则} \end{cases} \quad (5.2.6)$$

式中,  $\rho_{\min}$  为  $\rho$  的最小值, 可以防止  $\rho$  过小降低算法的收敛速度。同时, 为了提高蚁群算法的全局搜索能力, 提高其搜索速度, 还在每次循环结束时求出最优解, 并将其保留。改进后自适应蚁群算法的伪代码如下:

```
Begin
    初始化
    While (不满足算法终止条件);
        {For ( $a_k=1$ ;  $a_k < m$ ;  $a_k++$ )
            将  $m$  只蚂蚁随机放置于初始城市上
            For ( $i=1$ ;  $i < n$ ;  $i++$ )
                {For ( $a_k=1$ ;  $a_k < m-1$ ;  $a_k++$ )
                    蚂蚁  $a_k$  以概率  $p_{ij}^k$  选择下一城市
                }
            求出最佳结果
        }
```

```

将最佳结果赋予蚂蚁  $m$ 
If 最佳解 =  $N$  个循环以前的最佳解
根据公式(5.2.6)更新  $\rho$ 
更新  $\Delta\tau_{ik}$  ;
更新  $\tau_{ik}$  ;
 $\Delta\tau_{ik} = 0$  ;
 $N_c = N_c + 1$  ;
}
输出最佳结果
End

```

### 5.2.2.2 仿真算例

这里采用 TSPLIB 中的 Oliver30TSP 作为仿真算例。表 5.1 为基本蚁群算法在不同组合参数下的实验结果，而表 5.2 为自适应蚁群算法在不同组合参数下的实验结果。

表 5.1 基本蚁群算法在不同参数下的实验结果

$\alpha$	$\beta$	$\rho$	最短路径长度	迭代次数
1	4	0.5	395.2774	327
1	4	0.9	398.7039	341
2	2	0.5	390.5832	339
2	2	0.9	394.2361	308
4	1	0.5	316.6900	345
4	1	0.9	317.0540	322

表 5.2 自适应蚁群算法在不同参数下的实验结果

$\alpha$	$\beta$	$\rho_{min}$	最短路径长度	迭代次数
1	4	0.001	310.8758	162
1	4	0.5	322.5945	147
1	4	0.1	308.1685	150
2	2	0.001	271.1854	152
2	2	0.5	303.8851	143
4	1	0.1	287.8268	119
4	1	0.001	267.7795	168
4	1	0.1	290.4864	137

由表 5.1 和表 5.2 结果可见, 自适应蚁群算法在不同参数下所得到的最差结果小于基本蚁群算法的最优结果, 而其所需迭代次数也由基本蚁群算法的 300 代以上缩短为 200 代以内。图 5.1 和图 5.2 分别给出了自适应蚁群算法最好解和最差解的进化曲线。

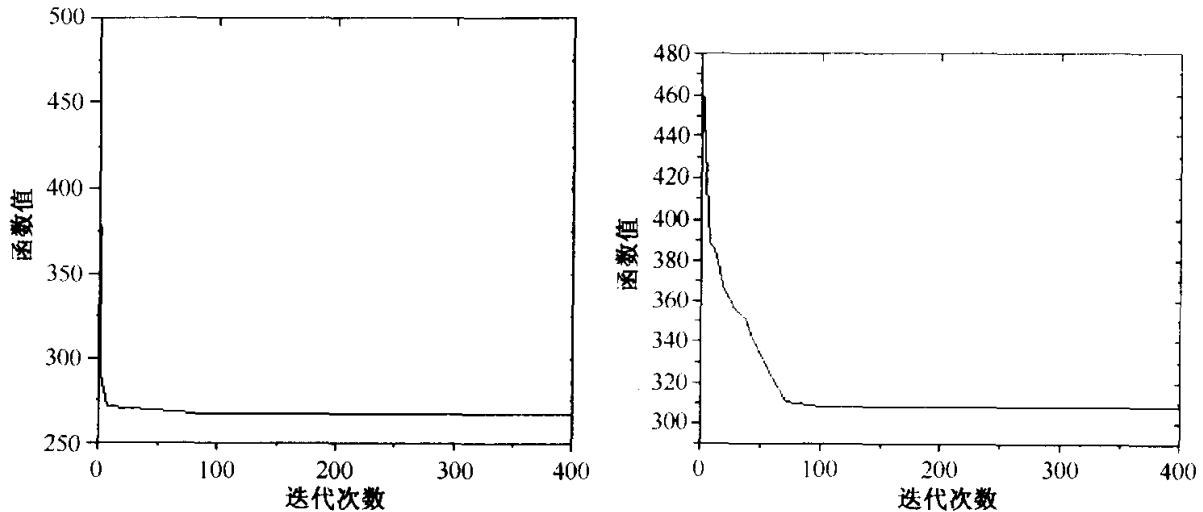


图 5.1 自适应蚁群算法最好解的进化曲线

图 5.2 自适应蚁群算法最差解的进化曲线

### 5.2.3 动态自适应调整信息素的蚁群算法

针对蚁群寻优过程中容易出现停滞和陷入局部最优等问题, 这里研究了一种根据蚁群算法搜索情况来自适应动态修改信息素的方法<sup>[17]</sup>, 可在一定程度上有效地解决扩大搜索空间和寻找最优解之间的矛盾, 从而使得算法跳离局部最优解。

#### 5.2.3.1 算法改进

这里采用时变函数  $Q(t)$  来代替调整信息素  $\Delta\tau_{ij}^k = Q/L_k$  中为常数项的信息素强度  $Q$ , 即选择

$$\Delta\tau_{ij}^k(t) = f(t) = \frac{Q(t)}{L_k} \quad (5.2.7)$$

由状态转移概率公式可知, 当  $\alpha=0$  时, 只是路径信息起作用, 算法相当于最短路径寻优, 从而有

$$p_{ij}^k = \eta_{ij}^\beta(t) \quad (5.2.8)$$

当  $\beta=0$  时, 路径信息的启发作用等于 0, 此时算法相当于盲目地随机搜索, 从而有

$$p_{ij}^k = \frac{\tau_{ij}^k(t)}{\sum \tau_{is}^k(t)} \quad (5.2.9)$$

选用时变函数代替常数项  $Q$ ，在路径上的信息素随搜索过程蒸发或增多的情况下，继续在蚂蚁的随机搜索和路径信息的启发作用之间继续保持“探索”和“利用”的平衡点。这里，可选择如下阶梯函数

$$Q(t) = \begin{cases} Q_1, & \text{若 } t \leq T_1 \\ Q_2, & \text{若 } T_1 < t \leq T_2 \\ Q_3, & \text{若 } T_2 < t \leq T_3 \end{cases} \quad (5.2.10)$$

式中， $Q_i$  对应阶梯函数的不同取值。 $Q(t)$ 也可选择连续函数，如  $\tanh(t)$ ，如图 5.3 所示。

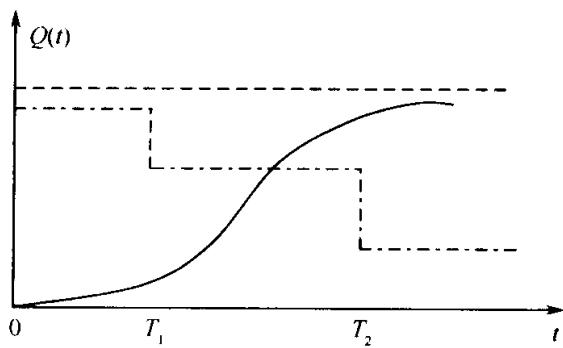


图 5.3 时变函数

如果一段时间内获得的最优解没有变化，说明搜索陷入某个极值点中（未必是全局最优解），此时可采用强制机制减小要增加的信息量，力图使其从局部极小值中逃脱出来，即减小公式 (5.2.7) 中的  $Q(t)$ ；在搜索过程的初始阶段，为了避免陷入局部最优解，缩小最优路径和最差路径上的信息量，需要适当抑止蚁群算法中的

正反馈，在搜索过程中可以加入少量负反馈信息量，如采取  $Q(t) = -0.0001$ ，以减小局部最优解与最差解对应路径上的信息素的差别，从而扩大算法的搜索范围。由于信息正反馈及信息素随时间衰减这两个因素的存在，在搜索陷入局部最优时，某组信息素相对其他路径弧段的信息素而言在数量上占据绝对的优势，因此本算法还对各路径弧段上的信息量做最大和最小值的限制，即对于  $\forall \tau_{ij}(t)$ ，有

$$fPhrm\_min \leq \min_{t \rightarrow \infty} \tau_{ij}(t) = \tau_{ij} \leq fPhrm\_max \quad (i, j \in [1, \dots, n]) \quad (5.2.11)$$

为了使算法对于不同的问题具有更强的适应性，有必要考虑路径弧段的归一化。对于路径弧段的归一化，多采用线性映射方法，即对于  $D = [d_{ij}]$ ，其中  $d_{ij} \in [d_{ij\min}, d_{ij\max}]$ ，将其映射到  $D' = [d'_{ij}]$ ，TSP 中路径弧段所对应的代价值不小于 0，因此， $d'_{ij} = a \cdot d_{ij} + b \in [0, 1]$ ，从而

$$\begin{aligned} d'_{ij\min} &= 0 = a \cdot d_{ij\min} + b \Rightarrow a = 1/(d_{ij\max} - d_{ij\min}) \\ d'_{ij\max} &= 1 = a \cdot d_{ij\max} + b \Rightarrow b = a \cdot d_{ij\min} \end{aligned} \quad (5.2.12)$$

### 5.2.3.2 仿真算例

这里以 TSPLIB 中的 Eil51TSP 作为仿真算例。设置  $\alpha=1$ ,  $\beta=5$ ,  $\rho=0.98$ ,

初始信息量  $f\text{Predefined\_phrm}=15$ , 最小信息量  $f\text{Phrm\_min}=1.0e^{-4}$ , 最大信息量  $\text{Phrm\_max}=15$ ,  $m=n$ 。

选取如下的  $Q(t)$  函数进行了实验, 并与基本蚁群算法获得的结果进行比较, 实验结果如表 5.3 所示。

$$Q(t) = \begin{cases} 0.5, & \text{若 } t < 100 \\ 1, & \text{若 } 100 \leq t < 200 \\ 2, & \text{若 } t \geq 200 \end{cases} \quad (5.2.13)$$

表 5.3 不同蚁群算法的仿真实验统计计算结果

算 法	平均解	最优解	最差解	平均值误差 (%)	已知最优解	平均运行时间(s)
自适应蚁群算法	427.960	426	431	0.469	426	100.64
基本蚁群算法	430.36	426	435	1.023	426	110

由表 5.3 可见, 改进后的自适应蚁群算法所获得的解明显优于基本蚁群算法, 平均运行时间也少于基本蚁群算法。

### 5.3 基于去交叉局部优化策略的蚁群算法

用蚁群算法求解 TSP, 若 TSP 规模较小, 一般能快速找到最优解, 但随着 TSP 规模的扩展, 求解过程会变得更加复杂。因此, 可考虑在蚁群算法中引入去交叉策略<sup>[18, 19]</sup>, 并对每代所构造的解进行局部优化, 以加快蚁群算法的收敛速度。

#### 5.3.1 判断交叉路径的原理和方法

判断两条路径是否存在交叉的原理如下。

平面内两条线段可能出现的情况如图 5.4 所示。

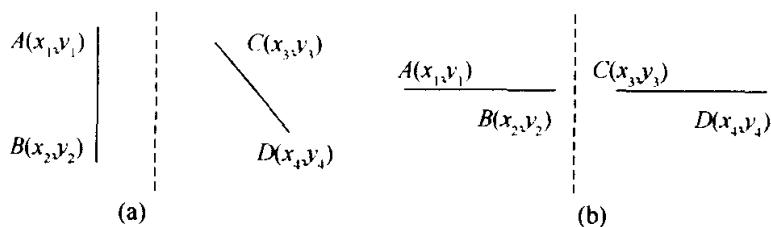


图 5.4 两条线段可能出现的情况

设交叉路径的 4 个顶点的坐标为  $(x_1, y_1)$ 、 $(x_2, y_2)$  和  $(x_3, y_3)$ 、 $(x_4, y_4)$ ,

则两条线段的直线方程分别为：

$$\text{线段 } AB : \begin{cases} x = x_1 + (x_2 - x_1) \cdot m \\ y = y_1 + (y_2 - y_1) \cdot m \end{cases} \quad 0 \leq m \leq 1 \quad (5.3.1)$$

$$\text{线段 } CD : \begin{cases} x = x_3 + (x_4 - x_3) \cdot n \\ y = y_3 + (y_4 - y_3) \cdot n \end{cases} \quad 0 \leq n \leq 1 \quad (5.3.2)$$

设  $D = \begin{vmatrix} x_2 - x_1 & x_3 - x_4 \\ y_2 - y_1 & y_3 - y_4 \end{vmatrix}$ , 当  $D=0$  时, 两线段  $AB$ 、 $CD$  平行或重合, 则

视为无交点; 而当  $D \neq 0$  时, 有

$$m = \frac{1}{D} \begin{vmatrix} x_3 - x_1 & x_3 - x_4 \\ y_3 - y_1 & y_3 - y_4 \end{vmatrix} \quad (5.3.3)$$

$$n = \frac{1}{D} \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix} \quad (5.3.4)$$

当  $0 \leq m \leq 1$  且  $0 \leq n \leq 1$  时, 线段  $AB$ 、 $CD$  有交点。反之, 则视线段  $AB$ 、 $CD$  无交点。

### 5.3.2 交叉路径的消除

如图 5.5 所示, 假设 TSP 中的旅行商先到达  $A$  点, 然后  $B$  点, 再次  $C$  点、 $D$  点, 则可以证明, 只有用  $A$  指向  $C$  且  $B$  指向  $D$  的连线来替代  $AB$ 、 $CD$  才是唯一的可行解。其证明过程如下所述。

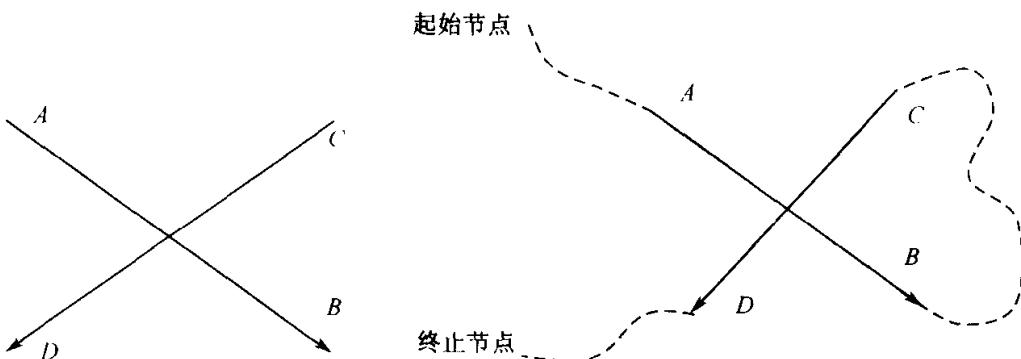


图 5.5 两条线段交叉时的情况

图 5.6 旅行商的交叉路径轨迹

如果旅行商从开始节点出发, 在  $A$ 、 $B$ 、 $C$ 、 $D$  四个节点当中首先到达  $A$  点, 下一个节点将到达  $B$  点, 然后在剩下的两个节点  $C$ 、 $D$  之中先到达  $C$  点, 再到达  $D$  点。由此, 该旅行商具体的行走路径如图 5.6 所示。

显然, 如果用  $A$  指向  $D$  且  $B$  指向  $C$  的连线来代替  $AB$ 、 $CD$ , 沿  $D$  的后继节点走下去将不经过  $B$ 、 $C$  两点而直接到达终点, 这样就形成了两个独立的环路, 如图 5.7 所示。

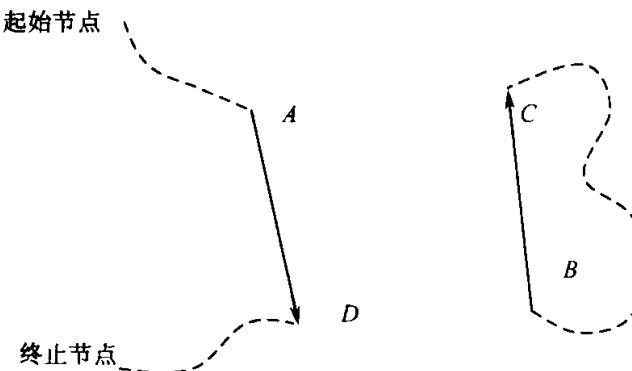


图 5.7 旅行商的非交叉路径轨迹示例

因此，只有用  $A$  指向  $C$ ,  $B$  指向  $D$  的连线来替代  $AB$ ,  $CD$  才是唯一的可行解。

实际算法中的操作为：从起始节点开始沿原有路径向后逐个遍历节点，直到遇到  $A$  点，连接  $A$  点至  $C$  点，随后将  $BC$  之间路径的方向反向，并将  $B$  点连接到  $D$  点，然后向下沿原有顺序遍历直至到达终止节点。由此，可将如图 5.5 所示的情况转换为如图 5.8 所示的情况。

下面举几个例子来进一步说明如何对 TSP 中  $n$  个节点的完全图进行不交叉处理。

首先，可从最简单的两节点情况进行分析，如图 5.9 所示。

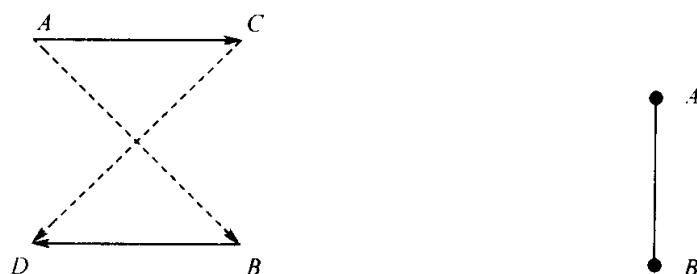


图 5.8 图 5.5 的演化情况

图 5.9 两个节点的情况

由图 5.9 可见，对于两个节点的情况，只要将这两个节点直接进行连接就能得到 TSP 的唯一最优解，不存在含有交叉路径的情况。

当 TSP 节点数为 3 个的时候，可按节点是否共线将其分为如图 5.10 所示的两种情形。

由图 5.10 可见，当三点不共线时，TSP 的解是唯一的最优解，不存在任何路段交叉的情形；而当三点共线时，TSP 的解也是唯一的最优解，但存在路径交叉的情况，而此时出现交叉情况是不可避免的。

当节点数增加时，情况即变得复杂起来。这里以 5 个节点的 TSP 作例子来说明实际求解过程中如何消除路径的交叉。最初给出的五节点 TSP 的解如图

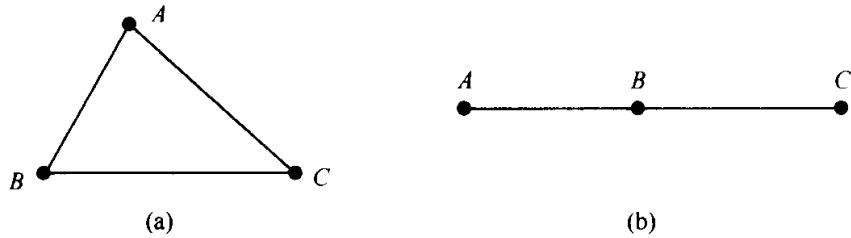


图 5.10 3个节点的情况

5.11(a)所示，其去交叉优化过程如下：

从节点  $A$  出发，发现  $AC$  和  $BD$  交叉，则  $AB$ 、 $CD$  取代  $AC$ 、 $BD$ ，即将图 5.11(a)优化成为图 5.11(b)；又  $BE$  与  $DA$  交叉，以  $BD$ 、 $EA$  取代  $BE$ 、 $DA$ ，即将图 5.11(b)优化成为图 5.11(c)；又  $BD$  与  $CE$  交叉，以  $BC$ 、 $DE$  取代  $BD$ 、 $CE$ ，即将图 5.11(c)优化成为图 5.11(d)。至此，图 5.11 中不再含有交叉的情形，从而完成了对该 TSP 解的优化。

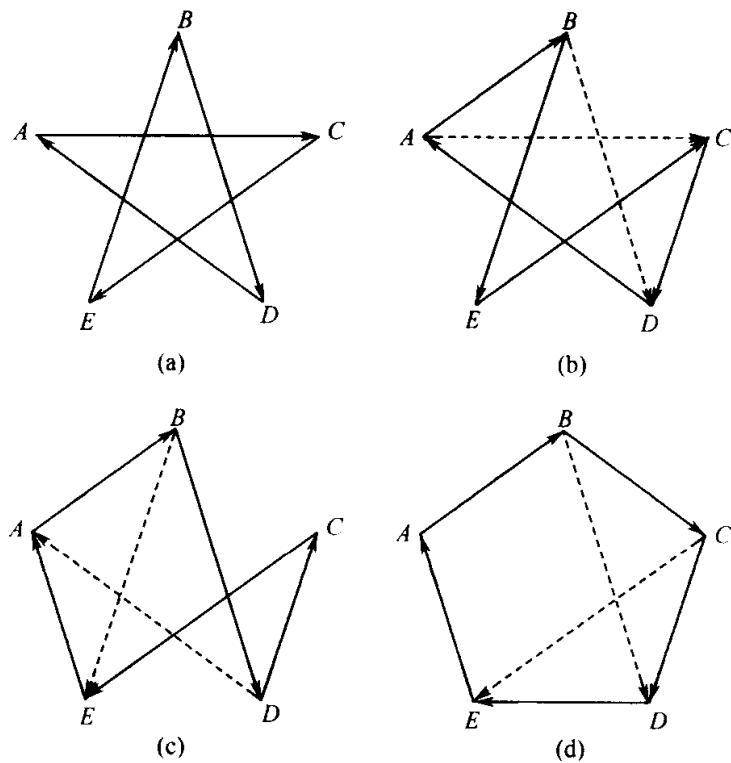


图 5.11 五节点 TSP 的去交叉优化过程

由上述分析可知，TSP 的最优解中绝对不含任何交叉线路的情形，因此，对于一个初始的大规模 TSP，可考虑对其进行预处理，以消除问题可行解中一些不会出现的路径，从而简化 TSP 的求解过程，并降低其计算复杂度。

### 5.3.3 对 TSP 进行预处理

这里以一个具体实例进行分析，如图 5.12 所示。

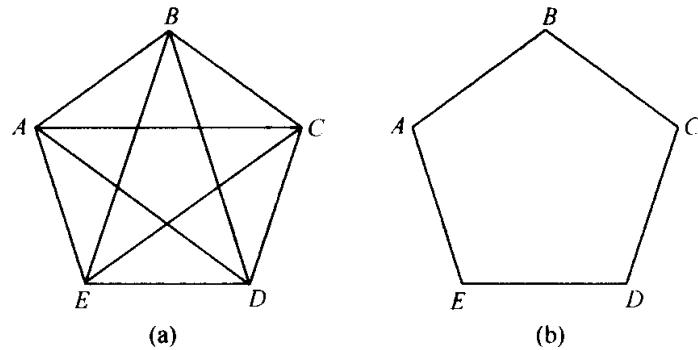


图 5.12 TSP 预处理实例

由图 5.12 (a) 可见，显然  $A$ 、 $B$ 、 $C$ 、 $D$ 、 $E$  是凸包顶点，若结果中存在不相邻的两个凸包顶点的连线，此解必然不是最优解。这里对图 5.12(a) 所示的 TSP 进行预处理，可得到如图 5.12(b) 所示的结果。由图 5.12(b) 可见，这已经是该问题的最优解了。该预处理实例说明对 TSP 进行预处理可大大简化问题的求解过程。

下面介绍一下与去交叉局部优化相关的求凸壳顶点的算法<sup>[20]</sup>：凸壳问题 (convex hull problem) 是几何学中的基本问题之一，许多实际问题都可以归纳成凸壳问题。设  $S$  是平面上的非空点集， $z_1$  和  $z_2$  是  $S$  中的任意两点， $t$  是 0 与 1 之间的任意实数。如果满足公式 (5.3.5) 的点  $z$  属于  $S$ ，则称  $S$  为凸集。

$$z = tz_1 + (1-t)z_2 \quad (0 \leq t \leq 1) \quad (5.3.5)$$

凸集  $S$  的凸壳是包含  $S$  的周长最小的凸多边形，凸壳必包含凸集，它具有许多好的特性，并具有非常广泛的应用。这里提出的算法思路是：首先求出平面点集各点纵、横坐标最大、最小值对应的点所围成的四边形，该四边形把点集分成 5 个部分。处在四边形内部的点不是凸壳的顶点，然后算法对四边形外部的点按横坐标排序，最后检查各点，由此可确定凸壳的顶点。

在 TSP 预处理算法中，从某一凸壳顶点开始，需要依次判断与其不相邻的下一个凸壳顶点是否在同一条直线上，若不是，则消去此线。然后依此类推，依次对其余的凸壳顶点进行同样处理。

### 5.3.4 仿真算例

选择如图 5.13 所示规模为 16 个均匀分布城市的 TSP 实例来进行仿真实验，其理论最优解为 160。

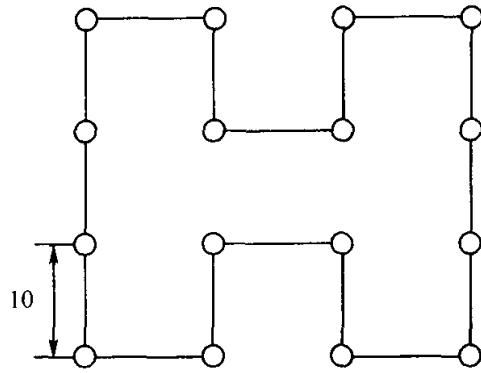


图 5.13 16 个城市的 TSP

设置  $\alpha=1.0$ ,  $\beta=1.0$ ,  $\rho=0.5$ ,  $m=16$ , 采用基本蚁群算法和改进蚁群算法分别求解最优解的统计结果如表 5.4 所示。

表 5.4 基于不同蚁群算法的 16 城市 TSP 求解结果统计比较

实验次数		1	2	3	4	5	6	7	8	9	平均
基本蚁群算法	迭代次数	8	23	11	14	15	12	25	10	22	15.56
	时间 (s)	10	26	13	16	17	15	27	12	25	17.89
改进蚁群算法	迭代次数	2	12	6	7	3	3	1	2	5	4.56
	时间 (s)	5	21	11	12	7	8	4	6	10	9.33

由表 5.4 可见, 去交叉局部优化策略改进了蚁群算法的局部搜索能力, 并提高了算法的全局收敛速度。

## 5.4 基于信息素扩散的蚁群算法

在自然界中, 蚂蚁之间的信息传递主要是通过信息素扩散完成的。先前蚂蚁对后面蚂蚁的行为发生影响时, 后面蚂蚁一般不在先前蚂蚁的运动轨迹上, 而是与该运动轨迹有着或大或小的距离。当距离比较小时, 后面蚂蚁的行为受影响较大。而现有的多数蚁群算法模型并没有考虑信息素的扩散, 对其有效性也没有一个完整的理论分析<sup>[21]</sup>。本节研究了一种与真实蚁群系统更加相符的基于信息素扩散的蚁群算法<sup>[22]</sup>, 旨在更进一步真实地体现蚁群信息系统, 以提高蚁群算法的全局收敛性能。

### 5.4.1 基于信息素扩散的蚁群算法原理

在蚂蚁进行路径搜索时, 如果找到一段很短的子路径 (子解), 它就释放出相

应浓度的信息素，该信息素一方面直接影响位于子解的两个城市上的蚂蚁，另一方面它会以该路径为中心向外扩散，影响附近其他蚂蚁的行为，使它们在寻找路径时以更大的概率在下一步选择此路径。通过这种基于信息素扩散的协作方式，其他蚂蚁在选择下一城市时所受的干扰会减小，从而可提高算法的收敛速度。

下面以图 5.14 所示的实例来说明基于信息素扩散蚁群算法的原理。

在图 5.14 中，假设蚂蚁甲欲从  $c$  到  $g$ ，其中  $ab$  为一距离很短的路径。若原先在路径  $bc$  和  $dc$  上的信息量相同，由于  $dc$  比  $bc$  短，蚂蚁甲将以更大的概率选择  $d$  为下一步要到达的位置。如果在此之前，蚂蚁乙刚好从  $b$  走到  $a$ ，则其将产生相应浓度的信息素，然后向周围进行扩散。由于距离远近的不同，路径  $bc$  上的信息量将大于路径  $dc$  上的信息量，当大到一定程度时，蚂蚁甲将以更大的概率选择  $b$  为下一步要到达的位置，从而使蚂蚁甲选择最短路径  $cbag$  的干扰减小。

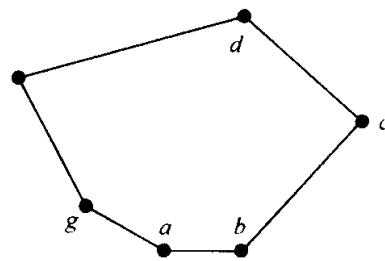


图 5.14 基于信息素扩散的  
蚁群算法原理示意图

## 5.4.2 算法设计

采用 Ant-Quantity 作为基本模型，在其中引入了所提出的信息素扩散模型。

### 5.4.2.1 信息素扩散模型

信息素在空气中的浓度与距离之间的关系如图 5.15 所示。

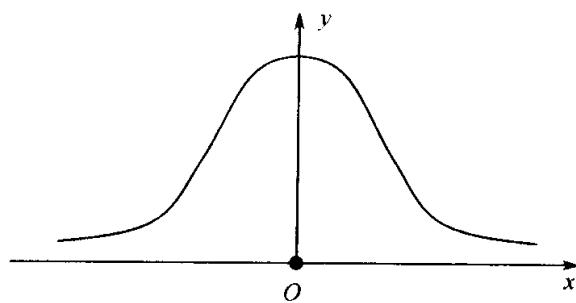


图 5.15 信息素扩散示意图

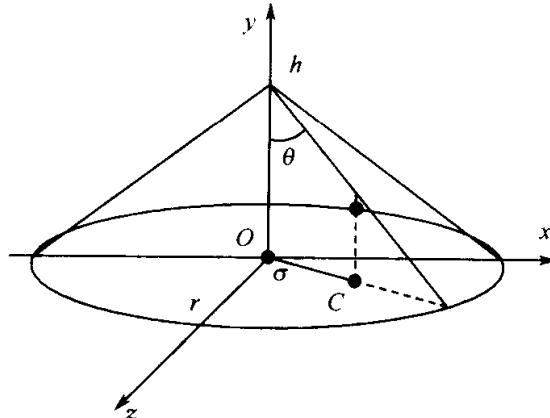


图 5.16 简化后的信息素扩散模型

图 5.15 中，横坐标  $x$  表示与产生信息素的信源之间的距离，纵坐标  $y$  表示信息量， $O$  点为信源的位置。由图 5.15 可见，空气中信息量随着与信源之间距

离的增大而减小，近似服从高斯分布。为了便于计算，这里对图 5.15 所示的扩散模型作了简化，具体如图 5.16 所示。

在简化后的信息素扩散模型中， $O$  点的信息量为  $D_{\max}$ ，参数  $\theta$  为一锐角，在基于信息素扩散的蚁群算法模型中保持不变， $h$  是圆锥体的高， $r$  表示扩散范围的半径，其大小为  $h \cdot \tan\theta$ 。以图 5.16 中的  $C$  点为例，在该点的蚂蚁所接收到的信息量  $D_C$  可用下式描述

$$D_C = D_{\max} \cdot \frac{h \cdot \tan\theta - \sigma}{h \cdot \tan\theta} \quad (5.4.1)$$

式中， $\sigma$  表示  $C$  点到  $O$  点的距离。公式 (5.4.1) 表明蚂蚁距离信源越远，则其所接收到的信息素越少。

#### 5.4.2.2 用于求解 TSP 的信息素扩散模型

假定蚂蚁  $k$  刚走过的两个城市  $i$  和  $j$  之间的距离为  $d_{ij}$ ，我们认为此时该蚂蚁将分别以  $i$  和  $j$  为中心向周围扩散信息素，其中  $i$  点和  $j$  点的信息量均为  $D_{\max}$ 。扩散的结果将形成形状如图 5.16 所示且分别以  $i$  和  $j$  为底面中心的圆锥体（简化的信息量场）。对于其他任意城市  $l$ ，如果它在蚂蚁  $k$  所产生信息素的扩散范围之内，则能求出扩散到该城市的由蚂蚁  $k$  所产生的信息量  $D_{il}^k$  和  $D_{jl}^k$ 。这里直接用  $D_{il}^k$  和  $D_{jl}^k$  去更新路径  $(i, l)$  和  $(j, l)$  上的信息量，即得到第  $k$  只蚂蚁在一次循环中留在各有关路径上的信息量分别如公式 (5.4.2)～(5.4.4) 所示。

$$\Delta\tau_{ij}^k = \frac{Q}{d_{ij}} \quad (5.4.2)$$

$$\Delta\tau_{il}^k = D_{il}^k \quad (5.4.3)$$

$$\Delta\tau_{jl}^k = D_{jl}^k \quad (5.4.4)$$

其中，公式 (5.4.2) 表示如果第  $k$  只蚂蚁在时刻  $t$  到  $t+1$  之间经过路径  $(i, j)$ ；公式 (5.4.3) 表示如果第  $k$  只蚂蚁在时刻  $t$  到  $t+1$  之间经过城市  $i$  但不经过城市  $l$ ；公式 (5.4.4) 表示如果第  $k$  只蚂蚁在时刻  $t$  到  $t+1$  之间经过城市  $j$  但不经过城市  $l$ 。也就是说，在原来的 Ant-Quantity 模型中，每只蚂蚁每走一步只改变其刚好经过的那段路径上的信息量，但在基于信息素扩散的蚁群算法模型中，该蚂蚁可能会改变多条路径上的信息量。这种改进可大大提高蚂蚁群体之间的合作效果，增强了蚁群算法的有效性，同时改进后的算法模型也更加忠实于自然界中真实的蚁群系统。

下面来推导  $D_{il}^k$  和  $D_{jl}^k$  的计算公式：令  $h = \bar{d}^{\omega+1}/(d_{ij})^\omega$ ，其中  $\omega$  为大于 1 的可调常数， $d$  为各城市间的平均距离。令  $D_{\max} = \gamma \cdot \Delta\tau_{ij}^k$ ，其中  $\gamma$  是小于 1 的可调常数。当以  $i$  为中心向外扩散信息素时，公式 (5.4.1) 中的  $\sigma = d_{il}$ ；而以  $j$  为中心

向外扩散信息素时  $\sigma = d_{jl}$ 。将  $\Delta\tau_{ij}^k$ 、 $h$ 、 $D_{\max}$  和  $\sigma$  代入公式 (5.4.1)，可得  $D_{il}^k$  和  $D_{jl}^k$  分别如公式 (5.4.5) 和式 (5.4.6) 所示

$$D_{il}^k = \begin{cases} \gamma \cdot Q/d_{ij} \left( 1 - \frac{d_{il} \cdot (d_{il})^\omega}{\bar{d}^{\omega+1}} \operatorname{ctan}\theta \right), & \text{若 } d_{il} < \bar{d}^{\omega+1}/(d_{ij})^\omega \cdot \tan\theta \\ 0, & \text{否则} \end{cases} \quad (5.4.5)$$

$$D_{jl}^k = \begin{cases} \gamma \cdot Q/d_{ij} \left( 1 - \frac{d_{jl} \cdot (d_{jl})^\omega}{\bar{d}^{\omega+1}} \operatorname{ctan}\theta \right), & \text{若 } d_{jl} < \bar{d}^{\omega+1}/(d_{ij})^\omega \cdot \tan\theta \\ 0, & \text{否则} \end{cases} \quad (5.4.6)$$

### 5.4.2.3 基于信息素扩散的蚁群算法流程

基于信息素扩散的蚁群算法流程可描述如下：

- (1) 初始化。
- (2) 随机选择每只蚂蚁的初始位置。
- (3) 按状态转移概率公式计算每只蚂蚁  $k$  将要转移的位置，可将其假设为  $j$ ，而将上一位置假设为  $i$ 。
- (4) 按公式 (5.4.2) 计算在路径  $(i, j)$  上新产生的信息量。
- (5) 按公式 (5.4.3) 计算从  $i$  扩散到其他路径  $(i, l)$  上的信息量。
- (6) 按公式 (5.4.4) 计算从  $j$  扩散到其他路径  $(j, l)$  上的信息量。
- (7) 若本次循环中每只蚂蚁都执行了第 (3) ~ (6) 步，则跳转到第 (8) 步，否则跳转到步骤 (3)。
- (8) 按信息素更新公式更新各条路径上的信息量。
- (9) 如果每只蚂蚁都遍历完所有路径，则跳转到第 (10) 步，否则跳转到第 (3) 步。
- (10) 如果满足算法结束条件，则跳转到第 (11) 步，否则跳转到第 (3) 步并开始新一轮的迭代。
- (11) 输出计算结果。

### 5.4.3 仿真算例

这里选用 TSPLIB 中的 Oliver30TSP 作为仿真算例。基于信息素扩散的蚁群算法在不同参数下的实验结果如表 5.5 所示。

图 5.17 是基于信息素扩散的蚁群算法求解 Oliver30TSP 时的最优解进化曲线。

表 5.5 基于信息素扩散的蚁群算法在不同参数下的实验结果

$\alpha$	$\beta$	$\rho$	$\omega$	$\gamma$	最短路径长度	算法收敛时所需的迭代次数
2	4	0.6	1	0.4	423.740563	76
2	4	0.9	1	0.4	423.911688	59
4	2	0.5	1	0.6	424.464270	64
4	2	0.9	1	0.6	424.464270	68

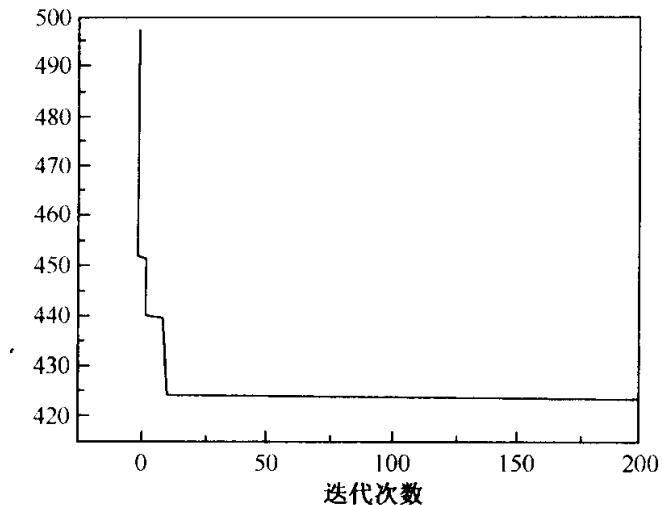


图 5.17 最优解的进化曲线

图 5.18 是基于信息素扩散的蚁群算法的解的多样性曲线，其多样性用公式 (5.4.7) 进行计算

$$DIV(n) = \sqrt{\frac{\sum_{k=1}^N (L_k(n) - \text{avg}(L(n))^2)}{N-1}} \quad (5.4.7)$$

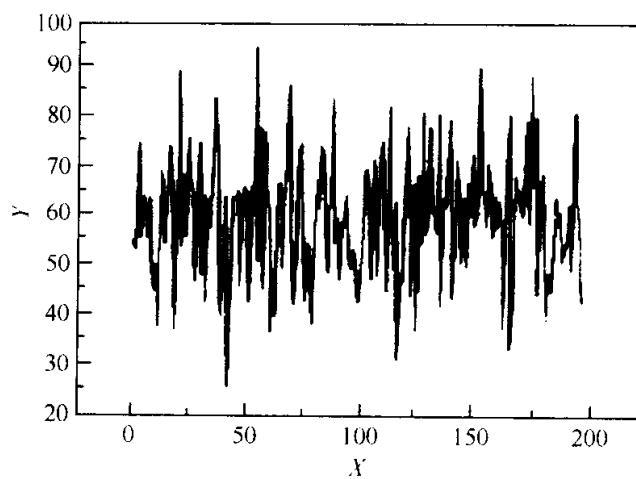


图 5.18 解的多样性曲线

式中,  $N$  表示蚂蚁数目, 此处取  $N=30$ ;  $L_k(n)$  表示第  $k$  只蚂蚁在第  $n$  次迭代所经过路径的长度;  $\text{avg}(L(n))$  则表示第  $n$  次迭代所有蚂蚁所经路径长度的平均值。

由图 5.18 可见, 在整个进化过程中, 解的多样性一直比较好, 因此基于信息素扩散的蚁群算法具有不断获得新的最优解的能力, 使得改进蚁群算法可获得全局最优解, 而不易陷入局部最优解。图 5.19 是基于信息素扩散的蚁群算法所找到的最优路径状态。

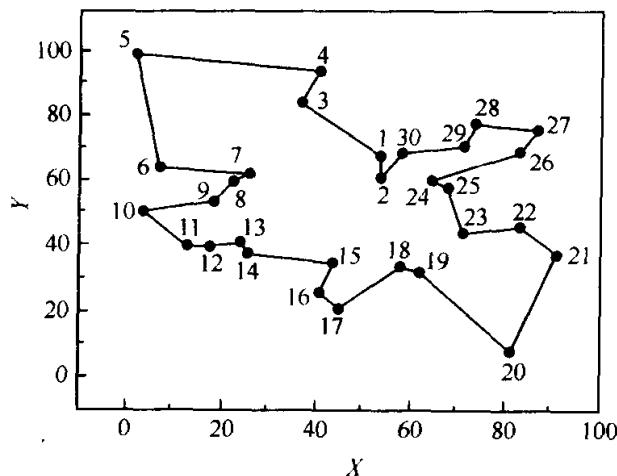


图 5.19 所找到的最优路径状态

## 5.5 多态蚁群算法

不论是 Dorigo M 提出的基本蚁群算法, 还是后来绝大部分学者所提出的改进蚁群算法, 都是基于单种蚁群、单种信息素的算法, 主要模拟了实际蚁群信息系统的一部分。而实际上, 真实蚁群社会中的蚁群是有组织、有分工的, 不同种类的蚁群有不同的信息素调控机制, 这种分工组织方式对蚁群完成复杂任务具有十分重要的作用。本节研究了一种新的含多种蚁群、多种信息素的多态蚁群算法 (polymorphic ant colony algorithm, PACA)。该算法引入了不同种类的蚁群, 每一蚁群都有不同的信息素调控机制, 并将局域搜索与全局搜索相结合<sup>[23]</sup>, 从而更符合蚁群的真实信息处理机制, 使算法的收敛速度得到大幅度提高。

### 5.5.1 蚁群信息系统的多态性

对蚁群社会的研究表明<sup>[24]</sup>, 真实蚁群社会中的所有蚂蚁各司其职, 又相互依赖协作, 形成一个有机整体。在执行某项任务时, 个体之间通过各自分泌的信息素相互联系。这里的“多态性”指蚁群社会所具有的多种状态的蚁群及信

息素。

### 5.5.2 现有蚁群算法多态性方面的不足

蚁群算法本质上是一种随机搜索算法，它是通过对候选解组成的群体的进化来搜索全局最优解。从多态性角度分析，基本蚁群算法主要存在如下不足：

(1) 单种蚁群、单种信息素没有考虑蚂蚁群的种类差异和动作差异，使所有蚂蚁都做同一种任务，不能完全反映真实蚁群社会的复杂性。如能在现有蚁群算法中，引入多种蚁群，不同蚁群的信息素调控不同，就能在蚂蚁搜索过程中，针对各具体路径选择合适的信息量，这无疑会加快寻优收敛速度。

(2) 在基本蚁群算法中，第  $k$  只蚂蚁第  $x$  步由城市  $i$  选择下一城市时，需按概率  $p_{ij}^k$  ( $j \in \text{tabu}_k$ ) 在剩下的  $m-x$  个城市中选择一个，多数情况下其搜索子空间的规模比较大。若能引入多种蚁群，则每种蚁群只需在自己的搜索空间内查找，这不仅大大缩减了搜索子空间的规模，而且选择效率也较高，同时也易于引入一些较好的局部搜索方法，如我们在实验中借鉴了文献 [25] 的结论，只需在剩下的  $\min\{m-x, \text{MAXPC}\}$  个城市中选择一个。这里 MAXPC 为最大可选城市数，MAXPC 的确定如表 5.6 所示，其数据来源于文献 [25]， $PC_i$  是以最优解中每一城市  $C_i$  为中心，作半径  $R$  的圆， $R$  从 0 不断扩大，直至取得  $C_i$  的近邻城市为止时所记录下圆内的城市数；MAXPC 为  $PC_i$  的最大值。

表 5.6 最大可选城市数统计结果

城市规模	MAXPC	$PC_i = 1$ 的城市数
20	8	11
100	9	55
144	13	61
1000	19	545

由表 5.6 可知，当城市规模  $\leq 100$  时， $\text{MAXPC} < 10$ ；当  $100 < \text{城市规模} \leq 1000$  时， $\text{MAXPC} < 20$ ；且此时  $PC_i = 1$  的概率基本上为 50% 以上。对于每个城市，如果从离其最近的 MAXPC 个城市中选择一个作为近邻城市，而不是从剩下城市中任选一个，也必然可以找到最优解。

(3) 在 Ant-Cycle 模型中

$$\Delta \tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{若第 } k \text{ 只蚂蚁在时刻 } t \text{ 和 } t+1 \text{ 之间经过 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (5.5.1)$$

蚂蚁在所有经过的路径  $(i, j)$  上都留下了信息素，增加了该路径上信息量，而路

径上信息量的增量计算是 PACA 的关键。如果只在可能是最优解组成部分（根据 MAXPC 判断）的路径上，增加适当信息量将会使算法的收敛速度大有提高，而这就需要使不同蚁群具有不同的信息素调控机制。

### 5.5.3 算法设计

在 PACA 中，将蚁群社会中从事劳动的蚂蚁定义为三类：侦察蚁、搜索蚁和工蚁。侦察蚁群的任务是以每个城市为中心作局域侦察，并以侦察素来标记侦察结果，以便为搜索蚁到达该城市后选择下一个城市时提供辅助信息；搜索蚁群的任务是进行全局搜索，每到一个城市，根据侦察素和各出口的信息素等信息来选择下一个城市，直至找到最佳路径为止，并将其标记，以便工蚁从最佳路径取食回巢；工蚁群的任务是从已标记好的最佳路径取食回巢。由于这里所定义的工蚁群与路径寻优无关，所以只需针对侦察蚁群和搜索蚁群设计各自的信息素调控机制。其中，侦察蚁群负责局部侦察，搜索蚁群负责全局搜索。

针对侦察蚁群的做法是：将  $m$  只侦察蚁分别放置在  $m$  个城市中，每只侦察蚁以所在城市为中心，侦察其他  $m-1$  个城市，并将侦察结果与已有的先验知识相结合，构成侦察素，并将其记为  $s[i][j]$ ，标记在从城市  $i$  到城市  $j$  的路径上。 $s[i][j](i, j=0, 1, 2, \dots, m-1; i \neq j)$  可通过下式计算

$$s[i][j] = \begin{cases} \frac{\tilde{d}_{ij}}{d_{ij}}, & \text{若城市 } j \text{ 在城市 } i \text{ 的 MAXPC 之内} \\ 0, & \text{否则} \end{cases} \quad (5.5.2)$$

式中， $\tilde{d}_{ij}$  表示以城市  $i$  为中心到其他  $m-1$  个城市的最小距离。并根据此结果，首先设置初始时刻各条路径上的信息量如下

$$\tau_{ij}(0) = \begin{cases} C \cdot s[i][j], & \text{若 } s[i][j] \neq 0 \\ \frac{C \cdot \tilde{d}_{ij}}{\hat{d}_{ij}}, & \text{否则} \end{cases} \quad (5.5.3)$$

式中， $\hat{d}_{ij}$  表示以城市  $i$  为中心到其他  $m-1$  个城市的最大距离； $C$  表示初始时刻各条路径上的信息量。

其次，该侦察素可为搜索蚁群在进行状态转移概率  $p_{ij}^k$  的计算、各路径上的信息量调整时提供辅助。

针对搜索蚁群的做法是：蚂蚁  $k(k=1, 2, \dots, n)$  在运动过程中的  $t$  时刻，从城市  $i$  转移到城市  $j$  的状态转移概率  $p_{ij}^k$  可按下式计算

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{s \notin \text{tabu}_k} [\tau_{is}(t)]^\alpha [\eta_{is}(t)]^\beta}, & \text{若 } j \notin \text{tabu}_k \text{ 且 } s[i][j] \neq 0 \\ 0, & \text{否则} \end{cases} \quad (5.5.4)$$

所有蚂蚁完成一次循环，各路径上信息量需要根据下式做调整

$$\tau_{ij}(t+1) = \begin{cases} \rho \cdot \tau_{ij}(t) + (1-\rho) \cdot \Delta\tau_{ij}, & \text{若 } s[i][j] \neq 0 \\ \rho \cdot \tau_{ij}(t), & \text{否则} \end{cases} \quad (5.5.5)$$

式中， $\Delta\tau_{ij}$  表示本次循环所有蚂蚁在路径  $(i, j)$  上所释放的信息量之和，且  $\Delta\tau_{ij} = \sum_{k=1}^n \Delta\tau_{ij}^k$ 。 $\Delta\tau_{ij}^k$  表示第  $k$  只蚂蚁在本次循环中留在路径  $(i, j)$  上的信息量，其值可由下式确定

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q \cdot (\tilde{d}_{ij}/d_{ij})}{L_k}, & \text{若蚂蚁 } k \text{ 经过 } (i, j), \text{ 且 } s[i][j] \neq 0 \\ 0, & \text{否则} \end{cases} \quad (5.5.6)$$

式中，每只搜索蚁根据侦察素只在可能是最优解组成部分的路径上留下适量的信息素（局部信息  $\tilde{d}_{ij}/d_{ij}$  与全局信息  $L_k$  相结合）；根据公式（5.5.4），每只搜索蚁只在可能是最优解组成部分（按  $s[i][j]$  是否为 0 决定）的路径上追加信息量。

由此，PACA 的流程可描述如下：

- (1) 初始化  $Q$ 、 $C$  和最大迭代次数。
- (2) 将  $m$  只侦察蚁分别放在  $m$  个城市中，每只侦察蚁以所在城市  $i$  为中心，侦察其他  $m-1$  个城市，按公式（5.5.2）计算侦察素，并将结果放入  $s[i][j]$  中。
- (3) 按公式（5.5.3）置初始时刻各条路径上的信息量。
- (4) 置迭代次数  $N_c$  初值为 0。
- (5) 随机选择每只搜索蚁的初始位置，并将该位置放入每只搜索蚁对应的  $\text{tabu}_k$  表中。
- (6) 按公式（5.5.4）计算每只搜索蚁  $k$  将要转移的位置，假设为  $j$ ，假设上一个位置为  $i$ ，并将  $j$  放入搜索蚁  $k$  对应的  $\text{tabu}_k$  表中，直至每只搜索蚁完成一个循环。
- (7) 计算各搜索蚁的目标函数值  $L_k$  ( $k=1, 2, \dots, n$ )，并记录当前最好解。
- (8) 若达到指定的迭代次数或者所求得的解在最近若干代中无明显改进，则跳转到第(11)步；否则跳转到第(9)步。
- (9) 按公式（5.5.5）更新各路径上的信息量。
- (10) 置  $\Delta\tau_{ij}$  为 0，置  $\text{tabu}_k$  表为空， $N_c \leftarrow N_c + 1$ ，跳转到第(5)步。
- (11) 输出最优解。

#### 5.5.4 仿真算例

选用 TSPLIB 中的 Oliver30TSP 作为仿真算例，分别用基本蚁群算法和 PACA 进行仿真实验。实验中，设置  $Q=100$ ,  $C=3$ ,  $\alpha=1$ ,  $\beta=3$ ,  $\rho=0.3$ , MAXPC=10。实验结果如表 5.7 和图 5.20 所示。表 5.7 中最短路径长度为最优

路径的长度，迭代次数是取 10 次实验的平均而得到的；图 5.20 是 PACA 所找到的最优路径状态。

表 5.7 ACA、PACA 的实验结果及比较

最短路径长度	迭代次数 (PACA)	迭代次数 (ACA)	时间比 (PACA/ACA)
424.635376	27	52	1 : 3
424.464270	41	126	1 : 4
423.911743	116	362	1 : 4
423.740601	137	>400	1 : ?

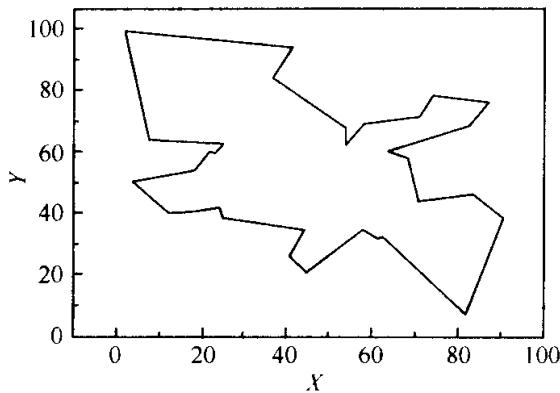


图 5.20 最优路径 (423.740601) 状态

多次仿真实验可知，用 PACA 平均 137 次迭代就能搜索到目前最好解 (423.740601)，而用基本蚁群算法在 400 次迭代内也得不到此解。就次优解而言，在取得相同结果的情况下，PACA 所需迭代次数比基本蚁群算法少得多。

## 5.6 基于模式学习的小窗口蚁群算法

通过对蚁群算法的研究发现，蚁群这个自适应系统与其他许多复杂自适应系统一样存在着模式，蚁群就是通过信息素来记忆、更换和组合这些模式以搜索最优路径的。在用蚁群算法求解 TSP 时，可通过限定蚂蚁每步移动的城市数目来降低解空间的规模，以提高解的质量，在此基础上提取模式，并将这些模式看做一个个整体、一个个“积木块”，然后再在这些“积木块”中寻找最优解。这种基于模式学习的小窗口蚁群算法思路是将基本蚁群算法所求解的问题放在较粗的粒度上进行计算，将一个高维空间转化为一个低维空间，充分利用了蚁群算法在低维空间具有较好的搜索性能这一特点<sup>[26~28]</sup>，从而在求解大规模组合优化问题

时，使蚁群算法的全局收敛性能有了很大提高。

### 5.6.1 蚁群算法中的模式学习

在  $n$  个城市的 TSP 中，TSP 的一个解可表示为一个循环排列  $V = (v_1, v_2, \dots, v_n, v_1)$ ，该循环排列表示  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n \rightarrow v_1$  的一条路径，其中  $v_i$  表示该路径上的第  $i$  个城市。设  $\Lambda$  为该 TSP 的一个解集，则有  $V \in \Lambda$ ，因此求解 TSP 即为在  $\Lambda$  中寻找一个最优解  $V^*$ 。

将解集  $\Lambda$  中两个任意解  $V_i, V_j$  的交集定义为  $H_{ij} = V_i \cap V_j$ ，其中  $H_{ij}$  表示  $V_i, V_j$  中相同排列的子集。不失一般性，设  $V_i, V_j$  分别表示为  $V_i = (\dots, v_k, \dots, v_m, \dots, v_w, \dots, v_u, \dots)$  和  $V_j = (\dots, v_w, \dots, v_u, \dots, v_k, \dots, v_m, \dots)$ ，其中  $V_i, V_j$  中的  $(v_m, \dots, v_u), (v_k, \dots, v_m)$  分别表示城市规模和排列顺序相同的子集。令  $h_{ij1} = (v_w, \dots, v_u)$ ,  $h_{ij2} = (v_k, \dots, v_m)$ ，则  $H_{ij} = (h_{ij1}, h_{ij2})$ 。如果  $V_i, V_j$  中有多个相同的子集，则  $H_{ij} = (h_{ij1}, h_{ij2}, \dots, h_{ijl})^\top$ ，其中  $1 \leq l \leq n/2$ ；若解  $V_i, V_j$  无相同的子集，则  $H_{ij} = \Phi$ 。

这里引入模式概念，将  $H_{ij}$  中的  $h_{ij1}, h_{ij2}, \dots, h_{ijl}$  称为模式，并将  $h_{ij1}, h_{ij2}, \dots, h_{ijl}$  看成一个个“积木块”。蚂蚁遍历完所有城市后会对最短的路径添加信息素，使得该路径上“好的模式”有更高的概率在下一次遍历中被选中。而在第二次遍历中，选中“好的模式”的路径也更有可能发现更短的路径。这样周而复始，通过信息素的正反馈，使“好的模式”不断被强化，而通过信息素的挥发丢弃那些“不好的模式”，这样不断缩小搜索空间，积累“好的模式”，通过“好的模式”拼搭结合，从而找到全局更优解。

模式的获取是通过不同次优解之间的交集实现的。TSP 解的表现形式是一组边的排列，将不同解中相同的局部排列片段提取出来作为待选模式，具体过程如图 5.21~图 5.23 所示。图 5.21 和图 5.22 是两个不同的解，但两个虚线圈内的局部排列是相同的。两个解的交集就是两个解中相同的部分，所以两个虚线圈内的部分是这两个解的交集。将这两个虚线圈内的路径看作一个整体，从原路径中删除，则原路径解 1 最终变成图 5.23，城市数目减少了。算法中增加的集合交集运算是线性的，因此不会对算法的计算时间产生较大的影响。然而，这样得到的模式未必就是全局最优解中的模式，但这些模式很可能出现在全局最优解中。

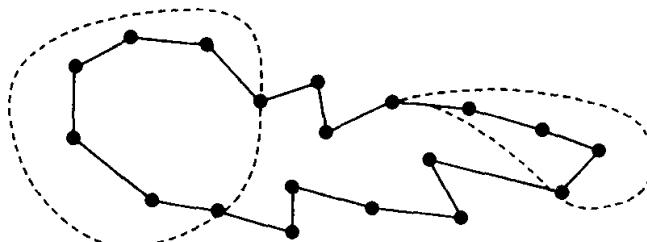


图 5.21 原路径解 1

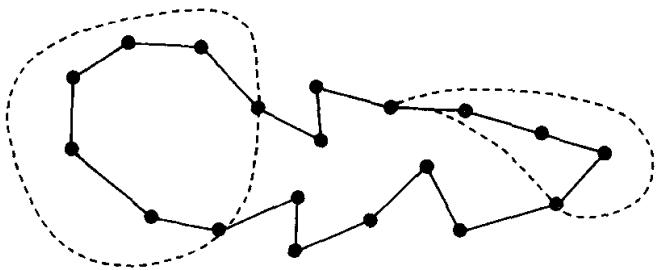


图 5.22 原路径解 2

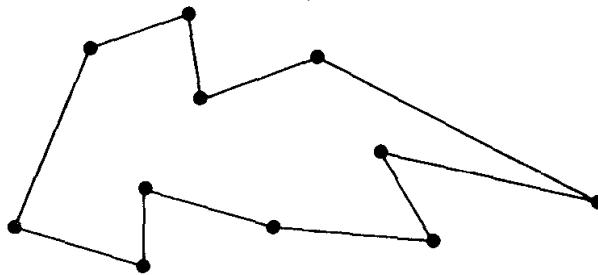


图 5.23 删去相同部分后的新解

因此，要在计算过程中对得到的模式不断进行调整改进，以期得到最优模式。

## 5.6.2 算法设计

在对 TSP 研究的过程中发现，TSP 解空间中绝大部分是劣质解，而且干扰着对优质解的寻找。如果能剔除这些劣质解，不但可以缩小解空间的搜索范围，而且还可减少劣质解的干扰，进而迅速发现全局最优解。

这里以 TSPLIB 中的 Oliver30TSP 为例来说明基于模式学习的小窗口蚁群算法原理，图 5.24 给出了 Oliver30TSP 的最优路径状态。由图 5.24 可见，与每个城市相连的另外两个城市是与该城市最近的几个城市中的两个，而绝不会出现如

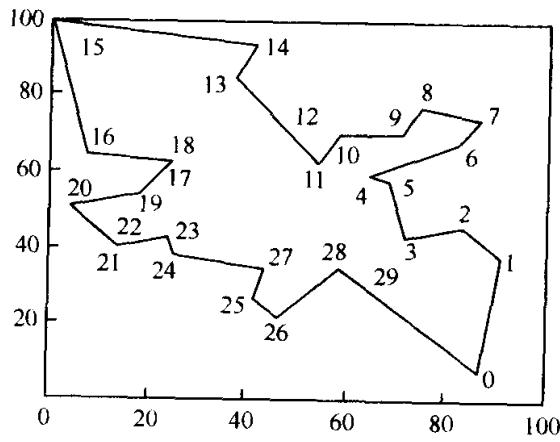


图 5.24 Oliver30TSP 的最优路径

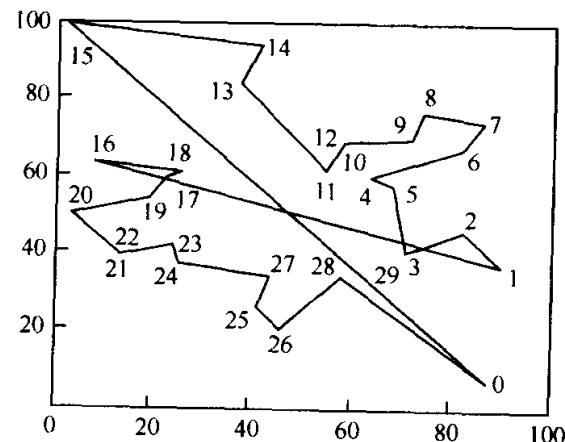


图 5.25 Oliver30TSP 的非最优路径

图 5.25 所示的最优解。更一般地说，在一个规模为  $N$  的 TSP 中，任何一个城市均有  $N-1$  个从该城市出发的路径，而在这  $N-1$  条路径中，只有最短几条路径中的一条才可能是组成最优解的路径之一，其他的较长路径不可能是最优解中的一条路径，因此可通过限定蚂蚁每次移动范围来达到剔除劣质解、缩小搜索范围的目的。

在算法中可为每个城市建一个数组  $\text{cityWin}[\text{window}]$ ，保存  $\text{window}$  个距离最近的城市。其中  $\text{window}$  表示窗口的大小。蚂蚁在窗口大小的选择上是根据 TSP 的规模而定的，规模越大，则窗口越大。蚂蚁每次移动就从  $\text{cityWin}[\text{window}]$  和  $\text{tabu}[k]$  的交集中选择移动的城市，并根据状态转移概率移动到下一个城市。若交集为空，则只在  $\text{tabu}[k]$  中选择。此外，该改进算法还引入了概率  $p_m$ ，以概率  $p_m$  用全局最优值更新信息素，其余以当前最优值更新信息素，并采用 2-opt 局部优化方法提高求解效率。

小窗口蚁群算法的具体步骤如下：

- (1) 初始化参数： $\alpha, \beta, \rho, Q, \tau_{\max}, \tau_{\min}, p_m, \text{window}, N_{c_{\max}}, \text{tabu}[k], \text{cityWin}[m]$ 。
- (2) 迭代次数  $N_c$ ++。
- (3) 一组蚂蚁开始循环。
- (4) 蚂蚁从  $\text{cityWin}[\text{window}]$  和  $\text{tabu}[k]$  中选择允许到达的城市，根据状态转移概率公式移动。
- (5) 若  $\text{tabu}[k]$  未满，则跳转到第 (4) 步继续。
- (6) 2-opt 局部优化路径。
- (7) 记录本次最短路径，置空  $\text{tabu}[k]$ 。
- (8) 根据  $p_m$ ，选择全局最优值或当前最优值，更新信息素。
- (9) 若  $N_c$  小于  $N_{c_{\max}}$ ，则跳转到第 (2) 步，否则，跳转到第 (10) 步。
- (10) 输出最优结果。

基于模式学习的蚁群算法是建立在小窗口蚁群算法基础之上的，模式学习蚁群算法采用主从式。在各个节点上运行 MMAS+2opt 算法，当其运行一段时间后，将得到的最好解传送给主程序。主程序从接收到的解中选出最好的几个解，并得到多个模式；然后将几个不同的模式传至各节点，MMAS+2opt 根据得到的模式在一个较粗的粒度上运行；运行一段时间后，再将最好的解传至主程序，用同样方法获取模式，并传至各节点；如此反复，直到传至主程序的最好结果不再改进为止。当结果不再改进时，表明已得不到更好的模式，必须对现有模式进行处理。由于模式是不断积累的，随着计算的进行，模式不断由小“积木块”聚合成大“积木块”。模式使计算粒度变粗，计算速度会逐渐加快，但却使解得到改进的可能性越来越小。因此，要将大“积木块”重新打碎成小“积木块”，让其再次重新组合成大“积木块”，以期得到更好的解；将模式调整后再传回各节点进行计算，直到满足终止条件。

在小窗口蚁群算法得到较好解的基础上，通过提取解中的模式，将一个个模式看做整体，以降低实际计算中城市的数目，并利用蚁群算法在低维空间具有较好的搜索性能这一特点，以提高算法的运行效率。首先，独立运行两次小窗口蚁群算法，得到两个较好的解  $V_i$ 、 $V_j$ ，比较这两个解，并将这两个解的交集  $H_{ij} = (h_{ij1}, h_{ij2}, \dots, h_{ijl})^T$  保存在一个二维数组  $\text{model}[\cdot][\cdot]$  中，将  $h_{ij1}, h_{ij2}, \dots, h_{ijl}$  看做一个整体，每个模式看做一个节点（城市）， $l$  个模式看做  $l$  个节点。在完成一次运算后，将得到的最优解和  $V_i$ 、 $V_j$  进行比较，再提取其中较好的两个模式重新进行计算，直到满足算法终止条件为止。

基于模式学习的蚁群算法的主要步骤如下：

- (1) 用 LWMMAS() 保存  $V_i$  和  $V_j$ 。
- (2) getModel[ $V_i$ ,  $V_j$ ]，将交集保存到二维数组  $\text{model}[\cdot][\cdot]$ 。
- (3) changeStructure(model)，改变城市数目。
- (4) 用 LWMMAS() 保存  $V_k$ 。
- (5) 将目前最好的两个解保存到  $V_i$  和  $V_j$ 。
- (6) 若不满足算法终止条件，则跳转到第 (3) 步。
- (7) 输出最优结果。

### 5.6.3 仿真算例

以 CHN144 为例，设置  $\alpha=4$ ,  $\beta=1$ ,  $Q=\tau_{\max}=1$ ,  $\tau_{\min}=0.000024$ ,  $\text{window}=5$ ,  $\rho=0.8$ ,  $m=n=144$ 。

已知 CHN144 的最好结果是 30380km，如图 5.26 (a) 所示；采用基于模式学习的小窗口蚁群算法得到的结果为 30349km，如图 5.26 (b) 所示。

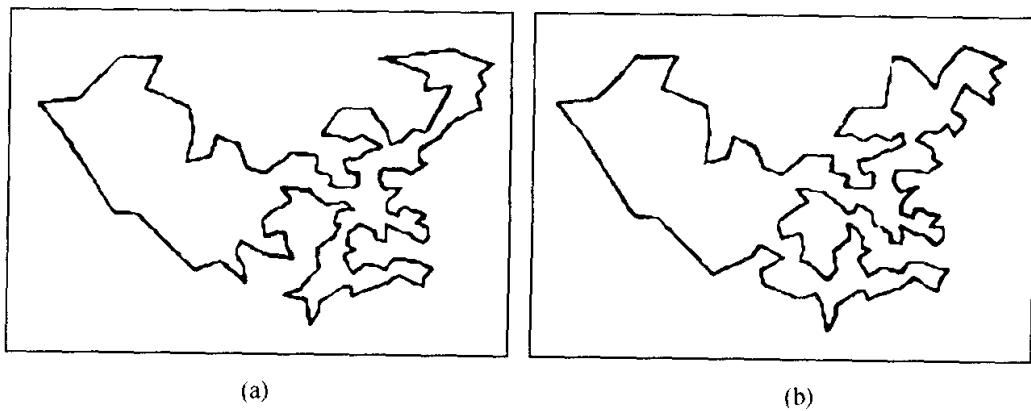


图 5.26 CHN144 实验结果

由图 5.26 可见，小窗口蚁群算法通过限定计算空间可在一定程度上改善解的质量，为基于模式学习的蚁群算法提供了较好的初始解。而基于模式学习的蚁

群算法在提取模式的基础上改变计算粒度，使改进后蚁群算法的求解性能有了一定的提高。

## 5.7 基于混合行为的蚁群算法

在利用基本蚁群算法求解优化问题时，要求使算法的搜索空间尽可能大，以寻找那些可能存在最优解的解空间，同时也要求充分利用有效的历史知识<sup>[29]</sup>，从而使蚁群算法以更大的概率收敛到全局最优解。

蚁群算法在寻优过程中，很大程度上受早期发现的较好解的影响，这些较好解以极大的概率引导蚁群走向局部最优解。为在加快蚁群算法收敛速度的同时又能避免停滞现象，本节研究了一种基于混合行为的蚁群算法（hybrid behavior based ant colony algorithm, HBACA）<sup>[30]</sup>。

### 5.7.1 算法设计

HBACA 采用具有不同行为特征的蚂蚁相互协作来发现所求问题的全局优化解，该算法模型由 4 部分组成，具体如图 5.27 所示。其中①表示蚂蚁，每只蚂蚁拥有自己的行为特征，该行为由规则集（rule set）④储存在环境（environment）③中的知识以及状态空间（state space）②一起决定。状态空间与待求解问题具有一一映射关系，蚂蚁在状态空间中移动，逐步构造出所求问题的可行解。图 5.27 中蚂蚁间的虚箭头表示蚂蚁之间不能进行直接通信，而是在移动过程中通过环境间接通信。蚂蚁利用通道（channel）读取写入信息到环境，环境储存了蚁群过去行动中所获取的历史知识。

**定义 5.7.1 蚂蚁行为** 蚂蚁在前进过程中，用以决定其下一步移动到哪一个状态的规则集合。通常情况下，蚂蚁行为可表示为  $\text{Action} = \{\Omega_i \mid i = 1, 2, \dots, n\}$ ，其中， $\Omega_i$  表示影响蚂蚁行为的第  $i$  个因素。

由于 HBACA 中影响蚂蚁行为的因素有规则集、环境、状态空间等，针对待求解的 TSP，可由定义 5.7.1 定义如下四种类型的蚂蚁行为：

行为 1 (Action 1)：蚂蚁以随机方式选择下一个要到达的城市。

行为 2 (Action 2)：蚂蚁以贪婪方式选择下一个要到达的城市，即

$$p_{ij}^k = \begin{cases} \frac{\eta_{ij}}{\sum_{s \in J_k(i)} \eta_{is}}, & \text{若 } j \in J_k(i) \\ 0, & \text{否则} \end{cases} \quad (5.7.1)$$

行为 3 (Action 3)：蚂蚁按下式选择下一个要到达的城市

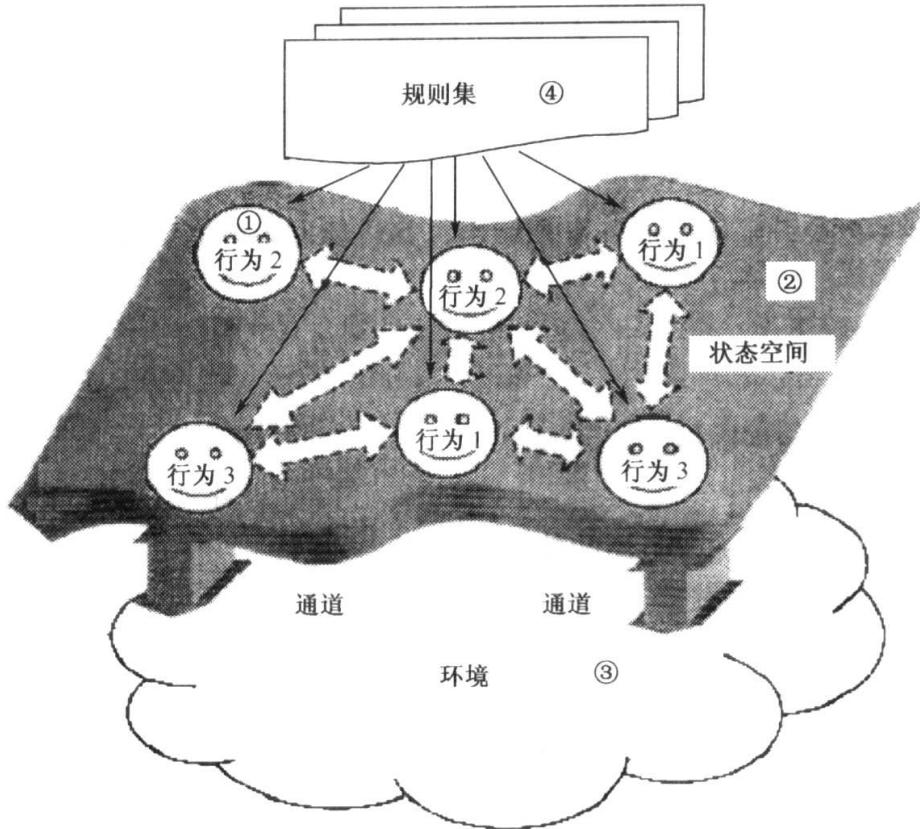


图 5.27 HBACA 的模型

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}}{\sum\limits_{s \in J_k(i)} \tau_{is}}, & \text{若 } j \in J_k(i) \\ 0, & \text{否则} \end{cases} \quad (5.7.2)$$

行为4(Action 4): 蚂蚁按下式选择下一个要到达的状态

$$j = \operatorname{argmax}(\tau_{is} \eta_{is}), \quad s \in J_k(i) \quad (5.7.3)$$

当所有蚂蚁完成解的构造之后，计算本次迭代的最优解；然后，将本次迭代的最优解与当前最优解进行比较，如果本次迭代的最优解小于当前最优解，则用本次迭代的最优解替换当前最优解；随后，将所有弧段上的信息量按下式进行更新

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \frac{Q}{L^{gb}} \quad (5.7.4)$$

式中， $L^{gb}$  表示当前最优解。由于 HBACA 中共定义了四种蚂蚁行为，因此可将蚁群按蚂蚁行为分成四个子蚁群，每个子蚁群中的蚂蚁具有相同的行为特征。设各子蚁群中的蚂蚁数目比例为  $r_1 : r_2 : r_3 : r_4$ 。

HBACA 的具体步骤可描述如下：

- (1) 初始化 HBACA 的参数  $r_1 : r_2 : r_3 : r_4, \rho, Q, N_{c_{\max}}, \tau_{ij}(0)$ 。

- (2) 生成  $m$  只蚂蚁，其中  $\frac{mr_1}{r_1+r_2+r_3+r_4}$  只蚂蚁按 Action 1 行动，另外  $\frac{mr_2}{r_1+r_2+r_3+r_4}$  只、 $\frac{mr_3}{r_1+r_2+r_3+r_4}$  只、 $\frac{mr_4}{r_1+r_2+r_3+r_4}$  只蚂蚁分别以 Action 2、Action 3、Action 4 行动。
- (3) 将 4 个蚁群随机地放到  $n$  个城市，并将该城市的索引添加到该蚂蚁的禁忌表中。
- (4) For 每只蚂蚁 do
- Repeat
- 按自己的行为规则选择下一城市；
- 移动到下一城市；
- 将该城市的索引加入自己的禁忌表
- Until 不能再向前移动
- End for
- (5) 计算每只蚂蚁的路径长度并找出本次迭代的最优解，如果本次迭代的最优解比当前最优解更好，则用本次迭代的最优解替代当前最优解。
- (6) 由公式 (5.7.4) 更新弧段上的信息量。
- (7) 置  $N_c = N_c + 1$ ；
- If  $N_c > N_{c_{\max}}$  then
- 输出最优解；
- 退出算法
- else
- 清空所有蚂蚁的禁忌表；
- 转第 (4) 步
- End if

### 5.7.2 仿真算例

这里以 TSPLIB 中的 Eil51TSP 作为仿真算例，实验分别针对挥发系数  $\rho$  和不同行为群体的比例  $r_1 : r_2 : r_3 : r_4$  进行。首先在固定挥发系数  $\rho$  的情况下，采用不同群体比例  $r_1 : r_2 : r_3 : r_4$  求解 Eil51TSP，其实验结果如表 5.8 所示。

由表 5.8 可见，蚁群中  $r_1$ （随机搜索）所占比例不能太大，否则对算法的搜索不利；另外，启发式因子在算法后期所起的作用减弱， $r_2$  也不应该太大。当  $r_1 : r_2 : r_3 : r_4$  取  $0.05 : 0.1 : 0.4 : 0.45$  时，HBACA 具有较好的求解性能。表 5.9 给出了信息素挥发系数  $\rho$  对 HBACA 的影响。

表 5.8 参数  $r_1 : r_2 : r_3 : r_4$  对 HBACA 的影响

$r_1 : r_2 : r_3 : r_4$	最好解	最差解	平均值	标准方差
0.05 : 0.1 : 0.25 : 0.6	426	436	429.263	1.950
0.05 : 0.1 : 0.4 : 0.45	426	432	428.731	1.259
0.05 : 0.2 : 0.5 : 0.25	427	441	431.379	2.826
0.1 : 0.1 : 0.2 : 0.6	427	443	433.175	3.084
0.1 : 0.1 : 0.4 : 0.4	428	442	432.943	3.183
0.1 : 0.2 : 0.4 : 0.3	428	443	432.691	3.329
0.2 : 0.1 : 0.2 : 0.5	434	452	446.385	6.845
0.2 : 0.2 : 0.3 : 0.3	432	449	448.578	7.073
0.2 : 0.3 : 0.3 : 0.2	436	472	451.658	11.546

表 5.9 信息素挥发系数  $\rho$  对 HBACA 的影响

$\rho$	最好解	最差解	平均值	标准方差
0.02	426	439	429.316	1.831
0.05	426	432	427.931	1.173
0.1	428	446	432.732	2.332
0.2	428	451	436.427	7.084
0.3	431	459	438.372	8.578
0.4	431	455	437.691	9.839
0.5	432	461	442.462	11.364
0.6	434	489	444.537	16.456
0.7	435	472	441.473	12.463
0.8	439	469	446.578	13.863

由表 5.9 可见, 当  $\rho=0.02$  及  $\rho=0.05$  时, HBACA 的求解性能较好。

下面利用 HBACA 的最优参数, 通过不同规模的 TSP 比较 HBACA 与基本蚁群算法的性能。其中基本蚁群算法的参数设置为  $\alpha=1.5$ ,  $\beta=4$ ,  $\rho=0.5$ ,  $Q=100$ ; HBACA 的参数为  $\rho=0.05$ ,  $r_1 : r_2 : r_3 : r_4 = 0.05 : 0.1 : 0.4 : 0.45$ ,  $Q=100$ , 两种算法的蚂蚁数均为 35。两种算法都运行 15 次后取其平均值, 表 5.10 给出了 HBACA 与基本蚁群算法的运行结果。

表 5.10 HBACA 与基本蚁群算法的比较

TSP 实例	最好解		最差解		平均值		标准方差	
	HBACA	ACA	HBACA	ACA	HBACA	ACA	HBACA	ACA
E151	428.1	434	431.2	439.3	433	456	3.45	9.24
Eil75	535	546	539.2	556.8	545	569	4.61	12.17
D198	15974	16608	16032.9	16832.3	16156	17123	79.43	249.73
KroA100	21282	21429	21453.7	21980.4	21763	22848	213.69	692.92
Att532	28131	29064	28293.5	30071.6	28512	31927	192.36	1212.67

从表 5.10 中的最优解和平均值可以看出, HBACA 明显优于基本蚁群算法, 标准方差也显示 HBACA 比基本蚁群算法具有更加稳定的求解性能。

## 5.8 带聚类处理的蚁群算法

蚁群“构造墓地”的行为是一种典型的聚类行为, 近几年来国内外许多学者在将聚类与蚁群优化相融合方面做了大量的研究工作<sup>[31~35]</sup>。

本节针对一类带聚类特征的 TSP, 研究了一种新型的带聚类处理的蚁群算法 (clustering processing ant colony algorithm, CPACA)<sup>[31]</sup>。CPACA 首先对 TSP 中的城市进行聚类处理, 将 TSP 问题分解成许多小规模的子问题 (子问题数目与城市的聚类数相等), 然后利用蚁群算法对每个子问题并行求解, 并将所有子问题的解按一定规则合并成待求解问题的解。由于该过程利用问题本身的聚类特征对所求问题进行分解, 并行求解每一个子问题, 从而极大地加快了算法的求解速度。

### 5.8.1 算法设计

#### 5.8.1.1 CPACA 基本原理

因为蚁群算法对小规模 TSP 具有极高的求解性能, 如果能将大规模 TSP 分解成一些小规模子问题, 对每个小规模子问题分别利用蚁群算法并行求解, 最后将每个小规模子问题的解合并成待求解问题的解, 这样将大大提高蚁群算法的求解性能。由此, 如何分解大规模 TSP 成为 CPACA 的关键。为了找出分解 TSP 的规律, 首先人为地构造两类具有不同分布特征的 TSP。图 5.28 为球状分布 TSP 的最优解, 城市可聚成 4 类; 图 5.29 为线状和球状混合分布 TSP 的最优解, 可聚成 3 类。

通过对图 5.28、图 5.29 的分析, 发现 TSP 最优解的结构具有如下规律:

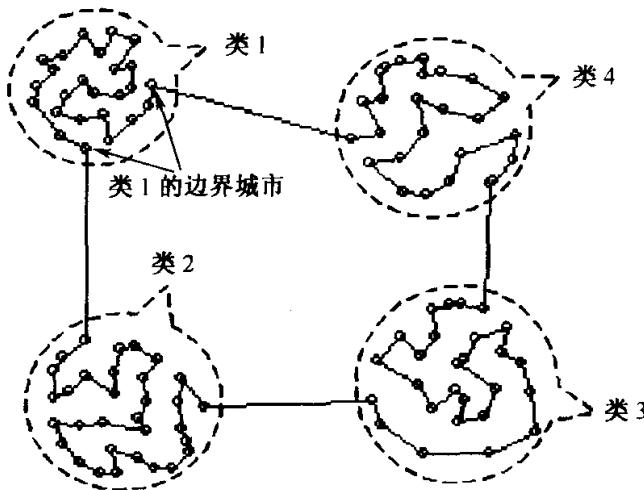


图 5.28 110 城市的 TSP (ACA 迭代 1000 次后的结果)

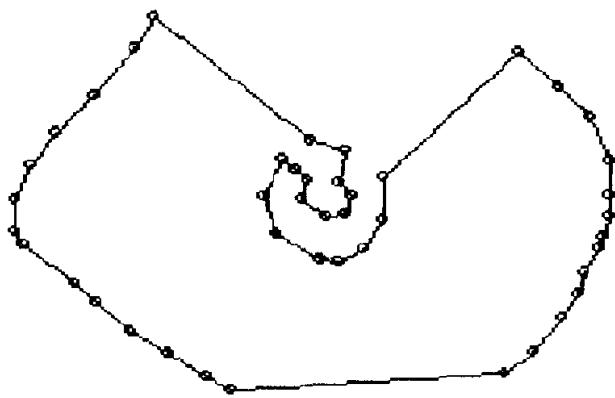


图 5.29 44 城市的 TSP (ACA 迭代 1000 次后的结果)

(1) 设规模为  $n$  的 TSP 按城市的分布可聚成  $c$  类，设类  $i(i=1, 2, \dots, c)$  中的城市数目为  $w_i$ ，显然有  $\sum_{i=1}^c w_i = n$ ，则在类  $i(i=1, 2, \dots, c)$  中，有且仅有两个城市  $u_1^i, u_{w_i}^i$  分别与另外两个不同的类相连。为方便起见，称城市  $u_1^i, u_{w_i}^i$  为类  $i$  的边界城市。

(2) 在类  $i(i=1, 2, \dots, c)$  中，有一条从边界城市  $u_1^i$  经过剩下的城市一次且仅一次最后到达边界城市  $u_{w_i}^i$  的路径，该路径是待求 TSP 解的组成部分，且是类  $i$  中从城市  $u_1^i$  到城市  $u_{w_i}^i$  的最短路径。因为如果该路径不是最短的，则 TSP 的解也非最优。

(3) 如果将类  $i(i=1, 2, \dots, c)$  本身看成一个超级城市  $i$ ，则由这  $c$  个超级城市构成一个新的 TSP，且该新 TSP 的解即构成所有类的连接顺序。例如在图 5.28 中，4 个超级城市（类）构成一个新的 TSP，每一类通过边界城市与其

他类的边界城市相连。因此，新 TSP 的最优解即构成这些超级城市连接的顺序。图 5.28 中的顺序为：城市 1（类 1）→城市 2（类 2）→城市 3（类 3）→城市 4（类 4）→城市 1（类 1）。该连接顺序是新 TSP 的一个解，否则待求解 TSP 的解非最优。

显然，具有不同分布特征的 TSP（图 5.29）也满足以上的三条规律。因此，从上述分析中受到启发，在求解带聚类特征的 TSP 时，首先对问题进行聚类处理，对所有的类，按规律（3）确定其连接顺序，再求出每类的边界城市，最后对类  $i$  ( $i=1, 2, \dots, c$ ) 采用蚁群算法并行求解，生成一条从边界城市  $u_1^i$  到边界城市  $u_{w_i}^i$  的最短路径，并将所有类中的路径按连接顺序合并起来，便生成了待求 TSP 的一个可行解。

### 5.8.1.2 CPACA 的实现

#### 1. 聚类算法的选择

在 CPACA 中，首先需根据 TSP 中城市的分布特征选择不同的聚类算法，且应为基于距离的聚类算法（distance-based clustering algorithm, DCA）。对于类内模式为球状分布的 TSP，选择 C-均值法，图 5.28 中的类内模式即为这种情况；对于非球状的类内模式，如条状和线状分布特性的 TSP，则选用近邻函数法。

#### 2. 类连接顺序

为了将所有类中的路径重新合并成最终问题的解，必须先求出各类的连接顺序。如图 5.28 所示，在 TSP 的最优解中，各类的连接顺序为：类 1 → 类 2 → 类 3 → 类 4 → 类 1。显然，为使 TSP 的解最优，类与类之间的所有连接边的长度之和应最小，因此可采用下面的方法确定类之间的连接顺序。

设可将 TSP 中的城市聚成  $U_1, U_2, \dots, U_c$  类 ( $c$  为类的个数)， $K_i$  ( $i=1, \dots, c$ ) 为每类的中心，其中  $K_i$  可能不是 TSP 中的城市。将城市  $K_i$  ( $i=1, \dots, c$ ) 看成一个 TSP，并用蚁群算法对其求解，设求得的解为  $K_{i1}, K_{i2}, \dots, K_{ik}, K_{i1}$  (其中  $i_1, i_2, \dots, i_c$  为  $1, 2, \dots, c$  的一个排列)，则排列  $i_1, i_2, \dots, i_c$  即为各类的连接顺序。

#### 3. 类中边界城市的确定

设类  $p$ 、类  $q$  ( $p, q=1, 2, \dots, c$ , 且  $p \neq q$ ) 按连接顺序相邻， $u_i^p$  ( $i=1, 2, \dots, w_p$ ) 为类  $p$  中的城市， $u_j^q$  ( $j=1, 2, \dots, w_q$ ) 为类  $q$  中的城市，则类  $p$ 、类  $q$  中的一个边界城市可由下式确定

$$\{u_k^p, u_l^q\} = \arg \min_{\substack{i=1,2,\dots,w_p \\ j=1,2,\dots,w_q}} d(u_i^p, u_j^q) \quad (5.8.1)$$

式中,  $u_i^p$  表示类  $p$  中的一个边界城市;  $u_j^q$  表示类  $q$  中的一个边界城市;  $U_p$ 、 $U_q$  分别表示类  $p$ 、类  $q$  中的城市集合;  $w_p$ 、 $w_q$  分别表示类  $p$ 、类  $q$  中的城市数目。

当各类的连接顺序确定后, 对任意类  $i$  只有一个直接前驱类和一个直接后继类, 利用公式 (5.8.1) 与其直接前驱类运算, 可以求出边界城市  $u_1^i$ , 通过其直接后继类, 可以求出第二个边界城市  $u_{w_i}^i$ 。

#### 4. 求解类内最短路径的蚁群算法

采用 CPACA 求解类内从第一个边界城市经所有剩余城市一次且仅一次后到达第二个边界城市的最短路径。与基本蚁群算法相比, 其主要区别在于最短路径的生成上。所有蚂蚁都从第一个边界城市出发, 逐步选择除第二个边界城市之外的城市, 从而生成问题的可行解。状态转移概率、信息素更新规则都与基本蚁群算法中的 Ant-Cycle 模型相同。

#### 5. 局部搜索策略

实验结果显示, 在搜索解的过程中, CPACA 能以极快的速度进入最优解所在的子空间, 而其后的搜索速度相对较慢, 故在 CPACA 中引入了局部搜索策略 2-opt。

CPACA 算法的实现步骤可描述如下:

- (1) 根据 TSP 中城市分布的先验知识, 选用 C-均值或近邻函数法对 TSP 中的城市进行聚类处理。设可将城市聚成  $U_1, U_2, \dots, U_c$  类 (其中  $c$  为类的个数), 且  $K_i (i=1, \dots, c)$  为每一类的中心, 其中  $K_i$  可以不是 TSP 中的城市。
- (2) 将  $K_i (i=1, \dots, c)$  看成一个 TSP, 并使用蚁群算法对其求解, 设求得的解为  $K_{i1}, K_{i2}, \dots, K_{ic}, K_{i1}$ , 此解即为合并各类中路径的连接顺序。
- (3) 按公式 (5.8.1) 计算各类中的边界城市。
- (4) 对每一类  $U_i (i=1, 2, \dots, c)$  的蚁群算法, 并行求解从边界城市  $u_1^i$  到边界城市  $u_{w_i}^i$  的一条经过所有剩余城市各一次的最短路径  $R_i (i=1, 2, \dots, c)$ 。
- (5) 对所有的类, 按连接顺序  $i_1, i_2, \dots, i_c, i_1$  通过类与类之间的边界城市连接起来, 便构成了 TSP 的一个可行解。
- (6) 对解进行 2-opt 局部搜索, 记录本次迭代的最好解。
- (7) 如果本次迭代最好解优于当前最优解, 则用其替换当前最优解。
- (8) 如果满足算法结束条件, 则退出算法, 否则跳转到第 (2) 步。

显然, 问题可聚的类越多, 则 CPACA 的求解效率越高, 但如果问题的聚类特性不明显, 则 CPACA 将退化为基本蚁群算法。

## 5.8.2 仿真算例

实验选用 TSPLIB 中的实例 Pr107、Pr136、D2103、U2319、Pr2392，通过基本蚁群算法与 CPACA 进行对比实验。对 Pr107，Pr136 问题用两种算法各运行 15 次，D2103、U2319、Pr2392 用两种算法各运行 4 次，得出实验结果如表 5.11 所示。

表 5.11 CPACA、基本蚁群算法求解不同规模 TSP 的结果对比

TSP 实例	ACA		CPACA		已知最优解
	所求的解	误差百分比 (%)	所求的解	误差百分比 (%)	
Pr107	44661	0.81	44514	0.48	44303
Pr136	100213	3.56	97218	0.46	96772
D2103	88306	9.77	81273	1.02	80450
D2319	262114	11.89	245341	4.73	234256
Pr2392	426218	12.75	392698	3.88	378032

图 5.30 是 CPACA 与基本蚁群算法两种算法求解 Pr136 问题时的收敛情况比较；图 5.31 为 CPACA 所求得的最优解；图 5.32 是 CPACA 求解 D2103 问题时，运行 4min 后所求得的该问题的全局最优解，而基本蚁群算法需要运行 3.5h 才能达到与该值相近的解。

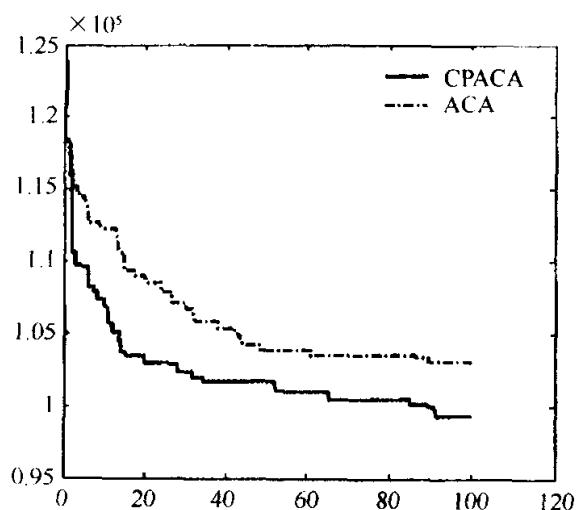


图 5.30 CPACA 与基本蚁群算法的收敛情况比较

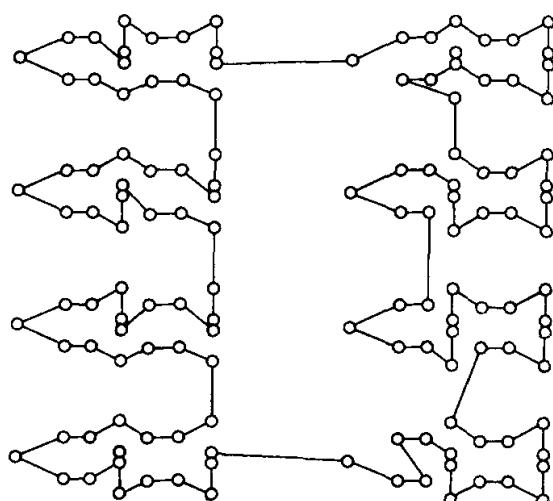


图 5.31 CPACA 求得 Pr136 问题的解

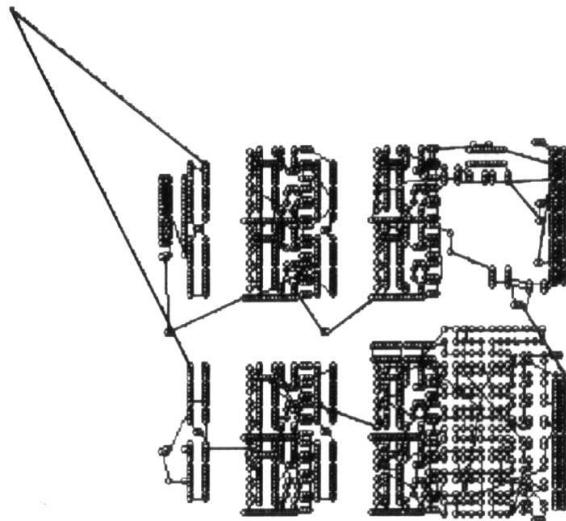


图 5.32 CPACA 求解 D2103 时的运行结果

由上述仿真实验结果可见，在求解带聚类特征的 TSP 时，CPACA 优于基本蚁群算法。

## 5.9 基于云模型理论的蚁群算法

云模型（cloud model）是一种新的实现定性概念和定量数值之间转换的有力工具，用来统一刻画基于语言值的定性概念和数值表示之间的相互映射关系<sup>[36]</sup>。云模型的数字特征可用  $E_x$ （expected value）、熵  $E_n$ （entropy）和超熵  $H_e$ （hyper entropy）三个数值表征，从而可将模糊性和随机性有机地集成在一起。

本节在介绍云模型基本理论的基础上，研究了一种利用云模型来有效限制蚁群算法陷入局部最优解的方法<sup>[37, 38]</sup>，实验表明，该改进策略可使蚁群算法的全局搜索速度和优化性能均得到明显改善。

### 5.9.1 算法设计

#### 5.9.1.1 云模型基本理论<sup>[39]</sup>

**定义 5.9.1 云** 设  $U$  是一个用普通集合表示的论域，即  $U=\{u\}$ ，关于论域  $U$  中的模糊集合，是指对于任意元素  $u$  都存在一个有稳定倾向的随机数  $\mu_{\tilde{A}}(u)$ ，称为  $u$  对  $\tilde{A}$  的隶属度。如果论域中的元素是简单有序的，则  $U$  可以看做是基础变量，隶属度在  $U$  上的分布叫做云；如果论域中的元素不是简单有序的，

而根据某个法则  $f$ , 可将  $U$  映射到另一个有序的论域  $U'$  上, 而在  $U'$  中有且仅有一个  $u'$  与  $u$  对应, 则  $U'$  为基础变量, 隶属度在  $U'$  上的分布也称为云。

云是用语言值表示的某个定性概念与定量表示之间的不确定性转换模型, 云模型每次产生的云滴具有不确定性, 某一个云滴也许无足轻重, 但云的整体反映了一个定性概念, 对应的数值域作为自变量。

### 1. 云模型的数字特征

云的数字特征用期望值  $E_x$ 、熵  $E_n$  和超熵  $H_e$  三个数值来表征, 构成了定性和定量的相互映射, 并以此作为知识表示的基础。不同熵和超熵的云模型及其数字特征如图 5.33 所示。

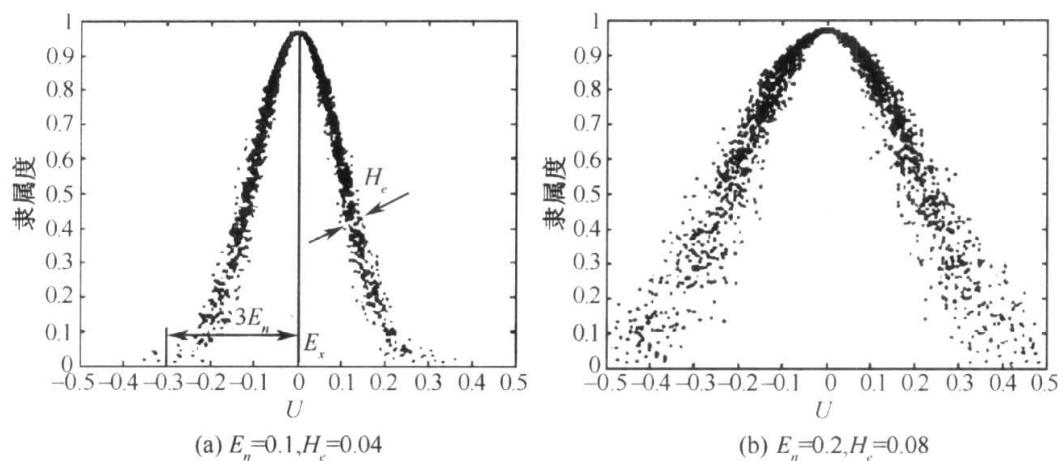


图 5.33 云模型及其数字特征

**定义 5.9.2 期望  $E_x$**  在数域空间内最能够代表这个定性概念的点, 反映了相应定性概念的信息中心值。

**定义 5.9.3 熵  $E_n$**  熵一方面反映了在数域空间可被语言值接受的范围; 另一方面还反映了在数域空间的点能够代表这个语言值的概率, 表示定性概念的云滴出现的随机性。熵揭示了模糊性和随机性之间的本质关联性。

**定义 5.9.4 超熵  $H_e$**  超熵是熵的不确定度量, 即熵的熵, 反映了在数域空间代表该语言值的所有点不确定度的凝聚性, 其大小间接地反映了云的厚度, 即云滴的凝聚度。

### 2. 云发生器

按云的产生机理和计算方向分, 有正向云和逆向云, 而正向云又分为基本云、 $U$  条件云和  $V$  条件云。

正向云发生器是根据已知正态云的数字特征期望值  $E_x$ 、熵  $E_n$  和超熵  $H_e$ , 产生满足上述正态云分布规律的二维云  $\text{Drop}(x, \mu)$ , 称为云滴。正向云通过输入

3个数字特征形成合乎条件的云滴，云发生器生成的若干云滴构成整个云，从而将定性概念通过不确定性转换云模型定量地表示出来；逆向云发生器是已知云中相当数量的云滴分布  $\text{Drop}(x, \mu)$ ，确定正态云的3个数字特征值期望值  $E_x$ 、熵  $E_n$  和超熵  $H_e$ 。

### 3. 定性关联规则

云模型的定性关联规则分为两大类，即单条定性关联规则和多条定性关联规则。

(1) 单条定性关联规则。通常单条定性关联规则的形式化描述为

$$\text{IF } A \text{ THEN } B \quad (5.9.1)$$

其中， $A$  和  $B$  为语言值表示的对象。对照语言原子与云的关系，便可运用云模型对象构造定性关联规则。根据上述各类云模型对象的概念，以带  $U$  条件和  $V$  条件的云模型对象构造定性关联规则，单条定性规则的构造原理如图 5.34 所示。

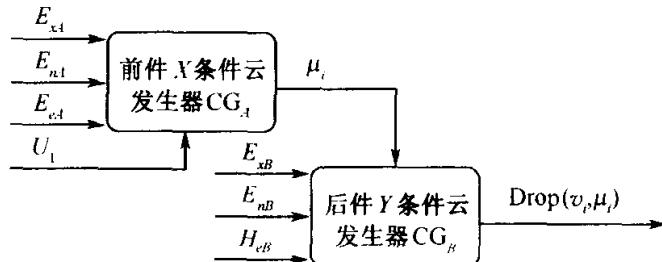


图 5.34 单条定性关联规则构造原理

在图 5.34 中， $CG_A$  表示对应输入平面语言值  $A$  的带  $U$  条件的云模型对象， $CG_B$  表示对应输出平面语言值  $B$  的带  $V$  条件的云模型对象。由于对应一个固定的输入值，输出空间中的  $\mu_i$  具有不确定性，因此这一推理系统的实现对不确定性具有良好的继承性和传递性。

(2) 多条定性关联规则。多条定性关联规则的形式化描述为

$$\left\{ \begin{array}{lll} \text{IF } & A_1 & \text{THEN } B_1 \\ \text{IF } & A_2 & \text{THEN } B_2 \\ \vdots & & \\ \text{IF } & A_n & \text{THEN } B_n \end{array} \right. \quad (5.9.2)$$

其中， $A_i$  和  $B_i$  ( $i=1, 2, \dots, n$ ) 为语言值表示的对象。多条定性关联规则的构造思路类似于单条定性关联规则云。

#### 5.9.1.2 改进策略

蚁群算法应用过程中易出现的停滞和扩散问题不容忽视<sup>[9, 40]</sup>，将各条寻优

路径上可能的残留信息素数量限制在  $[\tau_{\min}, \tau_{\max}]$ ，并在每次循环结束后，保留最优路径，一个循环中只有路径最短的蚂蚁才有权修改  $\tau_{ij}(t)$ 。修改策略在信息素更新之后，再按照下式进行阈值判断选择

$$\tau_{ij}(t+n) = \begin{cases} \tau_{\min}, & \text{若 } \tau_{ij}(t) \leq \tau_{\min} \\ \tau_{ij}(t), & \text{若 } \tau_{\min} < \tau_{ij}(t) \leq \tau_{\max} \\ \tau_{\max}, & \text{若 } \tau_{ij}(t) > \tau_{\max} \end{cases} \quad (5.9.3)$$

信息素挥发系数  $\rho$  和信息素强度  $Q$  用云模型理论进行调整的 4 条定性关联规则为：

[规则 1]：IF  $\rho > \rho_{\min}$  AND  $t \leq T$  THEN 选取较小的  $\rho$  和较小的  $Q$ ；

[规则 2]：IF  $\rho \leq \rho_{\min}$  AND  $t \leq T$  THEN 选取较大的  $\rho$  和较小的  $Q$ ；

[规则 3]：IF  $\rho > \rho_{\min}$  AND  $t > T$  THEN 选取较小的  $\rho$  和较大的  $Q$ ；

[规则 4]：IF  $\rho \leq \rho_{\min}$  AND  $t > T$  THEN 选取较大的  $\rho$  和较大的  $Q$ 。

上述关联规则可以用云模型的多规则生成器完成，这里以调整前的信息素挥发系数和时间或能表征它们变化趋势的指标作为输入参数来实现多规则生成器的  $U$  条件云，而以信息素强度云和调整后的信息素挥发系数云来实现多规则生成器的  $V$  条件云，从而这种云多规则生成器使随机过程和信息素强度、信息素挥发系数的优化设置有机地结合在一起。云模型多规则生成器的规则倾向性保证了蚁群算法的快速搜索能力和全局收敛性能；同时，云模型所蕴含的随机过程保证了总体上的最佳求解效果。

## 5.9.2 仿真算例

将基于云模型理论的改进蚁群算法与基本蚁群算法分别应用于 Chcl44TSP 进行仿真实验。对路径寻优采用升半正态云规则控制，云滴数为 500 个云滴，最优期望值  $E_x = 0.5$ ，熵  $E_n = 1.3$ ，超熵  $H_e = 0.05$ ，设置算法的参数  $\alpha = 1, \beta = 4, \rho_{\min} = 0.3, \Delta\tau_{ij}(0) = 0, N_{c_{\max}} = 300, m = 150, n = 144, \tau_{\max} = \tau_{ij}(0), \tau_{\min} = \tau_{\max}/(1064)$ 。改进前后蚁群算法的仿真实验结果比较如表 5.12 所示。

表 5.12 不同蚁群算法的仿真实验结果比较

算法类型	实际最优解	时间 (s)	平均最优解	相对误差 (%)
基本蚁群算法	30347	238	30855	1.674
改进蚁群算法	30347	127	30394	0.155

由表 5.12 可知，基于云模型理论的改进蚁群算法与基本蚁群算法相比较，前者寻到全局最优解的时间少了近 50%，并且相对误差大大减小，全局搜索速度和优化性能均得到了明显改善。

## 5.10 具有感觉和知觉特征的蚁群算法

针对蚁群算法中的早熟、停滞且收敛速度慢等缺陷，本节研究了一种具有感觉和知觉特征的蚁群算法（sensational and consciousness algorithm, SCA）<sup>[41]</sup>，该算法模拟蚂蚁的感觉和知觉行为，并受显意识和潜意识的相互作用选择路径，使之在初始阶段接受自己经验的影响，此后逐步有条件地接受信息量的影响，同时自适应地修改路径上的信息量。这样，可在加速收敛和防止早熟、停滞现象之间取得较好的平衡。

### 5.10.1 算法设计

科学研究表明，蚂蚁对于外界刺激物有着惊人的感觉和知觉能力。蚁群中单只蚂蚁的能力和智力非常有限，但是整个蚁群可以有足够的能力来完成多种复杂行为。它们可以在各自的感觉和知觉能力的支配下，很好地相互协调、分工、合作并完成任务。因此可在蚁群算法中模拟蚂蚁的这种感知现象，让蚂蚁根据感觉和知觉规律选路，以在加快收敛速度的同时保持解的多样性。根据感知规律，可将 SCA 中的蚂蚁搜索过程分为如下三个阶段：

#### 5.10.1.1 蚂蚁搜索的初始阶段

心理学研究指出<sup>[42]</sup>，感觉和知觉是客观事物作用于神经系统引起神经系统的活动而产生的。产生感觉和知觉的神经机构叫分析器，感受性是分析器对适宜刺激的感觉能力。感觉是大脑对当前直接作用于感觉器官的客观事物的个别属性的反映，而知觉则是其整体属性的反映，两者紧密相关。感受性是用感觉阈限的大小来度量的，感觉阈限是能引起感觉的、持续了一定时间的刺激量。心理学上把刚刚能引起感觉的最小刺激量定义为绝对感觉阈限（absolute sensor threshold, AST）。

在蚂蚁搜索过程的起始阶段，有的路径上有蚂蚁走过，有的路径还没有被走过。而蚂蚁的选路策略是它以较大的概率选择具有较大信息量的路径，这使得蚂蚁从搜索的一开始就以很大的概率集中于几条长度较短的路径上。实践中我们观察到，蚂蚁在搜索的初始阶段所得到的这些路径的整体长度往往都偏大，这样易导致所得的结果是局部最优而非全局最优。

为了避免蚂蚁从搜索的一开始就失去解的多样性，受 AST 原理的启发，当路径上的信息量未达到蚂蚁的 AST 时，让蚂蚁忽视该刺激物的存在，也就是让蚂蚁在搜索初始阶段的选路不受信息量的影响。只有当信息量积累到超过 AST

时，蚂蚁才在信息素的刺激下趋于选择信息量较大的路径。这样，可以让蚂蚁在初始阶段选择较多的不同路径，以获得多样化的解，避免蚂蚁陷入局部最优，让蚂蚁尽量少走“冤枉路”。大量的实验表明，这种方法有利于让蚂蚁向最优方向进行搜索并很快收敛，从而大大节约了计算时间。

### 5.10.1.2 蚂蚁搜索的中间阶段

刺激物引起感觉之后，如果刺激量发生变化也会引起感觉的变化，但并不是刺激的所有变化量都能引起感觉。由于经验和潜意识的作用，当刺激条件的改变幅度仅局限于一定的范围内时，包括特定感觉的知觉仍然保持相对不变，这就是知觉的“恒常性”。只有当刺激变化到一定量时才能感觉到差别，这里将能引起差别感觉的刺激物的最小变化量称为差别感觉阈限（contrast sensor threshold, CST）。早在 1846 年，德国心理学家 Weber E H 在研究 CST 时发现，CST 随原来刺激量的变化而变化，而且表现为一定的规律性。在一定的范围内 CST 与原来量的比值是一个常数。如果将它用公式来表示，设  $I$  表示原来刺激物的强度， $\Delta I$  表示 CST，那么当  $I$  小于某个强度阈限（intensity threshold, IT）时，有

$$\frac{\Delta I}{I} = K \quad (5.10.1)$$

式中， $K$  是一个常数，称之为 Weber 系数。这就是著名的韦伯定律（Weber's law）。一般而言，重量感觉的 Weber 系数为  $1/30$ ，听觉为  $1/10$ ，视觉为  $1/100$ 。

由于各条路径上的信息量也是在不断变化的，在将韦伯定律应用到蚁群算法的搜索过程中，需要改变基本蚁群算法中信息量一旦变化就直接影响蚁群选路的做法，认为蚁群在一定刺激强度范围内也存在一个 CST。当路径上信息素增加或减少的量在 CST 之下时，蚂蚁遵循知觉的恒常性规律，感受不到该路径上信息量的变化，该条路径被选择的概率主要依据于以前迭代中根据蚂蚁选路经验所形成的潜意识作用；反之，蚂蚁受其显意识控制，按照路径上所有蚂蚁信息量高低决定选路概率的大小。在具体实现此过程时，根据韦伯定律，取

$$CST = \tau_{ij}(t) \cdot K \quad (5.10.2)$$

式中， $K$  表示蚂蚁对信息量感觉的 Weber 系数，此处取值为  $1/50$ 。记

$$\theta_{ij}^k = \begin{cases} \frac{\alpha_{ij}^k}{\beta_{ij}^k}, & \text{若 } \tau_{ij} \leq CST \\ \tau_{ij}(t) \eta_{ij}, & \text{否则} \end{cases} \quad (5.10.3)$$

$$\alpha_{ij}^k = \frac{\tau_{ij}^k(t)}{\sum_{h \in \text{allowed}_k} \tau_{ih}^k(t)} \quad (5.10.4)$$

$$\beta_{ij}^k = \frac{\tau_{ij}(t) - \tau_{ij}^k(t)}{\sum_{h \in \text{allowed}_k} (\tau_{ih}(t) - \tau_{ih}^k(t))} \quad (5.10.5)$$

式中,  $\alpha_{ij}$  表示蚂蚁  $k$  的潜意识作用下路径  $(i, j)$  对它的吸引程度;  $\beta_{ij}$  表示该路径对其他蚂蚁的吸引作用。实际上, 当蚂蚁  $k$  按照其潜意识选择路径时,  $\beta_{ij}$  包含了所有其他蚂蚁所留信息量的作用, 干扰了蚂蚁  $k$  的潜意识, 构成了潜意识作用下蚂蚁  $k$  对路径  $(i, j)$  的排斥作用。蚂蚁  $k$  选择路径  $(i, j)$  的概率为

$$p_{ij}^k = \begin{cases} \frac{\theta_{ij}^k}{\sum_{h \in \text{allowed}_k} \theta_{ih}^k}, & \text{若 } j \in \text{allowed}_k \\ 0, & \text{否则} \end{cases} \quad (5.10.6)$$

当路径  $(i, j)$  上的信息量变化  $\Delta\tau_{ij}$  小于 CST 时, 蚂蚁则按照自己的潜意识作用选路。此时, 如果蚂蚁  $k$  在某一条路径上走过的次数越多, 它对这条路径越熟悉, 其潜意识作用下选择该路径的概率就大; 反之, 当一条路径上其他蚂蚁的信息量越多, 则潜意识中蚂蚁  $k$  对该路径就越排斥, 该路径被选择的概率就小。这种机制有效地防止了蚂蚁一味相信其他蚂蚁而造成盲从, 大大降低了大量蚂蚁聚集于少数局部较优路径上而造成早熟、停滞等现象发生的可能性。当信息量变化较大时, 蚂蚁在显意识作用下受整体信息量的影响而选路, 遵循“路径上走过的蚂蚁越多选择该路径的概率就越大”这一原则, 趋向于选择信息量较大的路径, 保证了所选路径的优越性, 而且加快了算法的收敛速度。

### 5.10.1.3 蚂蚁搜索的结束阶段

由于韦伯定律只能在刺激物的强度  $I$  小于某个特定的限度  $IT$  时才能适用, 在超过  $IT$  时并不适用, 故当路径上信息量达到相当大的程度时, 应改变蚂蚁的选路策略。为了确定  $IT$  的值, 首先估计各个路径上可能的最大信息量  $\tau_{\max}$ 。设蚂蚁搜索的最大遍历次数为  $N_{c_{\max}}$ , 蚂蚁总数目为  $m$ , 则最大信息量可以表示为

$$\tau_{\max} = N_{c_{\max}} \cdot m \times \text{适应度的数量级} \quad (5.10.7)$$

由 AST 的定义可知, 此处适应度的数量级应该和 AST 相当, 所以取

$$IT = h \cdot \tau_{\max} = h \cdot N_{c_{\max}} \cdot m \cdot AST \quad (5.10.8)$$

式中,  $h$  为一常数, 可取  $1/2$ 、 $2/3$  或  $3/4$  等。若某路径上的信息量较大, 则可能因为算法进入了局部最优的状态, 也可能因为整个蚁群的遍历行为已接近尾声。但由于上述搜索中间阶段的选路策略已经有效地避免了停滞现象的产生, 所以此时不大可能是因为陷入局部最优, 而是因为这些路径占有绝对优势。为加快收敛, 此时引用 Ant-Q 中的选路策略<sup>[6]</sup>来选择下一个城市  $j$

$$j = \begin{cases} \arg \max_{j \notin \text{tabu}_k} [\tau_{ij}(t) \mid j \in \text{allowed}_k], & \text{若 } q \leq q_0 \\ \text{公式(5.10.10),} & \text{否则} \end{cases} \quad (5.10.9)$$

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{k \notin \text{tabu}_k} [\tau_{ik}(t)]^\alpha [\eta_{ik}(t)]^\beta}, & \text{若 } j \in \text{allowed}_k \\ 0, & \text{否则} \end{cases} \quad (5.10.10)$$

通过大量的实验，确定选取随机数  $q_0$  值在 0.7 左右时较为合适。

#### 5.10.1.4 自适应的信息素更新策略

若干改进的蚁群算法在更新信息素时，要么采取只要蚂蚁遍历时经过该条路径就增加其信息量的方法<sup>[43, 44]</sup>；要么只让最优适应度路径上的信息量得以增加，其余路径上的信息量被削减。这些做法都采用了固定的信息量增减的比例，忽视了解的分布情况。这里提出了一种新的信息量自适应更新策略，即根据信息量的分布情况进行信息量的更新，以动态地调整各路径上的信息量分布，使之不至于过分集中或者分散，以在加速收敛的同时避免早熟。信息素的局部更新可采取如下策略

$$\tau_{ij}(t+1) = \begin{cases} \tau_{ij}(t) - \frac{5}{d_{ij}}, & \text{若此前已有 } m/2 \text{ 只蚂蚁选择了同一城市 } j \\ & \text{或 } m/4 \text{ 蚂蚁选择该城市后终止本次遍历} \\ \tau_{ij}(t) + \frac{1}{d_{ij}}, & \text{否则} \end{cases} \quad (5.10.11)$$

一次迭代中蚂蚁在某一城市选择下一城市时间间隔内信息素尚未被挥发，所以可在局部更新时不考虑信息素挥发因子<sup>[45, 46]</sup>。由于蚂蚁常常选择信息量较大的路径，当多只蚂蚁选中同一路径后，信息量增加的幅度太大就容易使多只蚂蚁集中到该路径，所以取  $1/d_{ij}$  为增加的信息量；若选择该路径的蚂蚁达到  $m/2$ ，或有  $m/4$  的蚂蚁选择该路径后因当前距离超过上一次的最优路径长度而终止遍历，则减少  $5/d_{ij}$  的信息量，以使其趋于各条路径信息量的平均值，从而使蚂蚁选择其他路径的可能性增加，让搜索得到的解趋于多样化。信息素按照下述公式进行更新

$$\tau_{ij}^k(t+1) = (1 - \rho)\tau_{ij}^k(t) + \psi_k \cdot \Delta\tau_{ij}^k(t) \quad (5.10.12)$$

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \sum_{k=1}^m \psi_k \cdot \Delta\tau_{ij}^k(t) \quad (5.10.13)$$

式中， $\psi_k$  表示第  $k$  只蚂蚁所对应的解对路径  $(i, j)$  上信息素更新的影响程度，其计算方法如下：设对经过路径  $(i, j)$  的总数为  $s$  的蚂蚁在本次迭代中遍历的路径全长由小到大进行排序，所得序号存于数组  $\text{rank}[]$  中， $\text{rank}[k]$  表示第  $k$  只蚂蚁所对应的序号，取

$$\psi_k = \frac{s}{2} - \text{rank}[k] + 1 \quad (5.10.14)$$

式中，若  $\text{rank}[k]$  值比较大，那么  $\phi_k$  就为负值，全局更新时该蚂蚁减少相应路径上的信息量，而且  $\text{rank}[k]$  值越大，即路径越长，路径上信息量减小的程度越大，从而使得越差路径上的信息量保留得越少，有利于保持较好路径上的信息；反之，若  $\text{rank}[k]$  值较小， $\phi_k$  就为正值，则蚂蚁在相应路径上增加信息量，且  $\text{rank}[k]$  越小，即路径越短，则  $\phi_k$  越大，信息量增加得就越多，从而有效地强化了短路径上的信息。这种自适应的信息量更新机制可动态地调整信息量，有效地实现蚂蚁的搜索速度和解多样性之间的动态平衡。

### 5.10.1.5 算法框架伪代码

综上所述，可将 SCA 的框架伪代码描述如下：

```

Main ( )
{
    (a) 初始化
        随机产生  $m$  个初始解，计算这  $m$  个初始解的适应度，即  $m$  只蚂蚁遍历完成的路径
        长度的倒数，记其最大适应度为  $f_{\max}$ 。取  $\text{AST} = C \cdot f_{\max}$ ，这里  $C$  为常数，取其值
        为 5。
        设经过路径  $(i, j)$  的初始解有  $s$  个，它们的总长度分别为  $L_1, L_2, \dots, L_s$ ，则路径
         $(i, j)$  上的初始信息量为
        
$$\tau_{ij}(0) = \sum_{k=1}^s \frac{1}{L_k}$$

    (b) 迭代过程
        While not 结束条件 do
            For  $i=1$  to  $n$  do //对  $n$  个城市循环
            {
                For  $k=1$  to  $m$  do //对  $m$  个蚂蚁循环
                    If (以城市  $i$  为起点的各条路径的平均信息量未达到 AST) then
                    {
                        蚂蚁在当前城市随机产生下一次访问的城市序号；按公式 (5.10.12) 更
                        新蚂蚁  $k$  的信息量；
                    }
                    else
                    {
                        If (以城市  $i$  为起点的各条路径上的平均信息量小于 IT) then
                            根据概率选择公式 (5.10.6) 选择下一个城市  $j$ 
                        else 根据公式 (5.10.9)、(5.10.10) 选择下一个城市  $j$ ；
                        根据公式 (5.10.11) 局部更新  $(i, j)$  上的信息素；
                    }
            }
}

```

End for  $k$

$m$  只蚂蚁中若有当前已遍历的路径长度已经超过上一次迭代得到的最优路径长度的，则停止该蚂蚁对下一城市的遍历；

}

End for  $i$

按公式 (5.10.13) 对所有蚂蚁的路径全局更新其信息素；

End while

}

## 5.10.2 仿真算例

用上述 SCA 与未改进的基本蚁群算法 (ACA) 对各种规模的 TSP 分别进行仿真实验。设置  $C=5$ ,  $K=1/50$ ,  $h=0.7$ , 运行 25 次, 每次最大迭代次数为 1500, 所测试的结果如表 5.13 和表 5.14 所示。表中的“平均时间”指一次运行中找到最好解的平均运行时间, “允许时间”指每次运行所允许执行的最长时间。

表 5.13 不同规模 TSP 的测试结果

TSP 实例	算 法	最 优	平 均	最 差	命中最优次数	平均时间 (s)
D198	ACA	15780	15780.9	15786	19	113.5
	SCA	15780	15780.3	15783	22	109.7
Lin318	ACA	42029	42029.2	42034	24	198.6
	SCA	42029	42029.0	42029	25	151.2
Pcb442	ACA	50778	50778.5	50782	19	584.1
	SCA	50778	50778.1	50781	24	376.8
Att532	ACA	27686	27686.2	27688	22	537.0
	SCA	27686	27686.2	27689	23	469.2

表 5.14 不同规模 TSP 达到最优解时所需的平均迭代次数

TSP 实例	ACA	SCA
D198	1659	1120
Lin318	2699	1833
Pcb442	4023	2976
Ry48p	201	175
Ft70	737	524
Kro124p	1018	712

图 5.35(a)~图 5.35(f) 给出了利用 SCA 和 ACA 求解不同规模 TSP 时所得最优解的进化曲线。

由图 5.35 不难看出, SCA 不但收敛速度较快, 而且具有较高的稳定性, 比较适合求解较大规模的 TSP。

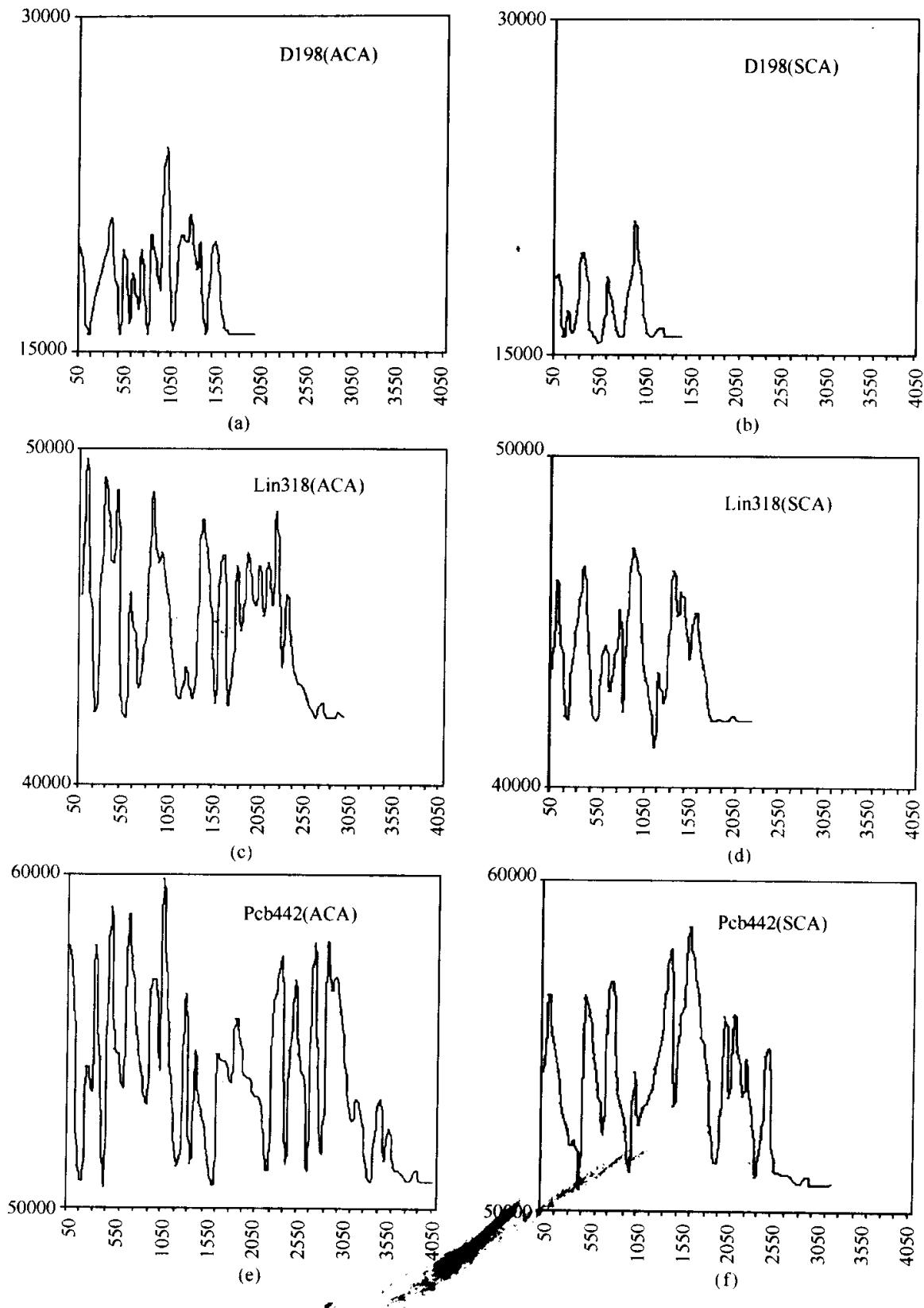


图 5.35 不同规模 TSP 的最优解进化曲线

由于参数  $C$ 、 $K$  和  $h$  决定了 AST、CST 和 IT 的取值，从而决定了蚂蚁的搜索

何时从初始阶段进入中间阶段、何时在潜意识的影响下选择路径以及何时进入选路的结束阶段，它们直接影响到蚁群算法的收敛速度及求解质量。为了分析这些参数对算法性能的影响，这里在等同条件下对不同规模的 TSP 分别运行 10 次，并在表 5.15~表 5.17 中用 D198 和 Ry48p 两个实例对这 3 个参数作了详细的实验分析。

对参数 C 取介于 0~10 之间的不同值进行测试，其结果如表 5.15 所示。

表 5.15 在 C 的不同取值下 D198 和 Ry48p 所得到的不同最优解

TSP 实例	C	10 次运行分别得到的最优解					方 差	平均时间(s)	平均迭代次数
D198	1	15780	15794	15832	15910	15780	98.108	156.3	1698
		15780	15780	15784	15780	16102			
	3	15862	15780	15962	15780	15780	56.605	195.6	1872
		15780	15782	15780	15780	15802			
	5	15794	15882	15832	15780	15852	34.505	166.9	1744
		15780	15822	15780	15780	15788			
	7	15780	15882	15782	15780	15782	64.510	201.5	2143
		15834	15988	15780	15780	15812			
	9	15932	15782	15868	15780	15986	71.252	293.9	2967
		15780	15798	15840	15794	15920			
Ry48p	1	14532	14422	14462	14428	14462	51.589	123.5	509
		14422	14422	14312	14422	14432			
	3	14542	14422	14438	14464	14430	37.500	117.2	475
		14422	14422	14486	14422	14428			
	5	14422	14508	14422	14484	14422	37.585	72.6	281
		14422	14516	14422	14422	14424			
	7	14422	14422	14648	14422	14422	67.344	128.3	517
		14426	14464	14422	14422	14424			
	9	14428	14422	14426	14422	14552	58.260	139.0	546
		14530	14466	14574	14422	14422			

由表 5.15 可见，C 取常数 1 时，虽然对实例 D198 运行算法消耗的时间较短，但所得最优解的稳定性较差。显然，这是因为 C 值太小导致蚂蚁过早进入中间阶段，而造成了解的局部收敛。而当 C 值取 9 时，算法收敛所需代数较多、时间较长，这是因为蚂蚁用于随机搜索的时间加长的缘故。从表 5.15 中还可看出，当 C=5 时，算法在解的稳定性和求解速度之间取得了较好的平衡，蚂蚁不会因 AST 取值较小而过早进入中间阶段，造成解的局部性；也不会因 AST 取值太大，使蚂蚁徘徊于选路的初始阶段，从而大大降低了收敛速度。

Weber 系数  $K(0 < K < 1)$  在不同取值下对问题 D198 和 Ry48p 求解所得最优解的比较如表 5.16 所示（取 C=5）。

由表 5.16 可见，当  $K$  取 1/50 左右时算法所得的解较稳定，且收敛所需的迭代次数较少，消耗的时间也相对较短。这是因为若  $K$  取值太大，CST 的值就

大，从而蚂蚁按状态转移概率公式选择路径时，会以极大的概率依照自身的潜意识执行，忽视了种群中其他个体的协同作用，虽然速度相对较快，但容易造成局部收敛；反之，若  $K$  取值太小，蚂蚁极大程度地依赖其他蚂蚁遗留的信息量而“盲从”于其他蚂蚁，造成选择路径过程的振荡而难于收敛。而当  $K$  取 1/50 时，算法可在上述两者之间取得很好的平衡。

表 5.16 在  $K$  的不同取值下 D198 和 Ry48p 所得到的不同最优解

TSP 实例	$K$	10 次运行分别得到的最优解					方 差	平均时间(s)	平均迭代次数
D198	1/20	15783	15910	15782	15780	15780	38.842	197.2	2587
		15825	15780	15788	15782	15791			
	1/30	15832	15780	15796	15812	15780	17.169	168.3	2149
		15780	15780	15782	15780	15780			
	1/40	15780	15780	15784	15798	15780	16.317	152.1	1975
		15834	15798	15780	15780	15782			
	1/50	15780	15782	15780	15780	15787	9.529	126.9	1572
		15805	15780	15792	15780	15804			
	1/60	15782	15780	15780	15780	15780	18.535	129.4	1604
		15780	15794	15780	15780	15842			
Ry48p	1/70	15780	15816	15780	15780	15780	34.917	127.2	1583
		15794	15780	15780	15896	15780			
	1/80	15796	15902	15780	15786	15780	35.898	149.0	1885
		15780	15780	15780	15780	15792			
	1/90	15863	15780	15982	15780	15814	60.676	142.2	1799
		15825	15788	15780	15784	15780			
	1/20	14428	14486	14472	14422	14512	31.875	69.1	286
		14422	14424	14422	14424	14422			
	1/30	14422	14422	14422	14488	14422	83.788	70.6	307
		14704	14426	14452	14422	14422			
	1/40	14422	14422	14422	14428	14422	21.804	62.4	252
		14424	14492	14422	14454	14422			
	1/50	14422	14432	14422	14422	14422	18.440	56.0	205
		14422	14424	14484	14422	14422			
	1/60	14470	14422	14422	14436	14422	26.740	61.5	241
		14422	14422	14504	14422	14422			
	1/70	14422	14432	14538	14422	14428	36.761	65.3	263
		14486	14454	14422	14422	14422			
	1/80	14474	14422	14482	14524	14422	35.361	59.4	230
		14428	14422	14422	14422	14482			
	1/90	14620	14422	14516	14422	14422	61.900	58.8	212
		14422	14466	14422	14428	14422			

参数  $h$  在不同取值下对问题 D198 和 Ry48p 求解所得最优解的比较结果如表 5.17 所示（取  $C=5$ ,  $K=5$ , 且  $0.5 \leq h < 1$ ）。

表 5.17 在  $h$  的不同取值下 D198 和 Ry48p 所得到的不同最优解

TSP 实例	$h$	10 次运行分别得到的最优解					方 差	平均时间(s)	平均迭代次数
D198	0.5	15812	15780	15782	15780	15782	9.594	125.2	1239
		15788	15780	15788	15794	15780			
	0.6	15788	15780	15782	15784	15780	4.079	127.4	1273
		15792	15780	15780	15782	15788			
	0.7	15783	15780	15782	15780	15780	1.077	110.3	1122
		15780	15780	15780	15782	15781			
	0.8	15784	15782	15780	15780	15784	5.618	148.6	1495
		15780	15790	15780	15780	15798			
	0.9	15792	15780	15780	15804	15788	7.440	172.1	1835
		15786	15782	15780	15780	15780			
Ry48p	0.5	14422	14512	14422	14454	14424	26.988	59.7	267
		14422	14448	14432	14424	14422			
	0.6	14422	14436	14422	14422	14422	7.057	57.3	232
		14442	14424	14434	14424	14422			
	0.7	14422	14434	14424	14424	14422	3.555	42.8	180
		14422	14424	14422	14428	14422			
	0.8	14432	14442	14422	14424	14432	6.437	62.4	258
		14422	14424	14422	14422	14422			
	0.9	14440	14424	14422	14422	14422	5.276	77.2	304
		14422	14424	14422	14424	14422			

由表 5.17 可见, 当  $h$  取 0.7 左右时, 算法的性能最高。这是因为若  $h$  值过小, 则  $\Delta T$  的值就较小, 蚂蚁搜索过程的绝大部分阶段按照公式 (5.10.9)、公式 (5.10.10) 选择路径, 算法以大于  $q_0$  的概率选择信息量最大的路径, 极易造成局部收敛, 从表中可以看出此时解的稳定性较差; 反之, 若  $h$  值太大, 则蚂蚁搜索过程几乎全部按照公式 (5.10.6) 选择路径而达不到提高收敛速度的目的。

由上述实验结果可见, 具有感觉和知觉特征的蚁群算法 (SCA) 找到最优解的平均时间较短, 同时搜索到最优解的次数也相对较高, 从而证明了该算法具有较好的全局收敛性和稳定性。

## 5.11 具有随机扰动特性的蚁群算法

针对基本蚁群算法计算时间较长和容易出现停滞现象的缺陷, 本节设计了一种随机扰动蚁群算法 (ant system with random perturbation behavior,

RPAS)<sup>[47]</sup>。该算法带有一定的自适应性，且具有很强的随机扰动特性。

### 5.11.1 算法设计

基本蚁群算法中的状态转移概率比较类似于遗传算法中的轮赌选择法，这里设计了如下更为简洁的状态转移策略

$$C_{ij}(k) = \begin{cases} (\tau_{ij}(k)\eta_{ij}(k))^\gamma, & \text{若 } j \notin \text{tabu}_k \\ 0, & \text{否则} \end{cases} \quad (5.11.1)$$

$$s = \max(C_{ij(k)}) \quad \text{所对应城市} \quad (5.11.2)$$

式中， $\gamma$  为具有倒指数的扰动因子，且  $\gamma > 0$ 。需要指出的是，公式 (5.11.1) 中的  $C_{ij}(k)$  不再是“状态转移概率”，而是路径  $(i, j)$  的“转移系数”，蚂蚁总是选择转移系数最大的路径，并具有一定的确定性。比较公式 (5.11.1) 和基本蚁群算法的状态转移概率公式，经分析发现，当取固定值时与基本蚁群算法相同，但仍不可避免出现停滞现象，故可采用可变的扰动因子。考虑到如下两种情况。

(1) 蚂蚁个体的运动总是沿着转移系数最大的路径移动。当群体规模较大时，很难在短时间内从大量杂乱无章的路径中找出一条较好的封闭路径，因此在最初的几次迭代中，为加速算法的收敛，应取较大的值方能使得较好路径上的信息量明显高于其他路径上的信息量。

(2) 若一直不变，必将导致某一路径上的信息量远远高于其他路径上的信息量，而该路径不一定是全局最优路径，从而导致后继搜索会出现停滞。由此，在后继搜索过程中应取较小的值，这样，一方面可提高路径选择的多样性（即起到一定的扰动作用），另一方面，可使收敛趋于平缓。

这里设计了如下倒指数关系曲线来描述扰动因子  $\gamma$

$$\gamma = a \times e^{b/k}, \quad k = 1, 2, \dots, N_{c_{\max}}; a > 0, b > 0 \quad (5.11.3)$$

式中， $a$ 、 $b$  表示扰动尺度因子； $N_{c_{\max}}$  表示最大迭代次数。特别地，为不失随机性，令  $a = a' \cdot X$ ，其中  $a' > 0$ ， $X$  是  $(0, 1)$  中均匀分布的随机数。由公式 (5.11.3) 可知，随着迭代次数的增加， $\gamma$  的值最终趋近于系数  $a$ ，而系数  $b$  的大小决定了倒指数曲线趋近于系数  $a$  的快慢程度。其倒指数关系曲线如图 5.36 所示。

针对基本蚁群算法易出现停滞现象，这里设计了如下的具有随机扰动特性的转移系数

$$C_{ij(k)} = \begin{cases} (\tau_{ij}(k)\eta_{ij}(k))^\gamma, & \tau_{ij}(k) = \max(\tau_{is}(k)), \quad s \notin \text{tabu}_k \\ (\tau_{ij}(k))^a \eta_{ij}(k), & \tau_{ij}(k) = \tau_{is}(k) - \max(\tau_{is}(k)), \quad \text{且 } U \leq p_m, s \notin \text{tabu}_k \\ (\tau_{ij}(k)\eta_{ij}(k))^\gamma, & \tau_{ij}(k) = \tau_{is}(k) - \max(\tau_{is}(k)), \quad \text{且 } U > p_m, s \notin \text{tabu}_k \\ 0, & \text{否则} \end{cases} \quad (5.11.4)$$

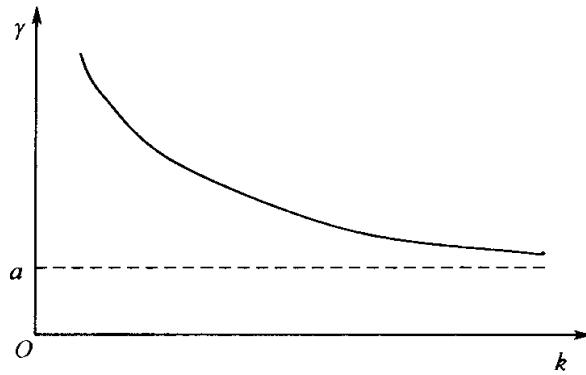


图 5.36 倒指数关系曲线

式中,  $p_m(0, 1)$  称为随机变异率;  $U$  是  $(0, 1)$  中均匀分布的随机数。

公式 (5.11.4) 表明, 某次迭代过程中某只蚂蚁有若干条路径可选, 对于信息量最大的那一条路径应用转移系数公式 (5.11.1), 而对于其他的可选路径, 则采用随机选择方式。该公式是确定性选择与随机性选择相结合的产物, 确定性选择导致蚂蚁总是选择转移系数最大的路径, 而随机性选择导致计算转移系数时具有较强的随机性, 正是两者的共同作用才使 RPAS 具有更强的全局搜索能力。

## 5.11.2 仿真算例

为了检验 RPAS 中随机扰动策略的性能, 这里对 TSPLIB 中的 Gr24TSP、Bayes29TSP 及 Gr48TSP 进行了计算, 并将其与基本蚁群算法的计算结果进行比较, RPAS 和基本蚁群算法的初始参数设置见表 5.18, 仿真计算结果如表 5.19 所示。

表 5.18 RPAS 和基本蚁群算法的初始参数设置

TSP	基本蚁群算法				RPAS					
	$\alpha$	$\beta$	$\rho$	Q	$\alpha$	Q	$\rho$	$p_m$	$a'$	b
24	1.0	5.0	0.5	50	10.0	10	0.85	0.4	5	2
29	1.5	4.0	0.6	50	10.0	10	0.85	0.4	5	2
48	1.5	4.0	0.6	50	10.0	50	0.80	0.2	5	2

表 5.19 RPAS 和基本蚁群算法的计算结果比较 (最大迭代次数均为 50 次)

TSP	基本蚁群算法		RPAS	
	时间 (s)	最短路径	时间 (s)	最短路径
24	5	1298	2	1278
29	8	2180	5	2042
48	52	5494	35	5265

图 5.37 给出了求解 Bayes29TSP 过程中，RPAS 和基本蚁群算法分别取较好组合参数时最好解的收敛曲线。

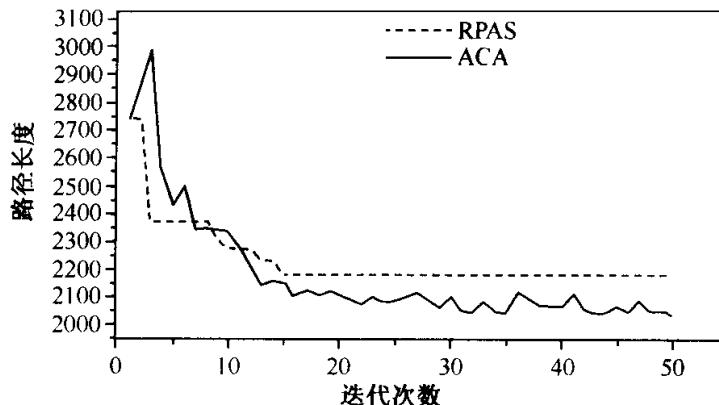


图 5.37 RPAS 和基本蚁群算法的收敛进程

由图 5.37 可见，基本蚁群算法在经过几次迭代后便陷入停滞状态，且计算结果较差，而 RPAS 由于具有很强的随机扰动特性，有效地避免了停滞现象的发生。

## 5.12 基于信息熵的改进蚁群算法

熵 (entropy) 的概念源于物理学的范畴，最早由德国物理学家 Clausius R 于 1854 年提出，当时是用以描述热力学系统的无序状态，而后被引入到其他多个学科，从而有了 Boltzmann 熵、信息熵、概率测度熵等<sup>[48]</sup>。其中，信息熵是由美国学者 Shannon C E 于 1948 年将热力学熵引入信息论而提出的，其为不确定方法的一个重要概念，常被用于较粗略地给出不确定性的度量。纵观熵一个半世纪的发展历史，可以说熵是对“不确定性”的最佳测度，是现代动力系统和遍历理论的重要概念。目前，熵在自然科学、社会科学、管理科学和经济学等科学领域具有非常广泛的应用，爱因斯坦曾将熵定律称之为“整个科学的首要法则”。

为了克服基本蚁群算法求解复杂组合优化问题时易出现早熟的缺陷，本节研究了一种基于信息熵的改进蚁群算法<sup>[49]</sup>，采用由信息熵控制的路径选择及随机扰动策略，实现了算法的自适应调节。

### 5.12.1 算法设计

#### 5.12.1.1 信息熵的基本概念及性质

对于离散型随机变量，其信息熵为

$$S = -k \sum_{i=1}^n p_i \ln p_i \quad (5.12.1)$$

其中,  $p_i$  表示各状态发生的概率,  $p_i \geq 0$ , 且  $\sum_{i=1}^n p_i = 1$ 。

信息熵具有以下性质:

- (1) 对称性:  $p_1, p_2, \dots, p_n$  的顺序改变, 熵值不变, 即  $S(p_1, p_2, \dots, p_n) = S(p_n, p_{n-1}, \dots, p_1) = S(p_2, p_1, \dots, p_n)$ ;
- (2) 非负性:  $S(p_1, p_2, \dots, p_n) \geq 0$ ;
- (3) 可加性: 相对独立的状态, 其熵的和等于和的熵;
- (4) 极值性:  $p_i = 1/n$  时, 熵值最大, 其值为  $\ln n$ 。

### 5.12.1.2 改进策略

因基本蚁群算法中各路径的信息量具有不确定性, 对于全局来讲, 蚂蚁选择哪条路径也表现出一定的不确定性, 而信息熵本身为不确定性的一种度量<sup>[48]</sup>, 故可尝试将信息熵引入蚁群算法, 利用与蚁群算法运行过程相关的信息熵值表示选择过程中的不确定性。采用控制信息熵的值来控制路径选择和局部随机变异扰动的概率, 便可实现蚁群算法的自适应调节。在信息熵值达到某要求值时, 使算法停止搜索, 得到问题的解。令  $p_m = \frac{\tau_m(t)}{\sum_{m \in s} \tau_m(t)}$ , 其中  $p_m$  表示路径  $m$  上的信息量占总信息量的比例, 且  $p_m \geq 0$ 。这里采用了如下信息素更新规则

$$\tau(t) = (1 - \rho)\tau(t-1) + \rho\alpha(t) \quad (5.12.2)$$

将  $p_m$  代入信息熵公式 (5.12.1), 便可得到信息熵的值, 确定出蚂蚁选择路径  $m$  的确定性程度。此定义结合了蚁群算法本身的特点, 将算法的运算过程与信息熵值结合在一起, 便于实现算法自适应调节。在算法运行之初, 各路径上的信息量相等, 信息熵值最大, 但随着某路径上信息素的增强, 熵值将逐渐减小, 此时如不加以控制, 熵值将最终变为 0, 即最终只有一条路径信息素最多, 被误认为是最终解, 从而造成早熟。故这里引入

$$\alpha'(t) = \frac{S_{\max} - S(t)}{S_{\max}} \quad (5.12.3)$$

$$\beta'(t) = 1 - \frac{S_{\max} - S(t)}{2S_{\max}} \quad (5.12.4)$$

式中,  $\beta'(t)$  表示最优路径被保持的概率;  $\alpha'(t)$  表示可允许适当小范围内选择路径的蚂蚁占总蚁群的比例。这样就可利用熵值的变化保证在算法运行的早期使  $\alpha'(t)$  较小, 以尽可能地搜索解空间, 后期使  $\alpha'(t)$  较大, 相当于局部寻优能力增强, 避免了过早停滞; 对于  $\beta'(t)$ , 运行初期值较大, 保证了尽可能寻找最优路径, 后期则适当减少, 增加随机操作的作用, 以避免早熟现象的产生。基于信息熵蚁群算法的程序结构流程如图 5.38 所示。

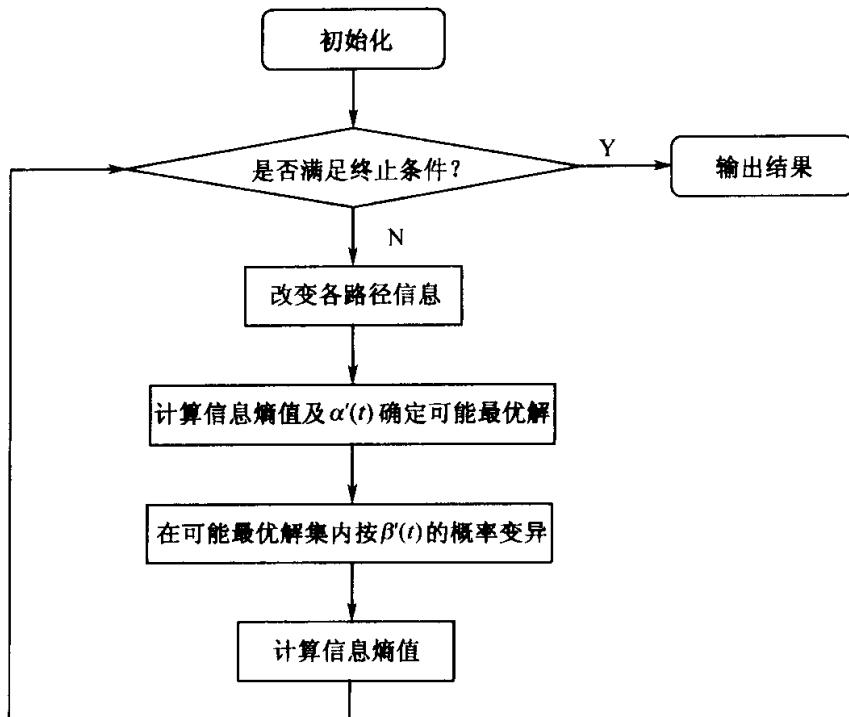


图 5.38 基于信息熵蚁群算法的程序结构流程

对于蚁群算法的终止原则，一般方法是直接规定最大迭代次数、计算所达到的数据精度或者计算机运行时间，但是此处采用了信息熵作为停机准则。因为某些复杂性问题很难确定最大迭代次数，根据算法运行过程中信息熵值的变化，可给定一稍大于 0 的信息熵值作为停机准则。

### 5.12.2 仿真算例

为了检验基于信息熵蚁群算法的求解性能，这里以 TSPLIB 中的 Bayes29TSP 和 Att48TSP 为仿真算例，并将其与基本蚁群算法的求解结果进行比较。设置  $\alpha=1.5$ ,  $\beta=4.0$ ,  $\rho=0.6$ ,  $Q=50$ ，停机准则为信息熵的值小于或等于 0.01，基于信息熵的蚁群算法和基本蚁群算法的计算结果如表 5.20 所示。

表 5.20 两种算法的计算结果比较

TSP 实例	基本蚁群算法		基于信息熵的蚁群算法	
	最短路径	时间(s)	最短路径	时间(s)
Bayes29	8	2180	5	2062
Att48	52	5494	33	5256

实验结果表明，利用信息熵控制蚁群的路径选择及随机扰动策略实现了算法

的自适应调节，较为有效地解决了早熟问题。改进后的蚁群算法具有较好的收敛性和稳定性，其鲁棒性很强，对其稍加修改后，便可用于求解其他组合优化问题。

## 5.13 本章小结

本章研究了离散域蚁群算法的若干改进策略，首先讨论了蚁群算法的自适应改进策略、去交叉局部优化策略、信息素扩散改进策略，然后介绍了多态蚁群算法、基于模式学习的小窗口蚁群算法、基于混合行为的蚁群算法、带聚类处理的蚁群算法、基于云模型理论的蚁群算法、具有感觉和知觉特征的蚁群算法以及具有随机扰动特性的蚁群算法，最后对一种基于信息熵的改进蚁群算法做了简要分析。虽然这些改进策略的侧重点和改进形式不同，但是其目的是相同的，即竭力避免蚁群算法陷入局部最优解，缩短搜索时间，进而提高蚁群算法的全局收敛性能。本章内容可为进一步改进蚁群算法提供借鉴。

## 参 考 文 献

- 1 Coloni A, Dorigo M, Maniezzo V, *et al.* Distributed optimization by ant colonies. Proceedings of the 1st European Conference on Artificial Life, 1991, 134~142
- 2 Dorigo M, Maniezzo V, Coloni A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- 3 Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66
- 4 Dorigo M, Caro G D, Gambardella L M. Ant algorithms for discrete optimization. Artificial Life, 1999, 5(2): 137~172
- 5 段海滨, 王道波, 朱家强等. 蚁群算法理论及应用研究的进展. 控制与决策, 2004, 19(12): 1321~1326, 1340
- 6 Gambardella L M, Dorigo M. Ant-Q: a reinforcement learning approach to the traveling salesman problem. Proceedings of the 12th International Conference on Machine Learning, 1995, 252~260
- 7 Dorigo M, Gambardella L M. A study of some properties of Ant-Q. Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, 1996, 656~665
- 8 Stützle T, Hoos H H. Improving the ant system: a detailed report on the MAX-MIN ant system. Technical Report AIDA-96-11, FG Intellektik, TH Darmstadt, 1996
- 9 Stützle T, Hoos H. MAX-MIN ant system and local search for the traveling salesman problem. Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, 1997, 309~314
- 10 Thomas S, Holger H H. MAX-MIN ant system. Future Generation Computer Systems, 2000, 16(8): 889~914
- 11 吴庆洪, 张纪会, 徐心和. 具有变异特征的蚁群算法. 计算机研究与发展, 1999, 36(10): 1240~1245
- 12 Watanabe I, Matsui S. Improving the performance of ACO algorithms by adaptive control of candidate

- set. Proceedings of the 2003 Congress on Evolutionary Computation, 2003, 2: 1355~1362
- 13 Gambardella L M, Dorigo M. Solving symmetric and asymmetric TSPs by ant colonies. Proceedings of the IEEE International Conference on Evolutionary Computation, 1996, 622~627
- 14 Dorigo M, Luca M. The Ant-Q: algorithm applied to the nuclear reload problem. Annals of Nuclear Energy, 2002, 29(12): 1455~1470
- 15 Wang Y, Xie J Y. Ant colony optimization for multicast routing. Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems, 2000, 54~57
- 16 王颖, 谢剑英. 一种自适应蚁群算法及其仿真研究. 系统仿真学报, 2002, 14(1): 31~33
- 17 覃刚力, 杨家本. 自适应调整信息素的蚁群算法. 信息与控制, 2002, 31(3): 198~201
- 18 黄岚, 王康平, 周春光等. 基于蚂蚁算法的混合方法求解旅行商问题. 吉林大学学报, 2002, 40 (4): 369~373
- 19 蔡利剑. 智能蚂蚁系统研究. 天津: 河北工业大学硕士学位论文, 2001
- 20 周培德. 求凸包顶点的一种算法. 北京理工大学学报, 1993, 13(1): 69~72
- 21 Dorigo M, Gambardella L M, Middendorf M, et al. Guest editorial: special section on ant colony optimization. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 317~319
- 22 黄国锐, 曹先彬, 王煦法. 基于信息素扩散的蚁群算法. 电子学报, 2004, 32(5): 865~868
- 23 徐精明, 曹先彬, 王煦法. 多态蚁群算法. 中国科学技术大学学报, 2005, 35(1): 59~65
- 24 吴坚, 王常禄. 中国蚂蚁. 北京: 中国林业出版社, 1995
- 25 全惠云, 文高进. 求解 TSP 的子空间遗传算法. 数学理论与应用, 2002, 22(1): 36~39
- 26 萧蕴诗, 李炳宇, 吴启迪. 求解 TSP 问题的模式学习并行蚁群算法. 控制与决策, 2004, 19(8): 885~888
- 27 萧蕴诗, 李炳宇. 小窗口蚁群算法. 计算机工程, 2003, 29(20): 143~145
- 28 李炳宇, 萧蕴诗. 基于模式求解旅行商问题的蚁群算法. 同济大学学报, 2003, 31(11): 1348~1352
- 29 Bullnheimer B, Hartl R F, Strauss C. A new rank-based version of the ant system: a computational study. Central European Journal for Operations Research and Economics, 1999, 7(1): 25~38
- 30 胡小兵, 黄席樾. 基于混合行为蚁群算法的研究. 控制与决策, 2005, 20(1): 69~71
- 31 胡小兵, 黄席樾. 对一类带聚类特征 TSP 问题的蚁群算法求解. 系统仿真学报, 2004, 16(2): 2683~2686
- 32 Chen Y F, Liu Y S, Fattah C A, et al. HDACC: a heuristic density-based ant colony clustering algorithm. Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004, 397~400
- 33 Liu S, Dou Z T, Li F, et al. A new ant colony clustering algorithm based on DBSCAN. Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, 2004, 3: 1491~1496
- 34 Wu B, Zheng Y, Liu S H, et al. CSIM: a document clustering algorithm based on swarm intelligence. Proceedings of the 2002 Congress on Evolutionary Computation, 2002, 1: 477~482
- 35 Peng Y Q, Hou X D, Liu S. The K-means clustering algorithm based on density and ant colony. Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003, 1: 457~460
- 36 Fan J H, Li D Y. Mining classification knowledge based on cloud models. Lecture Notes in Computer Science, 1999, 1574: 317~326
- 37 段海滨, 王道波, 于秀芬等. 基于云模型理论的蚁群算法改进研究. 哈尔滨工业大学学报, 2005, 37(1): 115~119
- 38 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 39 李德毅, 孟海军, 史雪梅. 隶属云和隶属云发生器. 计算机研究与发展, 1995: 32(6): 15~20

- 40 段海滨, 王道波. 一种快速全局优化的改进蚁群算法及仿真. 信息与控制, 2004, 33(2): 241~244
- 41 陈峻, 秦玲, 陈宏建等. 具有感觉和知觉特征的蚁群算法. 系统仿真学报, 2003, 15(10): 1418~1425
- 42 陈录生, 马剑侠. 新编心理学. 北京: 北京师范大学出版社, 1996
- 43 Dorigo M, Gambardella L M. Ant colonies for the traveling salesman problem. *Bio Systems*, 1997, 43(2): 73~81
- 44 Botee H M, Bonabeau E. Evolving ant colony optimization. *Complex Systems*, 1998, 1(2): 149~159
- 45 Talbi E G, Roux O, Fonlupt C, *et al.* Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, 2001, 17(4): 441~449
- 46 Gambardella L M, Dorigo M. An ant colony system hybridized with a new local search for the sequential ordering problem. *Informs Journal on Computing*, 2000, 12(3): 237~255
- 47 郝晋, 石立宝, 周家启. 求解复杂 TSP 问题的随机扰动蚁群算法. 系统工程理论与实践, 2002, 25(9): 88~91, 136
- 48 许国志, 顾基发, 车宏安. 系统科学. 上海: 上海教育出版社, 2000
- 49 李万庆, 李彦苍. 求解复杂优化问题的基于信息熵的自适应蚁群算法. 数学的实践与认识, 2005, 35(2): 134~139

# 第6章 连续域蚁群算法的改进研究

## 6.1 引言

在离散域组合优化问题的求解中，问题各分量的不同组合对应于多维离散域内的各个点，其中每个点的每一维分量对应于待优化问题的各个分量。离散域内优化问题的求解目标是在给定点集中设定相应的搜索算法，以使与问题最优解相对应的点（或点集）以递增的概率被选中，并最终收敛于与问题最优解相对应的点（或点集）。以典型的离散域优化问题——TSP为例，求解 TSP 的目标是，在总数为  $n!$  的离散点（即 TSP 的可行解总数）中，以较小的搜索代价寻求最短路径所对应的点（或点集）。

在离散域组合优化问题中，蚁群算法的信息量留存、增减和最优解的选取都是通过离散的点状分布求解方式来进行的；而在连续域优化问题的求解中，其解空间是一种区域性的表示方式，而不是以离散的点集来表示的。因此，连续域寻优蚁群算法与离散域（以 TSP 为例）寻优蚁群算法之间主要存在着以下不同之处。

(1) 从优化目标来说，求解 TSP 的蚁群算法要求所搜索出的路径最短且封闭；而用于连续域寻优问题的蚁群算法要求所求问题的目标函数值达到最优，目标函数中包含各蚂蚁所走过的所有节点的信息以及系统当前的性能指标信息。

(2) 从信息素更新策略来说，求解 TSP 的蚁群算法是根据路径长度来修正信息量，在求解过程中，信息素是遗留在两个城市之间的路径上，每一步求解过程中的蚁群信息素留存方式只是针对离散的点或点集分量；而用于连续域寻优问题的蚁群算法将根据目标函数值来修正信息量，在求解过程中，信息素物质则是遗留在蚂蚁所走过的每个节点上，每一步求解过程中的信息素留存方式在对当前蚁群所处点集产生影响的同时，对这些点的周围区域也产生相应的影响。

(3) 从寻优方式来说，由于连续域问题求解的蚁群信息留存及影响范围是区间性的，而非点状分布，所以在连续域问题求解中，蚁群判断寻优方式还应考虑总体信息量与蚂蚁个体当前位置所对应特定区间内的信息量累计比较值。

(4) 从行进方式来说，蚁群在连续解空间中的行进方式不同于离散解空间点集之间跳变的行进方式，而应是一种微调式的行进方式。

Bilchev G A 等<sup>[1]</sup>最早提出了一种连续蚁群算法，求解问题时先使用遗传算法对解空间进行全局搜索，然后利用蚁群算法对所得结果进行局部优化；高尚

等<sup>[2]</sup>提出了一种基于网格划分策略的连续域蚁群算法，该算法与网格划分法的不同之处在于前者利用了每一点的信息，而后者只利用了最小值的信息；Wang L 等<sup>[3~7]</sup>将离散域蚁群算法中的“信息量留存”过程拓展为连续域中的“信息量分布函数”，并定义了应用于连续函数寻优问题的改进蚁群算法；Li Y J 等<sup>[8~10]</sup>在借鉴遗传算法求解连续域优化问题编码方法、精英策略以及混合算法中的区域搜索思想的基础上，提出了一种用于连续域优化问题求解的自适应蚁群算法；段海滨等<sup>[11~14]</sup>提出了一种基于网格划分策略的自适应连续域蚁群算法；陈峻等<sup>[15,16]</sup>将所求问题解的每一分量的可能值组成一个动态的候选组，并记录候选组中每一可能值的信息量，进而提出了一种基于交叉变异操作的连续域蚁群算法；杨勇等<sup>[17,18]</sup>提出了一种用于求解连续域优化问题的嵌入确定性搜索蚁群算法，该算法在全局搜索过程中，利用信息素强度和启发式函数确定蚂蚁移动方向，而在局部搜索过程中嵌入了确定性搜索，以改善寻优性能，加快收敛速度；Dréo J 等<sup>[19,20]</sup>提出了一种基于密集非递阶的连续交互式蚁群算法 (continuous interacting ant colony algorithm, CIACA)，该算法通过修改信息素的留存方式和行走规则，并运用信息素交流和直接通信两种方式来指导蚂蚁寻优；Pourtakdoust S H 等<sup>[21]</sup>提出了一种仅依赖信息素的连续域蚁群算法；张勇德等<sup>[22]</sup>提出了一种用于求解带有约束条件的多目标函数优化问题的连续域蚁群算法，该算法将信息素交流和基于全局最优经验指导这两种寻优方式相结合，将当前发现的所有非支配解保存起来，进而用这些解来指导蚂蚁朝着分布较为稀疏的区域进行寻优，以保证解的分布性能，并提高了算法的收敛速度；Wen Y 等<sup>[23, 24]</sup>提出了一种结合遗传优化的动态窗口蚁群算法，该算法对于多变量强非线性的大规模复杂多阶段决策问题具有优良的求解性能。

## 6.2 基于网格划分策略的连续域蚁群算法

网格划分就是在变量区域内打网格，在网格点上求约束函数与目标函数的值，对于满足约束条件的点，再比较其目标函数的大小，从中选择较小者，并把该网格点作为一次迭代的结果；然后在求出的点附近将分点加密，再打网格，并重复前述计算与比较，直到网格的间距小于预先给定的精度。基于这种网格划分思想，本节研究了一种基于网格划分策略的连续域蚁群算法<sup>[2]</sup>。

### 6.2.1 无约束非线性最优化问题

假设要求解的无约束非线性最优化问题为

$$\min f(x_1, x_2, \dots, x_n) \quad (6.2.1)$$

## 6.2.2 算法设计

基于网格划分策略的连续域蚁群算法的思路为：首先可根据所求连续域优化问题的性质估计出所求变量的取值范围  $x_{j\text{lower}} \leq x_j \leq x_{j\text{upper}}$  ( $j = 1, 2, 3, \dots, n$ )。在变量区域内打网格，空间上的网格点对应于一个状态，蚂蚁在各个空间网格点之间移动，并根据各网格点的目标函数值留下不同的信息量，以此影响下一批蚂蚁的移动方向。循环一段时间后，相邻节点间的目标函数差值（即评价函数值）小的网格点信息量比较大，然后找出信息量较大的空间网格点，并缩小变量范围，在此点附近进行蚁群移动。不断重复这一过程，直到满足算法的停止条件为止。

假设参数变量分成  $N$  等份，从而可把  $n$  个变量变成  $n$  级决策问题，每一级有  $N+1$  个节点，这样共有  $(N+1) \times n$  个节点从第 1 级到第  $n$  级之间连接在一起，组成了解空间内的一个解，如图 6.1 所示。

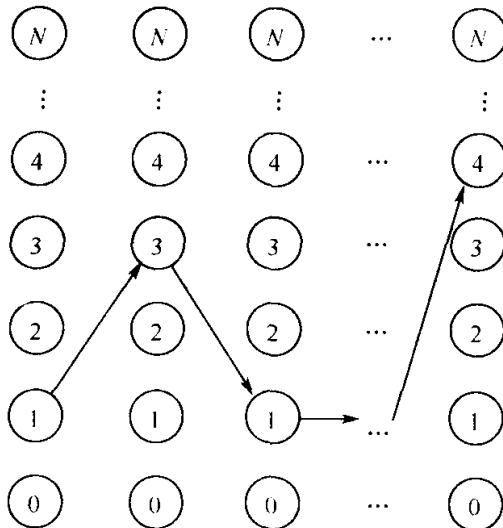


图 6.1 状态空间解

由图 6.1 可见，状态空间所示的状态为  $(1, 3, 1, \dots, 4)$ ，则其所对应解为

$$\begin{aligned}
 (x_1, x_2, x_3, \dots, x_n) = & \left( x_{1\text{lower}} + \frac{x_{1\text{upper}} - x_{1\text{lower}}}{N} \times 1, \right. \\
 & x_{2\text{lower}} + \frac{x_{2\text{upper}} - x_{2\text{lower}}}{N} \times 3, \\
 & x_{3\text{lower}} + \frac{x_{3\text{upper}} - x_{3\text{lower}}}{N} \times 1, \dots, \\
 & \left. x_{n\text{lower}} + \frac{x_{n\text{upper}} - x_{n\text{lower}}}{N} \times 4 \right)
 \end{aligned} \tag{6.2.2}$$

蚂蚁从第 1 级到第  $N$  级之间的状态转移概率可按下式进行计算

$$P_{ij} = \frac{\tau_{ij}}{\sum_{i=1}^N \tau_{ij}} \quad (6.2.3)$$

假设  $\tau_{ij}$  为第  $j$  级第  $i$  个节点的信息量，其更新方程为

$$\tau_{ij}^{\text{new}} = (1 - \rho) \tau_{ij}^{\text{old}} + \frac{Q}{f} \quad (6.2.4)$$

式中， $f$  为目标函数值。

改进后蚁群算法的具体实现步骤如下：

- (1) 估计出各变量的取值范围： $x_{j\text{lower}} \leq x_j \leq x_{j\text{upper}}$  ( $j=1, 2, \dots, n$ )。
- (2) 将各变量分成  $N$  等份， $h_j = \frac{x_{j\text{upper}} - x_{j\text{lower}}}{N}$  ( $j=1, 2, \dots, n$ )。
- (3) 若  $\max(h_1, h_2, \dots, h_n) < \epsilon$ ，则算法停止，最优解为  $x_j^* = \frac{x_{j\text{lower}} + x_{j\text{upper}}}{2}$  ( $j=1, 2, \dots, n$ )；否则跳转到第 (4) 步。
- (4) 循环次数  $N_c \leftarrow 0$ ，给  $\tau_{ij}$  矩阵赋相同的数值，设置  $Q, \rho, N_{c_{\text{max}}}$  的初始值。
- (5) 假设蚂蚁数为 num\_ant，对每只蚂蚁按公式 (6.2.3) 选择下一节点。
- (6) 按更新方程修改信息量， $N_c \leftarrow N_c + 1$ 。
- (7) 若  $N_c < N_{c_{\text{max}}}$ ，则跳转到第 (5) 步；否则，找出  $\tau_{ij}$  矩阵中每列最大的元素所对应的行 ( $m_1, m_2, \dots, m_n$ )，并缩小变量的取值范围： $x_{j\text{lower}} \leftarrow x_{j\text{lower}} + (m_j - \Delta) h_j$ ， $x_{j\text{upper}} \leftarrow x_{j\text{lower}} + (m_j + \Delta) h_j$ ，其中  $j = 1, 2, \dots, n$ ，跳转到第 (2) 步。

### 6.2.3 仿真算例

对于如下典型的连续域优化问题



图 6.2 改进蚁群算法的仿真过程

$$\min f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2.2)^2 + 1 \quad (6.2.5)$$

设置  $[x_{1\text{lower}}, x_{1\text{upper}}] = [0, 2]$ ,  $[x_{2\text{lower}}, x_{2\text{upper}}] = [1, 3]$ ,  $\text{num\_ant} = 20$ ,  $\epsilon = 0.001$ ,  $N = 10$ ,  $Q = 10$ ,  $N_{c_{\max}} = 100$ , 基于网格划分策略的连续域蚁群算法的仿真计算过程如图 6.2 所示。

图 6.2 中的“0”表示迭代过程中变量范围的中点值。算法最后求得的最优解  $(x_1, x_2) = (1.0366, 2.1928)$ , 与真实解  $[x_1^*, x_2^*] = [1, 2.2]$  很接近。

### 6.3 基于信息量分布函数的连续域蚁群算法

在连续域优化问题的求解中, 各单蚁智能体通过散布与其所在解空间位置优劣程度相关的信息量分布函数, 对蚁群的总体运动方向做出影响, 而蚁群的总体运动方向是在特定区域内, 对整个蚁群的信息量分布状态进行考察之后决定的<sup>[7]</sup>。蚁群运动的总体效果反映在连续解空间内, 并逐步收敛到最优解所在的邻近区域, 各单蚁的信息量分布函数对整个解空间所处区域均有影响, 其影响程度随各单蚁所在解空间位置距离的增加而递减。

本节通过将离散域蚁群算法中的“信息量留存”过程拓展为连续域内的“信息量分布函数”, 进而利用改进后的蚁群算法求解连续域优化问题。

#### 6.3.1 算法设计

连续域内蚁群算法的寻优过程在蚁群初始分布后, 还应包括信息量分布函数给定、信息量分布状态分析、蚁群移动方向决策等循环过程。

这里以一维空间函数  $y=f(x)$  的最大值(最小值)寻优为例, 对一维连续域内蚁群算法的函数优化问题进行了研究。而对于多维空间内的函数寻优, 可在此基础上作相应扩展即可。此处所定义的基于信息量分布函数的连续域蚁群算法如下。

(1) 将蚁群在解空间内按一定方式作初始分布。

首先根据问题定义域的大小决定合适的蚁群规模, 即蚂蚁数目  $N$  的大小。然后将问题的定义域进行  $N$  等分, 并在  $N$  个子区间的中部放置一只单蚁  $i (i=1 \sim N)$ 。而每只单蚁又带有一个随其坐标位置变化的移动子区间, 该单蚁则处于移动子区间的正中。各移动子区间的长度与问题定义域的  $1/N$  相等, 即将定义域  $N$  等分后所得的子区间长度相等。当各单蚁处于各子区间的中间位置时, 定义各子区间内的蚂蚁数目为 1。当各单蚁移动时, 根据其所带移动子区间与相邻两子区间的重叠程度变化, 定义这两个相邻子区间内实际蚂蚁数目变化。

根据以上定义可知, 如果问题的定义域为  $[\text{Start}, \text{End}]$ , 则当蚂蚁数目为  $N$

时，各子区间长度为

$$D_{RL} = \frac{\text{End} - \text{Start}}{N} \quad (6.3.1)$$

由此，每只单蚁所带的移动子区间长度  $D_{MRL} = D_{RL}$ ，而蚁群的初始坐标分布为

$$x_i = \text{Start} + \left( \frac{i}{N} - \frac{1}{2} \right) D_{RL} \quad (6.3.2)$$

其所处子区间  $i$  的左边界为

$$x_{il} = \text{Start} + (i - 1) D_{RL} \quad (6.3.3)$$

右边界为

$$x_{iR} = \text{Start} + i D_{RL} \quad (6.3.4)$$

当单蚁移动  $\Delta x$  时，由于相邻子区间与其所带移动区间的重合度变化  $\Delta x$ ，则定义相邻两区间内相应于此单蚁移动的实际蚂蚁数目  $N_{iR}$  的变化为

$$\Delta n = \frac{\Delta x}{D_{MRL}} = \frac{\Delta x}{D_{RL}} \quad (6.3.5)$$

即向右移动时，右边子区间内的实际蚂蚁数目增加  $\Delta n$ ，而左边子区间内的实际蚂蚁数目减少  $\Delta n$ ；向左移动时，情况则与之相反。

(2) 根据蚁群所处解空间位置的优劣决定当前蚁群的信息量分布。

根据蚁群当前位置  $x_i$  处函数值  $f(x_i)$  的大小，按寻优问题类别的不同，决定其所留下的相应信息量分布函数的峰值  $M_i$  的大小，并给出相应的信息量分布函数。例如特定区间内的函数最小值寻优，可按照下式定义相应的信息量分布函数峰值

$$M_i = C - f(x_i) \quad (6.3.6)$$

式中， $C$  为根据具体  $f(x_i)$  的大体范围所设定的常数，且满足  $C > f(x_i)$ 。这样，对于较小的函数值，其信息量分布函数的峰值反而大。再如函数最大值的寻优，当  $f(x_i) > 0$  时，则可定义

$$M_i = C_1 f(x_i) \quad (6.3.7)$$

式中， $C_1$  为根据具体问题而设定的正常数。当  $f(x_i) < 0$  时，可定义

$$M_i = \frac{C_3}{C_2 - f(x_i)} \quad (6.3.8)$$

$C_2$  和  $C_3$  的设定同上。对于一维空间内的连续函数寻优问题，可定义单蚁所对应的信息量分布函数为

$$T_i(x) = \frac{M_i e^{-k_i(x-x_i)}}{[1 + e^{-k_i(x-x_i)}]^2} \quad (6.3.9)$$

这样的信息量分布函数呈草帽形，其峰值为  $M_i$ ，中心点偏移值为  $x_i$ ，根据实际问题所定义的波形压缩系数为  $k_i$ 。

(3) 根据当前蚁群散布的总信息量分布和上一循环中信息量的遗留及挥发情

况，决定各子区间内应有的蚂蚁数目。

首先，按照下式求得当前蚁群散布的总信息量分布函数在各子区间内的积分值

$$IN_i = \int_{x_{iL}}^{x_{iR}} \sum_{i=1}^N T_i(x) dx \quad (6.3.10)$$

各子区间实际总信息量为

$$I_i = IN_i + \eta \cdot I_{iLast} - E_v \quad (6.3.11)$$

它是当前蚁群在该子区间内散布的信息量  $IN_i$  加上上一次总信息量的遗留部分 ( $\eta I_{iLast}$ ,  $\eta$  为信息量留存系数)，再与所设定的信息素挥发常熟  $E_v$  相减所得的结果。然后，按照下式求取实际总信息量在整个问题区间的总和

$$I_\Sigma = \sum_{i=1}^N I_i \quad (6.3.12)$$

由此，根据各子区间实际总信息量  $I_i$  占  $I_\Sigma$  的比例，便可按照下式求得当前蚁群分布条件下各子区间应有的蚂蚁数目

$$N_{iM} = \frac{I_i}{I_\Sigma} N \quad (6.3.13)$$

(4) 根据各子区间内应有的蚁群分布与当前蚁群分布之间的差别，决定蚁群的移动方向，并加以移动。

首先，根据已求得的各子区间内应有的蚂蚁数目  $N_{iM}$ ，以所考察之蚁当前所处区间为界进行求和操作，求出被考察之蚁所处区间  $i$  以左应有蚂蚁数目之和  $N_{iML}$  及所处区间  $i$  以右应有蚂蚁数目之和  $N_{iMR}$ ，并以此作为判定被考察之蚁移动方向的依据条件。其中

$$N_{iML} = \sum_{j=1}^{i-1} N_{jM}, \quad N_{iMR} = \sum_{j=i+1}^N N_{jM} \quad (6.3.14)$$

这里，还需根据已知各子区间内实际蚂蚁数目  $N_{iR}$ ，以所考察之蚁当前所处区间为界进行求和操作，求出被考察之蚁所处区间  $i$  以左实际蚂蚁数目之和  $N_{iRL}$  及所处区间  $i$  以右实际蚂蚁数目之和  $N_{iRR}$ 。其中

$$N_{iRL} = \sum_{j=1}^{i-1} N_{jR}, \quad N_{iRR} = \sum_{j=i+1}^N N_{jR} \quad (6.3.15)$$

然后，根据被考察之蚁所处区域及其左右实际蚂蚁数目与应有蚂蚁数目之间的差别，决定该蚁的运动方向，并作  $\Delta x$  的坐标变化。其运动规则如表 6.1 所示。

表 6.1 被考察之蚁坐标变化运动规则

规则	$N_{iRL} ? N_{iML}$	$N_{iR} ? N_{iM}$	$N_{iRR} ? N_{iMR}$	被考察之蚁坐标变化值
1	<	=	>	$-\Delta x$
2	>	=	<	$+\Delta x$

续表

规则	$N_{iRL} ? N_{iML}$	$N_{iR} ? N_{iM}$	$N_{iRR} ? N_{iMR}$	被考察之蚁坐标变化值
3	=	>	<	$+\Delta x$
4	>	>	<	$+\Delta x$
5	<	>	=	$-\Delta x$
6	<	>	>	$-\Delta x$
7	<	>	<	$\pm \Delta x$

在完成一次蚁群整体移动之后，又可返回第（2）步，进行相应的信息量分布和蚁群移动操作。如此循环往复，直至产生最优解为止。

### 6.3.2 仿真算例

假设所求解的连续函数为

$$y = f(x) = 5x^6 - 36x^5 + 82.5x^4 - 60x^3 + 36 \quad (6.3.16)$$

显然，公式 (6.3.16) 所示的函数为多极值函数，已知其局部最优点位于  $x=1$ ,  $x=2$  和  $x=3$  处。在  $x=2$  处函数取得局部最大值，在  $x=1$  和  $x=3$  处函数取得局部最小值，其中在  $x=3$  处为全局最小值。函数的寻优区间为  $[0, 3.5]$ ，如图 6.3 所示。

为表征算法的收敛特征，定义该算例中蚁群的总体求解误差为

$$E_{\Sigma} = \sum_{i=1}^N |x_i - 3| \quad (6.3.17)$$

这里需要选择的主要参数有：蚁群规模（蚂蚁数目  $N$ ），信息量分布函数的压缩系数  $k_i$ ，系数  $\eta$ ，寻优步长  $\Delta x$ ，信息素挥发常数  $E_V$  等。各函数峰值  $M_i$  取为  $[C - f(x_i)]$ 。

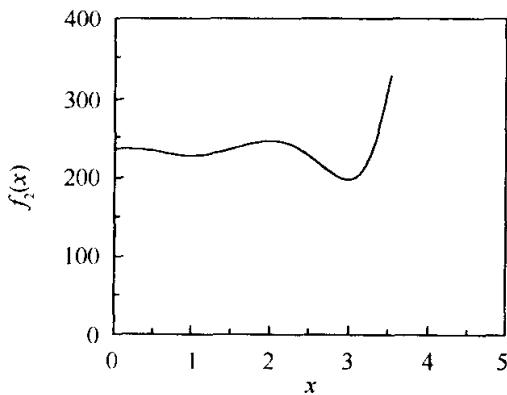


图 6.3 函数的寻优区间

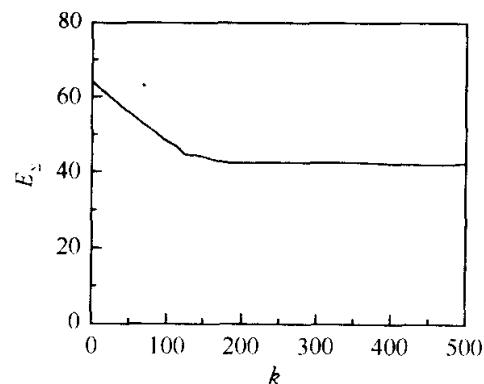


图 6.4 连续域蚁群算法第一次寻优过程的误差变化过程

总体而言，整个蚁群都能从初始的均匀分布状态按所给算法以一定精度趋于

最优解。当求解空间随寻优过程逐步缩小时，经 5 次寻优过程后，其最终的总体求解误差  $E_{\Sigma}=0.037$ ，即每只单蚁的平均求解误差为 0.14%。在 5 次寻优过程中， $N=9$ ，循环次数均取 500， $\eta$  均取 0.01， $E_V$  均取 50， $M_i$  表达式中的常数  $C=150$ 。

这里以第一次寻优过程为例，给出所定义蚁群算法的总体求解误差变化过程如图 6.4 所示，其中  $k$  代表蚂蚁数目。

典型的 5 次寻优过程的蚁群算法参数设定如表 6.2 所示。

表 6.2 连续域蚁群算法的参数设定

次 数	寻优区间	压缩系数 $k_i$	寻优步长 $\Delta x$	总体求解误差 $E_{\Sigma}$	蚁群的最终分布
1	[0, 3.5]	6.00	0.008	1.290	[2.6, 3.2]
2	[2.4, 3.4]	1.80	0.008	0.884	[2.8, 3.1]
3	[2.7, 3.2]	2.80	0.004	0.297	[2.95, 3.05]
4	[2.9, 3.1]	6.90	0.004	0.096	[2.97, 3.02]
5	[2.95, 3.05]	14.05	0.003	0.037	[2.99, 3.01]

## 6.4 连续域优化问题的自适应蚁群算法

本节首先研究了一种用于连续域优化问题的自适应蚁群算法，该算法采用了新的基于目标函数值的启发式信息素分配策略，并借鉴了遗传算法求解连续域优化问题的编码方法和精英策略<sup>[25]</sup>，以及混合算法中的区域搜索思想<sup>[26]</sup>。随后介绍了一种基于网格划分策略的自适应连续域蚁群算法，并将这两种自适应蚁群算法在同一初始条件下进行了仿真实验。

### 6.4.1 算法设计

#### 6.4.1.1 连续域优化问题到有向图搜索问题的映射

设待求解的连续域优化问题为

$$\begin{cases} \min J = f(x) \\ x_{\min} \leqslant x \leqslant x_{\max}, x \in R^n \end{cases} \quad (6.4.1)$$

可将候选解  $x$  用字长为  $N$  的二进制数表示为

$$x \Leftrightarrow \{b_N, b_{N-1}, \dots, b_1\} \quad (6.4.2)$$

式中， $b_j \in \{0, 1\}$ ， $j = 1, 2, \dots, N$ ， $b_1$  为最低位， $b_N$  为最高位。定义有向图  $(C, \Phi)$ ，其中  $C=(V, S)$ ，节点组合

$$V = \{v_s, v_N^0, v_{N-1}^0, \dots, v_1^0, v_N^1, v_{N-1}^1, \dots, v_1^1\} \quad (6.4.3)$$

集合中节点  $v_s$  称为起始节点，节点  $v_i^0$  和  $v_i^1$  分别用于表示公式 (6.4.2) 中

二进制位  $b_j$  取值为 0 和 1 时的状态。定义有向弧集合为

$$S = \{(v_N^0, v_{N-1}^0), \dots, (v_j^0, v_{j-1}^0), (v_j^0, v_{j-1}^1), (v_j^1, v_{j-1}^0), (v_j^1, v_{j-1}^1), \dots, (v_2^1, v_1^1)\} \quad (6.4.4)$$

即对于  $j=2, 3, \dots, N$ , 在所有节点  $v_j^0$  和  $v_j^1$  处, 分别有且仅有同时指向节点  $v_{j-1}^0$  和  $v_{j-1}^1$  的有向弧。

蚁群算法搜索时, 蚂蚁  $A_i$  从起始点  $v_s$  出发, 沿有向弧组成的路径历经  $N$  个节点, 构成一条由节点序列  $\{v_N^{i_N}, v_{N-1}^{i_{N-1}}, \dots, v_1^{i_1} \mid i_1, \dots, i_N \in [0, 1]\}$  组成的路径  $\omega_i$ , 对应于二进制串  $\{b_N^i, b_{N-1}^i, \dots, b_1^i\}$ , 经数制转换为一个候选解  $x_i$ , 即有映射

$$\Phi(\omega_i) = \{b_N^i, b_{N-1}^i, \dots, b_1^i\} \Leftrightarrow x_i = \frac{X_i}{2^N - 1} \cdot (x_{\max} - x_{\min}) + x_{\min} \quad (6.4.5)$$

式中,  $X_i$  为串  $\{b_N^i, b_{N-1}^i, \dots, b_1^i\}$  对应的二进制值。由此, 保证了在这种映射中, 所有的搜索路径都满足约束条件  $x_{\min} \leq x \leq x_{\max}$ 。

#### 6.4.1.2 最佳路径信息素的增强策略

基本蚁群算法的信息素更新策略所存在的一个主要问题是, 当与最优解所对应路径还未被穿越时, 该路径上的信息素只挥发而得不到增强, 其值会越来越小。这样, 在下一次搜索中, 最优解所对应路径上的节点被选取的概率相对较小, 从而导致错误的引导信息, 造成大量的无效搜索<sup>[8, 10]</sup>。基于此, 这里采用了以下信息素更新策略。

设  $W^*(t)$  为第  $t$  个搜索周期的最佳路径, 并且与该路径所对应的目标函数值满足下式

$$f^*(t) \leq f^*(t-1) \quad (6.4.6)$$

式中,  $f^*(t)$  表示与最佳路径  $W^*(t)$  所对应的第  $t$  个搜索周期的最佳目标函数值。信息素的更新增量为

$$\Delta\varphi_{i,j}(t) = \begin{cases} \psi(f^*(t)), & \text{若 } (i, j) \in W_k^*(t) \\ 0, & \text{否则} \end{cases} \quad (6.4.7)$$

由此, 有下式成立

$$\varphi_{i,j}^*(t) \geq \varphi_{i,j}(t) \quad (6.4.8)$$

#### 6.4.1.3 基于位编码的信息素增强分配策略

在基本蚁群算法中, 信息量和路径弧段与其所对应的搜索顺序无关, 因此其本质上是一种信息素均匀分配的策略。根据连续域优化问题到有向图搜索问题的转换方法, 在路径的节点序列中, 顺序靠前的节点对应于解中二进制编码较高的位, 由此可能带来候选解的大幅度变化; 而顺序在后的节点之间路径上的信息素

影响恰好相反。从而这种信息素均匀分配策略导致下一搜索周期中，会在该路径所对应候选解的基础上进行新的搜索，且对于不同尺度的搜索步长具有相同的选取概率。而合理的搜索机制应该为：对于较好的候选解，下一次搜索应该倾向于在较小的区域进行，这种区域搜索策略可保证在概率意义上，搜索过程具有较好的收敛性<sup>[26]</sup>；而对于较差的解，应该在此基础上尽可能进行较大步长的搜索，从而保证算法进行扩展搜索空间内的大范围搜索，以保证解的全局最优性。

基于以上分析，这里提出了一种与路径弧段在搜索过程中顺序相关的信息素增量分配策略：假设第  $t$  个搜索周期中，搜索路径  $W_s(t)$  中与路径弧段  $(i, j)$  中节点  $i$  相对应的候选解  $x_s$  二进制编码的第  $k$  位， $f_s(t)$  为  $W_s(t)$  所对应的目标函数值。由此，根据在较好解附近进行较小尺度搜索的原则，可采用以下信息素更新策略

$$\Delta\varphi_{i,j}(t,k) = \frac{1}{1 + 2^{f_s^*(L-k)}} \quad (6.4.9)$$

式中， $L$  表示候选解二进制编码的编码长度。在搜索周期  $t$  结束时，若所得的候选解越好，其所对应的节点被搜索的概率就越大，则在搜索周期  $t+1$  中越倾向于在原候选解基础上，进行解空间内较小范围的搜索，避免失去以前搜索所提供的有效信息。

#### 6.4.1.4 自适应的蚁群搜索信息素更新方法

结合公式 (6.4.7) 和公式 (6.4.9)，可得一种自适应的蚁群搜索信息素更新方法<sup>[27]</sup>，具体如下式所示

$$\varphi_{i,j}(t+1,k) = \begin{cases} \lambda\varphi_{i,j}(t,k) + (1-\lambda) \frac{1}{1 + 2^{f^*(L-k)}}, & \text{若 } f^*(t) \leq f^*(t-1) \\ \lambda\varphi_{i,j}(t,k), & \text{否则} \end{cases} \quad (6.4.10)$$

这种更新策略保证了在搜索过程中，搜索路径上的信息素分配与解的最优性成正相关关系，越好的候选解所对应的路径上有越多的信息素，从而为后续搜索提供正确的启发式信息。同时，通过根据位编码信息对当前最优路径上的弧段合理地分配信息素，保证算法在下一搜索周期中能以较大的概率搜索到最好解。这样，算法就会从过去的搜索中增强了自学习能力，进而提高了搜索效率。

#### 6.4.2 基于网格划分策略的自适应连续域蚁群算法<sup>[11, 12]</sup>

这里，对基于网格划分策略的连续域蚁群算法做了以下三点改进：

- (1) 蚁群算法应用过程中易出现的停滞和扩散问题不容忽视，将各条寻优路

径上可能的残留信息素数量限制在 $[\tau_{\min}, \tau_{\max}]$ 。每次循环结束后保留最优路径，一个循环中只有路径最短的蚂蚁才有权修改 $\tau_{ij}(t)$ ，并按照下式进行阈值判断选择

$$\tau_{ij}(t+n) = \begin{cases} \tau_{\min}, & \text{若 } \tau_{ij}(t) \leq \tau_{\min} \\ \tau_{ij}(t), & \text{若 } \tau_{\min} < \tau_{ij}(t) \leq \tau_{\max} \\ \tau_{\max}, & \text{若 } \tau_{ij}(t) > \tau_{\max} \end{cases} \quad (6.4.11)$$

(2) 为了提高蚁群算法的求解效率，避免计算结果陷入局部最优状态，对信息素挥发系数 $\rho$ 值采取如下式所示的自适应控制策略

$$\rho(t+1) = \begin{cases} 0.9 \cdot \rho(t), & \text{若 } 0.9 \cdot \rho(t) > \rho_{\min} \\ \rho_{\min}, & \text{否则} \end{cases} \quad (6.4.12)$$

(3) 为了提高搜索速度，该寻优策略中采用了两只蚂蚁从两个极限点同时搜索的优化方案，这种并行处理策略可有效地提高蚁群算法的全局收敛速度。

### 6.4.3 仿真算例

这里以连续域优化问题中一个典型的指数正弦组合优化问题为例，分别对本章第 6.4.1 小节所提出的自适应蚁群算法和基于网格划分策略的自适应连续域蚁群算法进行了仿真研究：

$$\begin{cases} \min J = 5e^{-0.5x} \sin(30x) + e^{0.2x} \sin(20x) + 6 \\ x \in [0, 8] \end{cases} \quad (6.4.13)$$

上述连续域优化问题的目标函数曲线如图 6.5 所示。

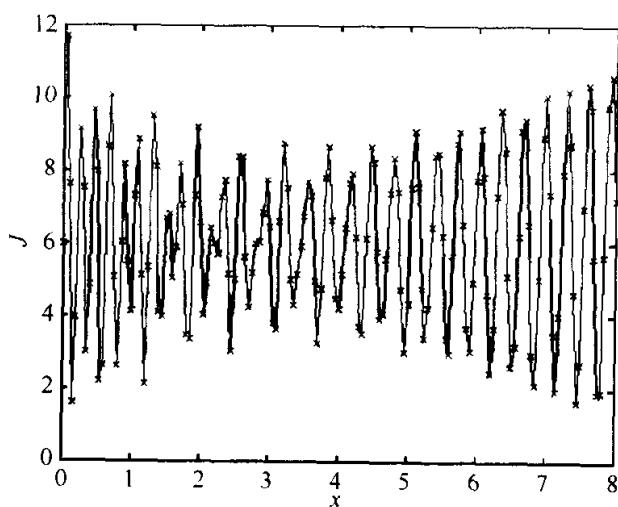


图 6.5 连续域优化问题的目标函数曲线

由图 6.5 可见，该目标函数具有多个局部极小点，因此在连续域优化问题中非常具有代表性。设置 $m=10$ ,  $\alpha=1$ ,  $\beta=2.5$ ,  $\rho_{\min}=0.2$ ,  $\tau_{ij}=1$ ,  $\Delta\tau_{ij}(0)=0$ ,

$N_c=0$ ,  $N_{c_{\max}}=20$ ,  $Q=30$ 。仿真计算后将两种算法的实验结果进行了对比，其比较结果如表 6.3 所示。

表 6.3 自适应蚁群算法与网格划分自适应蚁群算法的实验结果比较

算法类型	最优目标值	蚁群规模	搜索周期	运算次数	minJ 均值	鲁棒性能(%)
自适应蚁群算法	1.3652	10	20	1000	1.4403	5.5011
网格划分自适应蚁群算法	1.3652	10	20	1000	1.4095	3.2449

由表 6.3 可见，在同样的初始条件下，基于网格划分策略的自适应连续域蚁群算法与本章第 6.4.1 小节所提出的自适应蚁群算法相比而言，前者在目标函数均值、鲁棒性能等方面比后者更加优良。

## 6.5 基于交叉变异操作的连续域蚁群算法

本节研究了一种基于交叉变异操作的连续域蚁群算法，该算法对解的每一分量的可能取值组成一个动态的候选组，并记录候选组中每一个可能值的信息量。在蚁群算法的每一次迭代中，首先根据信息量选择解分量的初值，然后使用交叉、变异操作来确定全局最优解的值。

### 6.5.1 算法设计

考虑如下有约束的非线性规划问题

$$\min G(x_1, x_2, \dots, x_n) \quad (6.5.1)$$

使得

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \geq b_i, i = 1, 2, \dots, r \quad (6.5.2)$$

这里目标函数  $G$  为任一非线性函数，约束条件构成一个凸区域。这里可采用不等式变换的方法求得包含这个凸区域的最小的  $n$  维立方体<sup>[26]</sup>。设该立方体为

$$l_i \leq x_i < u_i, i = 1, 2, \dots, n \quad (6.5.3)$$

设蚂蚁数目为  $m$ ，在选取  $m$  个初始解以后，其第  $i$  个分量的  $m$  个取值构成了该分量的候选组。这里将解的  $n$  个分量看成  $n$  个顶点，在第  $i$  个顶点到第  $i+1$  个顶点之间有  $m$  条连线，代表第  $i$  个分量候选组中的  $m$  个不同的候选值，并记其中第  $j$  条连线为  $(i, j)$ ，在  $t$  时刻  $(i, j)$  上的信息量记为  $\tau_{ij}(t)$ 。每只蚂蚁都从第一个顶点出发，按照一定的规则依次选择  $n-1$  条连线。到达第  $n$  个顶点后，再从  $m$  条连线中选取某一条连线到达终点。每只蚂蚁所走过的路径代表一个解，其  $n$  条连线表示它的  $n$  个分量。为了使解的分布具有多样性，在各个分量选取  $m$

个值后，对其实行交叉、变异操作，并将所得到的值作为新一代解的相应分量。在得到  $m$  个解后，要根据其适应度值更新各条边上的信息量。每一次迭代后，要对各个分量的候选组中的值动态更新，并在新产生的值和候选组的值中选取信息量较高的  $m$  个值作为新的候选组。重复上述迭代过程，直至满足算法停止条件为止。

综上所述，基于交叉变异操作的连续域蚁群算法具体步骤可描述如下<sup>[15,16]</sup>：

(1) 初始化。

随机产生  $m$  个初始解，计算这  $m$  个初始解的适应度，由这  $m$  个初始解得到各个分量值的候选组，并根据候选组中的值按其所在解的适应度计算其信息量。

(2) 迭代过程。

```

While not 结束条件 do
    ① For i=1 to n do //对 n 个分量循环
        For k=1 to m do //对 m 只蚂蚁循环
            第 k 只蚂蚁根据状态转移概率在第 i 个分量的候选组内选择该分量的初值;
            End for k
            对所选择的第 i 个分量的 m 个初值施行交叉、变异操作生成第 i 个分量的新一代 m 个不同的候选值，并加入候选组;
            End for i
            根据所选择的分量构成 m 个新一代解，并计算新一代解的适应度值;
    ② 修改各分量的候选组中各候选值的信息量;
    ③ 取各分量的候选组中的信息量较高的 m 个值作为新的候选组
End while

```

在上述算法中，蚂蚁在得到新一代的解以后，算法则相应地更新各分量候选组中相应值的信息量  $\tau_{ij}(t)$ ，并按下式更新  $\Delta\tau_{ij}^k$

$$\Delta\tau_{ij}^k = \begin{cases} W \cdot f_k, & \text{若蚂蚁 } k \text{ 的解的第 } i \text{ 个分量选中第 } j \text{ 个候选值} \\ 0, & \text{否则} \end{cases} \quad (6.5.4)$$

式中， $W$  为一常数； $f_k$  表示第  $k$  只蚂蚁所对应解的适应度。

(3) 自适应的交叉和变异操作。

上述算法的迭代过程中，对第  $i$  个分量所选择的  $m$  个初值进行了交叉、变异等操作。为了加速收敛并防止局部优化，可采用具有自适应性的交叉、变异操作，根据解的适应度值  $f$  可按照下式定义其相对适应度  $F$

$$F = \begin{cases} \frac{f - f_{\min}}{f_{\max} - f_{\min}}, & \text{若 } f_{\max} \neq f_{\min} \\ 1, & \text{否则} \end{cases} \quad (6.5.5)$$

式中， $f_{\max}$  和  $f_{\min}$  分别表示本代群体中最大和最小的适应度值。在交叉操作中，设  $x_i(1)$  和  $x_i(2)$  为进行交叉操作的第  $i$  个分量的两个初值，其所在解的适应度分

别为  $f_1$ 、 $f_2$ ，记  $f_{\text{ave}} = (f_1 + f_2)/2$ 。这里以  $f_{\text{ave}}$  来衡量  $x_i(1)$  和  $x_i(2)$  的整体适应度，也作为交叉操作所产生的结果  $x'_i(1)$  和  $x'_i(2)$  的适应度值。可根据  $f_{\text{ave}}$  相应的相对适应度  $F_{\text{ave}}$  来决定实际交叉概率  $p_c$ ，取  $p_c = p_{\text{cross}}(1 - F_{\text{ave}})$ ，其中  $p_{\text{cross}}$  为系统预设的交叉概率。这样，对于来自整体适应度较大的一对解的分量值，其实行交叉的概率较小；反之，实行交叉的概率就较大。随机产生  $p \in [0, 1]$ ，若  $p > p_c$ ，则进行交叉操作。取  $b = 2 \times (1 - F_{\text{ave}})$ ，产生两个随机数  $c_1, c_2 \in [-b, b]$ ，使  $c_1 + c_2 = 1$ ，将  $x_i(1)$  和  $x_i(2)$  作仿射组合产生交叉结果值  $x'_i(1)$  和  $x'_i(2)$  来替换  $x_i(1)$  和  $x_i(2)$ ，其表达式为

$$x'_i(1) = c_1 x_i(1) + c_2 x_i(2), \quad x'_i(2) = c_2 x_i(1) + c_1 x_i(2) \quad (6.5.6)$$

由此，对于来自整体适应度较大的一对解的分量值，它们实行交叉的幅度较小；反之亦然。

在对某个分量值  $x$  进行的变异操作中，可根据  $x$  所在解的相对适应度值  $F$  来决定实际变异概率  $p_m$ ，取  $p_m = p_{\text{mutate}}(1 - F)$ ，其中  $p_{\text{mutate}}$  表示算法中预设的变异概率。这样，对于来自整体适应度较大的解分量值实行变异的概率较小，可使较优的解分量尽可能予以保留。随机产生  $p \in [0, 1]$ ，若  $p > p_m$ ，则进行变异操作。设对  $x_i$  进行变异操作得到  $x'_i$ ，记  $x_i$  值的上、下界分别为  $u_i, l_i$ ，为保证交叉操作的结果  $x'_i$  仍然在子区间  $[l_i, u_k]$  中，设

$$d_i = \max\{u_i - x_i, x_i - l_k\} \quad (6.5.7)$$

取

$$x'_i = \begin{cases} x_i + \delta(F, t)d_i, & \text{若 } l_i - x_i \leq \delta(F, t)d_i \leq u_i - x_i \\ x_i - \delta(F, t)d_i, & \text{否则} \end{cases} \quad (6.5.8)$$

式中， $\delta(F, t)$  是一个  $[-1, 1]$  间的随机数，它趋近于 0 的概率随  $F$  和  $t$  的增大而增加，其表达式为

$$\delta(F, t) = r(1 - F)^{1+\lambda t} \quad (6.5.9)$$

式中， $r$  是一个  $[-1, 1]$  间的随机数， $\lambda$  为决定非一致性程度的参数，它具有调节局部搜索区域的作用，其取值可在  $[0.0001, 0.0003]$  之间。这样，对于来自相对适应度较大的解的分量值，其变异的区域较小，成为局部搜索；反之，变异的区域较大，则构成全局搜索。同时，随着迭代次数的增多，分量值的变异幅度逐渐变小，这样可使收敛过程在迭代次数较多时得到适当的控制，以加速收敛。

## 6.5.2 仿真算例

这里用基于交叉变异操作的连续域蚁群算法和遗传算法分别对文献 [28] 附录中所示的带约束非线性优化问题  $G_1 \sim G_5$  进行了测试。设置  $m = 50$ ， $p_{\text{cross}} = 0.9$ ， $p_{\text{mutate}} = 0.05$ 。对每一个问题进行 10 次计算，它们达到最优解平均所需的代数和计算时间如表 6.4 所示。

表 6.4 基于交叉变异操作的连续域蚁群算法与遗传算法的性能比较

问 题	达到最优解所需的代数		达到最优解所需的时间(s)	
	遗传算法	连续域蚁群算法	遗传算法	连续域蚁群算法
G <sub>1</sub>	923.99	675.33	73.68	66.85
G <sub>2</sub>	4845.71	3479.54	365.70	245.72
G <sub>3</sub>	5672.68	4946.42	478.06	371.53
G <sub>4</sub>	5284.01	3127.10	433.24	269.34
G <sub>5</sub>	4697.24	3766.49	336.38	252.49
平均	4284.726	3198.976	337.434	241.186

由表 6.4 可见, 对上述问题使用遗传算法平均需要 4284.726 次迭代才能达到最优解, 而基于交叉变异操作的连续域蚁群算法平均只需 3198.976 次迭代便可达到, 说明改进后蚁群算法具有较高的搜索较优解的能力, 大大节约了计算时间。

## 6.6 嵌入确定性搜索的连续域蚁群算法

本节研究了一种用于求解连续域优化问题的嵌入确定性搜索蚁群算法。该算法在全局搜索过程中, 利用信息素强度和启发式函数确定蚂蚁的移动方向; 而在局部搜索过程中, 嵌入了确定性搜索, 以改善寻优性能, 加快收敛速率。

### 6.6.1 算法设计

设优化函数为  $\text{Max}Z=f(X)$ ,  $m$  只蚂蚁随机分布在定义域内, 每只蚂蚁都有一个邻域, 其半径为  $r$ 。每只蚂蚁在自己的邻域内进行搜索, 当所有蚂蚁完成局部搜索后, 蚂蚁个体根据信息素强度和启发式函数在全局范围内进行移动<sup>[17]</sup>; 完成一次循环后, 则进行信息素更新。

#### 6.6.1.1 局部搜索

设新的位置点为  $X'$ , 如果新的位置值比原来目标函数值大, 则取新位置, 否则舍去。局部搜索是在半径为  $r$  的区域内进行的, 且  $r$  随着迭代次数的增加而减少。于是, 有

$$X_i = \begin{cases} X'_i, & \text{若 } f(X'_i) > f(X_i) \\ X_i, & \text{否则} \end{cases} \quad (6.6.1)$$

#### 6.6.1.2 全局搜索

设  $\text{Act}(i)$  为第  $i$  只蚂蚁选择的动作,  $f_{\text{avg}}$  为  $m$  只蚂蚁的目标函数平均值,

则有

$$\text{Act}(i) = \begin{cases} \text{全局随机搜索,} & \text{若 } f(X_i) < f_{\text{avg}}, q < q_0 \\ S, & \text{否则} \end{cases} \quad (6.6.2)$$

式中,  $0 < q < 1$ ,  $0 < q_0 < 1$ ,  $S$  按如下转移规则选择动作

$$p(i, j) = \frac{\tau(j) e^{-d_{ij}/T}}{\sum \tau(j) e^{-d_{ij}/T}} \quad (6.6.3)$$

式中,  $d_{ij} = f(X_i) - f(X_j)$ , 且当  $i \neq j$  时,  $d_{ij} < 0$ ; 而当  $i = j$  时,  $d_{ij} = 0$ 。公式 (6.6.3) 保证了第  $i$  只蚂蚁按概率向其他目标函数值更大的蚂蚁  $j$  的邻域移动, 其中系数  $T$  的大小决定了这个概率函数的斜率。

蚂蚁向某个信息素强度高的地方移动时, 可能会在转移路途中的一个随机地点发现新的食物源, 这里将其定义为有向随机转移。第  $i$  只蚂蚁向第  $j$  只蚂蚁的邻域转移的公式为

$$X_i = \begin{cases} X_j \text{ 的邻域取随机值,} & \text{若 } \rho < \rho_0 \\ \alpha X_j + (1 - \alpha) X_i, & \text{否则} \end{cases} \quad (6.6.4)$$

式中,  $0 < \rho < 1$ ,  $\rho_0 > 0$ ,  $\alpha < 1$ 。

### 6.6.1.3 信息素强度更新规则

全局搜索结束后, 要对信息素强度进行更新。如果有  $n$  只蚂蚁向蚂蚁  $j$  处移动 (包括有向随机搜索), 则有

$$\tau(j) = \beta \tau(j) + \sum_{i=1}^n \Delta \tau_i \quad (6.6.5)$$

式中,  $\Delta \tau_i = 1/f(X_i)$ ,  $0 < \beta < 1$  是遗忘因子。

### 6.6.1.4 嵌入确定性搜索

随机性搜索算法存在着求解效率较低、求解结果较分散等缺陷, 因此有必要引入确定性搜索, 对其加以改进。这里考虑使用确定性搜索中的直接法, 直接法只利用函数信息而不需要利用导数信息, 甚至不要求函数连续, 适用面较广, 易于编程, 从而避免了复杂的计算。常用的直接法包括网格法、模式搜索法、二坐标轮换法等, 本节采用了模式搜索法中的步长加速法<sup>[29]</sup>。

#### 1. 步长加速法

步长加速法是在坐标轮换法的基础上发展起来的, 包括探测性搜索和模式性移动两部分。首先依次沿各坐标方向探索, 称之为探测性搜索; 然后经此探测后求得目标函数的变化规律, 从而确定搜索方向并沿此方向移动, 称之为模式移

动。重复以上两步，直到探测步长小于充分小的正数  $\epsilon$  为止。其具体步骤如下：

(1) 选定初始点  $X^0$ , 初始步长  $t^0$ ,  $\epsilon$ , 置  $k=0$ ,  $X^0 = [x_1^0, x_2^0, \dots, x_i^0, \dots, x_n^0]^T$ ,  $t^0 = [t_1^0, t_2^0, \dots, t_i^0, \dots, t_n^0]^T$ 。如果已知变量的上下界,  $t^0$  中的步长可取为

$$t_i^0 = 0.1(\bar{x}_i - x_i), \quad i = 1, 2, \dots, n \quad (6.6.6)$$

(2) 探测搜索：记  $x^{k,i}$  为第  $k$  次沿  $i$  坐标方向 ( $e_i$ ) 探测所得点, 对于不同情况, 将有如下几种选择

$$X^{k,i} = \begin{cases} X^{k,i-1} + t_i^k e_i, & \text{若 } f(X^{k,i-1} + t_i^k e_i) < f(X^{k,i-1}) \\ X^{k,i-1} - t_i^k e_i, & \text{若 } f(X^{k,i-1} - t_i^k e_i) < f(X^{k,i-1}) \leq f(X^{k,i-1} + t_i^k e_i) \\ X^{k,i-1}, & \text{若 } f(X^{k,i-1}) \leq f(X^{k,i-1} - t_i^k e_i) \end{cases} \quad (6.6.7)$$

记  $X^{k,0} = X^k$ , 探测结束时有  $X^{k,n} \neq X^{k,0}$ , 则转向模式移动; 若  $X^{k,n} = X^{k,0}$ , 表明以  $X^k$  出发按原步长未搜索到成功前进方向, 可缩短步长再试, 取  $t^k = 0.5t^k$ , 直到  $\|t^k\| \leq \epsilon$ , 这时表明  $X^k$  已经接近最小点。

(3) 模式移动: 当第  $k$  次探测搜索结束时, 有  $X^{k,n} \neq X^{k,0}$ , 则可以沿  $X^{k,n} - X^k$  方向移动, 该方向近似于负梯度方向, 记其终点为  $\bar{X}^{k+1}$ , 则有

$$\bar{X}^{k+1} = X^{k,n} + a^k(X^{k,n} - X^k) \quad (6.6.8)$$

式中,  $a^k$  可以直接取 1, 也可用一维搜索确定。如果取  $a^k$  为 1, 则公式 (6.6.8) 变为

$$\bar{X}^{k+1} = 2X^{k,n} - X^k \quad (6.6.9)$$

再从  $\bar{X}^{k+1}$  出发进行一轮探测搜索得到  $\bar{X}^{k+1,n}$ , 此时有如下两种情况。

情况 1: 若  $f(\bar{X}^{k+1,n}) < f(\bar{X}^{k,n})$ , 表明模式移动成功, 令

$$X^{k+1} = \bar{X}^{k+1,n} \quad (6.6.10)$$

情况 2: 若  $f(\bar{X}^{k+1,n}) \geq f(\bar{X}^{k,n})$ , 表明模式移动失败, 应退回模式移动的出发点

$$X^{k+1} = X^{k,n} \quad (6.6.11)$$

## 2. 嵌入确定性搜索的蚁群算法

嵌入确定性搜索的蚁群算法, 是在局部搜索时以一定的概率利用步长加速法进行确定性搜索。局部搜索规则如下

$$R = \begin{cases} \text{用步长加速法进行局部确定性搜索,} & \text{若 } v < v_0 \\ \text{按公式(6.6.1)进行局部随机搜索,} & \text{否则} \end{cases} \quad (6.6.12)$$

式中,  $v$  是随机数且  $0 < v < 1$ ;  $v_0$  是系数且  $0 < v_0 < 1$ 。

嵌入确定性搜索蚁群算法的具体步骤如下:

初始化
Loop
每只蚂蚁处于每次循环的开始位置;

```

Loop;
每只蚂蚁利用公式 (6.6.12) 进行局部搜索;
Until 所有蚂蚁完成局部搜索
Loop
每只蚂蚁进行全局搜索, 按公式 (6.6.2) ~ (6.6.4) 选择要进行的动作;
Until 所有蚂蚁完成全局搜索
按公式 (6.6.5) 进行信息素强度更新;
Until 中止条件

```

## 6.6.2 仿真算例

为了验证嵌入确定性搜索的连续域蚁群算法的有效性, 这里采用了如下 Ackley 测试函数

$$F = 20 \exp\left(-0.2 \sqrt{\frac{1}{2} \sum_{j=1}^2 x_j^2}\right) + \exp\left(\frac{1}{2} \sum_{j=1}^2 \cos(2\pi x_j)\right) - 22.71828, \\ -10 \leq x_j \leq 10, j = 1, 2 \quad (6.6.13)$$

上述函数的精确最优解为  $(x_1^*, x_2^*) = (0, 0)$  处的最大值  $F(x_1^*, x_2^*) = 0$ 。用嵌入确定性搜索的蚁群算法计算  $F$  函数 20 次, 设置  $m = 50$ ,  $v_0 = 0.1$ ,  $q_0 = 0.2$ ,  $T = 10$ ,  $\rho_0 = 0.6$ ,  $\beta = 0.7$ , 每次计算的循环次数为 120, 其计算结果如表 6.5 所示。

表 6.5 用改进蚁群算法求解 Ackley 函数  $F$  的计算结果

	最优点位置	最优值	平均最优值
函数 $F$	$(-0.000236, -0.000283)$	-0.001044	-0.00117

实验时记录了初始时刻、第 40 次循环、第 80 次循环和第 120 次循环的蚂蚁分布情况, 分别如图 6.6~图 6.9 所示。

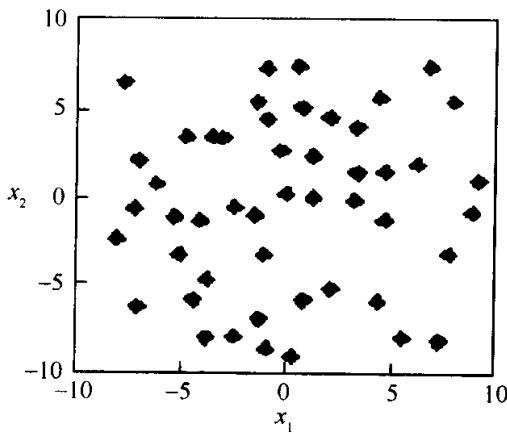


图 6.6 初始时刻

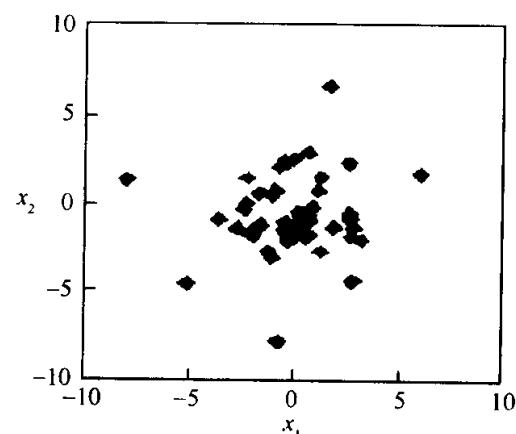


图 6.7 第 40 次循环

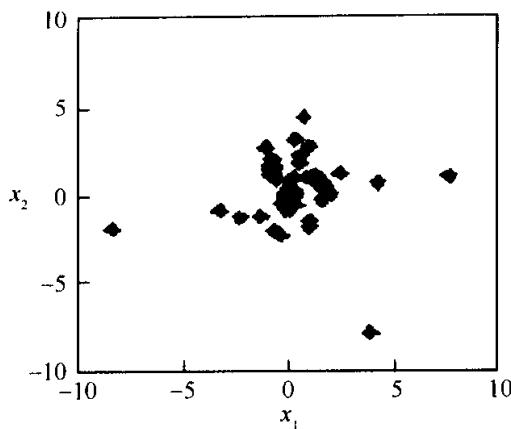


图 6.8 第 80 次循环

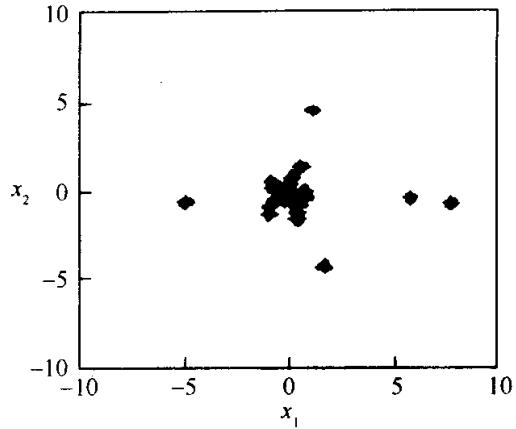


图 6.9 第 120 次循环

由上述仿真结果可见，蚂蚁逐渐向全局最优点附近转移。另外，通过全局和局部的随机搜索，增加了蚂蚁在整个连续域分布的随机性和多样性，改善了连续域蚁群算法的全局收敛性能。

## 6.7 基于密集非递阶的连续交互式蚁群算法 (CIACA)

本节研究了一种基于密集非递阶的 CIACA<sup>[19]</sup>，该算法通过修改蚂蚁信息素的留存方式和行走规则，并运用信息素交流和直接通信两种方式来指导蚂蚁寻优，其改进思想源自于对自然界中真实蚁群行为<sup>[30]</sup>和求解连续域优化问题蚁群算法机理<sup>[31]</sup>的进一步研究。

### 6.7.1 算法设计

#### 6.7.1.1 密集非递阶的概念

##### 1. 生物学定义

“密集非递阶 (dense heterarchy)”这一概念最早由 Wilson E D 于 1988 年在描述蚁群中的信息流时首次提出<sup>[32]</sup>，“蚁群是一个特殊的层次结构，可称之为非递阶结构。这意味着较高层次单元的性质在一定程度上影响着较低的一层，而被较高层次影响后的较低层次单元会反过来影响较高层次”。基于这一思想，这里提出了基于信息素轨迹交流和蚂蚁个体之间直接通信的两种通信通道。实际上，这两种通道对于蚁群算法而言非常重要，也是理解非递阶概念的一个很好范例。蚂蚁之间是通过“信息流”进行通信的，这样便使得这种非递阶结构变得更加密集。

这里构建了一个密集连接的网络，该网络并不是层次连接的，而是非递阶连接。形象而言，在军队中，层次连接指的是将军把命令传给上校，而上校把命令

传给上尉，其余以此类推。而在蚁群算法中，“蚁后”并不把命令传给其他蚂蚁，她作为一名劳动分子也是蚁群网络中的普通一员（如图 6.10 所示），这种没有“论资排辈”的系统具有很强的自组织功能。

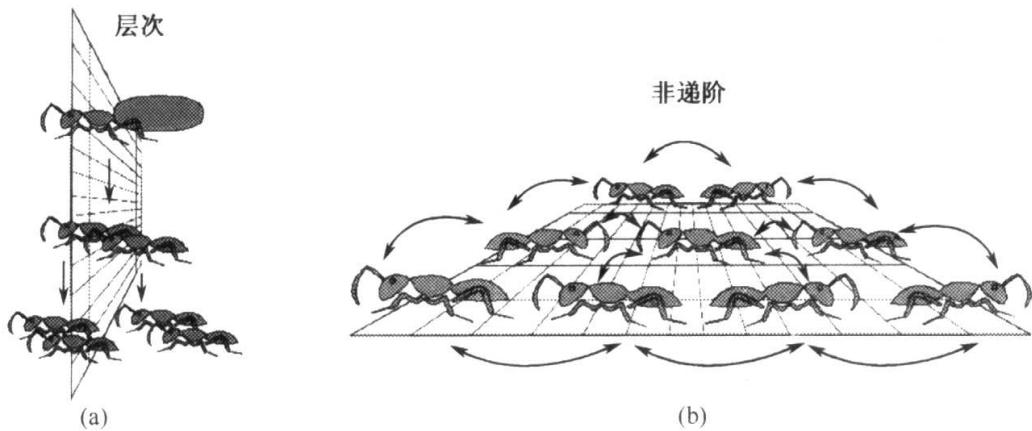


图 6.10 层次结构与非递阶结构示意图

“密集非递阶”是用来描述蚁群从环境中接收“信息流”方式的一个基本概念，每只蚂蚁都可在任意时刻与其他蚂蚁进行联络，而蚁群中的信息流是通过多个通信通道传输的。

## 2. 非递阶算法

在将“密集非递阶”应用于求解连续域优化问题之前，这里先介绍一个简单的非递阶算法。该算法利用了通信通道的基本思想，一个通信通道是信息素的存放地，可用来传递多种信息，如图 6.11 所示。

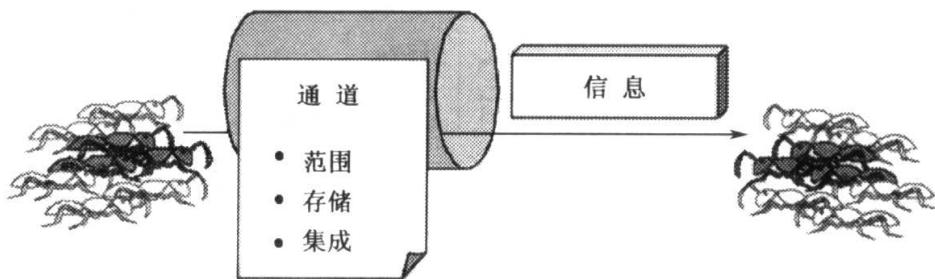


图 6.11 信息通道示意图

信息通道的基本性质如下。

(1) 范围：即蚁群中信息素的交流方式，蚁群中的某一子群可与另一子群进行信息交流。

(2) 存储：即信息素在系统中的驻留方式，信息素可在某一时间段内被一直保留。

(3) 集成：即信息素在系统中的进化方式，信息素可通过一个外部过程被一只或多只蚂蚁更新，也可不更新。

上述性质都集聚于同一信息通道，这样就形成了许多不同种类的信息通道。蚂蚁通信中所传递的信息具有多种形式，有时很难描述某些特殊类别的信息。

这里以“信息素存留”通道的性质为例，由于每只蚂蚁都有信息素，因此这里的“范围”就是所有的蚁群；由于这是一种启发式搜索，因此在一定的时间段内，信息素将被一直存留，这就是基本性质中所说的“存储”；由于信息素挥发作用，通道允许信息随着时间的推移而被破坏，这就是基本性质中所说的“集成”。

### 6.7.1.2 CIACA 通信通道

由非递阶概念可见，CIACA 的设计思路非常简单。这里按照所采用通信通道的不同，定义了三种版本的 CIACA。

#### 1. 信息素交流的 CIACA

第一个版本 CIACA 的设计思路是竭力接近于 Bilchev G A 等<sup>[1]</sup>率先提出的用于求解连续域优化问题的改进蚁群算法。该算法受蚂蚁的信息素存留启发而设置了一个通信通道，每只蚂蚁在其搜索空间内的某一节点上释放一定量的信息素，节点上的信息量与其所搜索到的目标函数值成正比。这些信息节点能够被蚁群中的所有个体察觉，并会逐渐消失。蚂蚁根据路径距离和路径上的信息量来决定是否选择这些信息节点。蚂蚁会向着信息素点集云的重心  $G_j$  移动，而重心位置依赖于第  $i$  个节点上第  $j$  只蚂蚁的“兴趣”  $\omega_{ij}$ ，其表达式如下

$$G_j = \sum_{i=1}^n \left( \frac{x_i \omega_{ij}}{\sum_{i=1}^n \omega_{ij}} \right) \quad (6.7.1)$$

$$\omega_{ij} = \frac{\bar{\delta}}{2} \cdot e^{-\theta_i \cdot \delta_{ij}} \quad (6.7.2)$$

式中， $n$  表示节点数目； $x_i$  表示第  $i$  个节点的位置； $\bar{\delta}$  表示蚁群中两只蚂蚁之间的平均距离； $\theta_i$  表示第  $i$  个节点上的信息量； $\delta_{ij}$  表示从第  $j$  只蚂蚁到第  $i$  个节点之间的距离。值得注意的是，处于信息节点上的蚂蚁并不径直地向信息素点集云的重心移动。事实上，每只蚂蚁都有在蚁群中均匀分布的参数调整范围，每只蚂蚁都得到一个允许范围内的随机距离，蚂蚁会以随机距离为度量向着其重心位置移动，但是某些干扰因素可能会影响蚂蚁所到达的最终位置。

从非递阶概念的角度来描述上述行为，则该 CIACA 中信息素交流通道的性质如下：

(1) 范围：当蚁群中某只蚂蚁留下一定量的信息素后，其他后继蚂蚁都能觉察到该信息素的存在。

(2) 存储：某一时间段内信息将被一直保留于蚁群系统之中。

(3) 集成：由于信息素的挥发作用，随着时间的推移信息素将被更新。

## 2. 利用个体之间的直接通信的 CIACA

每只蚂蚁都能给另一只蚂蚁发送“消息”，这意味着该通信通道的范围是“点对点”式的。蚂蚁可将已经接收到或将要接收到的信息存储到栈中，而栈中的信息可被随机读取，这一过程得益于从大规模多智能体系统通信问题的工作机理中所得到的启发<sup>[33]</sup>。此处所发送的“消息”是信息发送者的位置，即目标函数值。信息接收者会将发送者所发送来的“消息”与其自身的信息相比较，以决定它是否要向信息发送者的位置移动。最终位置将出现在一个以信息发送者为中心、信息接受者范围为半径的超球体内，然后信息接受者将“消息”进行压缩并将其随机发送给另一只蚂蚁。此时，该 CIACA 中的信息通信通道具有如下性质：

(1) 范围：当蚁群中的某只蚂蚁发出“消息”之后，仅有一只蚂蚁可以觉察此“消息”。

(2) 存储：某一时间段内信息可以“记忆”的形式保存在蚁群系统之中。

(3) 集成：所存储的信息是静态的。

## 3. 二者协同的 CIACA

前述两种版本的 CIACA 算法具有很大的不同，自组织的作用可将较低层次的个体整合成较高层次的整体。基于这一思想，可将上述两种版本 CIACA 算法中的简单通信通道融合于一个系统之中，由于通信通道没有并发机制，因此这一点实现起来比较容易。

### 6.7.1.3 CIACA

CIACA 的程序结构流程如图 6.12 所示。

由图 6.12 可见，CIACA 主要包括如下 3 个步骤：

(1) 设置参数。

(2) 算法开始。

(3) 在算法满足一定的结束条件时，算法结束。

蚂蚁根据其在通信通道系统中所处理的感知信息进行移动，这里需要设置 4 个参数，具体如下：

(1)  $\eta \in [0, +\infty)$ ：系统中蚂蚁的数目，其值可通过下式获得

$$\eta = \eta_{\max} (1 - e^{-d/p}) + \eta_0 \quad (6.7.3)$$

式中,  $d$  表示目标函数的维数;  $\eta_{\max}$  表示最大蚂蚁数目, 一般设置  $\eta_{\max}=1000$ ;  $\eta_0$  表示目标函数维数为 0 时的蚂蚁数目, 一般设置  $\eta_0=5$ ;  $p$  表示蚂蚁数目的相对重要性, 一般设置  $p=10$ 。

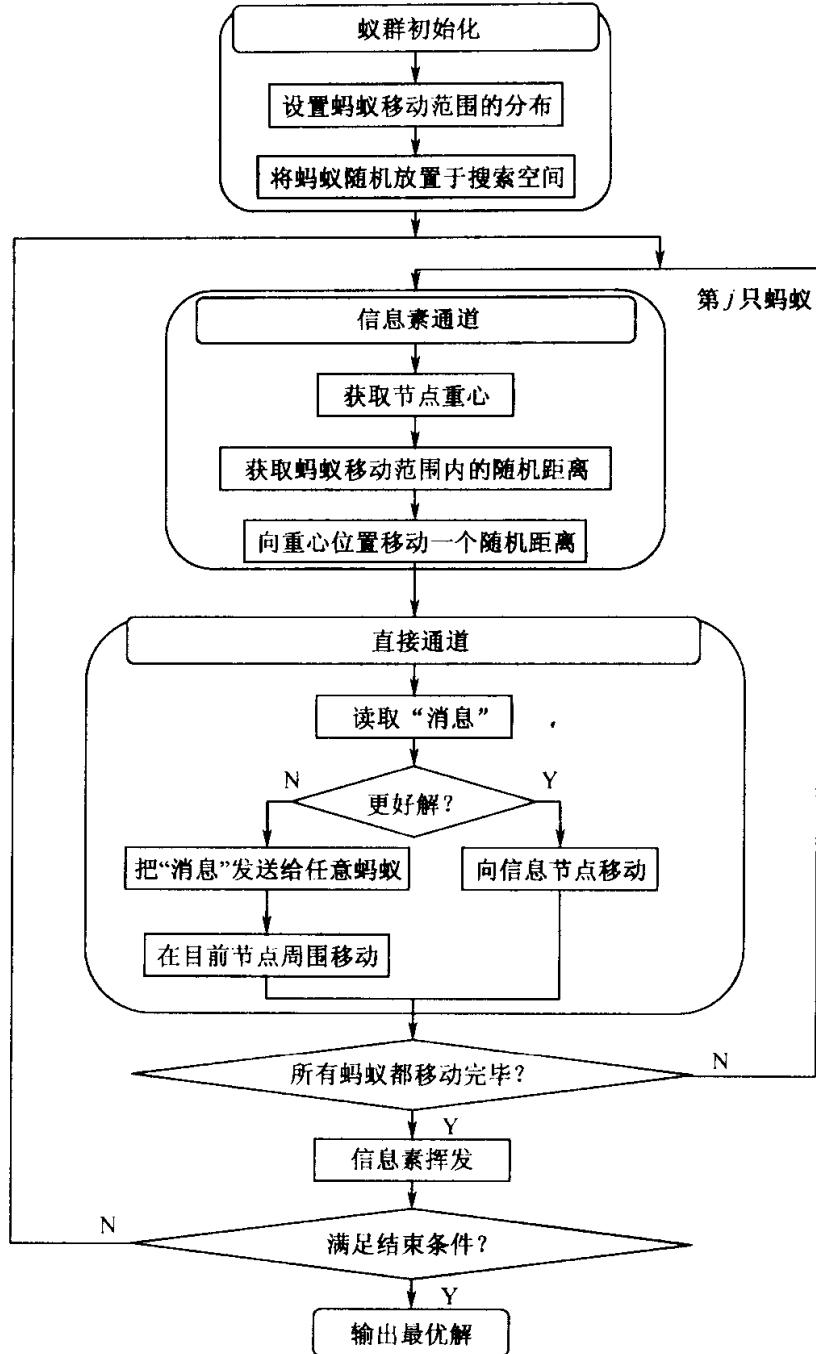


图 6.12 CIACA 的程序结构流程

(2)  $\sigma \in [0, 1]$ : 搜索空间度的百分比, 用来定义蚂蚁移动范围分布的标准偏差, 其经验值为 0.9。

- (3)  $\rho \in [0, 1]$ ：用来定义信息素的持久性，其经验值为 0.1。  
(4)  $\mu \in [0, +\infty)$ ：“消息”的初始数目，其值可通过公式  $\mu = \eta \cdot 2/3$  获得。

### 6.7.2 仿真算例

为了验证 CIACA 的有效性，可采用如下二维连续函数作为仿真算例

$$f(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 \quad (6.7.4)$$

其最优解为  $f(0, 0) = 0$ 。用

CIACA 对其求解时，设置搜索空间为  $[-50, 100]$ ，蚂蚁数目为 10，其他参数设置如前所述。函数  $f(x_1, x_2)$  最优解附近的三维曲线如图 6.13 所示。

CIACA 中的两种通道扮演着互相补充的角色，直接通信会产生一种增强模式，这是由于其过于注重最优节点而没有考虑以前遍历的区域；与之相反，信息素交流（由于其“存储”性质）允许其解呈现多样性。图 6.14 给出了在两种通道协同工作模式下，CIACA 分别在起始点、130 代以及 250 代时求解函数  $f(x_1, x_2)$  的情况。

由图 6.14 可见，多数蚂蚁倾向于聚集在其搜索空间内的全局最优解周围，

然而也有一部分蚂蚁会聚集在局部最优解附近。图 6.15 给出了只利用信息素交流时 CIACA 的运行情况，图 6.15 中的 (a)、(b)、(c)、(d) 分别表示算法在 101、102、103 和 104 次迭代时的情况。

由图 6.15 可见，蚂蚁在最优解所在的重心周围移动的同时，也在不断开辟新的搜索空间。

对于简单的  $f(x_1, x_2)$  函数而言，由于直接通信允许 CIACA 以较快的速度收敛，因此它更加适合解决函数优化问题（如图 6.16 所示），但 CIACA 容易陷于局部最优解。由两种通道协同工作时的仿真结果可见（如图 6.17 所

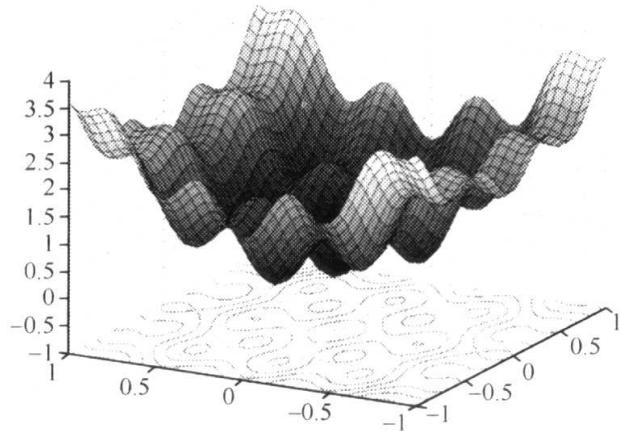


图 6.13 函数  $f(x_1, x_2)$  最优解附近的三维曲线

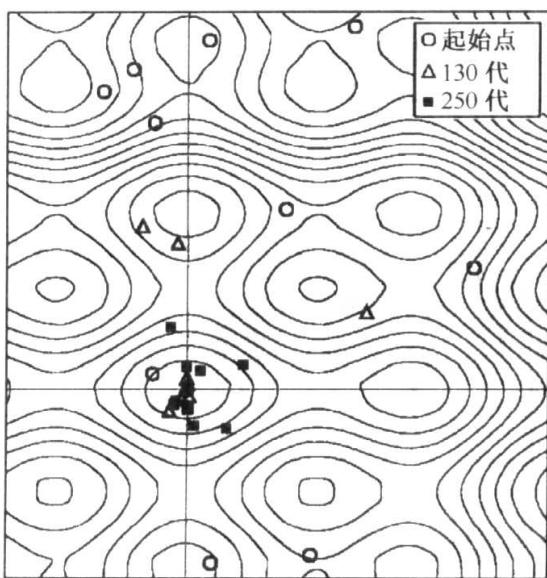


图 6.14 两种通道协同工作模式下 CIACA 求解函数  $f(x_1, x_2)$  的运行情况

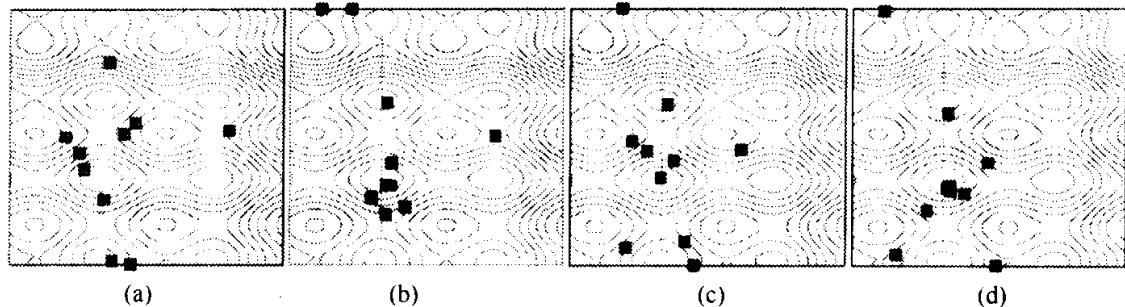


图 6.15 只利用信息素交流时 CIACA 的运行情况

示), 其标准偏差存在着振荡, 这意味着蚁群倾向于聚集到某一值后又表现出扩散趋势。

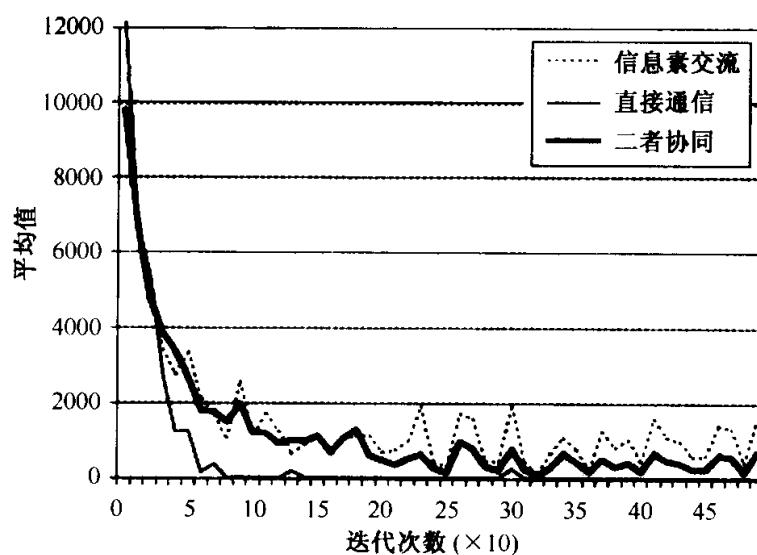


图 6.16 目标函数平均值的进化曲线

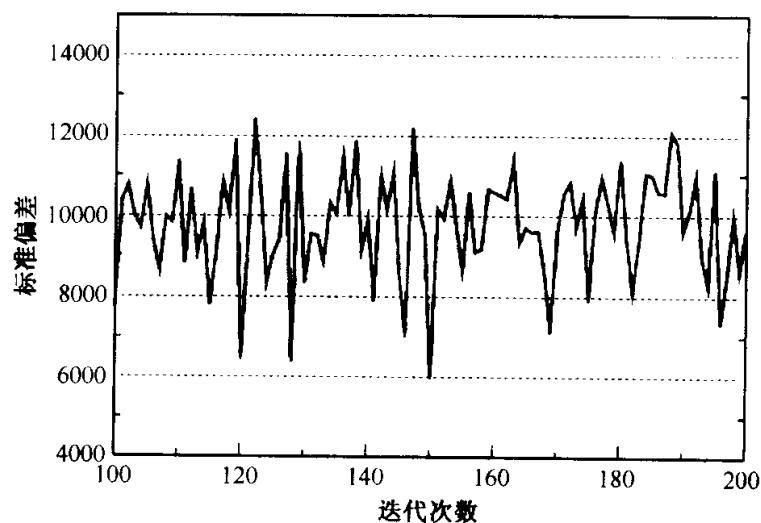


图 6.17 目标函数标准偏差的进化曲线

而在单模式条件下，目标函数的标准偏差的振荡值会明显减小（如图 6.18 所示）。

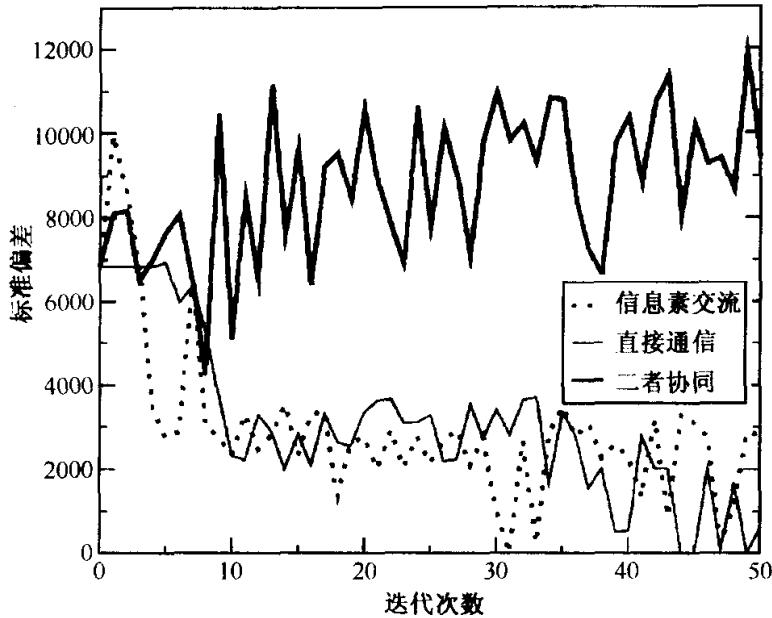


图 6.18 不同模式下的目标函数标准偏差

## 6.8 多目标优化问题的连续域蚁群算法

在科学研究与工程实践中，很多连续域内的决策问题都涉及对带有多个约束条件的多个目标进行同时优化这一复杂情况。在求解多目标优化问题 (multi-objective optimization problem, MOP) 时，各个目标之间往往是相互冲突的。由于 MOP 解的多样性，不仅要求所得的解能够收敛到 Pareto 前沿，而且还需保持群体的多样性。在利用蚁群算法求解 MOP 时，蚂蚁之间的信息素正反馈交流方式会使所求解倾向于集中在解空间内的某一区域，从而不利于保持群体多样性。

本节针对上述问题研究了一种用于求解带有多个约束条件 MOP 的连续域蚁群算法。该算法定义了连续域中信息量的留存方式和蚂蚁的行走策略，并将信息素交流和基于全局最优经验指导两种寻优方式相结合，将当前发现的所有非支配解保存起来，进而用这些解来指导蚂蚁朝着散布较为稀疏的区域寻优，以保证解的分布性能，并提高了蚁群算法的收敛速度，同时维持了群体的多样性。

### 6.8.1 算法设计

假设蚂蚁只在其所处位置释放信息素，这些留有信息素的位置可以被蚁群中

所有的蚂蚁察觉，而且蚂蚁可根据信息量的大小及距离来选择下一步的觅食方向。蚂蚁所释放信息素与其所表示解的优劣成正比，而由于 MOP 解的多样性，且不存在绝对的最优解，从而不容易比较解的改进程度。为此，可通过比较解之间的 Pareto 支配关系来决定蚂蚁所释放的信息量大小<sup>[22]</sup>。同时，蚂蚁在觅食过程中不断更新自己的位置，同时采用了一种基于最优经验的寻优方式。蚂蚁在寻优过程中除了受到同伴遗留信息素的影响外，还受到整个蚁群最优经验的影响，蚂蚁在二者的共同作用下完成寻优任务。

### 6.8.1.1 基于信息素交流的寻优方式

在 MOP 中，没有绝对的最优解，解的优劣是相对的。对于要选择移动方向的蚂蚁  $i$ ，比较其与蚁群中各只蚂蚁所表示解的 Pareto 支配关系。这种 Pareto 支配关系决定了同伴在它们所处位置上所释放的信息量。如果蚂蚁  $j$  ( $j=1, 2, \dots, N$ ,  $N$  为蚁群规模) 表示的解  $x_j$  为非可行解，则说明蚂蚁  $j$  所在的位置对寻优没有帮助，故释放很少的信息素。如果  $x_j$  为可行解且支配  $x_i$ ，则说明选  $x_j$  作为寻优方向，有利于算法朝着 Pareto 前沿或者可行解的方向进化。因此，蚂蚁  $j$  在其所处的位置上释放大量信息素，以吸引蚂蚁  $i$  前来寻优。按照这一思路，可将蚂蚁  $j$  在其所处位置上释放的信息量  $\theta_j$  定义如下

$$\theta_j = \begin{cases} \lambda_1, & \text{若 } x_j \text{ 为非可行解} \\ \lambda_2, & \text{若 } x_j \text{ 为非可行解且 } x_i < x_j \\ \lambda_3, & \text{若 } x_j \text{ 为非可行解且 } x_i \text{ 与 } x_j \text{ 为非约束支配关系} \\ \lambda_4, & \text{若 } x_j \text{ 为非可行解且 } x_i > x_j \end{cases} \quad (6.8.1)$$

式中， $i, j=1, 2, \dots, N$ ，且  $i \neq j$ ； $N$  表示蚁群规模； $\lambda_1, \lambda_2, \lambda_3, \lambda_4$  为 4 个不同的参数，且  $\lambda_4 > \lambda_3 > \lambda_2 > \lambda_1$ 。蚂蚁  $i$  的寻优方向与蚁群中其他蚂蚁所处位置的信息量及距离有关，可将其状态转移概率定义如下

$$P_j = \frac{\theta_j \delta_{ij}}{\sum_{j=1}^m \theta_j \delta_{ij}}, \quad j \neq i, j = 1, 2, \dots, N \quad (6.8.2)$$

式中， $\delta_{ij} = 1/d_{ij}$ ， $d_{ij}$  表示当前蚂蚁  $i$  与蚂蚁  $j$  之间的距离。

### 6.8.1.2 基于全局最优经验指导的寻优方式

如果仅依靠蚂蚁之间的信息素交流来进行寻优，则算法搜索所需要的时间较长，且不易保持群体的多样性。这里研究了另一种寻优方式，即在全局最优经验指导下进行寻优。在改进后的蚁群算法中，设立一个外部集合 BP，用来保存整个蚁群当前所发现的所有非支配解。在集合 BP 中寻找散布最为稀疏的非支配

解，其所在位置即为当前蚂蚁的寻优方向。

假设当前集合 BP 中有  $p$  个非支配解  $x = (x_1, x_2, \dots, x_p)$ ，则可按照下式计算每个解到其他解的距离

$$d_{ij} = \sqrt{\sum_{t=1}^K (f_t(x_i) - f_t(x_j))^2} \quad (6.8.3)$$

式中， $i=1, 2, \dots, p$ ,  $j=1, 2, \dots, p$ , 且  $i \neq j$ 。按照下式计算共享函数值

$$S(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{\sigma_{\text{share}}}, & \text{若 } d_{ij} < \sigma_{\text{share}} \\ 0, & \text{否则} \end{cases} \quad (6.8.4)$$

式中， $\sigma_{\text{share}}$  表示小生境半径。对非支配解  $i$  可按照下式求其小生境数

$$\text{niche}(i) = \sum_{j=1}^p S(d_{ij}), \quad j \neq i, i = 1, 2, \dots, p \quad (6.8.5)$$

小生境数  $\text{niche}(i)$  的最小非支配解  $i$  所在的位置就是当前蚂蚁的寻优方向。

### 6.8.1.3 蚂蚁行进策略

定义蚂蚁的活动范围为  $r$ ，如果蚂蚁与目标点之间的距离大于  $r$ ，则蚂蚁只能朝着目标点的方向移动长度为  $r$  的一段距离。值得注意的是，蚂蚁在移动后，并不能直接到达目标点，还要受到一个随机扰动因子  $\varphi$  的影响。扰动因子  $\varphi$  的值对算法的收敛有一定的影响，较大的  $\varphi$  值有利于增大搜索范围，而较小的  $\varphi$  值有利于局部寻优。因此，在算法的开始阶段，给  $\varphi$  一个较大的取值范围，以加大算法的搜索范围；随着迭代次数的增加，可线性地减小  $\varphi$  的取值范围，以加强算法的局部收敛能力。

### 6.8.1.4 算法实现

改进后蚁群算法的具体实现步骤如下：

- (1) 随机生成规模为  $N$  的初始蚁群 POP，计算 POP 中每只蚂蚁的目标函数值  $f_i(x), i=1, 2, \dots, k$ ，以及约束函数值  $e_j, j=1, 2, \dots, J$ 。
- (2) 初始化外部集合 BP，它的初始值为 POP 的所有可行解中的非支配解，即  $X_f = \{x \in \text{POP} \mid e(x) \leq 0\}$ ,  $\text{BP} = \{x \in X_f \mid \exists x' \in X_f, \text{使得 } x' < x\}$ 。
- (3) 设置迭代次数  $N_c = 0$ 。
- (4) 令  $i = 1$ 。
- (5) 随机产生一个  $[0, 1]$  范围内的随机数  $p$ ，将它与参数  $p_0$  比较。当  $p \leq p_0$  时，对当前蚂蚁  $i$  使用基于全局最优经验指导的方式寻优；当  $p > p_0$  时，则对蚂蚁  $i$  采用信息素交流的方式寻优。

- (6) 在蚂蚁的活动范围内移动蚂蚁  $i$ , 并在其最终位置上加一个随机扰动因子  $\varphi$ 。重新评价蚂蚁  $i$ , 计算其目标函数值以及约束函数值。
- (7) 更新最优经验结合 BP。如果蚂蚁  $i$  是可行解, 且对于集合 BP 来说是非支配的, 则将蚂蚁  $i$  加入集合 BP, 并删除 BP 中被  $i$  所支配的解。
- (8)  $i = i + 1$ , 如果  $i \leq N$ , 则跳转到第 (5) 步。
- (9)  $N_c = N_c + 1$ , 如果  $N_c < N_{c_{\max}}$ , 则跳转到第 (4) 步; 否则, 算法结束。

### 6.8.2 仿真算例

为验证改进后蚁群算法的有效性, 这里采用一组常用的基准函数 (TNK 问题) 来测试算法的性能。由于很难找到类似的方法与本节算法进行性能上的比较, 此处选择了非支配排序遗传算法 (non-dominated sorting genetic algorithm, NSGA)。为了体现比较的公平性, 改进后蚁群算法和 NSGA 采用相同的群体规模 (均为 60), 而且在每组测试函数中均迭代相同的次数。NSGA 变量采用实编码, 交叉概率为 0.9, 变异概率为 0.2。改进后蚁群算法的参数设置为:  $\lambda_1 = 0.01$ ,  $\lambda_2 = 0.1$ ,  $\lambda_3 = 2$ ,  $\lambda_4 = 5$ ,  $p_0 = 0.6$ 。

在 TNK 问题测试中, 以图形化的方式给出了这两种算法生成的 Pareto 前沿, 并对所得解的数量、分布性能以及散布范围等<sup>[34]</sup>做了比较。

(1) 间距评估: 可用来测度所得 Pareto 前沿上相邻解间距离的变化情况。其定义为

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (\bar{d} - d_i)^2} \quad (6.8.6)$$

式中,  $d_i = \min_j (|f_1^i(x) - f_1^j(x)| + |f_2^i(x) - f_2^j(x)|)$ ,  $i, j = 1, 2, \dots, n$ ,  $n$  为算法获得的 Pareto 前沿上向量的个数;  $\bar{d}$  为所求得  $d_i$  的平均值, 即  $\bar{d} = \frac{1}{n} \sum_{i=1}^n d_i$ 。当所获得的解越接近均匀散布时, 间距  $S$  的值越小。

(2) 最大散布范围评估: 可用来测度目标空间中的两个极值解的距离。其定义为

$$D = \sqrt{\left( \max_{i=1}^n f_1^i - \min_{i=1}^n f_1^i \right)^2 + \left( \max_{i=1}^n f_2^i - \min_{i=1}^n f_2^i \right)^2} \quad (6.8.7)$$

$D$  值越大, 表明算法所获得解的散布范围越广。

TNK 问题可描述为:

$$\begin{aligned} \min f_1(x) &= 4x_1^2 + 4x_2^2 \\ f_2(x) &= (x_1 - 5)^2 + (x_2 - 5)^2 \\ \text{s. t. } e_1(x) &\equiv (x_1 - 5)^2 + x_2^2 \leq 25 \\ e_2(x) &\equiv (x_1 - 8)^2 + (x_2 + 3)^2 \geq 7.7 \\ 0 &\leq x_1 \leq 5, 0 \leq x_2 \leq 3 \end{aligned} \quad (6.8.8)$$

TNK 问题的求解难度在于它的 Pareto 前沿是由三段不连通的 Pareto 曲线构成, 且 Pareto 曲线是非凸的。图 6.19 为两种算法迭代 100 次后的结果。

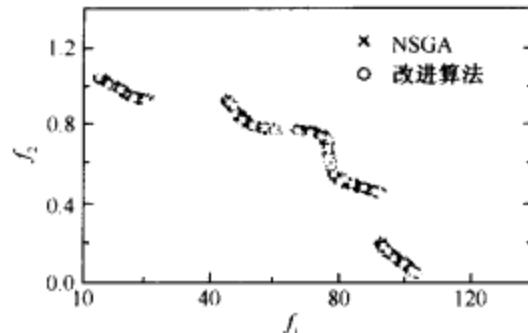


图 6.19 两种算法对 THK 问题的求解结果

图 6.19 表明两种算法都能很好地逼近 Pareto 前沿。表 6.6 给出了两种算法在间距及最大散布范围上的比较结果。

表 6.6 两种算法在间距及最大散布范围上的比较

算 法	解的个数	间距 S	最大散布范围 D
NSGA	78	0.0078	1.21
改进蚁群算法	136	0.0051	1.41

由仿真结果可见, 两种算法获得解的数量相当, 而且分布都比较均匀, 但在曲线的首段和尾段, 改进蚁群算法获得解的数量更多。

## 6.9 复杂多阶段连续决策问题的动态窗口蚁群算法

本节针对具有强非线性的大规模复杂多阶段连续决策问题, 研究了一种结合遗传优化的动态窗口蚁群算法<sup>[23]</sup>。该算法将各阶段容许决策值映射为一个层状构造图中的有限节点集, 其中每一层节点对应某一阶段容许决策集的子集, 该子集用实数编码遗传优化进行动态筛选, 以减小算法的搜索空间。

### 6.9.1 算法设计

#### 6.9.1.1 多阶段动态决策的蚁群搜索

用蚁群搜索来求解目标函数具有可加性和单调性的多阶段连续决策问题时, 首先将每一阶段的决策变量  $u(t)$  在其容许决策集合范围内离散化为一系列的离

散格点  $u_l(t) \in U(t)$ ,  $l=1, 2, \dots, q^r$  (设  $r$  维决策向量的每一维都均匀离散为  $q$  个格点), 称之为离散化决策变量; 记阶段  $t$  所有决策变量离散格点集合为  $U_d(t) \in U(t)$ , 称之为离散化容许决策集合。然后将各阶段的离散化决策变量  $u_l(t)$  的取值与构造图  $G$  中的节点做一对一映射, 可有

$$\phi: u_l(t) \in U_d(t) \leftrightarrow v_i^t \in V^t \quad (6.9.1)$$

由上述映射可导出如图 6.20 所示的层状构造图。

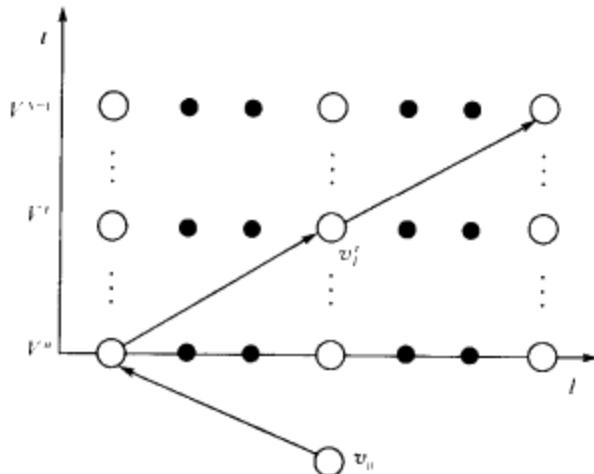


图 6.20 层状构造图

定义图 6.20 中每一水平层的节点为决策层, 对应某阶段离散化容许决策集合,  $V^t$  为第  $t$  层的节点集合,  $v_i^t$  为该层的第  $i$  个节点。在该层状构造图中, 蚂蚁从初始节点  $v_0$  出发, 重复地选择上一层的某一节点, 最终形成一条路径  $\{v_0, v_{n(0)}, \dots, v_{n(t)}, \dots, v_{n(N-1)}\}$ ,  $n(t)$  为第  $t$  阶段所选择的节点序号, 该路径对应一个可行解。

由于蚂蚁逐阶段选择决策值以构造问题的可行解, 故只需定义相邻层之间的连接, 即  $(v_i^t, v_j^{t+1})$ , 并定义其连接代价为

$$\begin{aligned} \text{Cost}(v_i^t, v_j^{t+1}) &= [J(t+1) - J(t)] + P(t+1) \\ &= L[x(t+1), u(t+1), t+1] + P(t+1) \\ &= \begin{cases} \infty, & \text{若 } P(t+1) = \infty \\ L[x(t+1), u(t+1), t+1], & \text{若 } P(t+1) = 0 \end{cases} \quad (6.9.2) \end{aligned}$$

式中,  $P(t+1)$  表示一个由约束条件决定的函数, 如果解不满足优化模型中的某个约束条件, 则令  $P(t+1)=\infty$ , 阻止蚂蚁选择该连接; 否则, 令  $P(t+1)=0$ 。连接代价函数的第一部分等于选择节点  $v_j^{t+1}$  后所增加的目标函数值, 连接  $(v_i^t, v_j^{t+1})$  的代价只能在蚂蚁到达节点  $v_i^t$  后确定, 连接  $(v_i^t, v_j^{t+1})$  上的局部启发信息  $\eta_j^a(k)$  可定义为连接代价  $\text{Cost}(v_i^t, v_j^{t+1})$  的倒数。

图 6.20 中, 如各阶段离散化容许决策集合的规模为  $q^r$ , 则所形成的层状构造图中每一层的节点总数就为  $q^r$ , 层状构造图规模随  $r$  呈指数增长。同时, 蚂蚁在每一层都要对下一层的所有节点进行选择, 构造一个解的计算量也呈指数增长。

### 6.9.1.2 搜索窗口的产生和更新

对于高维决策变量的多阶段连续决策问题, 其各阶段离散化容许决策集合的规模会“指数膨胀”, 让蚂蚁每个决策阶段在整个离散化容许决策集合中进行选择是困难的。因此, 这里不再将各阶段的决策变量离散化并全部映射为层状构造图中的节点, 而只按照某种优化规则从容许决策集合  $U(t)$  中抽取决策变量  $u(t)$  的  $w$  个不同取值进行节点映射。称由这  $w$  个决策变量取值构成的集合  $U_s(t) \in U(t)$  为蚂蚁在容许决策集  $U(t)$  中的搜索窗口,  $U_s(t)$  的元素个数  $w$  为搜索窗口宽度。这样, 蚂蚁只在每一阶段搜索窗口  $U_s(t)$  的  $w$  个节点上进行路径搜索。

在蚂蚁搜索的构造图中, 称所有指向某节点的有向连接为该节点的上游连接。一个节点的上游连接上的信息素越多, 其被蚂蚁选中构造新解的概率就越大。由于信息素还反映了相应决策值所构成解的优劣, 这也表明在该阶段选择该决策值构成的解“较有希望”成为最优解。

对于连续决策变量, 可采用实数编码遗传优化来更新搜索窗口。在若干次蚁群搜索迭代后, 首先确定搜索窗口  $U_s(t)$  中每个节点(也即其所对应的决策值)的适应度值。根据上述分析, 可定义节点适应度值  $G$  为其上游连接上的信息量平均值

$$G[u_i(t)] = G(v_i) = \frac{1}{N_v} \sum_{v \in V^{t-1}} \tau(v, v_i) \quad (6.9.3)$$

式中,  $N_v$  表示节点  $v_i$  上游连接的总数;  $\tau(v, v_i)$  表示节点  $v_i$  上游连接的信息量;  $v$  属于层状构造图中  $U_a(t-1)$  所映射的决策层  $V^{t-1}$ 。

然后, 对搜索窗口  $U_s(t)$  中的元素进行选择、交叉和变异操作, 用得到的下一代决策值更新原有的搜索窗口, 并再赋给相应决策层中的节点。因为原节点所对应的决策值发生了变化, 所以要重新定义节点上游连接的信息量。又由于节点新对应的决策值是通过上一代的两个父值经过交叉和变异操作产生的, 由此, 可按照下式定义新的上游连接信息量为两个父值适应度的几何平均

$$\tau_c(v, v_i) \Big|_{v \in V^{t-1}} = \frac{d_{\phi_1} G_{p_1} + d_{\phi_2} G_{p_2}}{d_{\phi_1} + d_{\phi_2}} \quad (6.9.4)$$

式中,  $\tau_c(v, v_i)$  表示节点  $v_i$  上游连接的新信息量;  $G_{p_1}$  与  $G_{p_2}$  分别表示节点  $v_i$  所对应的新决策值在遗传操作上的两个上一代决策值(父值)的适应度值, 其值由公式 (6.9.3) 给出;  $d_{\phi_1}$  与  $d_{\phi_2}$  分别表示该节点所对应的决策值与这两个父值在

解空间中的距离，反映了该决策值对两个父值中模式的继承程度。

经过若干次蚁群迭代后，遗传操作会根据连接上的信息素分布情况调整蚂蚁在各个阶段的搜索窗口，一方面使蚂蚁在构造解时避免了对各个阶段的所有决策值进行比较，而是根据公式（6.9.3）和遗传优化原理在各个阶段的决策域中进行优选，可提高搜索效率；另一方面，根据公式（6.9.4）重新分配连接的信息素，使信息素不只集中在节点的某一条上游连接上，而是在节点的所有上游连接上进行扩散。因此，当  $\tau_c(v, v_i)$  较大时，到达上游节点  $v (v \in V_{(t+1)})$  的蚂蚁选择节点  $v_i$  的可能性都较大。根据上述方式重新分配信息素，可有效地解决算法停滞问题。

### 6.9.1.3 算法的步骤

动态窗口蚁群算法的主要步骤如下。

(1) 初始化。

抽取每一阶段的可行决策值构成搜索窗口，并视为遗传优化的一个群落，根据公式（6.9.1）将搜索窗口中的决策变量值映射为蚁群搜索层状构造图中的相应层节点。

(2) 进行若干次蚁群搜索迭代。

蚁群通过层状构造图生成可行解，并根据对解的评价，更新层状构造图中各连接上的信息量。

(3) 进行若干次遗传操作，更新各阶段的搜索窗口。

根据公式（6.9.3）计算原搜索窗口中各节点所对应决策值的适应度值。在各阶段搜索窗口中运用遗传操作生成新一代的决策值，更新搜索窗口中的节点，并根据公式（6.9.4）初始化这些节点上游连接的信息量。

## 6.9.2 算法的计算效率分析

在本节所研究的动态窗口蚁群算法中，假设有  $M$  蚂蚁各自迭代了  $C$  次，则总迭代次数为  $CM$  次，蚂蚁构造一个解的计算量为  $O\left(\sum_{t=0}^{N-1} n_t\right)$ ，其中  $n_t$  为第  $t$  层节点的个数，所以全部计算量为  $O\left(CM \sum_{t=0}^{N-1} n_t\right)$ 。

采用动态窗口后，设各阶段搜索窗口宽度均为  $w$ （等于遗传优化中的种群规模），并且每隔  $T$  次蚁群迭代（计算量为  $O\left(TM \sum_{t=0}^{N-1} n_t\right)$ ），对搜索窗口  $U_S(t)$  进行一次遗传迭代更新，这需要进行  $w/2$  次遗传操作来更新  $U_S(t)$  中的所有元素。

更新  $N$  阶段搜索窗口的计算量共为  $O(Nw^2)$ 。

表 6.7 对动态窗口蚁群算法和基本蚁群算法各迭代  $C$  次的计算效率进行了比较, 这里设各阶段离散化容许决策集合的规模为  $q^r$ 。

表 6.7 动态窗口蚁群算法和基本蚁群算法的计算效率比较

算法类型	节点个数	连接个数	生成一个解的计算量	总计算量
动态窗口蚁群算法	$Nw+1$	$w+(N-1)w^2$	$Nw$	$O[C(TMw+Nw^2)]$
基本蚁群算法	$Nq^r+1$	$q^r+(N-1)q^{2r}$	$Nq^r$	$O(CMNq^r)$

由表 6.7 可见, 动态窗口搜索蚁群算法的计算量与基本蚁群算法计算量的比值(设  $C$ 、 $N$  等算法参数相等)约为

$$\frac{O[C(TMw+Nw^2)]}{O(CMNq^r)} \approx \frac{C(TMw+Nw^2)}{CMNq^r} = \frac{TMw+w^2}{Mq^r} \quad (6.9.5)$$

窗口宽度  $w$  可以根据具体问题来设置, 一般当  $r$  较大时远小于  $q^r$ , 故这个比值远小于 1。因此, 尽管动态窗口蚁群算法采用遗传优化来产生搜索窗口, 但是其总体计算量并不大。

## 6.10 本章小结

本章首先系统分析了连续域寻优蚁群算法与离散域寻优蚁群算法的不同之处, 随后重点研究了基于网格划分策略、信息量分布函数、自适应搜索、交叉变异操作、嵌入确定性搜索、连续交互式的连续域蚁群算法改进策略; 紧接着对一种用于求解带有约束条件的多目标函数优化问题的连续域蚁群算法进行了分析; 最后针对大规模复杂多阶段决策问题研究了一种结合遗传优化的动态窗口蚁群算法。本章内容可为进一步拓展蚁群算法在连续域优化问题中的应用提供一些有益的参考。

## 参 考 文 献

- 1 Bilchev G A, Parmee I C. The ant colony metaphor for searching continuous spaces. Lecture Notes in Computer Science, 1995, 993: 25~39
- 2 高尚, 钟娟, 莫述军. 连续优化问题的蚁群算法研究. 微机发展, 2003, 13(1): 21~22, 69
- 3 Wang L, Wu Q D. Ant system algorithm for optimization in continuous space. Proceedings of the 2001 IEEE International Conference on Control Applications, 2001, 395~400
- 4 Wang L, Wang X P, Wu Q D. Ant system algorithm based Rosenbrock function optimization in multi-dimension space. Proceedings of the 2002 International Conference on Machine Learning and Cybernetics, 2002, 2: 710~714

- 5 Wang L, Wu Q D. Further example study on ant system algorithm based continuous space optimization. Proceedings of the 4th World Congress on Intelligent Control and Automation, 2002, 3: 2541~2545
- 6 Wang L, Wu Q D. Performance evaluation of ant system optimization processes. Proceedings of the 4th World Congress on Intelligent Control and Automation, 2002, 3: 2546~2550
- 7 汪镭, 吴启迪. 蚁群算法在连续空间寻优问题求解中的应用. 控制与决策, 2003, 18(1): 45~48, 57
- 8 Li Y J, Wu T J. An adaptive ant colony system algorithm for continuous-space optimization problems. Journal of Zhejiang University SCIENCE, 2003, 4(1): 40~46
- 9 Li Y J, Wu T J, Hill D J. An accelerated ant colony algorithm for complex nonlinear system optimization. Proceedings of the 2003 IEEE International Symposium on Intelligent Control, 2003, 709~713
- 10 李艳君, 吴铁军. 连续空间优化问题的自适应蚁群系统算法. 模式识别与人工智能, 2001, 14(4): 423~427
- 11 段海滨, 王道波, 于秀芬. 一种求解连续空间优化问题的改进蚁群算法. 北京航空航天大学学报, 待发表
- 12 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 13 Duan H B, Wang D B, Zhu J Q. A novel method based on ant colony optimization algorithm for solving ill-conditioned linear systems of equations. Journal of System Engineering and Electronics, 2005, 16(3): 606~610
- 14 段海滨, 王道波, 于秀芬等. 一种改进的蚁群算法用于约束非线性规划问题求解. 四川大学学报, 2004, 41(5): 973~977
- 15 陈峻, 沈洁, 秦玲. 蚁群算法求解连续空间优化问题的一种方法. 软件学报, 2002, 13 (12): 2317~2323
- 16 陈峻, 沈洁, 秦玲. 蚁群算法进行连续参数优化的新途径. 系统工程理论与实践, 2003, 23(3): 48~53
- 17 杨勇, 宋晓峰, 王建飞等. 蚁群算法求解连续空间优化问题. 控制与决策, 2003, 18(5): 573~576
- 18 杨勇. 化工批处理过程调度的建模与优化. 杭州: 浙江大学博士学位论文, 2003
- 19 Dréo J, Siarry P. Continuous interacting ant colony algorithm based on dense hierarchy. Future Generation Computer Systems, 2004, 20(5): 841~856
- 20 Dréo J, Siarry P. A new ant colony algorithm using the heterarchical concept aimed at optimization of multimimima continuous functions. Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science, 2002, 2463: 216~221
- 21 Pourtakdoust S H, Nobahari H. An extension of ant colony system to continuous optimization problems. Proceedings of the 4th International Workshop on Ant Algorithms/ANTS2004, Lecture Notes in Computer Science, 2004, 3172: 294~301
- 22 张勇德, 黄莎白. 多目标优化问题的蚁群算法研究. 控制与决策, 2005, 20(2): 170~173, 176
- 23 Wen Y, Wu T J. Dynamic window search of ant colony optimization for complex multi-stage decision problems. Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics, 2003, 4091~4097
- 24 闻育, 吴铁军. 求解复杂多阶段决策问题的动态窗口蚁群算法. 自动化学报, 2004, 30(6): 872~879
- 25 玄光南. 遗传算法与工程设计. 北京: 科学出版社, 2000
- 26 Preux Ph, Talbi E G. Towards hybrid evolutionary algorithms. International Transactions in Operational Research, 1999, 6: 557~570
- 27 Xiong W Q, Wei P. A kind of ant colony algorithm for function optimization. Proceedings of the 2002 International Conference on Machine Learning and Cybernetics, 2002, 1: 552~555
- 28 Shen J, Chen L. A new approach to solving nonlinear programming. Journal of Systems Science and Systems Engineering, 2002, 11(1): 28~36

- 29 汪树玉, 杨德栓. 优化原理、方法与工程应用. 杭州: 浙江大学出版社, 1999
- 30 Monmarché N, Venturini G, Slimane M. On how pachycondyla apicalis ants suggest a new search algorithm. Future Generation Computer Systems, 2000, 16(8): 937~946
- 31 Mathur M, Karale S B, Priye S, et al. Ant colony approach to continuous function optimization. Industrial and Engineering Chemistry Research, 2000, 39(10): 3814~3822
- 32 Wilson E D, Hölldobler B. Dense heterarchy and mass communication as the basis of organization in ant colonies. Trend in Ecology and Evolution, 1988, 3: 65~68
- 33 Hewitt C. Viewing control structures as patterns of passing messages. Journal of Artificial Intelligence, 1977, 8(3): 323~364
- 34 Eckart Z, Kalyanmoy D, Lothar T. Comparison of multi-objective evolutionary algorithms: empirical results. Evolutionary Computation, 2000, 8(2): 173~195

## 第7章 蚁群算法的典型应用

### 7.1 引言

自 Dorigo M 等首次将蚁群算法应用于 TSP 以来，国内外许多学者对其进行了大量的研究工作，将其推广到了诸多优化领域，并已经取得了相当丰富的研究成果。

本章在前述章节的基础上，着重对蚁群算法在车间作业调度问题、网络路由问题、车辆路径问题、机器人领域、电力系统、故障诊断、控制参数优化、参数辨识、聚类分析、数据挖掘、图像处理、航迹规划、空战决策、岩土工程、化学工业、生命科学、布局优化等若干领域内的典型应用情况做了系统介绍。本章各小节将着重强调蚁群算法与实际问题的切入点以及算法的改进策略，同时给出了必要的仿真算例，以验证改进后蚁群算法可行性和有效性。

### 7.2 车间作业调度问题

车间作业调度问题 (job-shop scheduling problem, JSP) 既是实际生产中的一个重要问题，也是一个典型的 NP-hard 问题，目前已成为 CIMS 领域内的重要研究课题。Rodammer F A 等<sup>[1]</sup>、Jackek B 等<sup>[2]</sup>对调度方法、复杂性分析和数学规划模型进行了综述，并将解决 JSP 的方法分为精确算法和近似算法两大类。一般意义的 JSP 特征模型可描述如下：

- (1) 存在  $j$  个工作 (job) 和  $m$  个机器 (machine)；
- (2) 每个工作由一系列操作 (或任务/task/operation) 组成；
- (3) 操作的执行次序遵循严格的串行顺序；
- (4) 在特定时间，每个操作需要一个特定机器完成；
- (5) 每台机器在同一时刻不能同时完成不同的工作；
- (6) 在同一时刻，同一工作的各个操作不能并发执行；
- (7) 如何求得从第一个操作开始到最后一个操作结束之间的最长时间间隔 (makespan)。

继遗传算法、禁忌搜索算法、模拟退火算法相继被应用于解决 JSP 之后，国内外许多学者在应用蚁群算法解决多种类型的 JSP 方面做了大量的研究工作<sup>[3~17, 19~21]</sup>，并取得了许多研究成果。

### 7.2.1 蚁群算法与混流装配线调度

混流装配线 (sequencing mixed models on an assembly line, SMMAL) 是指在一定时间内，在一条生产线上生产出多种不同型号的产品，产品的品种可以随顾客需求的变化而变化<sup>[17]</sup>。SMMAL 是 JSP 的具体应用之一。

#### 7.2.1.1 问题描述

这里采用 Toyota 公司提出的调度目标函数，以汽车组装为例，即在组装所有车辆的过程中，所确定的组装顺序应使各零部件的使用速率均匀化。如果不同型号的汽车消耗零部件的种类大致相同，那么原问题可简化为单级 SMMAL 调度问题。由此，该问题可描述为

$$\min \sum_{j=1}^D \sum_{i=1}^n \sum_{p=1}^m (ka_p - b_{ip} - \beta_{j-1,p})^2 x_{ji} \quad (7.2.1)$$

$$x_{ji} = \begin{cases} 1, & \text{如果车型 } i \text{ 在调度中的 } j \text{ 位置} \\ 0, & \text{否则} \end{cases} \quad (7.2.2)$$

$$a_p = \frac{\sum_i d_i b_{ip}}{D} \quad (7.2.3)$$

式中， $i$  表示车型数的标号； $n$  表示需要装配的车型数； $m$  表示装配线上需要的零部件种类总数； $p$  表示生产调度中子装配的标号； $a_p$  表示零部件  $p$  的理想使用速率； $j$  表示车型调度结果（即排序位置）的标号； $D$  表示在一个生产循环中需要组装的各种车型的总和； $d_i$  表示在一个生产循环中车型  $i$  的数量； $b_{ip}$  表示生产每辆  $i$  车型需要零部件  $p$  的数量； $\beta_{j-1,p}$  表示在组装线调度中前  $j-1$  台车消耗零部件  $p$  的数量和  $\beta_{jp} = \beta_{j-1,p} + x_{ji}\beta_{ip}$ ，且  $\beta_{0,p} = 0$ 。

#### 7.2.1.2 算法应用

调度的目标函数如公式 (7.2.1) 所示，这里采用如图 7.1 所示的搜索空间定义，列表示排序阶段（用  $j$  表示），行表示每个阶段可供选择的车型（用  $i$  表示），圆圈的大小表示选择概率的大小，蚁群算法就是不断改变圆圈的大小，最终寻找满意的可行解。这里假设有 3 种车型 A、B、C 排序，每个生产循环需 A 型车 3 辆、B 型车 2 辆、C 型车 1 辆，则每个循环共需生产 6 辆车 ( $D=6$ )。列表示 6 个排序阶段，行表示有 3 种车型可以选择。图 7.1 (a) 所示的是搜索的初始状态，圆圈由局部搜索值和信息素综合而成；图 7.1 (b) 经过若干次迭

代之后，搜索空间变化，此时最可能的可行解是 B-A-C-A-B-A。

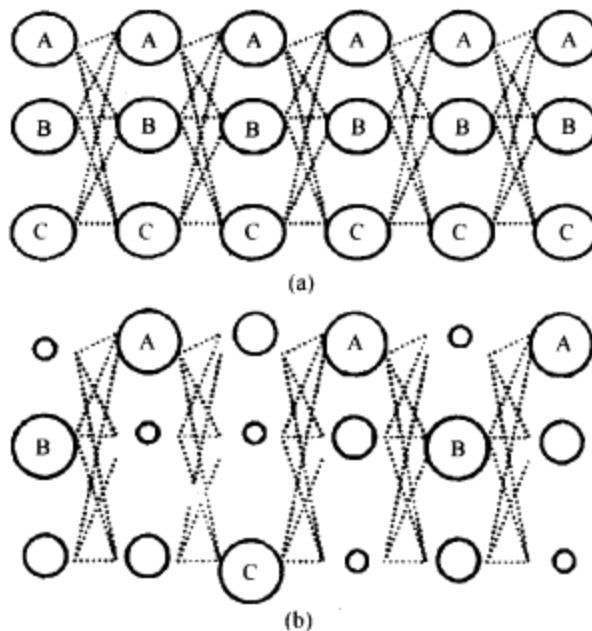


图 7.1 简单 SMMAL 排序的搜索空间举例

利用这种表示方法可以降低问题描述的维数，其问题规模为  $O(n \cdot m)$ ，其中  $n$  是车型数， $m$  是排序长度。

### 1. 局部搜索 ( $\eta_{ij}$ ) 的计算

$$\eta_{ij} = \frac{Q}{\sum_{k=1}^D \sum_{i=1}^n \sum_{p=1}^m (ka_p - b_{ip} - \beta_{k-1,p})^2 x_{ki}} \quad (7.2.4)$$

局部搜索  $\eta_{ij}$  采用的是贪婪法（与目标追随法一致）。目标追随法的思路是，每一步均从当前可选择策略中选取，使目标函数值增加最少的策略，即在确定第  $n+1$  台车辆的车型时，如果有多种车型可供选择，则从中选择一种车型，使第  $n+1$  台车辆组装时各零部件的使用速率最为均匀。若每步只考虑当前的状态，而不考虑全局状态，这样得到的结果常常为局部最优解。但是蚁群算法为每个可选择的车型  $i$  计算  $\eta_{ij}$ ，最后再结合信息素的作用对车型做出选择。

### 2. 状态转移概率

状态转移概率公式如下

$$p_{ij}^k(t) = \begin{cases} \frac{\alpha\tau_{ij} + (1-\alpha)\eta_{ij}}{\sum_{j \notin \text{tabu}_k} (\alpha\tau_{ij} + (1-\alpha)\eta_{ij})}, & \text{若 } (i) \notin \text{tabu}_k \\ 0, & \text{否则} \end{cases} \quad (7.2.5)$$

### 3. 信息素更新规则

$$\Delta\tau_{ij}^k = \begin{cases} \tau_0 \left(1 - \frac{Z_{\text{cutr}} - LB}{\bar{Z} - LB}\right), & \text{如果车型 } i \text{ 在调度中的 } j \text{ 位置} \\ 0, & \text{否则} \end{cases} \quad (7.2.6)$$

式中,  $LB$  表示目标函数的下限值;  
 $\bar{Z}$  表示当前目标函数的平均值;  
 $Z_{\text{cutr}}$  表示当前的目标函数值。这种  
动态标记的方法可在搜索过程中加  
大可行解间信息素的差别, 避免算  
法早熟, 如图 7.2 所示。

而  $\Delta\tau_{ij} = \sum_{k=1}^{n-\text{ant}} \Delta\tau_{ij}^k$ , 进而可将  
信息素更新策略表示为

$$\tau_{ij}(t+n) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij} \quad (7.2.7)$$

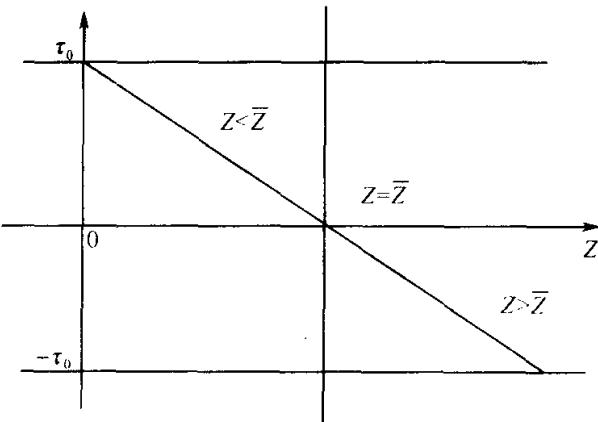


图 7.2 线性动态标注

#### 7.2.1.3 算例分析

这里采用了文献 [18] 中的算例数据, 具体如表 7.1 所示。

表 7.1 各车型的物料单和需求的子装配数

车 型	每个生产循环生产 的各种车型数	子 装 配									
		$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$x_9$	$x_{10}$
A	5	0	17	9	0	4	0	0	18	0	0
B	2	12	13	0	11	0	0	0	1	17	17
C	3	2	4	0	19	0	12	6	4	9	3
D	3	0	15	0	19	0	15	9	9	0	6
E	3	0	0	5	7	8	10	4	0	0	0
每个生产循环需求的子装配数		30	168	60	157	44	111	57	131	61	61

对参数设置经过多种组合实验, 得到蚁群算法最优组合参数为  $\rho=0.9$ ,  $\alpha=0.2$ ,  $Q=20000$ ,  $N_{\text{max}}=400$ ,  $n=5$ , 目标函数值为 2859.8, 排序结果为 C-A-D-

E-B-A-D-E-A-C-A-B-E-D-A-C。

用蚁群算法进行了实验，并与文献 [18] 中所给出的目标追随法、遗传算法和模拟退火算法结果相比较，具体如表 7.2 所示。

表 7.2 蚁群算法与其他启发式算法的对比

算法名称	目标函数	结果改善的百分比 (%)
目标追随法	3293	13
遗传算法	3073	6
模拟退火算法	3162	10
蚁群算法	2859.8	—

由表 7.2 可见，蚁群算法的求解性能优于其他启发式算法。

### 7.2.2 蚁群算法与混流车间动态调度

本小节在不做静态假设和确定性假设的前提下，研究了一种较复杂混流车间的蚂蚁动态调度算法<sup>[19, 20]</sup>。该算法针对柔性制造系统的特点，可求解具有如下特征的混流车间调度问题：

- (1) 车间由多个生产阶段组成，每个阶段都存在着多台加工能力不同的机床。
- (2) 待加工工件的种类  $N$  是一个有限的数值，但每类产品的工件数量可能有多个，根据市场需求或者车间生产计划，车间对各种产品的生产应保持一定的比例。
- (3) 待加工的工件在加工过程中动态到达。
- (4) 由于待加工工件并未预先给定，调度算法的优化目标则为在给定时间内所完成的产品数量。

在蚂蚁动态调度算法中，利用信息素表征机床对加工任务的吸引力，通过奖励机制，使其反映加工路线的优劣；通过工件的试错，找到并增强最优解；通过信息素挥发机制，淘汰劣质解。

#### 7.2.2.1 算法应用

生产实际中的车间调度与自然界中的蚂蚁觅食存在着许多相似之处，其对应关系如表 7.3 所示。

表 7.3 车间调度与蚁群探路的对应

车间调度	蚂蚁觅食
开始(伪工序)	蚁穴
结束(伪工序)	食物源
工件	蚂蚁
机床	一段路径
选择加工机床	选择路段
加工时间的差异	不同路段的长短差异
优化的调度,如工件加工时间较短	较短的路径,即蚂蚁觅食时间较短

在蚂蚁动态调度算法中,信息素反映了机床加工某任务对调度性能的改进程度。若机床不能完成该任务,则令信息量为0,并在加工过程中根据试错结果不断更新。为了利用最新的试错信息决策,蚂蚁动态调度算法并不是在加工之前就为工件指定加工路径,而是在加工过程中动态地寻求。

对任一工件,为加工任务 $j$ 选择机床 $i$ 的概率 $P[i]$ 如下式所示

$$P[i] = \frac{Q[i][j]}{\sum_{i=1}^n Q[i][j]} \quad (7.2.8)$$

式中,  $Q[i][j]$ 表示机床 $i$ 对任务 $j$ 的信息量。

信息素的更新规则如下:

(1) 初始时,可将机床 $i$ 对某一任务 $j$ 的信息量为

$$Q[i][j] = \begin{cases} \text{常数 } C, & \text{若机床 } i \text{ 能完成任务 } j \\ 0, & \text{否则} \end{cases} \quad (7.2.9)$$

(2) 当工件加工完毕后,更新曾加工该工件机床的信息量。

在增大这些历史机床对工件自身任务的信息量同时,对于机床可加工的其他任务的信息量予以一定的减少。针对机床 $i$ 曾加工工件的任务 $k$ ,则有

$$Q[i][j] = \begin{cases} Q[i][j] + I(m), & \text{若 } j = k \\ Q[i][j] + \alpha(m), & \text{否则} \end{cases} \quad (7.2.10)$$

式中,  $I(m)$ 和 $\alpha(m)$ 均为工件总加工时间 $m$ 的减函数,且 $0 < \alpha(m) \leq 1$ 。

(3) 信息素挥发。

每经过一个时间单位,所有的信息素都按一定比例挥发,即

$$Q = Q \cdot \beta \quad (7.2.11)$$

式中,  $\beta$ 表示信息素的挥发比,且 $0 < \beta \leq 1$ 。

根据上述规则,则可得到较复杂混流车间的蚂蚁动态调度算法如下:

```

Begin
(1) 根据公式 (7.2.9) 初始化所有信息素;
(2) While 当前工件状态未检查 //while 循环检查所有工件
    If 当前工件为新工件, 或者当前工件的当前任务加工已经完毕
        then
            依据各机床信息素, 利用公式 (7.2.8) 为下一加工任务随机选择机床;
    else if 当前工件加工完毕
        then
            根据公式 (7.2.10), 更新历史机床的信息量;
            令当前工件为下一工件
    End while
(3) 根据公式 (7.2.11) 进行信息素挥发;
(4) 用户控制; //在算法运行中允许用户动态更改及退出算法
(5) 时间单位加 1, 转公式 (7.2.9)
End

```

### 7.2.2.2 算例分析

利用蚂蚁动态调度算法对两阶段的柔性制造系统进行了仿真调度（如图 7.3 所示）。每个生产阶段都有两台不同的机床，每台机床都有输入和输出缓冲区，可存放一个待加工的工件，而车间缓冲可存放两个工件，工件停留在缓冲区内等待加工。

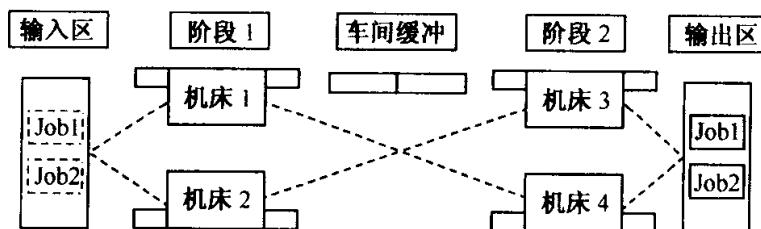


图 7.3 车间的布局

两类待加工工件都必须依次经过两个阶段的加工。当机床在不同加工任务间切换时，需要对其进行调整，实验中假定所有机床的调整时间都是 10 个时间单位，两类待加工工件按照确定的比例动态到达。为了避免车间阻塞，这里设定工件的到达速度略低于车间最快吞吐速度。

实验采用的性能指标是 2000 个时间单位内完成的工件数量。由于蚂蚁动态调度算法在机床选择上存在着随机性，其调度结果并不是固定值，下面给出 10

次调度后的平均数据。本实验中所采用的信息素更新规则参数如下

$$C = 10; \quad I(m) = \frac{30000}{m^3}; \quad \alpha(m) = \frac{1.2}{\sqrt[4]{I(m)}}; \quad \beta = 0$$

### 1. 确定加工时间混流车间调度实验

在该实验中，各机床对工件的加工时间是预先知道的确定值，具体如表 7.4 所示。

表 7.4 车间加工时间表

加工时间		阶段 1		阶段 2	
		机床 1	机床 2	机床 3	机床 4
工件 1	任务 1 (首工序)	25	40	—	—
	任务 2 (次工序)	—	—	[10, 20]	30
工件 2	任务 3 (首工序)	25	10	—	—
	任务 4 (次工序)	—	—	35	25

实验分别考察了在不同的工件混合比下，蚂蚁动态调度算法和启发式算法的性能。用于对比的启发式算法如下：

(1) 最早完成时间 (earliest completion time, ECT)：当前一任务加工完毕后，为下一加工任务选择能最早地将其加工完毕的机床。

(2) 最早开始时间 (earliest start time, EST)：当前一任务加工完毕后，为下一加工任务选择能最早开工的机床，即机床前一工件加工完毕，且调整好的时间最早。

(3) 最早准备时间 (earliest ready time, ERT)：当前一任务加工完毕后，为下一加工任务选择最早就绪的机床，即机床前一工件加工完毕的时间最早。

蚂蚁动态调度算法与其他启发式算法实验结果如表 7.5 所示。

表 7.5 确定加工时间调度实验结果

算法类型	1 : 1		4 : 3		3 : 4		2 : 3	
	工件 1	工件 2	工件 1	工件 2	工件 1	工件 2	工件 1	工件 2
蚂蚁动态调度算法	72. 1	69. 7	89. 9	66. 6	61. 7	82	59. 4	89. 1
ECT	64	65	80	58	60	79	58	86
ERT	56	56	70	51	54	70	48	69
EST	65	65	70	51	51	67	48	69

由表 7.5 可见，在不同工件混合比的情况下，蚂蚁动态调度算法的性能均优于其他启发式算法。

## 2. 随机加工时间混流车间调度实验

在生产实践中，加工时间往往与机床的状态以及工人操作有关，很难保持一个固定的数值，往往会在一定范围内变动。传统的调度算法都依靠事先的计算来为工件指定加工路径，因而无法求解随机加工时间的调度问题。由于蚂蚁动态调度算法是动态选择机床的，因而适合求解随机加工时间的混流车间调度问题。

在本实验中，每个工件的实际加工时间为表 7.6 所给出区间内的随机数，且这一时间只有在实际加工时才能确定，实验采用的工件混合比为 1:1。

表 7.6 用蚁群算法求解随机车间调度问题

加工时间		阶段 1		阶段 2		2000 时间的生产量
		机床 1	机床 2	机床 3	机床 4	
工件 1	任务 1 (次工序)	[20, 30]	[35, 45]	—	—	76.7
	任务 2 (次工序)	—	—	[10, 20]	[25, 35]	
工件 2	任务 3 (次工序)	[20, 30]	[5, 15]	—	—	73.6
	任务 4 (次工序)	—	—	[30, 40]	[20, 30]	

由表 7.6 可见，对于随机加工时间，蚂蚁动态调度算法亦能迅速找到较优的加工路径，并取得良好的调度效果。

## 3. 蚂蚁动态调度算法的自适应性实验

车间运行总要受到各种因素的干扰，如机床故障、车间输入的变化等，一个实用的调度算法应能根据车间内、外部环境的变化自动做出调整。在确定加工时间混流车间调度实验基础上所进行的适应性实验结果如图 7.4 和图 7.5 所示。

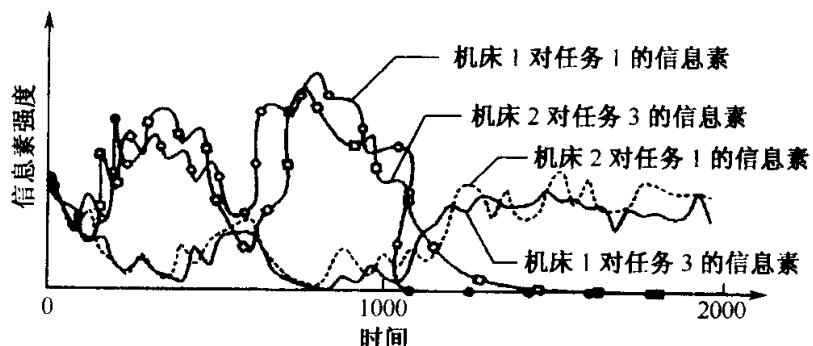


图 7.4 蚂蚁动态调度算法对车间配置变化的适应性

图 7.4 给出了当车间配置变化时信息素强度的变化情况。在  $t=1000$  之前，机床 1 对任务 1 的信息素强度很高，而机床 2 对任务 1 的信息素强度接近 0，这意味着几乎所有的任务 1 都会由机床 1 加工。同样，机床 2 专注于任务 3 的加

工。在  $t=1000$  时，通过算法中的用户控制，将机床 1 对任务 1 的信息素设为 0，以模拟故障的产生，表示今后机床 1 无法加工任务 1。可以看到，再经过一定的适应后，蚂蚁动态调度算法重新找到了稳定的加工路径和次序，即机床 2 加工任务 1，机床 1 加工任务 3。

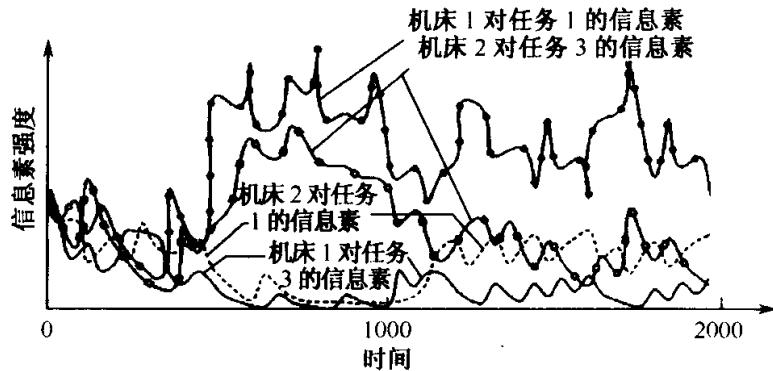


图 7.5 蚂蚁动态调度算法对车间输入变化的适应性

图 7.5 给出了当车间输入变化时信息素强度的变化情况。在  $t=1000$  之前，输入工件的混合比是 1 : 1，蚂蚁动态调度算法指派由机床 1 加工任务 1，机床 2 加工任务 3。在  $t=1000$  时，将工件混合比改为 2 : 1 后，待加工的任务 1 增加，这时机床 2 对任务 1 和任务 3 的信息素交互波动，表明机床 2 将同时加工任务 1 和任务 3，而机床 1 仍专门加工任务 1。

适应性实验表明，蚂蚁动态调度算法具有良好的自适应性，对于车间内外部环境的变化能够自动调节，并能找到新的满意解。

### 7.2.3 双向收敛蚁群算法与车间作业调度

本小节在 JSP 图形化定义的基础上，借鉴精英策略的思路，研究了一种使用多种挥发方式的双向收敛蚁群算法<sup>[21]</sup>。

#### 7.2.3.1 JSP 图形化定义

为了便于比较双向收敛蚁群算法与基本蚁群算法求解 JSP 的性能，这里采用 Muth 和 Thompson 于 1963 年所提出的 JSP6×6 基准问题，如表 7.7 所示（表中  $t$  表示任务所需时间）。

表 7.7 中， $Job_1$  的第一个任务  $Task_1$  需在  $Machine_3$  上完成，历时 1 个时间单位；第二个任务  $Task_2$  在  $Machine_1$  上完成，历时 3 个时间单位；其余依此类推。 $Job$  中的后续任务不能在前面任务完成之前启动。求解 JSP，就是针对每个  $Machine$ ，调度其上的任务次序。

表 7.7 Muth & Thompson 6×6 基准问题

项目	$m, t$					
Job <sub>1</sub>	3,1	1,3	2,6	4,7	6,3	5,6
Job <sub>2</sub>	2,8	3,5	5,10	6,10	1,10	4,4
Job <sub>3</sub>	3,5	4,4	6,8	1,9	2,1	5,7
Job <sub>4</sub>	2,5	1,5	3,5	4,3	5,8	6,9
Job <sub>5</sub>	3,9	2,3	5,5	6,4	1,3	4,1
Job <sub>6</sub>	2,3	4,3	6,9	1,10	5,4	3,1

可将上述 Muth & Thompson 问题用矩阵表示如下

$$T = \begin{bmatrix} m_3 & m_1 & m_2 & m_4 & m_6 & m_5 \\ m_2 & m_3 & m_5 & m_6 & m_1 & m_4 \\ m_3 & m_4 & m_6 & m_1 & m_2 & m_5 \\ m_2 & m_1 & m_3 & m_4 & m_5 & m_6 \\ m_3 & m_2 & m_5 & m_6 & m_1 & m_4 \\ m_2 & m_4 & m_6 & m_1 & m_5 & m_3 \end{bmatrix} \quad (7.2.12)$$

$$P = \begin{bmatrix} t_1 & t_3 & t_6 & t_7 & t_3 & t_6 \\ t_8 & t_5 & t_{10} & t_{10} & t_{10} & t_4 \\ t_5 & t_4 & t_8 & t_9 & t_1 & t_7 \\ t_5 & t_5 & t_5 & t_3 & t_8 & t_9 \\ t_9 & t_3 & t_5 & t_4 & t_3 & t_1 \\ t_3 & t_3 & t_9 & t_{10} & t_4 & t_1 \end{bmatrix} \quad (7.2.13)$$

其中，矩阵  $T$  表示每个工作的任务调度顺序；矩阵  $P$  表示相应的时间间隔。由此，可将上述问题转换成如图 7.6 所示的结构。

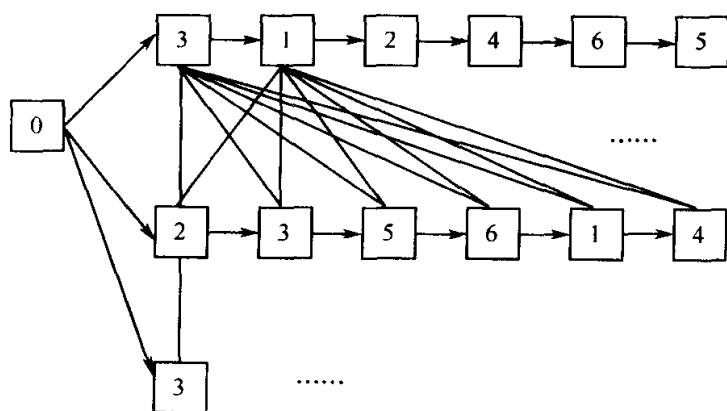


图 7.6 JSP 的图形化定义

图 7.6 所示的结构由 37 个节点组成，增加了一个虚拟起点 0，从点 0 开始可以提供通往  $Job_1, Job_2, \dots, Job_6$  的单向通路，此外的节点  $P_{i,j}$  ( $1 \leq i, j \leq 6$ ) 表示 Machine 矩阵  $T$  中相应位置的点，即  $T_{ij}$  中的 Machine 代号。因此，图 7.6 中的第  $i$  行（除起始点 0 以外）表示表 7.7 中的第  $i$  个  $Job$ 。每个工作内的各个任务由有向弧连接，各个工作之间的任务由无向弧连接。

### 7.2.3.2 算法应用

#### 1. 相关定义

对图 7.6 所示的图结构做如下定义：

**定义 7.2.1 操作 (operation) 用节点表示，其定义为**

$$Operation = \langle sTime, eTime, isDone, NEXT \rangle \quad (7.2.14)$$

式中， $sTime$  和  $eTime$  分别表示本操作的开始和结束时间； $isDone$  表示本节点代表的操作是否已经完成，或者蚂蚁是否已经走过节点； $Next$  表示下一步的结构，包含了从本节点能够到达的节点表和到达这些节点的路径上的信息量。

**定义 7.2.2 图 (graph) 为操作的集合**

$$Graph = \langle Operation, Machine \rangle \quad (7.2.15)$$

式中， $Machine$  为表 7.7 中操作所占用机器的矩阵。在  $Operation$  节点中已经包含了节点之间的连接，因此不需要定义节点之间的关系。

**定义 7.2.3 虚拟起始点 (startpoint) 为**

$$Startpoint = \langle NEXT, nextMachine \rangle \quad (7.2.16)$$

虚拟起始点只连接每个工作的第一个操作。操作所占用的时间可由  $NEXT$  结构中的  $Operation$  体现，因此虚拟起始点不占用操作时间。

#### 2. 算法描述

大量实验证明，没有信息素挥发的蚁群算法很难得到即使是局部的最优解<sup>[22]</sup>。下面具体介绍这里所提出的双向收敛蚁群算法。

(1) 生成一代蚂蚁。

根据算法规定的数量放出蚂蚁，为保证操作符合工作的调度顺序，在每只蚂蚁寻找路径的过程中，首先判断目的节点的前驱是否已经完成，在前驱已经完成并且本身尚未完成的所有节点中，使用信息量作为概率选取下一步目标。为保证蚂蚁遍历的次序符合  $Job$  操作的次序要求，这里采用如下原则：

- ①  $Job$  中同一行的节点完成后不能直接转向自己的非直接后继点；
- ② 使用评价函数计算所经路径代表的时间间隔 Makespan 时，遵守任务的先后次序，使蚂蚁行走时路径不代表任务次序；
- ③ 次序的含义在计算 Makespan 时被加到路径上。

在双向收敛蚁群算法中，蚂蚁无须每走一步都对下一步的机器占用和工序次序做判断，大大降低了计算量，提高了算法的效率。同时，双向收敛蚁群算法在计算状态转移概率的过程中，不按照传统的方法将路径长度考虑在内。假定蚂蚁行走的过程中不会重复已经走过的路径，则蚂蚁选择下一条可行路径的状态转换概率为

$$p_{ij}(t) = \frac{\tau_{ij}(t)}{\sum_{j \in \text{allowednodes}} \tau_{ij}(t)} \quad (7.2.17)$$

### (2) 评价和激励。

当本代蚁群中的所有蚂蚁完成对有向图中所有节点的遍历后，使用评价函数对得到的所有路径进行评价，并按如下规则更新信息素

$$\tau_{ij}(t + \Delta t) = (1 - \rho)\tau_{ij}(t) + \Delta\tau_{ij}(t + \Delta t) \quad (7.2.18)$$

上式中的第一部分表示每一代蚂蚁走完全程后所有信息素的挥发；第二部分表示信息素的修改，其计算公式如下

$$\Delta\tau_{ij}(t + \Delta t) = \begin{cases} \frac{Q}{f(\text{bestRoad})}, & t+n \text{ 过程中的最优值} \\ -\frac{Q'}{f(\text{worstRoad})}, & t+n \text{ 过程中的最差值} \\ 0, & \text{其他情况} \end{cases} \quad (7.2.19)$$

式中， $Q$  表示单位路径上的信息量； $Q'$  表示单位路径上用于惩罚最差值的信息量； $f()$  表示路径评价函数。

基本蚁群算法使用参数调节的方法避免算法陷入局部最优，这种方法取决于具体的参数数值，往往导致一套参数对应于一个具体问题，降低了算法的通用性。而双向收敛蚁群算法从两个方向进行反馈，很大程度上避免了对参数的依赖，同时加快了算法的收敛速度。

### (3) 循环执行。

若满足算法结束条件，则退出循环并输出计算结果；否则，跳转到第(1)步。

## 3. 评价函数

评价函数的基本思路是：规定在每只蚂蚁所经过的路径上，前继节点的开始时间不会落在后续节点之后。由于蚂蚁在寻径过程中已经考虑到了操作在工作中的先后次序，结果中的节点串将表示不同工作的时间次序。

具体算法如下：

```
// 初始化第一个操作的时间
O1. startTime=0;
O1. endTime=O1. executeTime;
MO1. Time=O1. endTime;
```

```

Makespan=0;
For (int i=2; i<=m * n; i++) //根据蚂蚁的路径找到相应操作的起止时间
{
    Oi. startTime=Oi-1. endTime; //本次操作只能在前面操作已经完成的情况下执行
    找到 Oi 对应的机器 MOi ;
    If (MOi. Time>Oi. startTime) then //如果这时机器被占用
        Oi. startTime=MOi. Time; //等机器释放后开始
        Oi. endTime=Oi. startTime+Oi. executeTime;
        MOi. Time=Oi. endTime;
    }
    For (int j=1; j<=NumberOfMachine; j++) //求出完成所有工作所需时间
    {
        If Makespan<Mj. Time then
            Makespan=Mj. Time
    }
}

```

使用双向收敛蚁群算法可以求出每只蚂蚁遍历图的所有节点之后所得出的 JSP 的解。根据这个解，对所有蚂蚁路径中最优的一条和最差的一条使用公式 (7.2.19) 更新信息素。

### 7.2.3.3 算例分析

这里采用 Muth 和 Thompson 的  $6 \times 6$  基准问题作为仿真算例。表 7.8 比较了双向收敛蚁群算法和基本蚁群算法所得到的解，所得结果为 5 次求解后的平均值。

表 7.8 双向收敛蚁群算法和基本蚁群算法最优解的比较

算法类型	蚂蚁数目	循环次数	蚂蚁总数	$\rho$	收敛结果
基本蚁群算法	36	3000	108 000	0.010	55
双向收敛蚁群算法	200	311	62 200	0.017	55

由表 7.8 可见，双向收敛蚁群算法使用了较多蚂蚁进行并行搜索，提高了搜索过程的挥发系数，从而在较少的代数中得到了 JSP 的解。

## 7.3 网络路由问题

网络路由方面的研究主要通过两个途径来提高服务质量 (quality of service,

QoS)：一条途径是节点控制；另一条途径是整网或局部网络控制。节点控制在单节点或单链路完成，主要控制业务对单节点共享资源的占用；整网或局部网络控制通常通过对路由与信令的控制达到对业务流或业务连接在网络中传输的直接控制。因为路由直接关系到网络性能，所以 QoS 路由成为解决 QoS 问题的核心技术之一，也是当今网络技术领域内的一个研究热点。

QoS 路由的任务就是在网络中寻找一条路径，使其能满足带宽、时延、时延抖动和费用的限制。Wang Z 等<sup>[23]</sup>证明了如果 QoS 路由至少包含两个限制时，它是一个 NP-C 问题。传统的路由算法很难有效地解决 NP-C 问题，因此一些学者基于蚁群算法提出了许多行之有效的解决方案<sup>[24~44]</sup>。

### 7.3.1 蚁群算法与 QoS 路由

随着 Internet 的飞速发展，对 QoS 路由的研究已经引起了众多关注。以前对 QoS 路由的研究大多采用的是平面网络，即所有路由器都在同一级内，一般称之为平面 QoS 路由；与之相对应的还有分级 QoS 路由，其基本思想是将路由器分成多个逻辑组，每个组又可包含更小的组。本小节将对基于蚁群算法的平面 QoS 路由和分级 QoS 路由分别进行研究。

#### 7.3.1.1 平面 QoS 蚂蚁路由算法

##### 1. 问题描述

可将网络模型表示为一个有向图  $G=(V, E)$ ，其中  $V$  是图中所有交换节点组成的集合， $E$  是图中所有边的集合，每一条边表示相邻两节点间的直达通信路径<sup>[40]</sup>。不失一般性，假定相邻两节点间最多仅有一条边。同时，假定  $B(l)$  表示链路  $l$  的可用带宽，对于一个路由请求  $w$ ，路由算法如果能够找到一条具有小费用的路径，同时满足如下 4 个条件，则此路由请求  $w$  就可接受。

(1) 在  $w$  的路由的每条路径  $l$  上，带宽可用限制为

$$B(l) \geq B_w \quad (7.3.1)$$

(2) 在  $w$  的路由上，端到端时延限制为

$$\sum_{n \in V_1} DN(n) \sum_{l \in E_1} + DL(l) \leq D_w \quad (7.3.2)$$

(3) 在  $w$  的路由上，端到端丢失率限制为

$$\prod_{n \in V_1} (1 - LR(n)) \geq (1 - L_w) \quad (7.3.3)$$

这里只考虑由于节点缓存器溢出引起的丢失。

(4) 在目的节点，端到端时延抖动限制为

$$NDV(d) \leq J_w \quad (7.3.4)$$

式中,  $B_w$ 、 $D_w$ 、 $L_w$  和  $J_w$  分别表示 QoS 要求的带宽、时延、丢失率和时延抖动限制;  $DN$  表示节点处理时延,  $DN: V \rightarrow R^+$ ;  $DL$  表示链路时延, 且  $DL: E \rightarrow R^+$ ;  $V_1$  表示  $w$  路由的节点集合;  $E_1$  表示  $w$  的路由链路集合  $LR$  是节点丢失率, 且  $LR: V \rightarrow R^+ \cup \{0\}$ ;  $DV$  表示在目的节点  $d$  的节点时延变化  $NDV: V \rightarrow R^+ \cup \{0\}$ , 其中  $R^+$  是正实数集合。

## 2. 算法设计

假定平面 QoS 蚂蚁路由算法中有  $m$  只蚂蚁, 并且采用了全局和局部信息素更新规则。

(1) 状态转移规则。

位于节点  $r$  的第  $i$  只蚂蚁依据下述规则来选择节点  $s$ :

如果  $q \leq q_0$ , 有

$$\rho_i(r, s) = \begin{cases} \max(\text{pheromone}(i, r, s)), & \text{若 } s \in J_i(r) \\ 0, & \text{否则} \end{cases} \quad (7.3.5)$$

否则, 有

$$\rho_i(r, s) = \begin{cases} \frac{\text{pheromone}(i, r, s)}{\sum_{u \in J_i(r)} \text{pheromone}(i, r, u)}, & \text{若 } s \in J_i(r) \\ 0, & \text{否则} \end{cases} \quad (7.3.6)$$

采用此定义来实现蚂蚁状态的转移可保证寻找优化路径时避免陷入局部最优。

(2) 信息素的局部更新规则。

对于第  $i$  只蚂蚁, 如果节点  $r, s$  是该蚂蚁所选路径上的两个相邻节点, 则信息素  $\text{pheromone}(i, r, s)$  用公式 (7.3.7) 来调节; 否则, 不调节。

$$\text{pheromone}(i, r, s) \leftarrow (1 - \alpha_0) \text{pheromone}(i, r, s) + \alpha_0 \cdot \text{cons} \quad (7.3.7)$$

若没有局部更新, 所有蚂蚁将在前一次的最好路径的有限相邻区域内搜寻。

(3) 信息素的全局更新规则。

对于第  $i$  只蚂蚁, 如果节点  $r, s$  是该蚂蚁所选路径上的两个相邻节点, 则信息素  $\text{pheromone}(i, r, s)$  按公式 (7.3.8) 来调节; 否则, 按公式 (7.3.9) 来调节。

$$\text{pheromone}(i, r, s) \leftarrow (1 - \alpha_1) \text{pheromone}(i, r, s) + \alpha_1 \cdot F \quad (7.3.8)$$

$$\text{pheromone}(i, r, s) \leftarrow (1 - \alpha_1) \text{pheromone}(i, r, s) \quad (7.3.9)$$

为了定义公式 (7.3.8) 中的限制函数  $F$ , 首先引入几个符号定义: 如果节点  $r$  是第  $i$  只蚂蚁所选路由的节点, 则  $N_r^i = 1$ , 否则  $N_r^i = 0$ ; 如果从节点  $r$  到节点  $s$  的边是第  $i$  只蚂蚁所选路由的边, 则  $P_{rs}^i = 1$ , 否则  $P_{rs}^i = 0$ ;  $LB_{rs}$ 、 $LC_{rs}$  和  $LD_{rs}$  分别表示从节点  $r$  到节点  $s$  的边的带宽、费用和时延;  $ND_r$  和  $NL_r$  分别表

示节点  $r$  的处理时延和丢失率。由此，可将限制函数  $F$  定义为

$$F = -F_1 + F_2 \quad (7.3.10)$$

$$F_1 = \sum_{i=1}^V \sum_{\substack{j=1 \\ j \neq i}}^V LC_{ij} \cdot P_{ij}^d \quad (7.3.11)$$

$$F_2 = A \cdot \sum_{i=1}^V \sum_{\substack{j=1 \\ j \neq i}}^V H(Z_{ij}) + B \cdot H(Z_2) + C \cdot H(Z_3) \quad (7.3.12)$$

如果  $Z < 0$ ,  $H(Z) = 0$ ; 否则,  $H(Z) = Z$ 。

$$Z_{ij} = P_{ij}^d \cdot LB_{ij} - B_w \quad (7.3.13)$$

$$Z_2 = D_w - (\sum \sum LD_{ij} \cdot P_{ij}^d + \sum N_i^d \cdot ND_i) \quad (7.3.14)$$

$$Z_3 = \prod_{i=1}^V (1 - N_i^d \cdot NL_i) - (1 - L_w) \quad (7.3.15)$$

式中,  $V$  表示网络的节点数目;  $A$ 、 $B$  和  $C$  分别表示带宽可用限制、端到端时延限制和端到端丢失率限制, 且为正实数;  $F_1$  表示费用限制;  $F_2$  表示 QoS 限制。

通过上述定义可见, 如果所选路由的总费用最小, 同时 QoS 限制也满足要求, 那么最优蚂蚁所选路由的各链路上的信息素应该增加更多。为了进一步提高算法的收敛性能, 这里做了以下两点改进:

- 在蚁群算法迭代中不考虑时延抖动, 而是在算法执行前进行处理;
- 通过删除带宽小于需求带宽的链路, 把网络滤成一个新的简单网络, 因此在  $F$  函数中设  $A = 0$ 。

平面 QoS 蚂蚁路由算法的具体步骤如下:

- (1) 如果  $NDV(d) > J_w$ , 那么退出; 否则, 跳转到第 (2) 步。
- (2) 通过删除带宽小于需求带宽的链路, 把网络滤成一个新的简单网络。
- (3) 初始化网络拓扑中各边的相应信息素。
- (4) 从蚁巢开始放出  $m$  只蚂蚁去寻找食物源, 在第一个时间单位, 每只蚂蚁从节点集合中随机地选择一个节点, 然后各蚂蚁通过重复应用状态转移规则来选择各自的路径。在选路过程中, 如果蚂蚁在到达目的节点前死亡, 另外一只和死亡的蚂蚁同类的蚂蚁重新放出来代替死亡的蚂蚁, 重新开始选择从源节点到目的节点的路径。当某只蚂蚁成功地完成路由选择后, 该蚂蚁所选路由各路径上的信息素根据局部更新规则进行更新。
- (5) 对所有的蚂蚁重复第 (4) 步, 直到  $m$  只蚂蚁都完成了第 (4) 步。
- (6) 选择建立了最小费用并满足 QoS 限制的路由的蚂蚁, 然后使用全局更新规则对该蚂蚁所选的路由的各路径上的信息素进行更新。
- (7) 重复第(4)~(6)步, 直到获得满意的结果。

### 3. 算例分析

图 7.7 是所考虑的网络拓扑, 每个顶点用  $\langle ndl, lr, dv \rangle$  表示, 其中的元素分

别表示节点时延、节点丢失率和节点时延变化。每条边用 $\langle \text{cost}, \text{bw}, \text{ldl} \rangle$ 表示，其中的元素分别表示边的费用、带宽和链路时延。

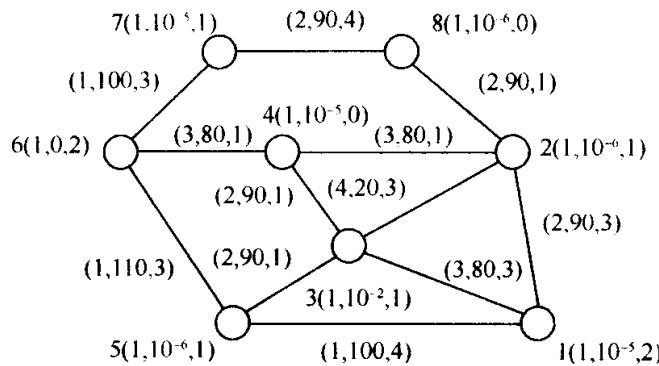


图 7.7 网络的拓扑及其参数

假定有 3 个单播路由请求  $(1, 6)$ 、 $(2, 6)$  和  $(3, 8)$ ，它们的 QoS 要求是： $B_w = 70$ ， $D_w = 8$ ， $L_w = 10^{-5}$ ， $J_m = 3$ ，经过简化的网络及其参数如图 7.8 所示。

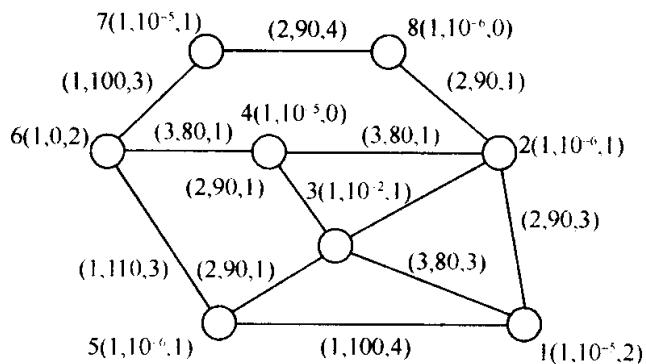


图 7.8 简化网络及其参数

设置  $m=20$ ， $\alpha_0=0.069$ ， $\alpha_1=0.079$ ， $\text{cons}=0.32$ ， $q_0=0.91$ ， $A=0$ ， $B=10$ ， $C=15$ ，信息素初始值=100.0。

通过仿真所获得的全局优化结果如表 7.9 所示，该仿真的平均迭代次数只有 134 次。此外，还可以发现，对于路由请求  $(1, 6)$ ，存在路由  $1 \rightarrow 5 \rightarrow 6$ ，虽然其费用仅为 2，但其时延达到了 8，因此没选择该路由。

表 7.9 平面 QoS 蚂蚁路由算法计算结果

路由请求 $(s, t)$	QoS 路由结果		
	选择的路由	费 用	时 延
(1, 6)	1→2→4→6	8	7
(2, 6)	2→4→6	6	3
(3, 8)	3→4→2→8	7	5

为了做进一步分析，对于相同的路由请求，还进行了 500 次仿真实验，其中发现最优解的几率为 89.6%，次优解的几率为 10.4%，没有无效解。这说明改进后的蚁群算法具有良好的寻优性能。

### 7.3.1.2 分级 QoS 蚂蚁路由算法

#### 1. 算法设计

这里以一个二级网络为例（如图 7.9 所示）。每个路由器即为一个 level-0 节点，由多个 level-0 节点和 level-0 链路形成的 LAN 称为 level-1 节点，level-1 节点之间通过 level-1 链路连接<sup>[41]</sup>。在大多数应用中，时延要求是最为重要的，为了方便起见，这里在研究 QoS 路由时仅考虑时延限制。

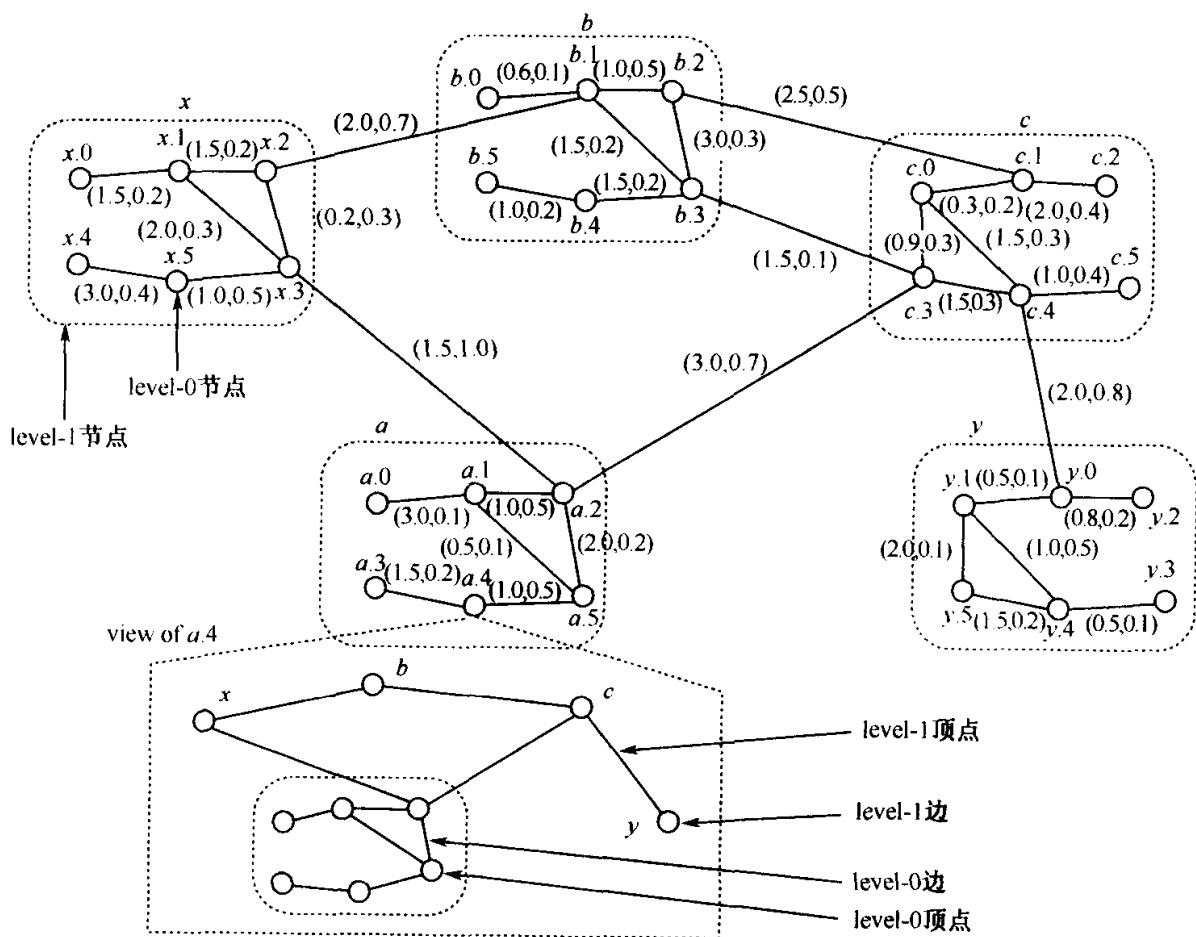


图 7.9 具有二级路由的网络（各边括号内的数表示时延、费用）

本小节所构造的分级 QoS 蚂蚁路由算法的具体步骤如下：

- (1) 初始化。
- (2) 根据呼叫的源-目的节点，确定此源-目的节点是否在同一级内：若在（如 *x.1*→*x.3*），直接在相应级 *x* 内调用蚁群算法进行选路（level-0 路由）；若不

在同一级内，则执行第（3）步。

（3）根据 level-1 节点的相邻关系，选最少 level-1 节点数的路由（称为 level-1 路由）。

（4）根据此路由，确定每个 level-1 节点内的源-目的节点，作为蚁群算法的输入，同时调用蚁群算法进行选路（称为 level-0 路由），并记录结果。

（5）根据第（3）、（4）步的结果，扩展出最终路由，判断是否满足要求：若满足，则结束此次选路；否则，判断是否还存在 level-1 路由：若存在，则采用最短路径法在剩余的可行路由中选择最短的一条，并将其定为新的 level-1 路由，跳转到第（4）步；否则，此次呼叫没有可用路由，算法结束。

下面用一个实例来详细说明分级 QoS 蚂蚁路由算法。例如寻找  $x.0 \rightarrow c.2$  的路由，如图 7.9 所示。

首先确定出 level-1 路由： $x \rightarrow b \rightarrow c$  或  $x \rightarrow a \rightarrow c$ 。假定选  $x \rightarrow a \rightarrow c$ ，则可得出  $(x.0, x.3), (a.2, a.2), (c.3, c.2)$  三个源-目的对，同时启动三个蚁群算法计算出满足要求的路由链路，假定为  $x.0 \rightarrow x.1 \rightarrow x.3, a.2 \rightarrow a.2, c.3 \rightarrow c.0 \rightarrow c.1 \rightarrow c.2$ ，根据此结果，可扩展出最终路由  $x.0 \rightarrow x.1 \rightarrow x.3 \rightarrow a.2 \rightarrow c.3 \rightarrow c.0 \rightarrow c.1 \rightarrow c.2$ ，若此路由满足要求就结束；否则，重新选定 level-1 路由，进行新的选路。

基于蚁群算法的 level-0 路由算法的状态转移规则和信息素局部更新规则同前所述，但对信息素全局更新规则中的限制函数  $F$  的定义做了如下改进。

$$F = \frac{F_2}{F_1} \quad (7.3.16)$$

$$F_1 = \sum_{i=1}^V \sum_{\substack{j=1 \\ j \neq i}}^V LC_{ij} \cdot P_{ij}^d \quad (7.3.17)$$

$$F_2 = H(Z) = \begin{cases} 0, & \text{若 } Z < 0 \\ 1, & \text{否则} \end{cases} \quad (7.3.18)$$

$$Z = \Delta_1 - \sum_{i=1}^V \sum_{\substack{j=1 \\ j \neq i}}^V LC_{ij} \cdot P_{ij}^d \quad (7.3.19)$$

式中， $\Delta_1$  表示 LAN 内节点的端到端时延限制。

## 2. 算例分析

为了验证分级 QoS 蚂蚁路由算法的可行性和有效性，这里针对图 7.9 所示的网络结构进行了计算机仿真。设置  $m = 10, \alpha_0 = 0.069, \alpha_1 = 0.079, \text{cons} = 0.32, q_0 = 0.91$ ，各边的信息量初始值 = 8.0。

假定有 3 个单播呼叫，分别为  $(x.0, x.5), (x.1, c.2)$  和  $(x.5, y.2)$ ，其时延要求相同，均为  $\Delta_1 = 5.0$  (LAN 内)， $\Delta = 10.0$  (总的端到端时延限制)。仿真结果如表 7.10 所示。

表 7.10 分级 QoS 蚂蚁路由算法的计算结果

呼 叫	路 径	费 用	时 延
(x. 0, x. 5)	x. 0→x. 1→x. 3→x. 5	1. 0	4. 5
(x. 1, c. 2)	x. 1→x. 3→a. 2→c. 3→c. 0→c. 1→c. 2	2. 9	9. 7
(x. 5, y. 2)	x. 5→x. 3→a. 2→c. 3→c. 4→y. 0→y. 2	3. 5	9. 8

为了对比，可以把此网络看成是一平面网络，时延限制  $\Delta=10.0$ ，采用分级 QoS 蚂蚁路由进行了仿真，仿真结果除了(x. 1, c. 2)外，其余均相同。此次 (x. 1, c. 2) 的路径为：x. 1→x. 2→b. 1→b. 3→c. 3→c. 0→c. 1→c. 2，费用为 2. 1，时延为 9. 7，比分级 QoS 路由算法具有更小的费用。

为了进一步测试分级 QoS 蚂蚁路由算法的性能，在这里随机给定图 7.9 中各边的参数，level-0 链路的时延值在(0, 3.0)之间，level-1 链路的时延值在(0, 5.0)之间，费用在(0, 1.1)之间，分别采用两种方法进行了 600 次测试，其实验统计结果如表 7.11 所示。

表 7.11 分级 QoS 蚂蚁路由算法与平面 QoS 蚂蚁路由算法的计算结果对比

路 由 方法	最 优 解	次 优 解	无 效 解
分 级 QoS 蚂 蚁 路 由 算 法	69. 5%	27. 4%	3. 1%
平 面 QoS 蚂 蚁 路 由 算 法	73. 1%	24. 1%	2. 8%

经多次实验发现，分级 QoS 蚂蚁路由算法的性能和平面 QoS 蚂蚁路由算法相差不多，但运算速度却比平面 QoS 蚂蚁路由算法提高了 30% 左右。对于连接更加复杂的网络，速度会提高更多，这在实际应用中是非常有意义的。

### 7.3.2 蚁群算法与拥塞规避路由

开放最短路径优先是现有网络普遍采用的一种路由协议，它采用的路由算法——链路状态路由 (link state-routing, LS) 算法只具有寻路功能，而不具有任何拥塞响应机制<sup>[43]</sup>。当一条链路即将或者已经发生拥塞时，只能简单地丢弃数据包，这使得网络资源无法得到充分的利用。

这里研究了一种基于蚁群算法的具有拥塞规避特性的路由算法。该算法能够迅速探索到两点之间的最优路径，同时能够预测到链路的拥塞状态，从而迅速寻找一条新的路径，将流量分散，以达到解除拥塞状态的目的。

### 7.3.2.1 算法设计

#### 1. 蚂蚁路由算法简介

为了在网络中应用蚂蚁寻路机制，文献 [44] 提出了一种蚂蚁路由算法，网络中的每个节点将路由表以一张概率表替换，概率值相当于信息素的强度，可把概率表称为信息素表。这里以  $P_{nd}$  表示信息素表中到目标节点  $d$  下一跳选择相邻节点  $n$  的概率。当需要转发数据包时，总是根据表中与数据包目标地址相对应的一条记录选择概率值最大的相邻节点作为下一跳。

蚁群算法依赖信息素表中的概率值更新来模拟蚂蚁遗留信息素的行为<sup>[45]</sup>。每个节点定期发送一个探测包——蚂蚁，用于收集节点和链路的状态信息。探测蚂蚁的目标节点是随机选择的。每到达一个节点，探测蚂蚁将更新该节点信息素表中目标地址为该蚂蚁源地址这一条记录中的概率值。

信息素表对于每个目标节点而言，其在初始化时选择各相邻节点的概率相等，蚂蚁行进中总是根据信息素表中记录的概率值随机选择一条路径。算法中概率增大的计算公式为

$$p = \frac{p_{\text{old}} + \Delta p}{1 + \Delta p} \quad (7.3.20)$$

而概率减小的计算公式为

$$p = \frac{p_{\text{old}}}{1 + \Delta p} \quad (7.3.21)$$

式中， $0 \leq \Delta p \leq 1$ ，取值与蚂蚁经历的时延成反比，而蚂蚁经历的时延又与其所经链路的拥塞状态有关。当一条链路拥塞时，蚂蚁经历的时延增大，选择这条链路的概率变小。为了不使开销太大，各个节点发送蚂蚁的频率都不高，这种依靠定期发送的蚂蚁来响应链路的拥塞情况，显然太慢，无法迅速缓解链路的拥塞。

#### 2. 基于蚁群算法的拥塞规避路由算法

该算法的目标是加速对符合约束条件的最优路径的探索和对链路拥塞的反应。前一个目标是基于蚂蚁的泛滥技术以及一个相应设计的概率更新计算公式来实现的；后一个目标则依赖于前一目标实现的效果，新路径探索得越快，也就越容易分散流量，从而实现了对拥塞的规避。

##### (1) 蚂蚁泛滥。

要探索一个网络中两节点之间的最优路径，一个方法是同时对两节点之间的各条路径进行比较。采用蚂蚁泛滥，一个源节点发送蚂蚁时，同时向其相邻节点发送蚂蚁。相邻节点收到蚂蚁后将在所有的链路上复制蚂蚁，但收到该蚂蚁的链路除外。为了防止蚂蚁总数无限制增长，一个源节点发送的蚂蚁以及由它复制得

到的所有蚂蚁都有一个相同的标识号，称这些具有相同标识号的蚂蚁为同一批蚂蚁。每个节点第一次收到一批蚂蚁中的一只时，将记录其标识号，并在所有链路上复制发送蚂蚁。当该节点再次接收到同一批蚂蚁时，将不再对其进行复制。通过蚂蚁泛滥，每个节点都能够收到蚂蚁，并且到达同一节点的多只蚂蚁必然是沿不同路径到达的。

### (2) 信息素表中概率值的更新计算。

针对蚂蚁的泛滥机制，这里提出了一个新的概率更新计算公式。该公式一方面能够体现所使用的寻路准则（最短距离路径和最大带宽路径），另一方面能够使源自同一节点的多只蚂蚁沿不同路径到达同一节点时，选择最优路径的概率值增长得最多或者减少得最少，从而使最优路径能够尽快凸现出来。

信息素表中增大概率的计算公式为

$$p_{n+1}^i = \begin{cases} \frac{\Delta p}{1 + \Delta p} p_n^i, & \text{若 } p_n^i \leq 1 - \Delta p \\ \frac{1 - \Delta p}{\Delta p} (p_n^i - 1) + 1, & \text{否则} \end{cases} \quad (7.3.22)$$

减小概率的计算公式为

$$p_{n+1}^i = \begin{cases} \frac{1 - \Delta p}{\Delta p} p_n^i, & \text{若 } p_n^i \leq \Delta p \\ \frac{\Delta p}{1 - \Delta p} (p_n^i - 1) + 1, & \text{否则} \end{cases} \quad (7.3.23)$$

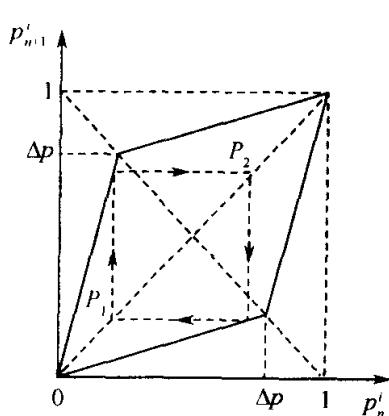


图 7.10 概率计算公式曲线图

式中， $p_n^i$  和  $p_{n+1}^i$  分别表示更新前后选择节点  $i$  的概率； $\Delta p$  为一参数， $0.5 \leq \Delta p < 1$ ，其取值取决于所要求的寻路准则以及到达蚂蚁所携带的信息。公式 (7.3.22) 和公式 (7.3.23) 所对应的曲线如图 7.10 所示。

由图 7.10 可见，概率值变化的幅度与  $\Delta p$  的大小紧密相关。 $\Delta p$  越大，概率值变化的幅度也越大；因此，可以用  $\Delta p$  的取值来体现所要求的寻路准则。当寻路准则是寻找最短路径，或者是寻找最大带宽路径时，可使用下述公式

$$p \propto \left[ \frac{1}{f(h_{op})} \right]^\alpha \left[ \frac{B_{path}}{B_{max}} \right]^\beta \quad (7.3.24)$$

式中， $h_{op}$  表示蚂蚁从源节点到当前节点所经历的跳数， $f(h_{op})$  则为  $h_{op}$  的一个增函数； $B_{path}$  表示蚂蚁所经历的所有链路的最小带宽； $B_{max}$  表示当前节点的所有相邻链路的最大带宽； $\alpha$ 、 $\beta$  是两个系数，分别表示距离和带宽对  $\Delta p$  的相对重要性。

概率更新计算公式 (7.3.22) 和公式 (7.3.23) 的曲线相对于直线  $p_n^i = p_{n+1}^i$  是轴对称的 (如图 7.10 所示), 这可以保证当有源自同一节点的多只蚂蚁沿不同路径到达同一节点时, 选择最优路径的概率值总是增长得最多或减少得最少。

图 7.11 给出了两只蚂蚁从节点 3 出发到达节点 1 的网络结构。

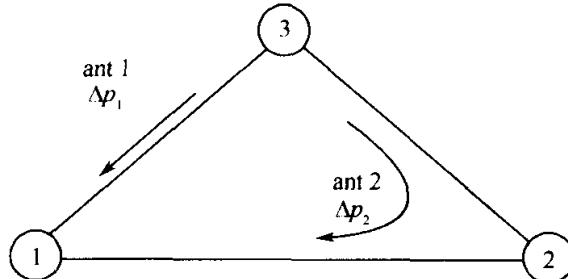


图 7.11 从节点 3 出发的两只蚂蚁到达节点 1

由图 7.11 可见, 由节点 3 发出的两只蚂蚁沿不同的路径到达节点 1, 因为路径 3-1 比路径 3-2-1 短, 故  $\Delta p_1 > \Delta p_2$ 。观察节点 1 处到达节点 3 选择路径 1-3 的概率  $P_{33}$  的变化过程, 可发现采用公式 (7.3.22) 和公式 (7.3.23), 不管两只蚂蚁到达的顺序以及到达前  $P_{33}$  的取值如何, 最终  $P_{33}$  的取值都是增大的 (如图 7.12(a) 所示); 而采用公式 (7.3.20) 和公式 (7.3.21), 最终  $P_{33}$  的取值则与蚂蚁到达的顺序以及到达前  $P_{33}$  的取值有关。图 7.12(b) 表示的一种情况就是  $P_{33}$  变小了, 与期望的结果不符。采用蚂蚁泛滥机制以及新的概率计算公式, 每个节点只要泛滥一次蚂蚁, 其他节点就能计算并获得其最优路径。

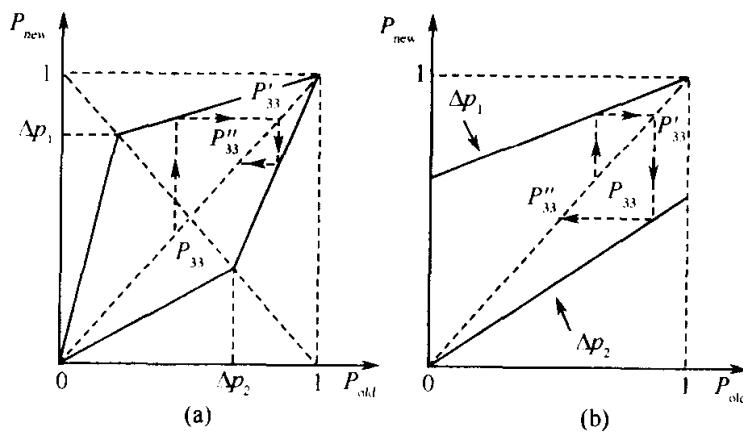


图 7.12 在不同的计算公式下  $P_{33}$  的变化过程

### (3) 拥塞规避。

对链路拥塞状态的判断, 采用与 RED 相同的方法<sup>[46]</sup>, 即计算队列的平均长度, 并与上限和下限阈值比较。当一个拥塞节点预测到一条相邻链路即将发生拥

塞，就立即向所有路径经过该拥塞链路的目标节点发送拥塞通告信息，要求探索新的路径。目标节点接收到信息后，立即向拥塞节点发送拥塞应答蚂蚁。拥塞应答蚂蚁以探测蚂蚁一样的方式更新所经节点的信息素表。但是，拥塞应答蚂蚁将避开所有的拥塞链路。当拥塞应答蚂蚁到达拥塞节点，表明存在一条新的非拥塞的路径。拥塞节点将立即切换到这条新的路径，实现流量的分散，减缓拥塞状态。当拥塞节点由拥塞状态变为空闲，再把路由切换回原来的路径。

### 7.3.2.2 算例分析

一般来说，可从以下两个方面评价路由算法：

- (1) 端到端的平均时延。
- (2) 网络丢包率，就是在网络中丢弃的包数占进入网络的总包数的比例。数据包在网络中被丢弃，主要是由网络拥塞引起的。

在 NS 仿真平台上，对基于蚁群算法的拥塞规避路由算法进行了仿真，仿真所用的网络结构如图 7.13 所示。

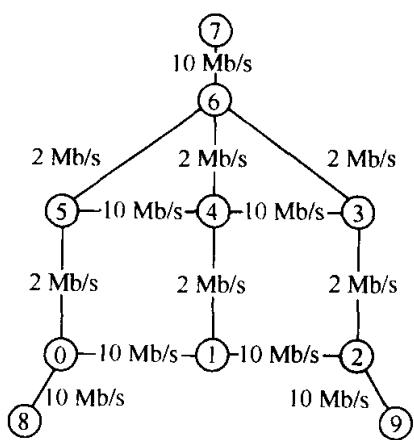


图 7.13 仿真网络结构图

由图 7.13 可见，链路的传输时延都是 10ms。在节点 8 和 9 处各有一个数据源，目的地址是节点 7。所使用的数据源是负指数分布开关数据源，在数据发送期间，将以恒定速率  $R$  发送 UDP 包。通过不断加大数据源的发送速率  $R$ ，使网络负载加大并发生拥塞。比较在不同的网络负载下链路状态路由算法和基于蚁群算法的拥塞规避路由算法（记为 LB）的性能。网络负载的定义如下

$$\text{网络负载} = \frac{\text{数据流的平均到达速率 } \lambda}{\text{网络的平均服务速率 } \mu} \quad (7.3.25)$$

在仿真中，数据流的平均到达速率为两个数据源的平均速率之和，网络的平均服务速率为从节点 8 和 9 到节点 7 的网络带宽 6Mb/s。

仿真时，节点 8 和 9 处的数据源的发送速率  $R$  一样，取值从 2Mb/s 开始，以 500Kb/s 为步长渐增，并且数据源的发送和停止的持续时间的平均值都是 100ms。各条链路判断拥塞的上限阈值为 0.6，下限阈值为 0.3。仿真结果如图 7.14 和图 7.15 所示。

由图 7.14 可见，基于蚁群算法的拥塞规避路由算法由于具有拥塞规避机制，使得网络丢包率大大降低。在数据包的平均时延方面，一方面使得节点 8-7 之间的平均时延减小了，另一方面使节点 9-7 之间的平均时延增大了。但两路数据流的平均时延更加接近了，所有数据包经历的时延更趋于一致（见图 7.15）。为了

更清楚地看到这一点，对网络负载为 0.75 情况下的数据包时延做了统计。

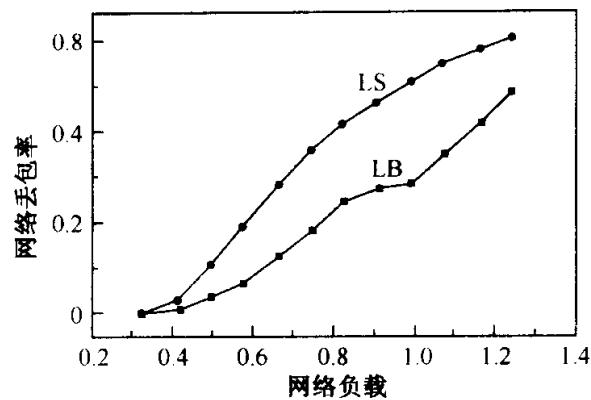


图 7.14 在不同的网络负载下，两种路由算法的网络丢包率

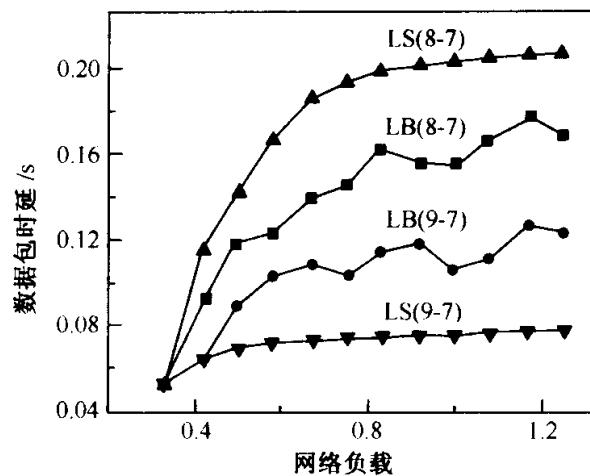


图 7.15 在不同的网络负载下，节点 8-7 以及 9-7 之间的数据包的平均传输时延

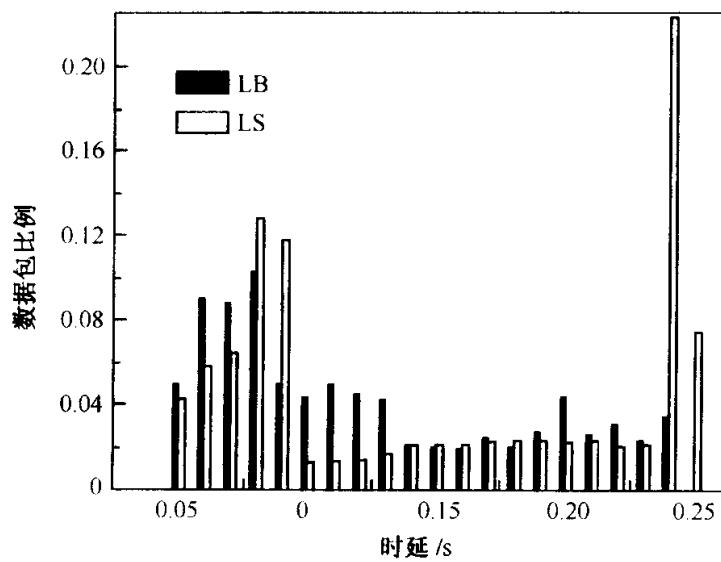


图 7.16 网络负载为 0.75 时，数据包传输时延的直方图

图 7.16 给出了在不同路由算法下，所有数据包时延的概率分布特性。采用 LB 算法，所有数据包的平均时延为 0.1171s；采用 LS 算法，所有数据包的平均时延为 0.1526s。

由图 7.16 可见，LB 算法有更多的数据包的时延分布在均值的附近范围内，而 LS 算法大部分数据包的时延都分布在远离均值的两端，即 LB 算法的时延方差比 LS 算法的小。若以不超过某个时延上限的数据包所占比率来看，LB 算法也优于 LS 算法。

## 7.4 车辆路径问题

在物流配送供应领域中，一个常见问题是：已知有一批客户，各客户点的位置坐标和货物需求已知，供应商具有若干可供派送的车辆，运载能力给定，每辆车都从起点出发，完成若干客户点的运送任务后再回到起点。现要求以最少的车辆数、最小的车辆总行程来完成货物的派送任务，该问题被称为车辆路径问题 (vehicle routing problem, VRP)，有时也称之为“车辆计划”、“货车派遣”等。

从本质上说，TSP 是 VRP 的基本问题，二者既相似又存在许多差别，因此在设计求解 VRP 的蚁群算法时，既要注意吸收用于求解 TSP 的蚁群算法的经验，又要充分考虑 VRP 的具体要求。而 VRP 的复杂程度远高于 TSP，应用蚁群算法求解 VRP 与 TSP 的不同之处主要体现在如下三个方面。

(1) 子路径构造过程的区别。

在 TSP 中，每只蚂蚁均要经过所有节点；而在 VRP 中，每只蚂蚁并不需要遍历所有节点。因此，在求解 VRP 的每次迭代中，每只蚂蚁移动次数是不确定的，只能将是否已回到原点作为路径构造完成的标志。

(2)  $\text{allowed}_k$  的区别。

$\text{allowed}_k$  的确定是蚂蚁构造路径过程中一个十分关键的问题。在 TSP 中，蚂蚁转移时只需考虑路径距离和信息量；而在 VRP 中，蚂蚁转移时不但要考虑上述因素，还需考虑车辆容量的限制。

(3) 可行解结构的区别。

在 TSP 中，每只蚂蚁所构造出来的路径均是一个可行解；而在 VRP 中，每只蚂蚁所构造的回路仅是可行解的“零部件”，各蚂蚁所构造的回路可能能够组合成一些可行解，也可能一个可行解都得不到。因此，研究如何设计算法以尽量避免无可行解现象的出现以及如何获取可行解是蚁群算法在 VRP 领域中应用的关键。

近几年来，许多学者利用蚁群算法对各种 VRP 其进行了大量的研究，设计了各种类型的蚁群算法<sup>[47~65, 69, 70]</sup>。本节首先研究了 VRP 的自适应蚁群算法实现方案，然后针对一种有时间窗车辆路径问题 (vehicle routing problem with

time window, VRPTW)<sup>[62]</sup>, 设计了一种改进的最大-最小蚂蚁系统 (MMAS)。

### 7.4.1 自适应蚁群算法与车辆路径问题

#### 7.4.1.1 算法设计

用于求解 VRP 的自适应蚁群算法的设计思路是：以求解 TSP 的蚁群算法为基础，充分考虑 VRP 的具体要求，对算法的选择机制、更新机制以及协调机制做进一步改进，引入自适应的转移策略和信息素更新策略，并融合节约法，以克服基本蚁群算法计算时间长、易出现停滞等缺陷。

##### 1. 转移规则

借鉴 Dorigo M 的 Ant-Q 算法思想，采用确定性选择和随机性选择相结合的选择策略。

##### 2. 信息素更新规则

**定义 7.4.1 蚂蚁吸引力** 设经过节点  $i$  的蚂蚁数目为  $R$ ，经过有向弧段  $(i, j)$  的蚂蚁数目为  $r$ ，则称  $r/R$  为有向弧段  $(i, j)$  的蚂蚁吸引力。

在进行信息素局部更新时，若每次施放的信息量  $Q$  为常量，则有向弧段  $(i, j)$  上的蚂蚁吸引力越大，经过弧段  $(i, j)$  的蚂蚁数目就越多，从而局部更新的次数也就越多。久而久之，会导致弧段之间的信息量差距过大，限制了算法搜索的全局性。因此，随着算法搜索状态的变化， $Q$  值应不断调整。其调整原则是，路径的蚂蚁吸引力越大，则  $Q(t)$  越小。不妨设  $Q(t)=Q(1-r/R)$ ，并且若  $R=0$ ，则  $Q(t)=Q$ 。

假设第  $k$  只蚂蚁在第  $q$  次迭代中的第  $f$  次转移时经过有向弧段  $(i, j)$ ，在此之前共有  $R_k$  只蚂蚁经过  $i$  点，其中有  $r_k$  只蚂蚁选择了有向弧段  $(i, j)$ 。算法的全局更新规则与求解 TSP 时类似，而局部更新规则设计如下

$$\tau_{ij}^{new} = (1 - \rho)\tau_{ij}^{old} + \Delta\tau_{ij}, \quad \forall i, j, \text{且 } i \neq j \quad (7.4.1)$$

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k \quad (7.4.2)$$

$$\Delta\tau_{ij}^k = \begin{cases} Q_1 \left(1 - \frac{r_k}{R_k}\right), & \text{若蚂蚁 } k \text{ 从 } i \text{ 移动到 } j \\ 0, & \text{否则} \end{cases} \quad (7.4.3)$$

##### 3. 可行解问题的研究

在 VRP 中，每只蚂蚁所构造的回路只是可行解的一个组成部分，各蚂蚁所

构造的回路可能能够组成一些可行解，但也可能一个可行解都得不到。若经常得不到可行解，则会造成大量的计算时间浪费。这里可采取以下策略。

(1) 大蚂蚁数策略：增加算法的蚂蚁数目  $M$ ，扩大组合范围，从而增加可行解产生的可能性。

(2) 蚂蚁初始分布均匀策略：在每次迭代前，将蚂蚁随机均匀分布于各个节点，从而增加发现可行解的机会。

(3) 近似解可行化策略：前两种策略的目的都是为了提高各蚂蚁所构造的子回路组合成可行解的可能性，是一些针对无可行解的“事前”预防性措施。然而，当问题规模较大时，会有大量的蚂蚁所找到的回路是相同的，使得实际上真正起作用的蚂蚁数目大大减少，从而严重影响产生可行解的可能性。针对这一问题，这里提出了近似解可行化策略，即在找不到可行解的情况下，选择一些与可行解接近的非可行解作为问题的近似解，再按照某些规则将近似解转变成可行解的过程。近似解可行化策略本质上是一种“事后”性质的治理措施。显然，采取这种措施后，一般可确保至少得到所求问题的一个可行解。

#### 4. 近似解的获取

记  $CNT(tabu_k)$  为  $tabu_k$  中的节点数，如若  $tabu_1 = \{0, 1, 3, 0\}$ ，则  $CNT(tabu_1) = 4$ 。

**定义 7.4.2 子回路互异** 若  $\forall v_i \neq \forall v_j$ ,  $v_i \in tabu_k$ ,  $v_j \in tabu_{k'}$ ,  $i \neq 0$ ,  $j \neq 0$ ,  $k \neq k'$ ，则称子回路  $tabu_k$  和  $tabu_{k'}$  互异，记为  $tabu_k \cap tabu_{k'} = \emptyset$ ，意即两个子回路除 0 点外，无其他相同节点。

**命题 7.4.1** 任意数量的子回路所构成的集合（记为  $S'$ ）必属于可行解、未满非可行解、溢出非可行解以及混合非可行解等四种情形之一。

**定义 7.4.3 可行解** 所谓可行解，指的是这样一种集合，该集合的元素是两两互异的子回路，并包括所有节点，即

$$S_1 = \left\{ tabu_r \mid r = 1, 2, \dots, R, \forall tabu_r \cap \forall tabu_{r'} = \emptyset, \sum_{r=1}^R CNT(tabu_r) = N + 2R - 1 \right\} \quad (7.4.4)$$

**定义 7.4.4 未满非可行解** 未满非可行解由两两互异的子回路组成，并存在一个或多个节点未在该解的路径中。

$$S_2 = \left\{ tabu_r \mid r = 1, 2, \dots, R, \forall tabu_r \cap \forall tabu_{r'} = \emptyset, r \neq r', \sum_{r=1}^R CNT(tabu_r) = N + 2R - 1 \right\} \quad (7.4.5)$$

记  $L_g(S_2)$  为  $S_2$  中已包含的节点数。显然， $L_g(S_2)$  越大，未满非可行解越接近可行解。设  $\bar{S}_2$  为未包含的节点集，则显然有  $L_g(S_2) + L_g(\bar{S}_2) = N$ 。将那些在所有未满非可行解中具有最大  $L_g(S_2)$  的未满非可行解称为最大未满非可行解。

**定义 7.4.5 溢出非可行解** 溢出非可行解虽然包括了所有节点，但它存在某些子回路不互异，即存在除 0 点外的公共节点。

$$S_3 = \left\{ \text{tabu}_r \mid r = 1, 2, \dots, R, \exists \text{tabu}_r \cap \text{tabu}_{r'} = \emptyset, \right. \\ \left. r \neq r', \sum_{r=1}^R \text{CNT}(\text{tabu}_r) = N + 2R - 1 \right\} \quad (7.4.6)$$

记  $L_p(S_3)$  为溢出非可行解  $S_3$  中的公共节点数。公共节点数的计算方法为：比较任意两个不互异的子回路，计算其除 0 点外的相同节点数量，公共节点数为所有这些节点数量的累加，进而有

$$\sum_{r=1}^R \text{CNT}(\text{tabu}_r) = N + 2R - 1 + L_p(S_3) \quad (7.4.7)$$

因此  $L_p(S_3)$  越小，溢出非可行解越接近可行解。显然，当  $L_p(S_3) = 0$  时，溢出非可行解就变为可行解了。将那些在所有溢出非可行解中具有最小  $L_p(S_3)$  的溢出非可行解称为最小溢出非可行解。

**定义 7.4.6 混合非可行解** 混合非可行解是指未包括所有节点，并存在某些子回路不互异的解，即

$$S_4 = \left\{ \text{tabu}_r \mid r = 1, 2, \dots, R, \exists \text{tabu}_r \cap \text{tabu}_{r'} = \emptyset, \right. \\ \left. \sum_{r=1}^R \text{CNT}(\text{tabu}_r) = N + 2R - 1 \right\} \quad (7.4.8)$$

**定义 7.4.7 近似解** 将所有最大未满非可行解和最小溢出非可行解称为近似解。按该定义，虽然当算法不存在可行解时，从理论上并不能保证一定存在近似解，但实践中同时不存在溢出非可行解和未满非可行解情况的可能性极小，因此几乎不对近似解的获取产生影响。

## 5. 近似解的可行化

在获得近似解以后，接下来的问题就是如何将近似解转变为可行解。对于最大未满非可行解，可行化策略就是采用某种方法将未在路径中的点插入到现有路径中。这里采用了节约算法<sup>[63]</sup>，其基本思想是首先将各点单独与源点相连，构成一条仅含一个点的路径；然后按照下式计算将点  $i$  与  $j$  在满足车辆容量约束的基础上连接在同一路径上的费用节约值

$$s(i, j) = s(j, i) = c_{i0} + c_{j0} - c_{ij} \quad (7.4.9)$$

由上式可见,  $s(i, j)$ 越大, 说明将  $i$  与  $j$  连接在一起时总路程减少越多。设计近似解可行化的节约法步骤如下:

(1) 将  $\bar{S}_2$  集中的节点单独与 0 点相连, 构成一些仅含一个点的路径, 并将它们添加到  $S_2$  中, 然后计算  $s(i, j)$ , 令

$$\Phi = \{s(i, j) \mid v_i \in \bar{S}_2, \forall j, s(i, j) > 0\} \quad (7.4.10)$$

(2) 在  $M$  内按  $s(i, j)$  从大到小的顺序排列。

(3) 若  $M = \Phi$ , 则算法终止, 此时有  $\bar{S}_2 = \Phi$ ; 否则对第一项  $s(i, j)$ , 考察对应的  $(i, j)$ , 若点  $j$  不在已构成的路径上, 即  $v_j \notin S_2$ , 或点  $j$  在已构成的路径上, 即  $v_j \in S_2$ , 但不是路径的内点, 跳转到第 (4) 步, 否则跳转到第 (7) 步。

(4) 考察点  $i$  与  $j$  连接后的路径上总货物量  $Q$ , 若  $Q \leq q$ , 跳转到第 (5) 步, 否则跳转到第 (7) 步。

(5) 连接点  $i$  与  $j$ , 修改相应的  $\text{tabu}_k$  以及  $S_2$ , 跳转到第 (6) 步。

(6) 从  $M$  中去掉所有以  $i$  为起始点的  $s(i, j)$ , 且若  $v_j \in \bar{S}_2$ , 则从  $M$  中去掉所有以  $j$  为起始点的  $s(j, k)$ , 跳转到第 (3) 步。

(7) 令  $M = M - s(i, j)$ , 跳转到第 (3) 步。

对于最小溢出非可行解, 也可采用节约法将其转变为可行解。只是在应用该方法之前, 需对其公共节点进行“擦除”处理, 即先去掉所有路径中的公共节点, 将溢出非可行解转换成未满非可行解, 再应用节约法将公共节点插入到路径中。

## 6. 算法步骤

- (1) 初始化各参数, 输入基础数据, 计数器  $N_c = 0$ 。
- (2) 将  $M$  只蚂蚁随机均匀地放到  $N$  个节点上, 得到节点  $i$  的蚂蚁集  $S_i$  和蚂蚁数  $b_i$ , 初始化  $\text{tabu}_k$  以及  $\text{allowed}_{k,l} = 0$  (已完成任务蚂蚁数); 初始点为 0 的蚂蚁的过程变量  $\text{Pro}[k] = 1$ , 初始点为非 0 点的蚂蚁的过程变量  $\text{Pro}[k] = 2$ 。
- (3) 在节点  $i$  取蚂蚁  $k$ , 判断其初始节点。若为 0 点, 则按转移规则确定节点  $j$ , 更新  $\text{tabu}_k$ ,  $S_i$ ,  $b_i$ 。且若  $\text{Pro}[k] = 1$ , 则  $\text{Pro}[k] = 2$ , 跳转到第 (4) 步; 而当  $\text{Pro}[k] = 2$  时, 若  $j$  点为初始节点, 则  $l++$ , 并跳转到第 (3) 步, 否则跳转到第 (4) 步。若为非 0 点, 则按转移规则确定转移节点  $j$ , 更新  $\text{tabu}_k$ ,  $S_i$ ,  $b_i$ 。并且当  $\text{Pro}[k] = 1$  时, 若  $j$  点为 0, 则  $\text{Pro}[k] = 2$ , 并跳转到第 (3) 步; 当  $\text{Pro}[k] = 2$  时, 若  $j$  点为初始节点, 则  $l++$ , 并跳转到第 (3) 步, 否则跳转到第 (4) 步。
- (4) 更新  $S_i$ , 重复第 (3) 步和第 (4) 步, 直到  $b_i = 0$ 。
- (5) 在所有蚂蚁都移动一次后, 按局部更新规则进行信息素的更新。
- (6) 更新所有节点的  $b_i$ , 若所有节点上蚂蚁数量  $b_i = 0$  或  $l = M$ , 跳转到第 (7) 步, 否则跳转到第 (3) 步。

(7) 由  $\text{tabu}_k$  生成路径集  $L = \{L_1, L_2, \dots, L_M\}$ , 寻找可行解, 得到可行解集  $A = \{A_1, A_2, \dots, A_p\}$ ; 若未发现可行解, 则采取近似解可行化策略, 并跳转到第(8)步。

(8) 计算本次搜索到的最优路径  $L^*(q)$ , 并得到迄今为止的最优路径,  $L^* = \min\{L^*(q), L^*\}$ , 按全局更新规则进行信息素更新。

(9) 若  $N_c \geq N_{c_{\max}}$ , 则算法终止, 并输出  $L^*$ ; 否则,  $N_c + +$ , 并跳转到第(2)步。

#### 7.4.1.2 算例分析

设有 19 个客户随机分布于弧段长为 10km 的正方形区域内。配送中心位于区域正中央, 其坐标为 (0, 0)。各客户的需求由计算机随机产生, 车辆载重量为 9t, 实验基础数据如表 7.12 所示。

表 7.12 实验基础数据

客户编号	0	1	2	3	4	5	6	7	8	9
横坐标(km)	0	0	0	-2	-3	3	-4	-4	1	1
纵坐标(km)	0	-1	3	-2	-3	-1	0	-1	-2	-1
配送量(t)	0	1.5	1.8	2.0	0.8	1.5	1.0	2.5	3.0	1.7
客户编号	10	11	12	13	14	15	16	17	18	19
横坐标(km)	1	3	-3	2	1	2	2	1	-3	-1
纵坐标(km)	3	4	0	0	-3	-1	1	-4	2	-1
配送量(t)	0.6	0.2	2.4	1.9	2.0	0.7	0.5	2.2	3.1	0.1

设置  $M=60$ ,  $N_{c_{\max}}=50$ ,  $\tau_{ij}=10$ ,  $\alpha=1$ ,  $\beta=1$ ,  $\rho=0.15$ ,  $Q_1=10$ ,  $Q_2=50$ ,  $Q_3=100$ 。运行 10 次, 其结果如表 7.13 所示。

表 7.13 实验计算结果

计算次序	1	2	3	4	5	6	7	8	9	10	平均
最小配送距离	47.37	41.86	43.1	41.99	41.86	43.21	41.94	43.1	44.32	42.58	42.73
使用车辆数目	4	4	4	4	4	4	4	4	4	4	4
首次搜索终解代数	26	39	26	32	35	28	41	37	19	39	32.3
计算时间	1.99	2.2	1.99	2.14	2.15	2.07	2.23	2.16	1.3	2.19	2.14
与最小的差值	1.51	0	1.24	0.13	0	1.35	0.08	1.24	2.46	0.72	0.87

由表 7.13 可见, 采用改进后的蚁群算法在 10 次求解中都可得到很高的求解

质量，所找到的最好解为 41.86km，对应的四条配送路径分别为

路线 1: 0→18→0；

路线 2: 0→1→17→14→8→0；

路线 3: 0→12→6→7→4→3→19→0；

路线 4: 0→2→10→11→16→7→5→15→9→0。

改进后蚁群算法的计算结果也很稳定，10 次求解中，最差解的配送里程只比最好解多 5.8%，且具有较高的计算效率，有效地解决了计算时间长的问题。

这里以第 2 次运行过程为例，得到改进后蚁群算法的进化过程如图 7.17 所示。

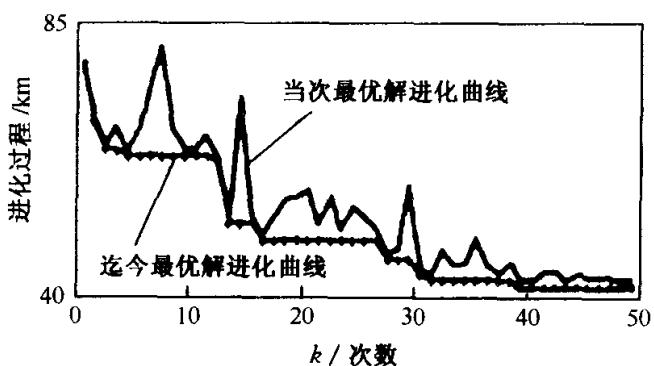


图 7.17 改进后蚁群算法的进化曲线

由图 7.17 可见，对于 VRP，利用改进蚁群算法可取得非常好的优化结果，且计算结果较为稳定。

## 7.4.2 最大-最小蚁群算法与有时间窗车辆路径问题

### 7.4.2.1 问题描述

VRPTW 可概述如下：有  $n$  个货物需求点（或称顾客），已知每个需求点的需求量及位置，用多辆汽车从中心仓库（或配送中心）到达这批需求点。要求必须在其时间窗口内为每个客户服务，如果汽车提前到了客户所在地，也必须等待，直到允许为该客户服务为止。每辆汽车载重量一定，每条路线不超过汽车载重量。每个需求点的需求必须且只能由一辆汽车来提供，目标是最小化总的汽车行驶距离和所需的汽车数目。

这里将最小化汽车总的行驶路程作为第一目标，而将最小化车辆数目作为第二目标。

### 7.4.2.2 算法设计

MMAS 与基本蚁群算法的主要区别在于通过将每条轨迹上的信息量限制在  $[\tau_{\min}, \tau_{\max}]$  之间<sup>[66~68]</sup>，较好地避免了搜索面的局部停滞（即早熟现象）。显而易见， $\tau_{\max}$  与  $\tau_{\min}$  值的设定是至关重要的。因为每次迭代路径上增加的最大信息素为  $1/L(s^{gb})$ ，其中  $L(s^{gb})$  为对应全局最好解的路径长度，所以每当更新最好解时，需要同时更新  $\tau_{\max}$  与  $\tau_{\min}$ 。 $\tau_{\max}$  与信息素挥发因子  $\rho$  及  $L(s^{gb})$  成反比，而与“精英蚂蚁”的数目成正比。这里可按照以下策略动态地确定  $\tau_{\max}(t)$  与  $\tau_{\min}(t)$ 。

(1) 在最初信息素还未得到更新时（即产生第一代解前），采用下式确定  $\tau_{\max}(t)$  和  $\tau_{\min}(t)$

$$\tau_{\max}(t) = \frac{1}{2(1-\rho)} \cdot \frac{1}{L(S^{gb})} \quad (7.4.11)$$

$$\tau_{\min}(t) = \frac{\tau_{\max}(t)}{20} \quad (7.4.12)$$

(2) 一旦信息素更新之后，即采用下式确定  $\tau_{\max}(t)$

$$\tau_{\max}(t) = \frac{1}{2(1-\rho)} \cdot \frac{1}{L(S^{gb})} + \frac{\sigma}{L(S^{gb})} \quad (7.4.13)$$

式中， $\sigma$  表示“精英蚂蚁”的数目。 $\tau_{\min}(t)$  的确定与公式 (7.4.11) 相同。

#### 1. MMAS 中路径的构造

每只蚂蚁选择下一城市时，在满足车辆容量和时间窗约束的前提下，需要考虑如下两方面因素<sup>[65]</sup>：

(1) 通往下一城市的路径长度以及路径上的信息量。

(2) 时间窗因素的择优性，由下一客户  $j$  的时间窗宽度和所在客户  $i$  到达下一客户  $j$  的时间等因素决定，该择优性的优先原则为需等待时间较短优先原则和时间窗较小优先原则。

综合上述两方面因素，第  $k$  条路径上的蚂蚁在城市  $v_i$  选择城市  $v_j$  的概率可由下式决定

$$P_{ij}^k = \begin{cases} \bar{\omega}_1 \frac{(\tau_{ij})^\alpha (\eta_{ij})^\alpha}{\sum_{h \in \Omega} (\tau_{ih})^\alpha (\eta_{ih})^\alpha} + \bar{\omega}_2 \frac{1/(|t_{ij} - a_j| + |t_{ij} - b_j|)}{\sum_{h \in \Omega} 1/(|t_{ih} - a_h| + |t_{ih} - b_h|)}, & \text{若 } v_j \in \Omega \\ 0, & \text{否则} \end{cases} \quad (7.4.14)$$

式中， $\Omega = \{v_j \mid v_j \text{ 为可被访问的城市}\} \cup \{v_0\}$ ， $v_0$  表示配送中心； $\bar{\omega}_1$  和  $\bar{\omega}_2$  表示权重系数，满足  $0 \leq \bar{\omega}_1, \bar{\omega}_2 \leq 1$ ，且  $\bar{\omega}_1 + \bar{\omega}_2 = 1$ ； $[a_j, b_j]$  表示客户  $j$  的时间窗； $t_{ij}$  表示由客户  $i$  到达客户  $j$  的时间（即开始为客户  $i$  服务的时刻 + 客户  $i$  所需的

服务时间 + 从客户  $i$  到  $j$  的路程消耗时间)。

## 2. 信息素的更新

信息素的更新有局部更新和全局更新两种方式。这里采用全局更新的方式，只将最好的蚂蚁用于信息素的更新，就是对在路径构造中排名前几个的“精英蚂蚁”进行信息素更新<sup>[69]</sup>。其更新规则如下

$$\tau_{ij}^{\text{new}} = (1 - \rho)\tau_{ij}^{\text{old}} + \sum_{u=1}^{\sigma-1} \Delta\tau_{ij}^u + \sigma\Delta\tau_{ij}^* \quad (7.4.15)$$

仅当轨迹  $(v_i, v_j)$  被排名第  $\mu$  的最好蚂蚁利用时，其信息素才增加  $\Delta\tau_{ij}^u$ ，而  $\Delta\tau_{ij}^u = (\sigma - \mu)/L_u$ ，其中， $L_u$  表示排名第  $\mu$  的路径长度。所有属于当前最好路径轨迹上的信息素都被加强  $\sigma\Delta\tau_{ij}^*$ ，其中， $\sigma\Delta\tau_{ij}^* = 1/L^*$ ， $L^*$  表示当前最好解路径的长度。

这种信息素更新方式收敛速度较慢，而且其全局优化性能也不明显。为了加快收敛速度，同时又不影响全局优化能力，在较短的时间内找到最优解，这里提出了一种改进的信息素更新策略。该策略保留全局的最好解，但为了扩大信息素更新的范围，“精英蚂蚁”取每次迭代结果中的前几位，这时的信息素更新仍然采用公式 (7.4.15)。在迭代过程中，对出现优于上代解的本代解给予激励，对劣于上代解的本代解给予惩罚，从而加快了其收敛速度。对更新过的路径所采取的激励与惩罚措施如下式所示

$$\tau_{ij}^{\text{new}} = \tau_{ij}^{\text{new}} + \tau_{ij}^{\text{new}} \times \frac{L_{\text{new}} - L_{\text{old}}}{L_{\text{old}}} \quad (7.4.16)$$

## 3. 局部优化

在蚁群算法中混入局部优化算法，对每代构造的解进行改进，可以进一步缩短解路线的长度，从而加快蚁群算法的收敛速度。这里只对每代最好解进行局部优化，局部优化作用时间为所有的蚂蚁已经构造完解，但信息素还未更新之前。这里采用 2-opt 的局部优化方法<sup>[70]</sup>。

## 4. 初始解的构造

在系统初始化时要确定  $\tau_{\min}$  和  $\tau_{\max}$  的值，而确定它们的值需要先明确  $L(s^{gb})$  的值，所以在最初就必须有一个较好的可行解。由于 VRPTW 的复杂性<sup>[71]</sup>，产生一个可行的初始解并不是一件容易的事情。这里采用一种基于客户时间窗下限较小优先的快速产生初始解的算法，其时间复杂度为  $O(n^2 \log_2 n)$ ，明显快于节约法  $O(n^4)$ 。该算法选择下一个客户的策略为，从当前路径的最后一个客户出发，到所有未访问过的客户中开始服务时间最早的那个客户。只有当开始服务时间超过这个客户的时间窗时，才需要开辟一条新路径，并从未访问过的客户中重

新选择出发点，将所有未访问过的客户中开始服务时间最早的那个客户作为新路径的第一个客户。算法的具体步骤描述如下：

- (1) 初始化。
- (2) 将所有未被访问过的客户放入集合 C 中。
- (3)  $C = \langle c_0, \dots, c_i, \dots, c_j, \dots, c \mid c \mid - 1 \rangle$ , C 中的元素按如下规则有序排列：对任意的  $i \leq j$ , 满足  $W(c_i, \text{当前路径 } R \text{ 的最后一个客户}) \leq W(c_j, \text{当前路径 } R \text{ 的最后一个客户})$ ; 令  $k=1$ 。
- (4) 如果集合 C 为空，则跳转到第 (7) 步。
- (5) 如果  $W(c_k, c_m) = T$ , 保存当前路径  $R$ ; 重新开辟一条路径, 从当前未被访问的客户中随机选择一个客户  $c_r$  作为新路径的出发点;  $C = C - c_r$ ; 跳转到第 (3) 步。
- (6) 如果客户  $c_k$  为当前路径  $R$  的合法客户, 将  $c_k$  加入到当前路径  $R$  中;  $C = C - c_r$ ;  $k = k + 1$ ; 跳转到第 (4) 步;
- (7) 结束并输出计算结果。

其中,  $W(c_i, c_m)$  的返回值为 “ $\max(c_m \text{ 的开始服务时间} + c_m \text{ 的服务所需时间} + \text{从 } c_m \text{ 到 } c_i \text{ 的行驶时间} a_{ci})$ ”。若到达  $c_i$  的时间晚于  $b_{ci}$ , 则  $W(c_i, c_m)$  返回一个很大的  $T$  值, 表示该客户不能加入路径。

#### 7.4.2.3 算例分析

将上述改进 MMAS 应用于 Marius M 等在文献 [72] 中所述的基准 VRPTW 实例, 这些实例共 56 个。这 56 个实例中的每个例子都含有 100 个客户和 1 个中心仓库, 并规定了车辆负载、客户的时间窗和车辆运行时间。实例分为 6 大类, 其中 R1 和 R2 中的客户为随机分布, C1 和 C2 中的客户有聚集趋势, RC1 和 RC2 的客户兼有 R 类和 C 类的特征。

实验开始, 在每个城市都放置一只蚂蚁, 蚂蚁的个数与客户的数目相同。设置  $\alpha = 1.0$ ,  $\beta = 5.0$ ; 信息素挥发系数  $\rho$  的取值在  $0.02 \sim 0.3$  之间;  $\sigma$  的取值一般在  $3 \sim 6$  之间较合适;  $\bar{w}_1$  和  $\bar{w}_2$  的设定需根据具体情况而定, 一般来说,  $0.5 < \bar{w}_1 < 1.0$ ,  $0 < \bar{w}_2 < 0.5$ ; 路径上信息素初始化为  $\tau_{\max}$ 。

在 MMAS 中, 一般将路径上的信息素初始化为上限值  $\tau_{\max}$ , 这样可使系统具有更好的全局搜索能力。为了说明这样做的好处, 这里将分别初始化为  $\tau_{\max}$  和  $\tau_{\min}$  的结果做了对比, 如图 7.18 所示。在图 7.18 中分别描述了这两种情况下对实例 RC101 所做实验中迭代次数与获得第一目标最好解的关系, 其中 DV 表示偏离最优目标已知最优解的百分比,  $N$  表示迭代次数。

由图 7.18 可见, 初始化为上限值虽收敛较慢, 但可获得更好的解。将未改进的 MMAS 与改进后的 MMAS 做了比较, 其比较结果如表 7.14 所示。

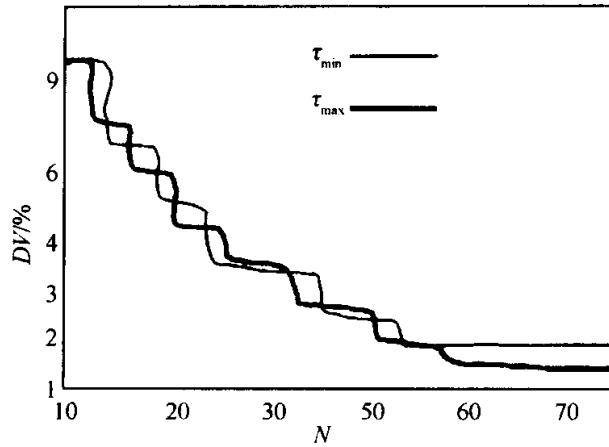


图 7.18 信息素初始化为  $\tau_{\max}$  和  $\tau_{\min}$  的结果对比

表 7.14 MMAS 获得的最好结果与历经的迭代次数

实例	未改进的 MMAS		改进后的 MMAS	
	DV (%)	N	DV (%)	N
RC201	2.4	64	0.7	51
RC203	2.3	57	1.3	48
RC205	2.3	69	1.1	63
RC207	2.6	55	1.0	53
C202	2.2	54	1.3	54
C204	1.9	67	1.0	62
C206	2.5	68	1.4	64
C208	2.3	58	0.9	57

由表 7.14 可见，改进后的 MMAS 具有更快的收敛速度和更好的计算结果。图 7.19 描述了在实例 RC102 中获得的第一目标值与 CPU 运行时间之间的关系。

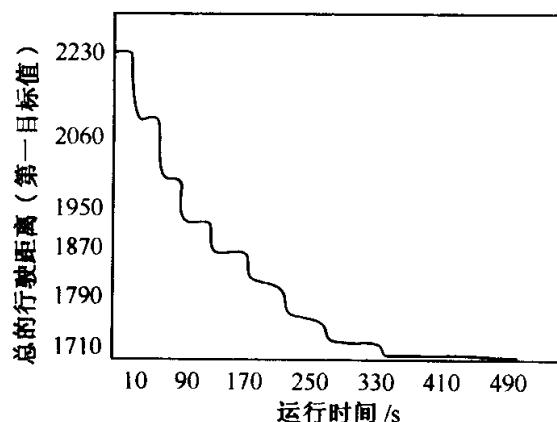


图 7.19 获得的第一目标值与 CPU 运行时间之间的关系

由图 7.19 可见，改进后的 MMAS 在运行初期就能使当前解得到很好的改善，并且收敛速度较快。实验结果与当前已知最优解的比较如表 7.15 所示，改进 MMAS 的平均值为对某一实例进行 10 次实验所得解的平均。表 7.15 中给出的解分为两部分，“/”之前为第一目标值，之后为第二目标值。

表 7.15 改进后 MMAS 的平均值及实际最优值与已知最优值的比较

实例	R101	R102	R103	R104	R105	R106
已知最优值	1650.80/19	1486.12/17	1292.68/13	1007.31/9	1377.11/14	1252.03/12
平均值	1701.58/18	1554.91/17	1338.57/13	1050.72/9	1412.14/13	1303.61/12
实际最优值	1660.71/18	1503.47/17	1307.74/13	1032.29/9	1374.03/13	1266.74/12
实例	R107	R108	R109	R110	R111	
已知最优值	1104.66/10	963.99/9	1194.73/11	1124.4/10	1096.72/10	982.14/9
平均值	1148.15/10	1002.17/9	1247.32/11	1161.25/10	1133.94/10	1025.88/9
实际最优值	1121.94/10	979.64/9	1217.09/11	1155.43/10	1120.57/10	1004.71/9

由表 7.15 可见，改进后 MMAS 的最好解已经很接近最优解，个别实例（如实例 R101 和实例 R105）的第二目标值还优于当前最好解，说明采用 MMAS 解决 VRPTW 是非常有效的。该系统还具有如下优点：

- (1) MMAS 可方便地解决其他 VRP，若没有时间窗的 VRP，只需将  $\omega_2$  设为 0 即可。
- (2) MMAS 可被用于多供货点 VRP (vehicle routing problem with multi-depot)。
- (3) 通过实时设定某条路径上信息素的上下限，可将其用于求解动态实时的 VRP。

## 7.5 机器人领域

机器人是一种具有高度灵活性的自动化机器，所不同的是这种机器具备一些与人或生物相似的智能能力，如感知能力、规划能力、动作能力和协同能力等。机器人技术作为 20 世纪人类最伟大的发明之一，至今已经历了 40 余年的发展历史。

机器人路径规划是指机器人按照某一性能指标搜索一条从起始状态到目标状态的最优或近似最优的无碰路径，它是实现机器人控制和导航的基础之一。一般可将机器人路径规划算法分为全局规划和局部规划两大类。多机器人系统是一个松散结构的分布式系统，其优点在于既可以独立工作，又可以在需要时进行协作。在任务未知的环境中，确定有哪些任务需要多个机器人协作完成是一个重要

而艰巨的问题。近年来，蚁群算法在机器人路径规划、多机器人协作、机器人控制等方面均取得了丰富的研究成果<sup>[73~87]</sup>。

限于篇幅，本节主要介绍了蚁群算法在机器人路径规划和多机器人协作这两个方面的典型应用。

### 7.5.1 蚁群算法与机器人路径规划

#### 7.5.1.1 算法设计

这里以复杂工作环境中的机器人（最优时间）路径规划问题为例<sup>[81]</sup>。为了利用蚁群算法有效地解决复杂工作环境中的机器人路径规划问题，本小节对基本蚁群算法做了如下改进。

##### 1. 定义了距离启发式信息概率

公式(7.5.1)决定了只依赖于距离信息时，编号为  $k$  的蚂蚁从当前点  $i$  向其周围某点的移动概率

$$\phi_{i,j}^k = \begin{cases} \frac{((\text{MaxDistance}_{A(i),e} - \text{Distance}_{j,e})\omega + \mu)^\lambda}{\sum\limits_{j \in \text{Available}(i)} ((\text{MaxDistance}_{A(i),e} - \text{Distance}_{j,e})\omega + \mu)^\lambda}, & \text{若 } j \in \text{Available}(i) \\ 0, & \text{否则} \end{cases} \quad (7.5.1)$$

式中， $\text{Available}(i)$  表示点  $i$  周围一个单位距离内非障碍区中点的集合，算法中的人工蚂蚁只能向前、后、左、右四个方向移动，因此，有  $0 < |\text{Available}(i)| \leq 4$ 。 $\text{Distance}_{j,e}$  表示从  $j$  点到终点  $e$  的距离，其值在算法执行前被预先计算出； $\text{MaxDistance}_{A(i),e}$  表示  $\text{Distance}_{j,e}$  中的最大值。因为  $\text{Available}(i)$  的各个  $\text{Distance}_{j,e}$  之间相差不到 2，所以需要对  $\text{Distance}_{j,e}$  重新定标，以体现它们之间的差别，否则启发式概率将变成一个随机函数，达不到应有的启发效果。这里引入三个标定系数  $\omega$ ， $\mu$  和  $\lambda$ 。根据多次实验，取  $\omega=10$ ， $\mu=2$ ， $\lambda=2$ 。

##### 2. 参数 $\alpha$ 和 $\beta$ 的确定

如前所述， $\alpha$  和  $\beta$  两个参数分别决定了信息量和启发式函数（能见度）的相对重要性。在基本蚁群算法中， $\alpha$  和  $\beta$  是常数；而在路径规划问题中，由于蚂蚁可经过的点太多（在一个  $400 \times 200$  位图表示的环境中有 80000 个点，而研究 TSP 一般达到几百规模就已经非常大了），很难确保每个点都获得信息素。因此在设计用于求解复杂工作环境中的机器人路径规划问题的蚁群算法时， $\alpha$  和  $\beta$  将随时间变化而做相应调整，即有

$$\alpha = \begin{cases} \frac{4t}{m}, & \text{若 } 0 \leq t < m \\ 0, & \text{若 } m \leq t \leq u \end{cases} \quad (7.5.2)$$

$$\beta = \begin{cases} \frac{3m - 1.5t}{m}, & \text{若 } 0 \leq t < m \\ 1.5, & \text{若 } m \leq t \leq u \end{cases} \quad (7.5.3)$$

式中,  $m$  为临界时刻, 在  $m$  时刻前, 由于各点上的信息量较少, 蚂蚁寻路过程中的主导因素为启发式因素, 这样能使更多的点获得信息素; 在  $m$  时刻后, 蚂蚁寻路过程中的主导因素变为信息素因素。从初始时刻 0 到临界时刻  $m$ ,  $\alpha$  值随时间线性递增,  $\beta$  值随时间线性递减; 从临界时刻  $m$  到终止时刻  $u$ ,  $\alpha$  和  $\beta$  值均为常数。

### 3. 对可行路径的修正处理

因为算法中的蚂蚁是在所设定的由像素点构成的位图环境中爬行, 当位图环境中的像素点过多(密)时, 蚂蚁在某段特定距离内的爬行路径可能变成曲线段, 从而人为地造成移动路径长度的增加。为避免这种情况的出现, 可人为地将蚂蚁在某段特定距离内弯曲的爬行路径“拉直”为一直线段, 从而减少移动路径的长度。

### 4. 距离启发式信息概率和转移概率的综合使用

从当前点到下一可行点的转移是由距离启发式信息概率和基于信息量的转移概率综合决定的, 而这里所采用的综合决定方法是基于比例选择策略的。通过比例选择, 可交替使用三种概率: 常规的基于信息量的状态转移概率, 公式 (7.5.1) 所示的距离启发式信息概率和综合考虑上述两种概率所产生的转移概率, 以决定下一个具体的可行移动点。

基于上述改进, 整个求解算法的具体步骤如下:

(1) 产生初始时刻的蚂蚁种群移动路径。

(2) 信息素的调整: 对所产生的每一条可行移动路径, 分别计算路径的长度和所对应信息素的增量, 再采用基本蚁群算法中的信息量更新公式, 对路径上各点所对应的信息素进行更新。

(3) 对产生的每一可行路径进行一定的修正处理: 即将蚂蚁所走的弯曲路径逐段拉直为一条由直线段连接的可行路径(即成为一折线)。将此可行路径与记录的目前最短路径进行比较, 如果路径长度更小, 则用该路径替换最短路径。对路径上所有点的信息素也根据第 (2) 步中的方法进行更新。如果当前时刻已达到预先设定的终止时刻, 则跳转到第 (5) 步。

(4) 下一时刻蚂蚁路径的产生: 综合使用当前点周围的距离启发式信息概率

和基于信息量的状态转移概率，产生由起点到终点的可行路径，并跳转到第(2)步。

(5) 算法结束：将当前路径作为最短路径输出。

### 7.5.1.2 仿真算例

下面给出一个仿真算例。设置  $W=10$ ,  $m=10$ ,  $\rho=0.7$ 。图 7.20 是一个给定的大小为  $400 \times 200$  像素的环境位图，并表示 10 只蚂蚁在第 10 个时刻累积的移动路径曲线。

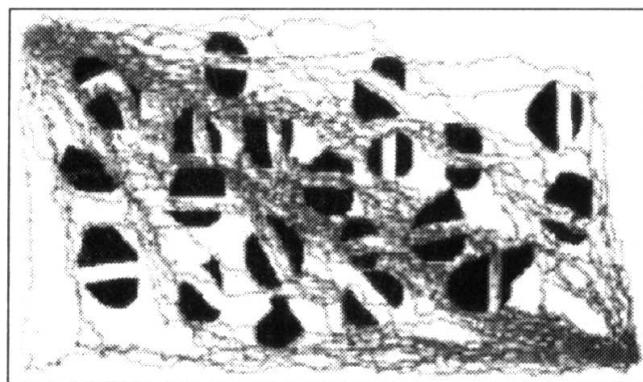


图 7.20 在第 10 个时刻累积的蚁群移动路径曲线

由图 7.20 可见，大多数蚂蚁所选择的路径都位于连接起点和终点线段两侧的一个限定范围内，这使得该区域内的信息量高于其他区域。图 7.21 是第 50 个时刻累积的蚁群移动路径曲线。

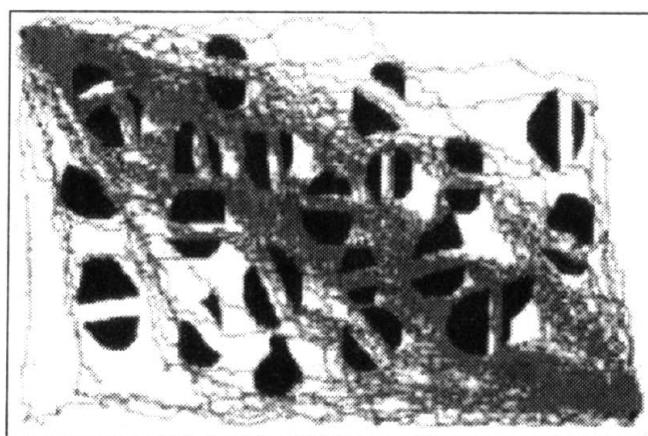


图 7.21 在第 50 个时刻累积的蚁群移动路径曲线

图 7.21 与图 7.20 相比，越来越多蚂蚁的移动路径落入了连接起点和终点线

段两侧的限定区域内，这导致其中所含的信息量渐渐高于其他区域。图 7.22 给出了在第 120 个时刻蚁群最终的收敛移动路径。

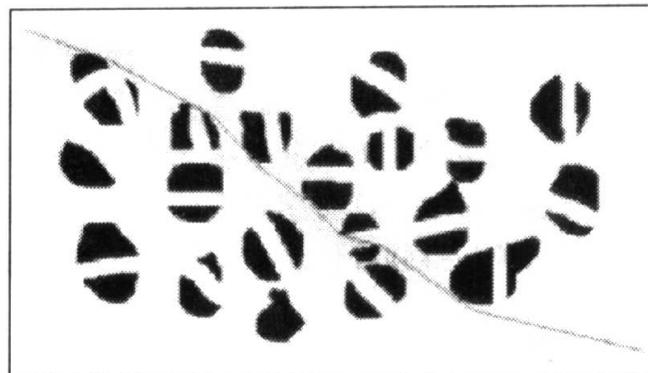


图 7.22 在第 120 个时刻蚁群最终的收敛移动路径曲线

由图 7.22 可见，所有蚂蚁经过的点所对应的信息量用灰度表示，而实线表示一条已被修正处理过的最短移动路径。图 7.23 表示最短移动路径长度的收敛曲线。

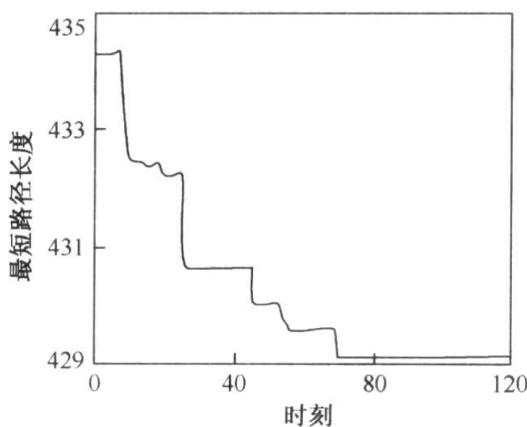


图 7.23 最短移动路径长度的收敛曲线图

由图 7.23 可见，通过使用所设计的蚁群算法，在第 70 个时刻就能得到全局收敛的路径解，而从第 70~120 个时刻没有得到更好的解，这表明算法的收敛效果趋于稳定。

## 7.5.2 蚁群算法与多机器人协作

### 7.5.2.1 算法设计

受到蚁群算法机理的启发，可将“外激励”的方法引入到多机器人系统中，

由机器人在工作过程中对不同的任务赋予不同的信息量，对其中难度较大的工作设置较大的信息量，以吸引其他机器人来进行协作。基于这一思想，这里设计了一种基于改进蚁群算法的多机器人协作策略<sup>[85, 86]</sup>。

多机器人系统中常用的局部通信方式有两种：一种是两个机器人之间进行单向或双向通信；另一种是采用“黑板原理”的通信方式，即系统中每个机器人都以一定的频率将一些和自己相关的简单信息写在“黑板”上一定的数据区域，同时从“黑板”上的某些数据区读取自己所感兴趣的信息。这里采用“黑板原理”的通信方式具有操作灵活、鲁棒性强等优点，不会由于局部的失误而影响全局的性能。

设  $n$  个机器人进入一包含  $m$  个任务的未知区域，它们开始进行独立的任务搜索。开始时刻对应于所有任务  $j$  ( $j = 1, 2, \dots, m$ ) 的信息量  $\tau_j$  为零。若机器人  $i$  ( $i = 1, 2, \dots, n$ ) 发现一个任务  $j$  时，它首先尝试着独自完成这一任务。若成功，机器人  $i$  得到一个奖励信号；若不成功，则将任务  $j$  的相关信息（如位置）写到“黑板”上，并将任务  $j$  对应的信息量  $\tau_j$  赋一大于零的值，于是有

$$\tau_j = \Delta\tau > 0 \quad (7.5.4)$$

同时，机器人  $i$  在任务  $j$  处等待一段时间。若机器人  $i$  没有发现任何任务，处于“空闲”状态（没有执行具体任务的状态）时，它每隔一段时间在“黑板”上的任务相关区进行搜索，如未发现与某个任务相关的信息量大于零，则继续在空间中搜索任务；若发现有  $k$  个任务的信息量  $\tau_s$  ( $s = 1, 2, \dots, k$ ) 大于零，则按照下式计算与之相关的概率

$$p_{is} = \frac{(\tau_s)^\alpha}{\sum_{s=1}^k (\tau_s)^\alpha}, \quad s = 1, 2, \dots, k \quad (7.5.5)$$

机器人在下一步以概率  $p_{is}$  选择任务  $s$

$$P(s = 0 \rightarrow s = 1) = p_{is} \quad (7.5.6)$$

若机器人选择执行任务  $s$ ，并且能够胜任，或者可与在  $s$  处等待的其他机器人一起完成任务，则立刻将“黑板”上与任务  $s$  对应的信息量  $\tau_s$  置零，以免其他机器人受到错误的指导；仍向  $s$  所在的区域靠拢，并且过多的机器人聚集在同一个区域可能会造成阻塞。若任务仍然未被完成， $\tau_s$  继续保持大于零。如果在两个机器人协作的情况下尚不能完成任务，则将  $\tau_s$  更新为

$$\tau_s = \tau_s + \Delta\tau \quad (7.5.7)$$

由此，难度越大的任务就会逐渐被赋予越来越大的信息量，从而吸引多个机器人进行协作。

机器人的“任务死锁”是指机器人执着于执行一件自己“力所不及”的任务，从而丧失了完成其他任务的可能。与之相类似，多机器人系统中，可能也会出现“任务死锁”。

假设存在这样一种情况：在未知环境中，存在着某一个任务  $s$ ，它的艰巨性很大，是系统中所有机器人协作也完成不了的。当一个机器人  $i$  首先发现这一任务，并且得出依靠自身力量难以胜任的结论时，这一任务会被赋予一定的信息量  $\tau_s$ ；随后，另一个机器人  $j$  受到  $\tau_s$  的吸引，前来协助机器人  $i$  完成这一任务，结果却发现在二者协作的情况下仍然不能完成任务，于是  $\tau_s$  的值继续增大。随着  $\tau_s$  的不断增加，越来越多的机器人受到它的吸引，前来加入协作，结果却是导致  $\tau_s$  的进一步加大。尽管设定当机器人在任务  $s$  处尝试一段时间后，若发现没有进展，就放弃这一选择，重新根据概率选择需要执行的任务，但是由于  $\tau_s$  的值足够大，使得与之对应的  $p_{is}$  也很大，因此当机器人放弃任务  $s$  之后，很有可能马上又受到任务  $s$  的吸引，再次尝试执行这一任务。这样，系统中的机器人很可能会全部被聚拢在一项无法完成的任务旁，结果使系统丧失了行动能力。为了防止上述局面的发生，这里加入了一个自适应衰减因子，使机器人具有摆脱“任务死锁”的状况。

当一个处于空闲状态的机器人发现环境中有  $s$  个任务的信息量大于零，分别为  $\tau_s (s = 1, 2, \dots, k)$ ，则按照下式计算与之相关的概率

$$p_{is} = \frac{(\tau_{is})^\alpha}{\sum_{s=1}^k (\tau_{is})^\alpha} \quad (7.5.8)$$

式中， $\tau_{is}$  表示机器人  $i$  所感触到的信息量，初始化时  $\tau_{is} = \tau_s$ 。经历一段时刻  $T$  后，机器人  $i$  若未能完成任务  $s$ ，则放弃  $s$ ，重新进行选择，同时引入衰减因子  $\lambda$ ，修改  $\tau_{is}$  为

$$\tau_{is} = \lambda \tau_{is} \quad (7.5.9)$$

$$\lambda = \mu^{\text{select}_{is}}, \quad 0 < \mu < 1 \quad (7.5.10)$$

式中， $\mu$  为一个小于 1 的常数， $\text{select}_{is}$  表示机器人  $i$  连续选择任务  $s$  的次数。由公式 (7.5.9) 和公式 (7.5.10) 可见，当  $\text{select}_{is}$  增大时， $\lambda$  减小， $\tau_{is}$  减小， $p_{is}$  随之减小，这就使得机器人放弃任务  $s$  而执行其他任务成为可能。

### 7.5.2.2 算例分析

仿真环境为一  $4m \times 4m$  的房间，其顶点坐标分别为  $(0, 0)$ 、 $(0, 4)$ 、 $(4, 0)$ 、 $(4, 4)$ ，如图 7.24 所示。

图 7.24 所示的房间内分布着 4 个箱子，质量分别为  $3kg$ 、 $5kg$ 、 $9kg$ 、 $3kg$ 。有 4 个机器人参与工作，其目的是在房间中找到这些箱子并将它们搬运回“家”（房间右部中间的正方形区域）。可以看出，至少需要两个机器人协作才能搬运  $5kg$  的箱子，而至少 3 个机器人协作才能将  $9kg$  的箱子搬回家。在开始工作之前，机器人对环境信息一无所知。机器人的速度为  $0.2m/s$ ，载重为  $3kg$ ；机器

人的传感器探测距离为 1m。

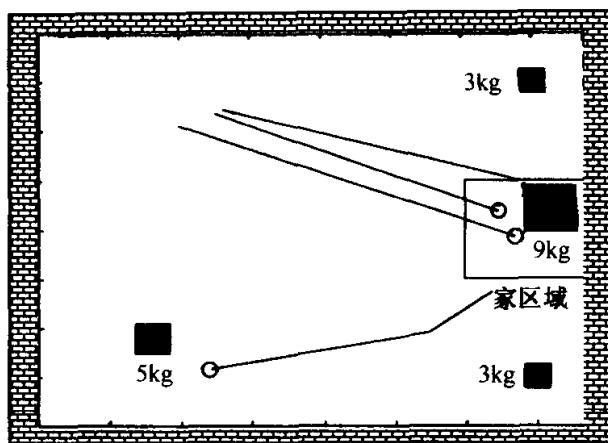


图 7.24 机器人工作阶段

图 7.24 中所示阶段，3 个机器人协同工作搬运一个 9kg 箱子，另一个机器人进行搜寻工作。机器人工作阶段如图 7.25 所示。

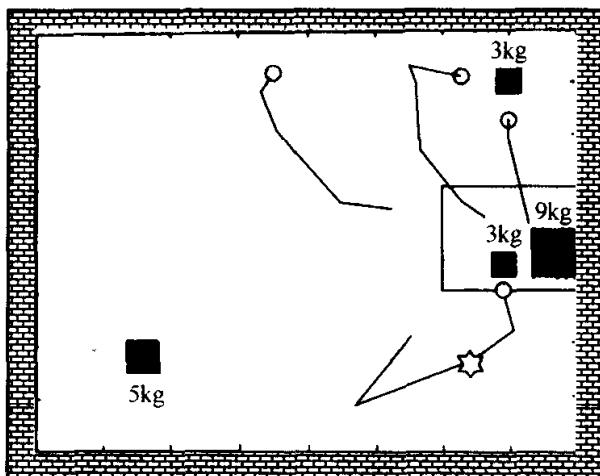


图 7.25 机器人工作阶段

图 7.25 中的右下角，一个机器人在运动过程中找到一个 3kg 的箱子，并将其搬运回家；右上角，两个机器人在运动过程中都发现了一个 3kg 的箱子，并向它靠拢，准备搬运它；中部上方，一个机器人在随机游荡，寻找任务。

设各机器人所具有的能力为：知道自己当前的位置  $X_i = (x_i, y_i)$ ；知道“家”的坐标范围；可以感知测量范围内的机器人或障碍物与自己的距离；可以探测出箱子的存在并能将其抓起；具有局部通信功能，采用“黑板原理”的通信方式。

采用行为主义的方法<sup>[87]</sup>，每个机器人所具有的基本行为有以下 3 种。

(1) 避障：当机器人视野中出现障碍物时，会随机地转动一个角度以绕开障碍。

(2) 游荡：机器人在每一步以一定的概率随机决定是转一定的方向还是沿当前朝向前进。

(3) 回家：机器人沿当前位置与家区域之间的最短路径向家运动。

实验开始时，4个机器人被随机的赋予一个初始坐标，之后，它们在房间中随机游荡，当它们的传感器探测范围内出现箱子时，它们就向之靠拢，尝试独立或与已经等候在那里的机器人协作共同将箱子推回家中。若成功，则将与这一箱子对应的信息量置零；反之，若不成功，则根据协作的机器人数来修改信息量。

首先采用不带衰减因子的策略，进行100次实验，每次实验中机器人的工作时间为30分钟。在全部的100次实验中机器人在规定时间内将所有的箱子搬运回家，占实验次数的100%。这说明改进后的蚁群算法可以成功地为多机器人系统实现协作规划。

此后改变了箱子的重量，用一个15kg的箱子取代了9kg的箱子。可以看出，4个机器人同时协作也不能将15kg的箱子搬运回家。但却可以将除去15kg箱子之外的其他箱子搬回。同样进行100次实验，仍然使用不带有衰减因子的协作策略。只有其中的28次实验，机器人将除去15kg箱子之外的其他3个箱子都搬运回家；有26次发生了“任务死锁”，即在其他箱子还没有搬运完时，4个机器人都被吸引到了15kg的箱子处，从而使系统失去了继续工作的能力。这说明“任务死锁”这一现象在实际情况中很有可能发生，并且会严重的破坏系统的性能。机器人的“任务死锁”状态如图7.26所示。

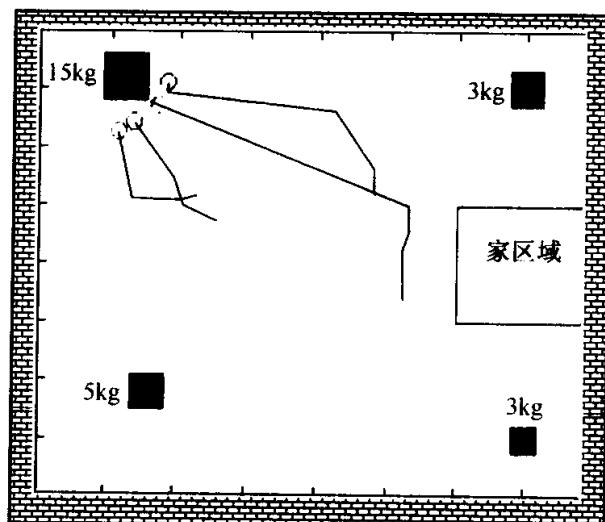


图7.26 机器人的“任务死锁”状态

换用带有衰减因子的策略，衰减因子为0.8，同样进行100次实验。在所有的实验中机器人在30分钟内将15kg箱子之外的所有其他箱子搬运回家，占全部实验次数的100%；没有发生“任务死锁”。

通过仿真可以看到，基于改进蚁群算法的协作策略可以帮助多机器人系统在未知环境中进行协作规划，而衰减因子的引入有效的防止了“任务死锁”状态的出现。

## 7.6 电力系统

电力系统是一个国家经济发展的命脉，而配电网是电力系统的重要组成部分，其投资及运行费用在整个电力系统费用中所占的比例十分可观。好的配电网规划方案可为电力公司节约大量的资金，而配电网网络结构的规划是一个离散的、非线性的多约束组合优化问题，长期以来，各国学者对这一问题做了大量的研究，提出了多种算法<sup>[88~90]</sup>。

机组最优投入问题（unit commitment, UC）是寻求一个周期内各个负荷水平下机组的最优组合方式及开停机计划，使运行费用为最小，它是一个高维数、非凸的、离散的非线性组合优化问题，很难找出理论上的最优解，但由于它能带来显著的经济效益，因此 UC 也是电力系统行业中的一个热点研究内容<sup>[91~94]</sup>。近些年来，用仿生优化算法解决电力系统的各种优化问题一直是一个非常活跃的研究方向，蚁群算法在电力系统领域也得到了很好的应用<sup>[95~102]</sup>。

本节利用蚁群算法的启发式正反馈机制对电力系统中的配电网网络规划和机组最优投入这两个典型问题进行了研究。

### 7.6.1 蚁群算法与配电网网络规划

#### 7.6.1.1 问题描述

配电网中每一段线路的建设费用可表示为<sup>[100]</sup>

$$C_k = l_k \cdot f(D_k) \quad (7.6.1)$$

式中， $C_k$  表示线路  $k$  的投资费用； $D_k$  表示线路  $k$  的线径； $f(D_k)$  表示线径为  $D_k$  时，单位长度线路的投资费用； $l_k$  表示线路  $k$  的长度。

配电网中每一段线路的网损可表示为

$$R_k = \alpha_1 \left( \frac{P_k}{U} \right)^2 g(D_k) l_k \tau \quad (7.6.2)$$

式中， $R_k$  表示线路  $k$  的网损； $P_k$  表示线路  $k$  的通过功率； $\alpha_1$  表示电价； $U$  表示电压； $\tau$  表示年损耗小时数； $g(D_k)$  表示线径为  $D_k$  时线路的电阻率；其余各变量的含义同式 (7.6.1)。

在变电站供电范围已知的情况下，配电网规划问题的数学模型可表示为

$$\begin{aligned} \min C &= \sum_{k \in N_k} (\omega C_k X_k + R_k) \\ \text{s. t. } &\begin{cases} I(l_k) \leq I_{\max}(l_k) \\ V_{\min} \leq V_s \leq V_{\max} \quad s = 1, 2, \dots, S \\ K_{\max} = S - 1 \end{cases} \end{aligned} \quad (7.6.3)$$

式中,  $\omega$  表示年等值系数;  $k$  表示线路的编号;  $N_k$  表示可能构成辐射网的线路编号的集合;  $X_k$  表示线路  $k$  为新修线路时取 1; 线路  $k$  为已有资源时 (如电缆沟、电杆等) 根据实际情况取  $0 \sim 1$  之间的小数;  $I(l_k)$  为线路  $l_k$  的电流;  $I_{\max}(l_k)$  为线路  $l_k$  的最大允许电流;  $s$  为需供电的负荷点编号;  $S$  为节点数目;  $V_{\min}$  为节点电压下限;  $V_{\max}$  为节点电压上限;  $K_{\max}$  为规划后线路的条数。

分析公式 (7.6.3) 可发现,  $C_k$  的值与  $l_k$  成正比,  $R_k$  的值与  $P_k^2 l_k$  成正比。若取不同线径导线单位长度造价, 电阻率为一平均值, 则公式 (7.6.3) 可近似等效为

$$\min C = \sum_{k \in N_k} (al_k X_k + bP_k^2 l_k) \quad (7.6.4)$$

式中,  $a$ 、 $b$  为常数。公式 (7.6.4) 又可等效为

$$\min C = \sum_{k \in N_k} (l_k X_k + \beta P_k^2 l_k) \quad (7.6.5)$$

式中,  $\beta = b/a$ 。公式 (7.6.5) 虽然有某种程度的简化, 但其求解目标是一个多约束的线路编号集合。

### 7.6.1.2 算法设计

改进蚁群算法的信息素处理方式为

$$\tau(l_k, t_{\text{new}}) = (1 - \rho(l_k, t_{\text{new}})) + (t_{\text{new}} - t_{\text{old}}) \cdot Q \quad (7.6.6)$$

$$\rho(l_k, t_{\text{new}}) = \frac{\tau(l_k, t_{\text{new}})}{1 + t_{\text{new}} - t_{\text{old}}} \quad (7.6.7)$$

式中,  $t_{\text{new}}$  表示当前时间;  $t_{\text{old}}$  表示前一时间;  $\tau(l_k, t_{\text{new}})$  表示当前时间街道  $l_k$  的信息量;  $\rho(l_k, t_{\text{new}})$  表示当前时间街道  $l_k$  的信息素挥发系数;  $Q$  表示单只蚂蚁在单位时间段内所遗留的信息素。

由于配电网网络结构规划是一个复杂的组合优化问题, 城市中街道众多, 若仅对“信息素”进行处理, 则将致使收敛速度很慢。这里将需供电的负荷点作为“食物”, 给城市中各条可能的街道赋予“味道”, 通过对“信息素”和“味道”的处理来模拟蚂蚁觅食的过程, 以求得配电网规划问题的最优解或次优解。在给出整条街道“味道”的求解公式之前, 先定义地理信息系统上任意一点的“味道”。地理信息上任意一点  $(x, y)$  的“味道”可表示为

$$S_{xy} = \sum_{i \in N} \frac{\beta W_i^2}{1 + d_i} \quad (7.6.8)$$

式中,  $S_{xy}$  表示任意一点  $(x, y)$  的“味道”;  $W_i$  表示第  $i$  号负荷点的大小;  $d_i$  表示第  $i$  号负荷点到  $(x, y)$  的距离;  $N$  表示所有需要供电负荷点编号的集合。公式 (7.6.8) 中, 负荷值取平方是考虑到网损同负荷值的平方成正比。为了使各条街道的“味道”能确实体现出差异, 应对距离值做处理, 尽可能使

$$0.5 < d_i < 10 \quad (7.6.9)$$

在配电网规划中，由于规划线路只能沿街道进行，因此“味道”也只能沿街道分布。

以图 7.27 为例，负荷点  $W_i$  造成  $C$  点的“味道”为

$$S_C = \beta W_i^2 (1 + l_A + l_B + l_C) \quad (7.6.10)$$

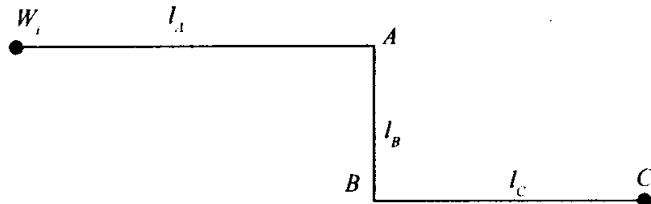


图 7.27 某街道示意图

实际计算中，为了简化编程，可用直线距离乘以地理复杂系数作为两点之间的街道长度。仍以图 7.27 中  $C$  点的“味道”为例，有

$$S_C = \frac{\beta W_i^2}{1 + d_{iC}\xi} \quad (7.6.11)$$

式中， $d_{iC}$  表示负荷点  $W_i$  到点  $C$  的直线距离； $\xi$  表示地理复杂系数。由此可得整条街道为

$$S_{l_C} = \frac{S_B + S_C}{l_C} + \theta_k \cdot l_C \quad (7.6.12)$$

式中， $l_C$  表示街道长度； $S_B$ 、 $S_C$  表示  $l_C$  两端味道； $\theta_k$  为系数，若街道  $l_C$  有已有资源，如电缆沟、电杆，则  $\theta_k$  为一正数，否则为 0。

在明确了“信息素”和“味道”的定义后，这里所确定的规划策略为：在街道有“信息素”时，蚂蚁选择哪条街道的概率由该条街道的“信息素”决定；在所有街道都没有“信息素”时，蚂蚁选择哪条街道的概率由该街道的“味道”决定。通过大批蚂蚁反复从电源点出发寻找负荷点，最终决定配电网规划问题的最优或近似最优网架结构。蚂蚁走过街道后留下的“信息素”可以通过公式 (7.6.6) 求得。

在实际计算中，由于网架的结构未知，公式 (7.6.4) 中系数  $a$ 、 $b$  的值难以确定， $\beta$  值也难以确定。因此，这里通过迭代计算同时确定网架结构及  $a$ 、 $b$  和  $\beta$  值。为了保证计算结果的可行性，这里约定所有蚂蚁在觅食过程中不能走自己已经走过的街道。

改进后蚁群算法的程序结构流程如图 7.28 所示。

图 7.28 中，蚂蚁在点  $r$  时沿街道  $l$  前进的概率  $P(l_k)$  的计算公式为

$$P(l_k) = \begin{cases} \frac{\tau(l_k, t_{\text{new}})}{\sum_{l_m \in L_r} \tau(l_m, t_{\text{new}})}, & \text{若 } \sum_{l_m \in L_r} \tau(l_m, t_{\text{new}}) > 0 \\ \frac{S_{l_k}}{\sum_{l_m \in L_r} S_{l_m}}, & \text{若 } \sum_{l_m \in L_r} \tau(l_m, t_{\text{new}}) = 0 \end{cases} \quad (7.6.13)$$

式中,  $L_r$  表示与点  $r$  相连接的街道所构成的集合, 但不包括蚂蚁到达  $r$  点所经过的街道;  $S_{l_k}$  表示街道  $l_k$  的“味道”。

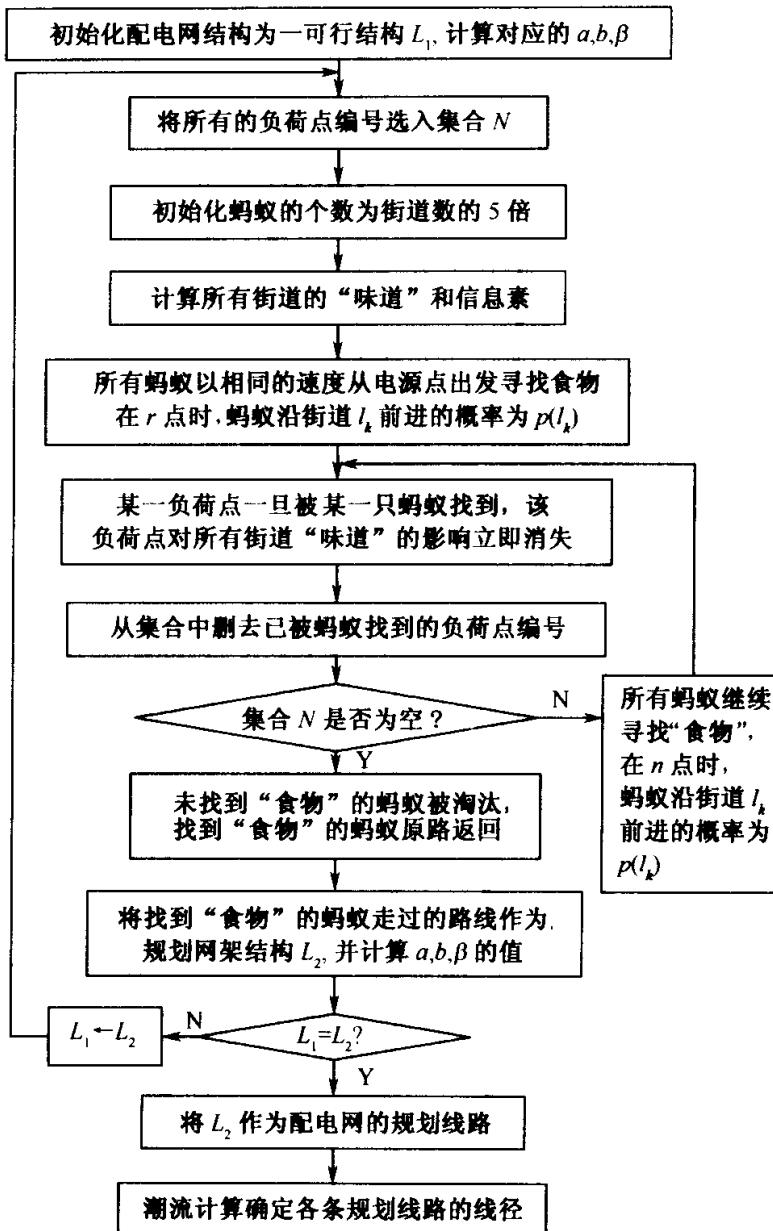


图 7.28 改进蚁群算法的程序结构流程

### 7.6.1.3 算例分析

图 7.29 所示是某城市一区域的实际街道情况，该区域有 33 条街道，21 个负荷点，各条街道上均无已有线路，变电站位置位于点 6。图 7.29 中的数字为节点编号。

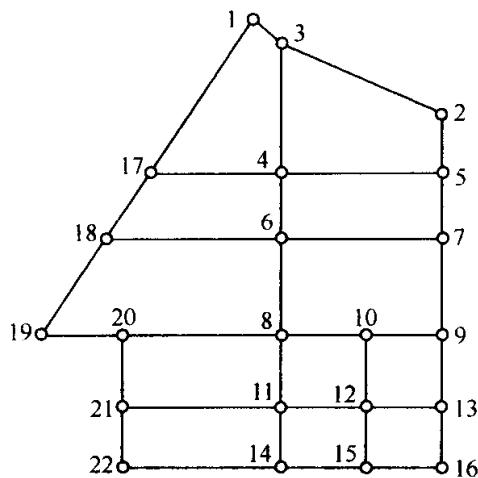


图 7.29 某城市街道图

各负荷点的大小及坐标如表 7.16 所示。

表 7.16 负荷表

节点编号	坐标(X, Y)	负荷值(kVA)	节点编号	坐标(X, Y)	负荷值(kVA)
1	(3.65, 9.15)	1000	12	(5.4, 3.05)	500
2	(6.6, 7.65)	315	13	(6.6, 3.05)	250
3	(4.05, 8.8)	315	14	(4.05, 2.1)	150
4	(4.05, 6.7)	150	15	(5.4, 2.1)	250
5	(6.6, 6.7)	200	16	(6.6, 2.1)	600
6	(4.05, 5.7)	0	17	(2, 6.7)	600
7	(6.6, 5.7)	1000	18	(1.2, 5.7)	150
8	(4.05, 4.2)	600	19	(0.35, 4.2)	150
9	(6.6, 4.2)	250	20	(1.6, 4.2)	200
10	(5.4, 4.2)	630	21	(1.6, 3.05)	200
11	(4.05, 3.05)	1000	22	(1.6, 2.1)	200

设有三种导线可选：LGJ-25/4，LGJ-5018，LGJ-120/25。在利用改进蚁群算法计算后，规划出的网架结构如图 7.30 所示，各条线路的编号由图 7.30 给出。

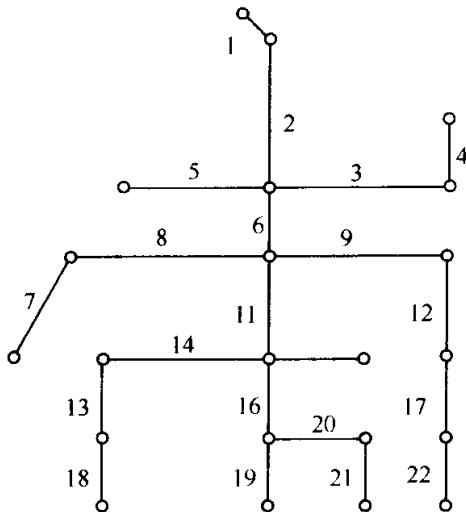


图 7.30 规划结果图

各条线路的导线型号及其所对应的线路编号如表 7.17 所示。

表 7.17 规划结果

导线型号	线路编号
LGJ-25/4	1, 2, 3, 4, 5, 7, 8, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
LGJ-50/8	9, 16
LGJ-120/25	6, 11

由表 7.17 可见，蚁群算法可有效地进行配电的规划，所得结果也比较切合实际。

## 7.6.2 蚁群算法与机组最优投入 (UC)

### 7.6.2.1 问题描述

UC 的目标函数中主要包括两个基本部分：① 机组的发电费用；② 机组的启动费用，同时还必须满足一定的约束条件<sup>[102]</sup>。

UC 本质上是一个多阶段最优决策问题，在每个时段机组有不同的组合方式；而用蚁群算法求解 TSP 也相当于一个求解多阶段的最优决策问题，每个阶段选择一个未曾经过的城市。为将 UC 转化成蚁群算法模式，这里利用了动态规划中状态和决策的概念。此外，考虑到蚁群算法中蚂蚁从某个城市出发到最终又回到该城市，构成了一条封闭路径；而对于 UC，这里给出了如下类似的定义。

**定义 7.6.1 “路径”** 从时段 1 到时段  $T$  的所有时段中，每一时段任取一

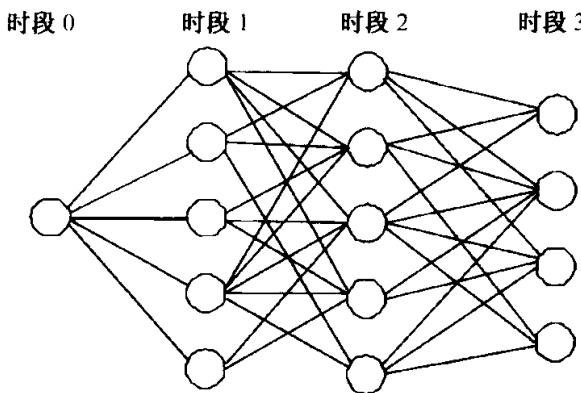


图 7.31 状态转移空间图

种机组组合状态，这样组成的一个决策集合称为一条“路径”。

根据上述概念，UC 可以转化为一个阶段搜索问题，如图 7.31 所示。

在图 7.31 中，时段 0 为初始时段，只有一个状态，进入时段 1 时，有几种可选状态，从时段 0 的状态到时段 1 的任意状态称为决策，不同的决策将导致运行费用的差别，求解目标就是要寻找一个从时段 0 到时段 T 的决策路径，使得总的决策费用最小。此外，时段 1 及以后的各时段，各状态的决策及决策数并不完全相同。

由此，可将 UC 模型可以转化为一个动态模型，即

$$F_t(U_t^l) = \min\{\phi_t^l(U_{t-1}^k, U_t^l)\} \quad (t \in T) \quad (7.6.14)$$

s. t. (1) 功率平衡约束

$$\sum_{i=1}^n P_i^t = P_D^t + P_L^t \quad (t = 1, \dots, T) \quad (7.6.15)$$

(2) 旋转备用约束

$$\frac{\sum_{i=1}^n \gamma_i P_{i\max} - P_D^t}{P_D^t} \geq R^t \quad (t = 1, \dots, T) \quad (7.6.16)$$

(3) 机组出力上下限约束

$$\underline{\gamma}_i P_i \leq P_i^t \leq \bar{\gamma}_i \bar{P}_i \quad (i = 1, \dots, n; t = 1, \dots, T) \quad (7.6.17)$$

(4) 机组最小允许开机、停机时间约束

$$(\gamma_i^t - \gamma_i^{t-1})(w_i^{t-1} - w_i^{\min}) \leq 0 \quad (i = 1, \dots, n; t = 1, \dots, T) \quad (7.6.18)$$

式中， $w_i^t = \gamma_i^t(w_i^{t-1} + 1)$ ， $w_i^t$  表示机组  $i$  在  $t-1$  时段已连续运行的时间。

$$(\gamma_i^t - \gamma_i^{t-1})(q_i^{t-1} - q_i^{\min}) \leq 0 \quad (i = 1, \dots, n; t = 1, \dots, T) \quad (7.6.19)$$

式中， $q_i^t = (1 - \gamma_i^t)(q_i^{t-1} + 1)$ ， $q_i^{t-1}$  表示机组  $i$  在  $t-1$  时段已连续停机的时间。

(5) 线路  $N$  安全性约束

$$P_l \leq P_{l\max} \quad (i = 1, \dots, NL) \quad (7.6.20)$$

公式 (7.6.14) 表示从  $t-1$  时段的状态  $k$  到  $t$  时段的状态  $l$  的最小运行费用。其中， $F_t(U_t^l)$  表示到  $t$  时段时的最小运行费用； $U_t^l$  表示  $t$  时段的  $l$  状态。公式 (7.6.15) ~ (7.6.20) 中， $n$  表示发电机台数； $T$  表示时段数； $NL$  表示线路数； $P_i^t$  表示第  $i$  台机组在第  $t$  时段的出力； $P_D^t$  表示全网负荷； $P_L^t$  表示网损； $R^t$  表示旋转备用率； $\gamma_i^t$  表示 0-1 变量，其中 0 表示停机，1 表示开机； $\underline{\gamma}_i$ 、 $\bar{\gamma}_i$  分别表示第  $i$  台机组的最大和最小出力； $w_i^{\min}$  表示机组最小启动时间， $q_i^{\min}$  表示

机组最小停运时间； $P_l$  表示线路有功功率； $P_{l\max}$  表示线路传输功率的限制值； $P_{i\max}$  表示第  $i$  台发电机的最大出力。由此，从  $t-1$  时段的  $k$  状态转移到  $t$  时段的  $l$  状态时运行费用的累积  $\phi_t^l(U_{t-1}^k, U_t^l)$  为

$$\phi_t^l(U_{t-1}^k, U_t^l) = \sum_{i \in G} F_i(P_i^t) + \sum_{i \in G} F_S(q_i^{t-1}) + C_P(k, l) + F_{t-1}(U_{t-1}^k) \quad (7.6.21)$$

式中， $\sum_{i \in G} F_i(P_i^t)$  表示  $t$  时段的发电费用； $\sum_{i \in G} F_S(q_i^{t-1})$  表示发电机的启动费用； $C_P(k, l)$  表示违反约束时的惩罚费用； $F_{t-1}(U_{t-1}^k)$  表示从时段 1 到时段  $t-1$  运行成本的累积。

### 7.6.2.2 算法设计

#### 1. 目标函数的转化

蚁群算法中，每个优化方案是由一只蚂蚁走过的路径表示的。为了便于求解，可把 UC 设计成类似于 TSP 的模式。机组所有可选的组合状态对应于 TSP 中的各个城市，而两个状态之间的决策对应于 TSP 中的两个城市之间的路径。相应地，UC 的目标函数可以写成 TSP 的模式，即

$$\min \left( \sum_{i=1}^{n-1} tc(s_{\pi(i)}, s_{\pi(i+1)}) + tc(s_{\pi(n)}, s_{\pi(1)}) \right) \quad (7.6.22)$$

式中， $tc(s_i, s_j)$  表示从状态  $i$  到状态  $j$  的转移费用； $\pi(i)$  表示第  $i$  时段所有可选状态集合。这样 UC 就可以像 TSP 一样，采用蚁群算法来对其进行求解。

#### 2. 约束条件的处理

##### (1) 等式约束。

进行功率平衡时要考虑网损的影响，这里采用直流潮流 B 系数法计算网损。利用该方法可直接建立网损与发电机有功之间的关系，即该系统的 B 系数： $B_L$ ， $B_{L0}$ ， $B_0$ 。进行经济调度时，按等耗量/费用微增率法分配各机组出力，以满足功率平衡约束。

##### (2) 不等式约束。

用 tabu 表来限制不满足某些约束的状态。需要说明的是：这里的 tabu 表与基本蚁群算法中的 tabu 表有所不同，这里的 tabu 表对应于时段  $t$ ，记录的是该时段所有允许转移的状态，而任意两个时段的 tabu 表互不约束，它们只与该时段负荷大小和机组过去的启停过程有关，因此可能出现允许转移的状态数不同的情况，并导致图 7.31 中所示决策及决策数的不同。

###### ① 旋转备用和机组最小允许开、停机时间约束。

对于满足上述两种约束的状态，在 tabu 表中将其置 1，否则置 0。

② 机组出力上下限约束。

进行经济调度时，当一台机组出力越限（上/下）时，将其出力定在限值（上/下）处，然后对剩余机组重新调度直到所有机组均满足出力上下限约束并保持功率平衡。

③ 线路安全性约束。

针对公式 (7.6.14) 所给的目标函数，通过引入惩罚项来处理违反安全约束的状态，从而构造出如下增广费用函数

$$\begin{aligned} \min & \sum_{t=1}^T \sum_{i=1}^n [F_i(P_i^t)\gamma_i^t + F_S(t)\gamma_i^t(1 - \gamma_i^{t-1})] \\ & + C_E \left( \max \left( 0, \sum_{k \in NL} (|T_k(0, P)| - T_k) \right) \right) \end{aligned} \quad (7.6.23)$$

式中， $NL$  表示正常工况下的线路集合， $C_E$  表示惩罚因子， $T_k$  表示线路传输功率。

### 3. 算法具体步骤

(1) 对负荷进行排序，从中选择最大负荷和最小负荷。对于  $n$  台机组将其组合成各种状态，计算每个状态下发电机的最大和最小出力之和，得到该状态有功的上限值和下限值。与负荷相比较，若一种状态的下限值大于最大负荷，或上限值小于最小负荷，则将该状态舍去；否则记录该状态及其有功的上、下限值。

(2) 判断初始状态  $S_0$ 。

(3) 蚁群算法的初始化。

(4) 进入下一时段：判断是否为最终时段  $T$ ，若是，则跳转到第 (6) 步，否则跳转到第 (5) 步。

(5) 按照前述的约束处理方法，形成各蚂蚁在当前时段的 tabu 表，并计算相应的启动费用作为蚁群算法中的路径长度。对所有蚂蚁，按状态转移规则选择下一城市  $j$ （状态），将蚂蚁  $k$  移至  $j$ ，计算此状态下各机组的出力及发电费用，并累加蚂蚁  $k$  的路径长度，跳转到第 (4) 步。

(6) 记录得到的最短路径，并按全局信息更新规则更新路径信息；

(7) 如果未达最大迭代次数，且未出现停滞现象，则令  $t = 0$ ，并跳转到第 (4) 步，进行下一次迭代，否则停止。

### 7.6.2.3 算例分析

本算例中的基准功率为 100MW，单线图、线路、负荷、网络结构及参数见文献 [103]，发电机参数见文献 [92] 和文献 [103]，各时段负荷等参数如表 7.18 所示。

表 7.18 各机组的出力及费用

序号	负 荷	发电机出力			总出力	费 用
		1	2	3		
1	2.10	0.50000	0.91981	0.76615	2.18596	3148.99268
2	2.00	0.81888	1.24237	0.00000	2.06125	2825.34399
3	1.80	0.69045	1.16537	0.00000	1.85582	2569.10425
4	2.50	0.54933	1.08076	0.95923	2.58932	3638.10107
5	2.30	0.00000	1.25637	1.16992	2.42629	3247.32617
6	2.35	0.00000	1.28060	1.19899	2.47959	3314.42798
7	1.95	0.78674	1.22309	0.00000	2.00983	2760.94678
8	1.75	0.65841	1.14615	0.00000	1.80456	2505.60742

计算时，设置  $\alpha = 0.2$ ,  $\rho = 0.5$ ,  $Q = 100.0$ ,  $N_{c_{\max}} = 300$ , 最大停滞次数为 10 次。

表 7.18 列出了各时段负荷、机组的出力及相应的运行费用（未包括启动费用）。“0”表示机组在该时段停机。应用蚁群算法得到 8 个时段的总费用为 24199.684, 其中发电费用 24009.852, 启动费用 189.832。图 7.32 显示了求解 UC 时蚁群算法的进化曲线。

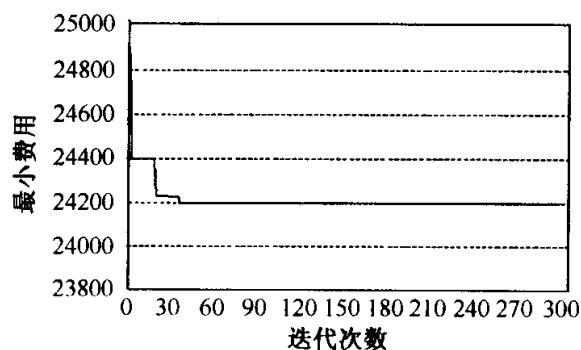


图 7.32 求解 UC 时蚁群算法的进化曲线

由图 7.32 可见，随着迭代次数的增加，费用在逐渐减小，大约 30 次以后费用不再降低，此时已得到了一个较好解。图 7.33 显示了各时段的负荷及相应的发电机总出力，其中柱状图表示某时段的负荷大小，而折线表示相应时段的发电机出力。

由图 7.33 可见，任意时段下发电机的出力总要大于该时段的负荷，即满足旋转备用约束。由仿真结果可见，利用蚁群算法求解 UC 是可行的、有效的。

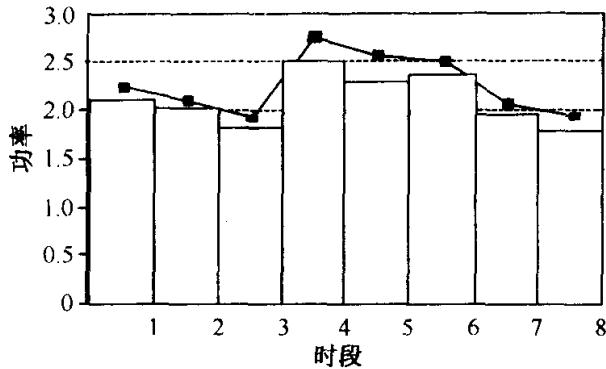


图 7.33 负荷及发电机出力曲线

## 7.7 故障诊断

建立复杂故障诊断系统的关键是如何获取和积累诊断知识、归纳总结诊断规则。为了尽早地发现故障，找出产生该故障的原因，利用机器学习将这些故障征兆自动地分类，将有利于快速、自动、正确地进行故障决策处理。因此，故障模式分类精确与否的重要性就不言而喻了。

近几年，许多学者在将蚁群算法与故障诊断相结合方面进行了一些有益的尝试。Chang C S 等<sup>[104]</sup>最早将蚁群算法应用于电力系统的故障诊断，达到了预想的诊断效果；针对故障诊断过程中故障征兆自动分类的困难性，孙京浩等<sup>[105]</sup>将故障的识别问题转化为求解带约束的最优化故障分类问题，然后通过在蚁群算法中引入近邻函数准则来解决这一最优化故障分类问题；覃方君等<sup>[106]</sup>同时考虑了故障源的重要度和故障检测的难易度两个因素，采用蚁群算法确定故障的最优检测次序，以指导多故障状态的决策；樊友平等<sup>[107]</sup>用蚁群算法来确定故障树的最优检测次序，并实时地、自适应地指导运载火箭控制系统的多故障状态决策，以实现漏电故障的快速定位，既提高了诊断效率，又保证了诊断软件的鲁棒性，取得了良好的实际应用效果。

本节主要研究了一种基于蚁群算法的故障识别分类算法，其主要思路是将故障识别问题转化为求解带约束的最优化聚类问题，并应用蚁群算法求解这一聚类问题。

### 7.7.1 基于蚁群算法的故障识别算法

任一复杂系统发生故障时，其输出或行为将产生与正常状态不同的征兆，这些不同的征兆对应着不同的故障原因。设  $D = \{D_1, D_2, \dots, D_L\}$  表示系统所有故障的非空有限集合， $B = \{B_1, B_2, \dots, B_H\}$  表示故障集合  $D$  引起的所有征兆的

非空有限集合，则该诊断问题可以表述成由已知的征兆集合  $B$  确定产生该征兆集合的故障原因集合的识别。故障征兆  $B$  是多维超空间点的集合，因此，故障诊断的知识获取问题可以转变成欧氏空间中点的聚类问题。

### 7.7.1.1 近邻函数准则<sup>[108]</sup>

单纯地采用样本距离作为故障的分类指标，往往容易将核函数不确定和不能用简单函数来表示的样本错误地划分，从而造成故障的误判。这里在蚁群算法中引入近邻函数准则来解决上述情况下的故障分类问题。

对于数据集中的任意两个样本  $y_i, y_j$ ，若  $y_j$  是  $y_i$  的第  $I$  个近邻，则称  $y_j$  对  $y_i$  的近邻系数为  $I$ 。类似地，若  $y_i$  是  $y_j$  的第  $K$  个近邻，则称  $y_i$  是  $y_j$  的近邻系数为  $K$ 。如果  $y_i$  和  $y_j$  互为最近邻，这时的近邻函数值最小，等于零。若用  $\alpha_{ij}$  表示  $y_i$  和  $y_j$  之间的近邻函数值，则有

$$\alpha_{ij} = I + K - 2 \quad (7.7.1)$$

若在聚类过程中  $y_i$  和  $y_j$  被分在同一类，那么此时的  $y_i$  和  $y_j$  是相互“连接”的。对于每个这样的“连接”存在着一个相应的“连接”损失。算法中，“连接”损失规定为两个样本间的近邻函数值  $\alpha_{ij}$ 。由于  $I, K$  总是小于或等于  $N-1$ ，因此  $y_i$  和  $y_j$  之间的近邻函数最大值为

$$\max \alpha_{ij} = N-1 + N-1 - 2 = 2N-4, \quad i \neq j \quad (7.7.2)$$

显然， $\alpha_{ij} > \max \alpha_{ij}$ ， $i \neq j$ 。这样规定“连接”损失的好处是使密度较为接近的点容易聚成一类。将总类内损失规定为

$$L_{IA} = \sum_{i=1}^N \sum_{j=1}^N \alpha_{ij} \quad (7.7.3)$$

同一类中的  $y_i$  和  $y_j$ ，由于存在着“连接”关系，所以  $\alpha_{ij}$  不等于零；而不同类的  $y_i$  和  $y_j$ ，由于不存在“连接”关系，所以  $\alpha_{ij}$  等于零，因此可对所有样本进行求和。这里，可将总类间损失定义为

$$L_{IR} = \sum_{i=1}^c \beta_i \quad (7.7.4)$$

式中， $c$  为常数； $\beta_i$  为其他类对  $i$  类的类间损失。显然，聚类的结果应使  $c$  个类的类间损失都为负。这里，可定义准则函数为

$$J_{NN} = L_{IA} + L_{IR} \quad (7.7.5)$$

聚类的结果应使准则函数  $J_{NN}$  取最小值。

### 7.7.1.2 算法描述

基于故障样本的分类与蚂蚁觅食及 TSP 的相似性，将每个故障样本数据看

成一只蚂蚁所需访问的地点，每只蚂蚁根据样本数据的近邻函数及样本之间的信息量大小，以一定的概率选择下一个访问的地点，并将其加入自身的数据列表里，直到每只蚂蚁对全部的数据样本都进行了一次唯一的访问为止，从而形成了全部数据样本的一个有序连接。在每个数据列表里，每个故障样本数据都只出现一次。

然后对每只蚂蚁的数据列表根据其样本之间的近邻函数值的大小，打断其中值最大的两个样本数据之间的连接，从而形成了最初的两个故障分类。对于已经进行了分类的故障样本数据，计算每一类故障之间的连接损失。依此类推，直到样本数据之间的连接损失最小为止，最终形成该蚂蚁的故障分类结果。

### 7.7.1.3 算法具体步骤

(1) 计算故障征兆样本数据的加权距离矩阵  $\eta$ ，使其元素  $\Delta_{ij} = \Delta(y_i, y_j)$  表示样本  $y_i$  和  $y_j$  间的加权 Euclidean 距离

$$\Delta_{ij} = \|D(X_i - X_j)\|_2 = \sqrt{\sum_{k=1}^m P_k (x_{ik} - x_{jk})^2} \quad (7.7.6)$$

式中， $P_k$  表示故障征兆样本数据中不同特征的加权因子。

(2) 利用矩阵  $\eta$  作为近故障样本的邻矩阵  $M$ ，其元素  $M_{ij}$  为样本  $y_i$  对  $y_j$  的近邻函数值。一般  $M$  为正定矩阵，由于样本点的近邻序数只能是  $1, 2, \dots, N-1$ ，所以  $M$  矩阵中各元素均为整数。

(3) 形成近邻函数矩阵  $L$ ，其元素为  $L_{ij} = M_{ij} + M_{ji} - 2$ ，如果  $y_i$  和  $y_j$  间有“连接”关系，则  $L_{ij}$  给出了它们之间的近邻函数值。置  $L$  矩阵的对角元素  $L_{ii}$  的值为  $2N$  或更大。

(4) 将每一故障样本数据看作蚂蚁所需访问的地点，每只蚂蚁根据故障样本数据的近邻函数矩阵  $L$ ，并由下式决定下一个将加入其数据列表的不在禁忌列表里的故障样本数据：

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [L_{ij}(t)]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha [L_{ik}(t)]^\beta}, & \text{若 } j \in \text{allowed}_k \\ 0, & \text{否则} \end{cases} \quad (7.7.7)$$

式中， $L_{ij}$  表示样本  $i$  和  $j$  之间的近邻函数值。

(5) 将加入到蚂蚁数据列表的故障样本数据加入其禁忌列表  $\text{tabu}_k (k = 1, 2, \dots, m)$ 。

(6) 判断是否所有的蚂蚁都对全部故障样本数据完成了一次访问。否则跳转到第(4)步。

(7) 由每只蚂蚁的数据列表中的故障样本数据的连接情况，根据样本之间的

近邻函数值的大小，打断当前有连接情况的近邻函数值最大的样本之间的连接，从而形成不同的故障分类。

(8) 由公式 (7.7.3) ~ (7.7.5) 计算所有故障类别的类内损失和类间损失及总的代价。重复第 (7) 步和第 (8) 步，直到每只蚂蚁的  $J_{NN}(k)$  最小，完成一次故障的识别。

(9) 取所有蚂蚁中  $J_{NN}(k)$  最小的蚂蚁的识别结果作为本次循环的求解结果。

(10) 根据信息素更新公式，调整故障征兆样本数据连接之间的信息素大小，其中

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{若第 } k \text{ 只蚂蚁在本次循环使样本数据 } i \text{ 和 } j \text{ 之间有连接} \\ 0, & \text{否则} \end{cases} \quad (7.7.8)$$

式中， $L_k$  表示总连接损失。

(11) 判断循环次数是否达到最大步数的给定值， $J_{NN}=L_{IA}+L_{IR}$  是否达到给定的最小值，若是则中止运行并给出最终解，否则跳转到第 (4) 步。

### 7.7.2 仿真算例

以某化学反应器的故障诊断为例，应用上述改进蚁群算法对其进行仿真研究。反应器的三个主要输入控制变量  $x_i (i=1, 2, 3)$  分别为温度、压力和入料流量。反应器的主要故障形态为低转化率，催化剂低选择性和催化剂熔结。表 7.19 列出了实际测得的 16 组 ( $X_1 \sim X_{16}$ ) 具有温度、压力、入料流量三维的过程样本数据及其对应的不同的故障状态。

表 7.19 反应器的输入变量及其故障状态对应表

序号	温度 ( $x_1$ )	压力 ( $x_2$ )	入料流量 ( $x_3$ )	故障状态
1	400	100	200	无故障
2	420	100	200	
3	380	100	200	
4	400	100	190	
5	400	90	200	
6	400	100	210	
7	400	100	180	
8	300	100	200	低转化率
9	325	90	200	
10	350	80	200	
11	360	100	190	
12	370	100	180	

续表

序号	温度( $x_1$ )	压効( $x_2$ )	入料流量( $x_3$ )	故障状态
13	400	100	250	催化剂低选择性
14	400	100	230	
15	550	100	200	催化剂熔结
16	525	100	180	

设置  $\alpha=1$ ,  $\beta=5$ ,  $\rho=0.9$ ,  $Q=200$ ,  $m=16$ , 最小连接损失为 100, 初始信息素  $S_0=10$ 。按照算法得出的一组运行结果为:  $(X_3, X_1, X_4, X_6, X_5, X_2, X_7)$ ,  $(X_{12}, X_{11}, X_{10}, X_9, X_8)$ ,  $(X_{14}, X_{13})$ ,  $(X_{15}, X_{16})$ , 可以看出, 故障识别的结果是完全正确的。

图 7.34 给出了故障识别过程中整个群体在每次循环中连接损失的标准偏差变化情况和群体连接损失的平均值变化情况。

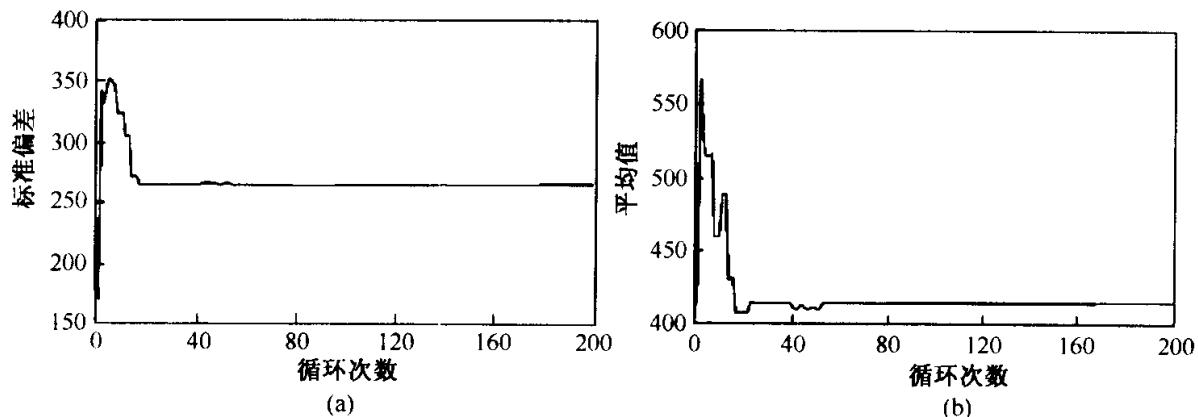


图 7.34 群体连接损失的标准偏差和平均值的变化情况

上述仿真结果充分说明了改进蚁群算法求解故障诊断问题的快速性和有效性。

## 7.8 控制参数优化

PID (proportional-integral-differential) 控制器本质上是一种对“过去”、“现在”和“未来”信息估计的控制算法。PID 控制早在 20 世纪 30 年代末期就已经出现, 由于其具有算法简单、使用方便、鲁棒性好、可靠性高等优点<sup>[109~111]</sup>, 目前在控制工程领域中仍有约 90% 的控制回路具有 PID 结构<sup>[112]</sup>。但是常规 PID 控制算法本身具有一定的局限性, 为了改善其性能, 国际上很多学者对 PID 控制算法进行了大量的研究和改进, 出现了非线性 PID (nonlinear PID, NLPID)、最优 PID、鲁棒 PID、智能 PID 以及自适应 PID 等许多新型的

PID 控制算法。任何 PID 控制器性能的好坏完全依赖于其控制参数的优化，目前除了传统的经验法和 Z-N 法以外<sup>[113]</sup>，采用遗传算法、蚁群算法、微粒群算法等仿生优化算法对 PID 控制参数进行优化已逐渐成为 PID 应用领域中一个新的研究内容<sup>[114~117, 122~127]</sup>。

本节以蚁群算法为基础，首先研究了一种具有不完全微分的 PID 控制器的参数整定方法，随后提出了一种新型的 NLPID 控制器的参数优化策略。

## 7.8.1 蚁群算法与最优 PID 控制

### 7.8.1.1 问题描述

一般来说，引入微分控制可提高系统的动态性能，但会使系统的输出对干扰非常敏感。为了克服这一缺陷，可在 PID 控制器中引入一个一阶惯性环节。该环节本质上是一个低通滤波器，其传递函数为

$$G(s) = \frac{1}{1 + T_f s} \quad (7.8.1)$$

式中， $T_f$  表示时间常数。具有一阶惯性环节的微分控制称为不完全微分控制，具有不完全微分的 PID 控制器的传递函数可表示为

$$U(s) = \left( K_p + \frac{K_p}{T_i s} + \frac{K_p T_d s}{T_f} \right) E(s) = U_p(s) + U_i(s) + U_d(s) \quad (7.8.2)$$

式中， $K_p$  为比例增益； $T_i$  为积分时间常数； $T_d$  为微分时间常数； $E(s)$  为系统的输入量与输出量之间的误差； $U(s)$  为控制量。在离散时间域内，PID 控制律可表示为

$$u(k) = u_p(k) + u_i(k) + u_d(k) = K_p e(k) + K_i \sum_{j=0}^k e(j) + u_d(k) \quad (7.8.3)$$

式中， $K_i = K_p T / T_i$ ； $T$  为采样周期。由公式 (7.8.3)， $U_d(s)$  可表示为

$$U_d(s) = \frac{K_p T_d s}{1 + T_f s} E(s) \quad (7.8.4)$$

将公式 (7.8.4) 转换成如下微分方程

$$u_d(t) + T_f \frac{du_d(t)}{dt} = K_p T_d \frac{de(t)}{dt} \quad (7.8.5)$$

在离散时间域内，公式 (7.8.5) 可以写成

$$\begin{aligned} u_d(k) &= K_d (1 - \lambda) [e(k) - e(k-1)] + \lambda u_d(k-1) \\ &= K_d (1 - \lambda) \sum_{j=1}^k \lambda^{k-j} \cdot \Delta e(j) + \lambda^k u_d(0) \end{aligned} \quad (7.8.6)$$

式中， $\lambda$  为常数，且  $\lambda = T_f / (T_f + T) < 1$ ； $u_d(0)$  为  $u_d(k)$  的初始值； $K_d = K_p T_d / T$ 。将公式 (7.8.6) 代入公式 (7.8.3)，可得

$$u(k) = K_p e(k) + K_i \sum_{j=1}^k e(j) + K_d (1-\lambda) \sum_{j=1}^k \lambda^{k-j} \cdot \Delta e(j) + \lambda^k u_d(0) \quad (7.8.7)$$

与传统的 PID 控制器相比，具有不完全微分的 PID 控制器的算法较复杂，但由于它具有良好的控制性能和鲁棒性能，近年来得到越来越广泛的应用。

### 7.8.1.2 算法设计

#### 1. 节点和路径的生成

以具有不完全微分的 PID 控制器的 3 个参数  $K_p$ 、 $T_i$  和  $T_d$  作为待优化变量，并设它们都有 5 个有效数位。根据这些参数在大多数 PID 控制器中的取值情况，令  $K_p$  的 5 个数位中小数点前占 2 位，小数点后占 3 位； $T_i$  和  $T_d$  的 5 个数位中小数点前各占 1 位，小数点后各占 4 位。为便于采用蚁群算法，把这 3 个参数的值抽象地表示在  $XOY$  平面上，方法为：画 15 条等间距、等长度且垂直于  $X$  轴的线段  $L_1, L_2, \dots, L_{15}$ ，如图 7.35 所示<sup>[116, 117]</sup>。其中  $L_1 \sim L_5$ 、 $L_6 \sim L_{10}$  和  $L_{11} \sim L_{15}$  分别表示  $K_p$ 、 $T_i$  和  $T_d$  的第 1~5 个数位，这些线段在  $X$  轴上的位置分别用数字 1~15 表示。然后，将这些线段都 9 等份，这样每条线段上就有 10 个节点，分别表示该线段所代表的数位可能取的 10 个值 0~9。在  $XOY$  平面上总共有  $15 \times 10$  个节点，用符号  $\text{Knot}(x_i, y_{i,j})$  表示一个节点， $x_i$  为线段  $L_i$  的横坐标 ( $i = 1 \sim 15$ )； $y_{i,j}$  为线段  $L_i$  上节点  $j$  的纵坐标 ( $j = 0 \sim 9$ )。每个节点都表示一个数值，它等于该节点的纵坐标值  $y_{i,j}$ ，例如，节点  $\text{Knot}(5, 8)$  表示  $K_p$  的第 5 个数位（即  $K_p$  值小数点后第 3 位）的值等于 8。

设某只蚂蚁从坐标原点  $O$  出发，当它爬行到线段  $L_{15}$  上任意一点时，完成一次循环，其爬行路径可表示为： $\text{Path} = \{O, \text{Knot}(x_1, y_{1,j}), \text{Knot}(x_2, y_{2,j}), \dots, \text{Knot}(x_{15}, y_{15,j})\}$ 。这里节点  $\text{Knot}(x_i, y_{i,j})$  位于线段  $L_i$  上。显然，这条路径所代表的  $K_p$ 、 $T_i$  和  $T_d$  的值可按照如下公式计算

$$\begin{cases} K_p = y_{1,j} \times 10^1 + y_{2,j} \times 10^0 + y_{3,j} \times 10^{-1} + y_{4,j} \times 10^{-2} + y_{5,j} \times 10^{-3} \\ T_i = y_{6,j} \times 10^0 + y_{7,j} \times 10^{-1} + y_{8,j} \times 10^{-2} + y_{9,j} \times 10^{-3} + y_{10,j} \times 10^{-4} \\ T_d = y_{11,j} \times 10^0 + y_{12,j} \times 10^{-1} + y_{13,j} \times 10^{-2} + y_{14,j} \times 10^{-3} + y_{15,j} \times 10^{-4} \end{cases} \quad (7.8.8)$$

图 7.35 显示了一只蚂蚁的爬行路径，该路径所表示的 PID 参数的值为： $K_p = 64.3780$ ， $T_i = 5.4267$ ， $T_d = 3.4254$ 。

#### 2. 目标函数的建立

为使系统具有良好的性能，建立目标函数时，应以系统的性能指标为基础。这里主要考虑系统单位阶跃响应的超调量  $\sigma$ 、上升时间  $t_r$  以及调整时间  $t_s$ ，可定

义目标函数  $F$  如下

$$F = \lambda_\sigma \left( \frac{\sigma}{\sigma_0} \right) + \lambda_{t_r} \left( \frac{t_r}{t_{r0}} \right) + \lambda_{t_s} \left( \frac{t_s}{t_{s0}} \right) \quad (7.8.9)$$

式中,  $\sigma_0$ 、 $t_{r0}$  和  $t_{s0}$  均为采用 Z-N 法得到的系统性能指标;  $\lambda_\sigma$ 、 $\lambda_{t_r}$  和  $\lambda_{t_s}$  为加权系数, 根据经验, 它们分别取 0.6、0.2 和 0.2。 $\sigma$ 、 $t_r$  和  $t_s$  的约束条件取为

$$\begin{cases} \sigma < \sigma_0 \\ t_r < t_{r0} \\ t_s < t_{s0} \end{cases} \quad (7.8.10)$$

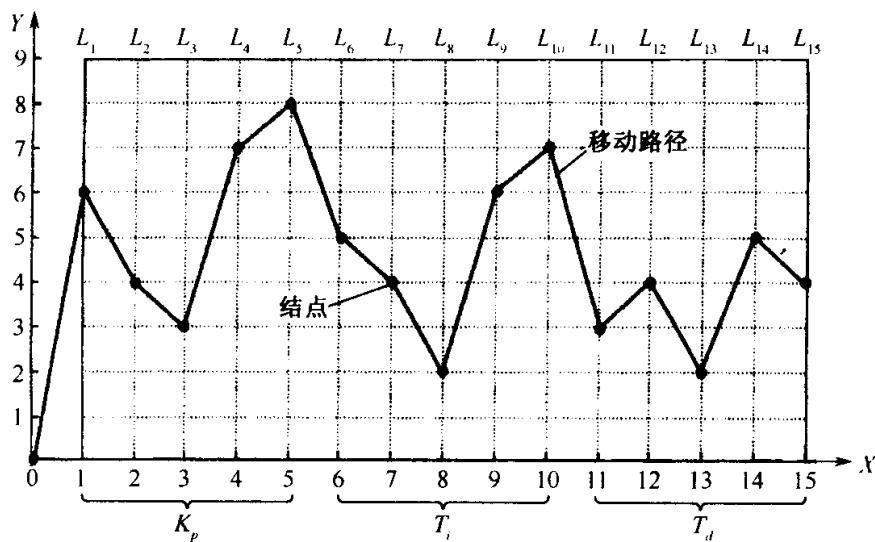


图 7.35 节点和路径生成示意图

### 3. 路径点的选择

假设每只蚂蚁从线段  $L_i$  上任一节点爬行到下一线段  $L_{i+1}$  上任一节点的时间相等, 与节点间距离无关。于是, 若所有蚂蚁都从坐标原点  $O$  出发, 则它们将同时到达每一条线段  $L_i$  ( $i=1 \sim 15$ ), 最后同时到达线段  $L_{15}$  上各自的终点, 完成一次循环。

在时刻  $t$  设蚁群移动到线段  $L_i$  上, 令  $b_j$  ( $j=0 \sim 9$ ) 为  $t$  时刻在  $L_i$  上节点  $j$  处的蚂蚁数, 则蚂蚁总数  $m$  可以表示为  $m = \sum_{j=0}^9 b_j(t)$ 。设  $\tau(x_i, y_{i,j}, t)$  表示  $t$  时刻在节点  $\text{Knot}(x_i, y_{i,j})$  上遗留的信息量, 初始时刻各节点上的信息量相等, 即  $\tau(x_i, y_{i,j}, 0) = c$  (其中,  $c$  为常数,  $i=1 \sim 15$ ,  $j=0 \sim 9$ ),  $\Delta\tau(x_i, y_{i,j}, 0) = 0$ 。设  $P_k(x_i, y_{i,j}, t)$  表示时刻  $t$  第  $k$  只蚂蚁由  $L_i$  上任一点向  $\text{Knot}(x_i, y_{i,j})$  爬行的概率, 则有

$$P_k(x_i, y_{i,j}, t) = \frac{\tau^\alpha(x_i, y_{i,j}, t) \eta^\beta(x_i, y_{i,j}, t)}{\sum_{j=0}^9 \tau^\alpha(x_i, y_{i,j}, t) \eta^\beta(x_i, y_{i,j}, t)} \quad (7.8.11)$$

式中,  $\eta(x_i, y_{i,j}, t)$  为节点  $\text{Knot}(x_i, y_{i,j}, j)$  上的能见度, 且

$$\eta(x_i, y_{i,j}, t) = \frac{10 - |y_{i,j} - y_{i,j}^*|}{10} \quad (7.8.12)$$

式中,  $y_{i,j}^*(i=1\sim 15, j=0\sim 9)$  按如下方式取值: 在蚁群算法的第一次循环中,  $y_{i,j}^*$  为通过 Z-N 法计算出的 PID 参数  $K_{p0}$ ,  $T_{i0}$  和  $T_{d0}$  的值映射在图 7.35 上的 15 个节点所对应的纵坐标值; 在以后各次循环中,  $y_{i,j}^*$  则为上一次循环中所产生的最优路径 (上一次循环中的最优性能指标) 所对应的 PID 参数  $K_p^*$ 、 $T_i^*$  和  $T_d^*$  的值映射在图 7.35 上的 15 个节点所对应的纵坐标值。

#### 4. 信息素的更新

假设在初始时刻  $t=0$ , 所有蚂蚁都位于坐标原点  $O$ , 那么经过 15 个时间单位后, 所有蚂蚁都从起始点爬到终点, 此时可根据如下公式调整各路径点上的信息量

$$\tau(x_i, y_{i,j}, t+15) = \rho\tau(x_i, y_{i,j}, t) + \Delta\tau(x_i, y_{i,j}) \quad (7.8.13)$$

$$\Delta\tau(x_i, y_{i,j}) = \sum_{k=1}^m \Delta\tau_k(x_i, y_{i,j}) \quad (7.8.14)$$

$$\Delta\tau_k(x_i, y_{i,j}) = \begin{cases} \frac{Q}{F_k}, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过 } \text{Knot}(x_i, y_{i,j}) \\ 0, & \text{否则} \end{cases} \quad (7.8.15)$$

式中,  $F_k$  表示第  $k$  只蚂蚁在本次循环中的目标函数值, 由公式 (7.8.9) 计算。

#### 5. 优化 PID 参数的步骤

利用蚁群算法获取最优 PID 参数的步骤可归纳如下。

(1) 利用 Z-N 法计算出 PID 参数  $K_{p0}$ ,  $T_{i0}$  和  $T_{d0}$  以及系统的性能指标  $\sigma_0$ ,  $t_r$  和  $t_s$ 。

(2) 设定蚂蚁数  $m$ , 并给每只蚂蚁  $k(k=1\sim m)$  各定义一个具有 15 个元素的一维数组  $\text{Path}_k$ 。在  $\text{Path}_k$  中依次存放第  $k$  只蚂蚁要经过的 15 个节点的纵坐标值, 可用来表示第  $k$  只蚂蚁的爬行路径。

(3) 令时间计数器  $t=0$ , 循环次数  $N_c=0$ , 设定最大循环次数  $N_{c_{\max}}$  以及初始时刻各节点上信息量  $(x_i, y_{i,j}, 0)$  的值  $c(i=1\sim 15, j=0\sim 9)$ , 令  $\Delta\tau(x_i, y_{i,j})=0$ , 将全部蚂蚁置于起始点  $O$ 。

(4) 置变量  $i=1$ 。

(5) 利用公式 (7.8.11) 计算这些蚂蚁向线段  $L_i$  上每个节点转移的概率; 根据这些概率, 采用赌轮选择方法为每只蚂蚁  $k(k=1\sim m)$  在线段  $L_i$  上选择一个节点, 并将蚂蚁  $k$  移到该节点, 同时将该节点的纵坐标值存入  $\text{Path}_k$  的第  $i$  个元素中。

(6) 置  $i=i+1$ , 若  $i \leq 15$ , 则跳转到第 (5) 步; 否则, 跳转到第 (7) 步。

(7) 根据蚂蚁  $k$  ( $k = 1 \sim m$ ) 所走过的路径, 即数组  $\text{Path}_k$ , 利用公式 (7.8.8) 计算该路径对应的 PID 参数  $K_p^k$ 、 $T_i^k$  和  $T_d^k$ ; 利用这些参数对系统进行计算机仿真并计算出系统的性能指标  $\sigma$ 、 $t_r$  和  $t_s$ ; 利用公式 (7.8.9) 计算蚂蚁  $k$  所对应的目标函数  $F_k$ ; 记录本次循环中的最优路径 (它对应着本次循环中的最优性能指标), 并将与其相对应的 PID 参数存入  $K_p^*$ 、 $T_i^*$  和  $T_d^*$ 。

(8) 令  $t \leftarrow t + 15$ ,  $N_c \leftarrow N_c + 1$ , 根据公式 (7.8.13) ~ (7.8.15) 更新每个节点上的信息量, 并将  $\text{Path}_k$  ( $k = 1 \sim m$ ) 中的所有元素清零。

(9) 若  $N_c < N_{c_{\max}}$  且整个蚁群尚未收敛到走同一条路径, 则再次将全部蚂蚁置于起始点  $O$  并跳转到第 (4) 步; 若  $N_c < N_{c_{\max}}$  但整个蚁群已收敛到走同一条路径, 则算法结束, 输出最优路径及其所对应的最优 PID 参数  $K_p^*$ 、 $T_i^*$  和  $T_d^*$ 。

### 7.8.1.3 仿真算例

这里将这种具有不完全微分的 PID 控制器来控制智能仿生人工腿中的执行电机。该电机的传递函数可表示为

$$G(s) = \frac{6068}{s(s^2 + 110s + 6068)} \quad (7.8.16)$$

对该系统进行计算机仿真。在仿真实验中, 取系统输入量为单位阶跃信号, 并取  $\rho = 0.5$ ,  $\lambda = 0.9$ ,  $T_f = 0.5s$ ,  $m = 10$ 。利用蚁群算法获得的 PID 最优参数  $K_p^* = 27.111$ ,  $T_i^* = 3.1504$ ,  $T_d^* = 0.0128$ 。系统的单位阶跃响应如图 7.36 所示。

系统单位阶跃响应的性能指标如表 7.20 所示。

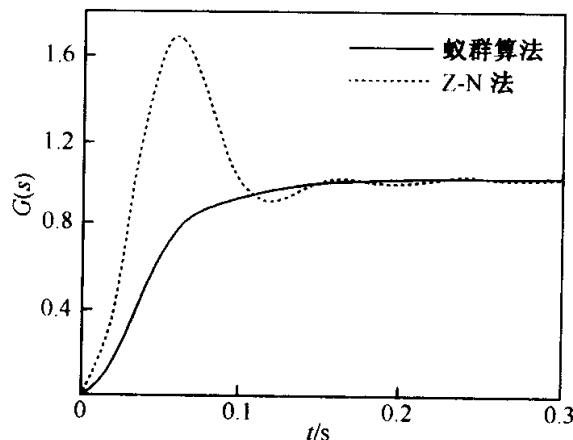


图 7.36 系统的单位阶跃响应

表 7.20 两种控制方法的 PID 参数和性能指标

PID 控制方法	$K_p$	$T_i$	$T_d$	$\sigma(\%)$	$t_s(s)$
Z-N 法	66.000	0.0403	0.0097	68.32	0.1884
蚁群算法	27.111	3.1504	0.0128	6.99	0.1730

由仿真结果可见，利用具有不完全微分的 PID 控制器，系统单位阶跃响应的超调量  $\sigma$  与采用常规 PID 控制器并利用经典的 Z-N 法整定其参数的方法所得的  $\sigma$  相比大幅度减少，调整时间  $t_s$ （取允许误差为 2%）也减小。

### 7.8.2 蚁群算法与 NLPID 控制

#### 7.8.2.1 问题描述<sup>[118~120]</sup>

把参考输入  $r(t)$  送入跟踪-微分器 I，提取两个信号  $x_{11}(t)$  和  $x_{12}(t)$ ， $x_{11}(t)$  跟踪  $r(t)$ ， $x_{12}(t) = \dot{x}_{11}(t)$ ；把被调量  $y(t)$  送入跟踪-微分器 II，提取两个信号  $x_{21}(t)$  和  $x_{22}(t)$ ， $x_{21}(t)$  跟踪  $y(t)$ ， $x_{22}(t) = \dot{x}_{21}(t)$ ，现采用如下 3 个量

$$\begin{cases} e_1(t) = x_{11}(t) - x_{21}(t) \\ e_2(t) = x_{12}(t) - x_{22}(t) \\ e_0(t) = \int_0^t e_1(\tau) d\tau \end{cases} \quad (7.8.17)$$

来替代常规 PID 控制器中的 3 个基本要素： $e(t) = r(t) - y(t)$ ,  $e_1(t)$ ,  $\int_0^t e(\tau) d\tau$ 。

然后通过  $e_0$ 、 $e_1$  和  $e_2$  的适当非线性组合来产生控制量  $u_1(t)$ ，随后再经过线性滤波器是使其变为  $u(t)$ ，该 NLPID 控制器系统结构如图 7.37 所示。

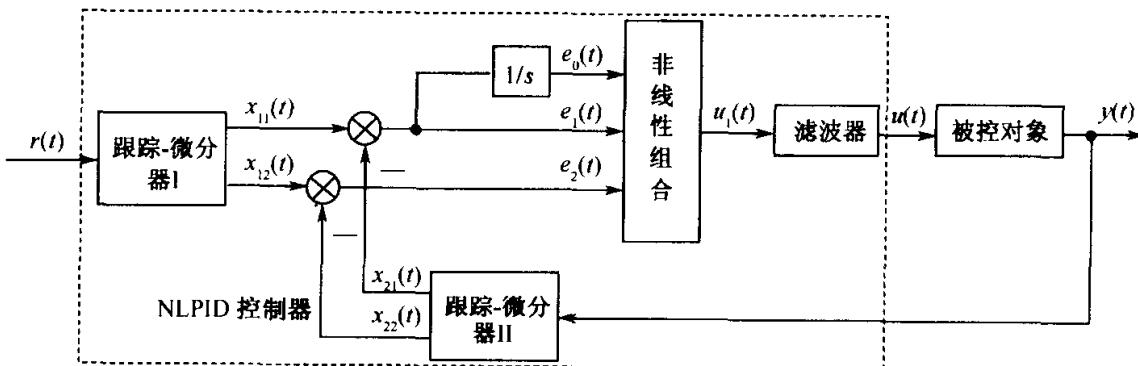


图 7.37 NLPID 控制器系统结构

在图 7.37 中，跟踪-微分器 I 主要安排理想的过渡过程  $x_{11}(t)$ ，并给出此过程的微分信号  $x_{12}(t)$ ，而跟踪-微分器 II 主要尽快复原被调量  $y(t)$  并给出其微分信号。此处，NLPID 控制器的基本要素不是直接取自输入-输出误差，而是输入信号和输出信号经非线性处理所得的新的差分及其微分、积分。

令

$$\text{sat}(x, \delta) = \begin{cases} \text{sgn}(x), & \text{若 } |x| \geq \delta \\ \frac{x}{\delta}, & \text{否则} \end{cases} \quad (7.8.18)$$

式中,  $\delta$  为常数。由此可构造如下 NLPID 控制器:

对于跟踪-微分器 I, 有

$$\begin{cases} \dot{x}_1 = x_2 \\ A_1 = x_1 - r(t) + \frac{x_2 |x_2|}{2R_1} \\ \dot{x}_2 = -R_1 \text{sat}(A_1, \delta_1) \end{cases} \quad (7.8.19)$$

式中,  $R_1$  是根据过渡过程速度要求来确定的;  $\delta_1$  是与积分步长和  $R_1$  有关的参数。同理, 对于跟踪-微分器 II, 亦有

$$\begin{cases} \dot{x}_3 = x_4 \\ A_2 = x_3 - y(t) + \frac{x_4 |x_4|}{2R_2} \\ \dot{x}_4 = -R_2 \text{sat}(A_2, \delta_2) \end{cases} \quad (7.8.20)$$

式中,  $R_2$  也是根据过渡过程速度要求来确定的, 它的取值通常比  $R_1$  的取值大;  $\delta_2$  是与  $R_1$ 、 $R_2$  和  $\delta_1$  有关的参数,  $\delta_2 = \frac{R_2}{R_1} \delta_1$ 。

同时, 定义误差积分为  $\dot{x}_5 = x_1 - x_3$ , 且  $e_0 = x_5$ ,  $e_1 = x_1 - x_3$ ,  $e_2 = x_2 - x_4$ 。令非线性函数

$$\text{fal}(e, \alpha', \delta) = \begin{cases} |e|^{\alpha'} \text{sgn}(e), & \text{若 } |e| \geq \delta \\ \frac{e}{\delta^{1-\alpha'}}, & \text{否则} \end{cases} \quad (7.8.21)$$

当  $e < |\delta|$  时,  $\text{fal}(e, \alpha', \delta)$  和  $e$  之间是线性关系, 从而当  $e$  接近 0 时,  $\text{fal}(e, \alpha', \delta)$  可实现平滑的控制。由此可得如下非线性组合

$$\begin{cases} u_1 = \beta_0 \text{fal}(e_0, \alpha', \delta) + \beta_1 \text{fal}(e_1, \alpha', \delta) + \beta_2 \text{fal}(e_2, \alpha', \delta) \\ \dot{x}_6 = -\rho_0(x_6 - u_1) \\ u = x_6 \end{cases} \quad (7.8.22)$$

式中,  $\beta_0$ 、 $\beta_1$ 、 $\beta_2$  表示 NLPID 控制器的控制系数;  $x_6$  表示经过滤波后施加于被控对象的控制量, 公式 (7.8.22) 中的滤波器用于提高系统的鲁棒性能和稳定性;  $\rho_0$  是根据滤波和跟踪的需要而定, 即比较  $u = u_1$  和  $u = x_6$  的效果来确定;  $\alpha'$  应取 0.5~1.0 之间的常数;  $\delta$  也是适当小的常数。由此可见, 这种 NLPID 控制器的关键是其控制系数  $\beta_0$ 、 $\beta_1$  及  $\beta_2$  的优化问题。

### 7.8.2.2 算法设计

这里以飞行仿真转台 (flight simulator) 为应用对象来阐述基于改进蚁群算

法的 NLPID 控制参数优化策略。

飞行仿真转台是航空、航天等国防领域中进行地面含实物半物理仿真和测试实验的关键硬件设备<sup>[121]</sup>，它可在实验室环境内实时地复现飞行器在空中的动力学特性和各种飞行姿态，其性能的优劣直接关系到仿真实验的逼真性和置信度。

基于改进蚁群算法优化 NLPID 控制参数的飞行仿真转台系统结构如图 7.38 所示。

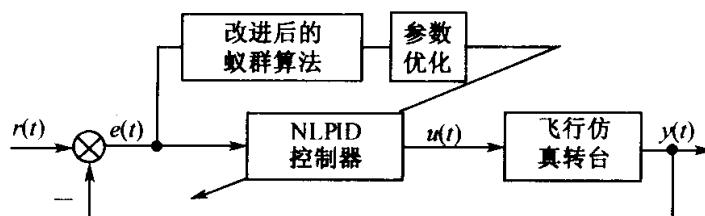


图 7.38 基于改进蚁群算法的 NLPID 飞行仿真转台系统结构

飞行仿真转台的 NLPID 参数优化问题可以归结为典型的连续空间优化问题<sup>[122~125]</sup>。而求解连续空间优化问题的蚁群算法思路一般为：首先可根据问题的性质估计一下最优解的范围，估计出所求变量的取值范围。在变量区域内打网格，空间的网格点上对应于一个状态，蚂蚁在各个空间网格点之间移动，根据各网格点的目标函数值，留下不同的信息量，以此影响下一批蚂蚁的移动方向。循环一段时间后，相邻节点间的目标函数差值（即评价函数值）小的网格点信息量比较大。根据信息量，找出信息量大的空间网格点，缩小变量范围，在此点附近进行蚁群移动。重复上述过程，直到满足算法的停止条件为止。

时间乘以误差绝对值积分（integrated time absolute error, ITAE）最小的性能指标是由 Graham D 和 Lathrop L C 最先提出的<sup>[128]</sup>，它较少考虑大的起始误差，强调超调量和调节时间，反映了控制系统的快速性和精确性，在控制领域中一直被普遍采用。其具体形式如下

$$J_{\text{ITAE}} = \int t |e(t)| dt = \min \quad (7.8.23)$$

将上式离散化，可得其差分方程为

$$J_{\text{ITAE}} = \sum_{k=0}^N |e(kT)| kT = \min \quad (7.8.24)$$

式中， $T$  表示仿真计算步长， $N$  表示仿真计算的总点数。

设蚂蚁总数为  $m$ ，寻优时，将蚂蚁按照随机原则散布在空间网格点上，对于每只蚂蚁  $i$ ，定义其评价函数值为  $i$  点的目标函数  $J_i$  和相邻为  $j$  点的目标函数  $J_j$  的差值，并记

$$\Delta J_{ij} = J_i - J_j, \quad \forall i, j \quad (7.8.25)$$

定义蚂蚁  $i$  的转移概率如下

$$P_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\Delta J_{ij}]^\beta}{\sum_{r \in \text{allowed}_l} [\tau_{ir}]^\alpha [\Delta J_{ir}]^\beta}, & \text{若 } j \in \text{allowed}_l \\ 0, & \text{否则} \end{cases} \quad (7.8.26)$$

寻优时，将蚂蚁按照随机原则散布在空间网格点上，并记录具有最好评价函数值的精灵蚂蚁。然后，按照公式 (7.8.26) 所给的空间状态转移概率移动各只蚂蚁。在搜索过程中嵌入了邻近搜索机制，即当  $\Delta J_{ij} > 0$  时，蚂蚁  $l$  按概率  $P_{ij}$  从其邻域  $i$  移动至邻域  $j$ ；而当  $\Delta J_{ij} \leq 0$  时，蚂蚁  $l$  进行自身的邻域搜索，以寻找更优解。

一次循环结束时，蚂蚁所移动路径上的信息素数量按照下式做相应调整

$$\begin{cases} \tau_{ij,\text{New}} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \\ \Delta\tau_{ij} = \sum_{l=1}^m \Delta\tau_{ij}^l \end{cases} \quad (7.8.27)$$

$$\Delta\tau_{ij}^l = \begin{cases} Q, & \text{若 } l \text{ 只蚂蚁在本次循环中经过路径 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (7.8.28)$$

式中， $J_l$  表示第  $l$  只蚂蚁在本次循环中的目标函数计算值。这里还引入了对  $\rho$  值采取自适应控制的策略<sup>[129]</sup>，以提高算法的求解效率，即将常数  $\rho$  改为如下阈值函数

$$\rho(t+1) = \begin{cases} 0.9 \cdot \rho(t), & \text{若 } 0.9 \cdot \rho(t) > \rho_{\min} \\ \rho_{\min}, & \text{否则} \end{cases} \quad (7.8.29)$$

为了提高搜索速度，该寻优策略中采用了两只蚂蚁从两个极限点同时搜索的优化方案，这种并行处理策略可有效地提高算法的全局收敛速度。

### 7.8.2.3 仿真算例

本仿真实验对象是某型高性能电动飞行仿真转台，其中断采样间隔是 0.0008s，设置  $\alpha = 1.5$ ,  $\beta = 4.5$ ,  $\rho_{\min} = 0.2$ ,  $\tau_{ij} = 1$ ,  $\Delta\tau_{ij}(0) = 0$ ,  $m = 20$ ,  $\rho_0 = 0.15$ ,  $R_1 = 150$ ,  $R_2 = 100$ ,  $\delta_1 = 0.002$ ,  $\delta_2 = 0.0013$ ,  $\delta = 0.01$ ,  $\alpha' = 0.6$ ,  $N_{c_{\max}} = 100$ ,  $Q = 300$ 。经过多次调试，确定控制系数  $\beta_0$ 、 $\beta_1$  及  $\beta_2$  的大致范围为  $\beta_0 \subset [0.01, 0.09]$ ,  $\beta_1 \subset [770, 790]$ ,  $\beta_2 \subset [0.1, 0.7]$ 。用改进后的蚁群算法连续进行了 10 次数字仿真，所得的最优解均值为  $\bar{\beta}_0^* = 0.03$ ,  $\bar{\beta}_1^* = 776.39$ ,  $\bar{\beta}_3^* = 0.54$ 。目标函数  $J$  的进化过程如图 7.39 所示。

由图 7.39 可见，尽管最优目标函数  $J$  在寻优的初期处于极大的发散状态，但是仍能以极快的速度收敛，从而寻到满意的结果。

图 7.40 给出了采用基于改进蚁群算法优化 NLPID 控制参数的飞行仿真转台系统跟踪带噪声随机信号时的运行情况。输入信号采用了如下带噪声的随机

信号

$$r(t) = 1.5\sin(2\pi t) + 0.5\cos(6\pi t) + \sigma(t) \quad (7.8.30)$$

式中,  $\sigma(t)$  为 5% 的白噪声, 即  $|\sigma(t)| < 0.05$ 。

由实验结果可见, 采用改进蚁群算法优化 NLPID 控制参数的飞行仿真转台系统可以从带噪声的输入信号中合理地提取微分信号, 对噪声具有很强的滤波作用, 整个系统响应速度快, 并且具有较强的鲁棒性。

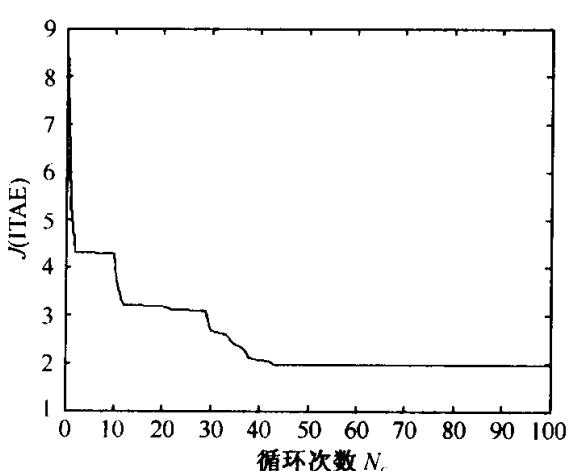


图 7.39 目标函数随迭代次数的进化曲线

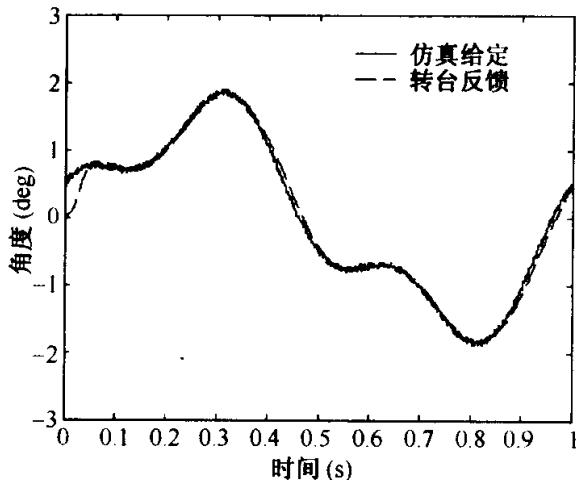


图 7.40 噪声信号输入下的飞行仿真  
转台跟踪响应

## 7.9 系统辨识

Zadeh L A 曾于 1962 年对“系统辨识”给出如下定义：“系统辨识是在对输入和输出观测的基础上, 在指定的一类系统内确定一个与被识别系统相等价的模型”。实际上, 我们不可能找到一个与实际系统完全等价的模型。但从实用的角度来看, 系统辨识就是从一组模型中选择一个模型, 按照某种准则, 使之能最好地拟合由系统的输入、输出观测数据体现出的实际系统的动态或静态特性。

近年来, 随着对仿生智能理论研究的不断深入及其在不同领域的广泛应用, 许多学者开始尝试把人工神经网络、遗传算法、蚁群算法、微粒群算法等应用于系统辨识中, 发展了很多新的系统辨识方法<sup>[130~133, 137]</sup>。本节基于蚁群算法的基本思想, 首先以连续空间内的一类线性系统为例, 研究了线性系统的参数辨识问题, 然后以 LuGre 摩擦模型为对象研究了非线性系统的参数辨识问题。

## 7.9.1 蚁群算法与线性系统参数辨识

### 7.9.1.1 算法设计

这里以连续空间内线性系统  $\dot{x} = A_p x + B_p u$  的参数辨识问题为例，研究了多维连续空间内基于蚁群算法的参数辨识问题<sup>[132, 133]</sup>。用于线性系统参数辨识的蚁群算法的总体框架如图 7.41 所示，其中被辨识系统参数为  $A_p$  和  $B_p$ ，而参照系统参数  $A_{S1}$  和  $B_{S1}$  的变化受蚁群在解空间内的寻优移动过程制约。在此辨识问题中，蚁群算法寻优的空间维数为矩阵  $A_p$  和  $B_p$  的元素个数之和，即所需辨识的参数个数之和。现假设矩阵  $A_p$  和  $B_p$  均为单元素，即蚁群个体是在  $A$  轴和  $B$  轴所构成的二维参数空间内进行参数辨识操作。

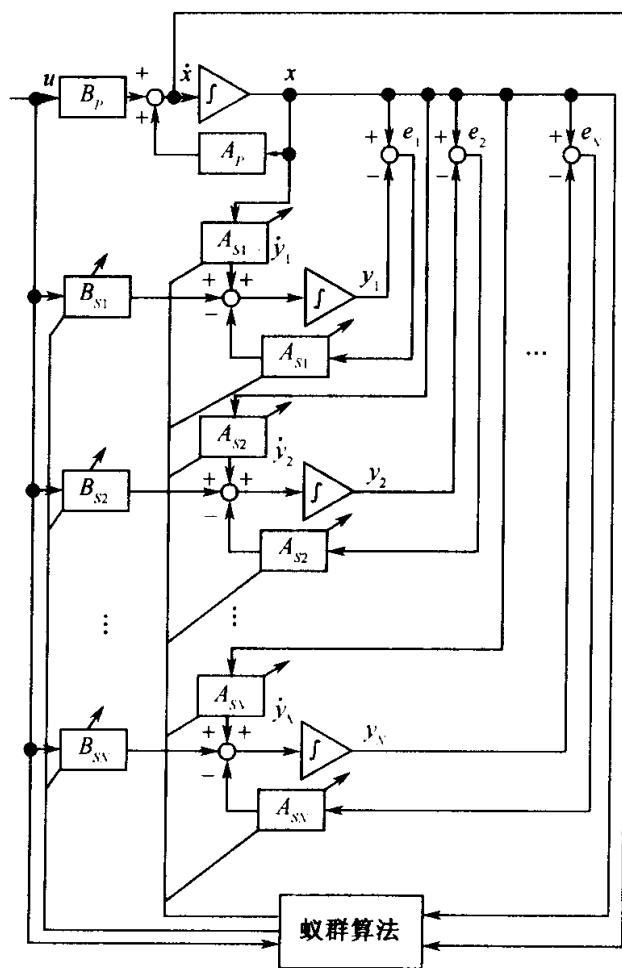


图 7.41 蚁群算法用于线性系统参数辨识的总体框图

用于线性系统参数辨识的蚁群算法具体步骤如下。

(1) 将蚁群在解空间内按照一定方式做初始分布。这里需根据问题定义域的

大小，即被辨识系统参数的可能范围的大小，决定合适的蚁群规模。这里采用的蚁群规模为  $N^2$  个，即以每  $N$  个单蚁为一组，在  $A$  轴方向上作均匀分布，而同一组内的  $N$  个单蚁又按照  $B$  轴方向在问题空间内做均匀分布。举例来说，如果被辨识系统问题的定义域为  $A : [\text{Start } A, \text{End } A]$ ,  $B : [\text{Start } B, \text{End } B]$  (矩形形状)，同时，蚁群分布又按照先变化  $B_{Si}$ ，后变化  $A_{Si}$  的均匀分布方式，则蚁群的初始坐标分布为

$$\begin{cases} A_{Si} = \text{Start } A + \left[ \text{Int}\left(\frac{i-1}{N}\right) + \frac{1}{2} \right] D_{AL} \\ B_{Si} = \text{Start } B + \left[ i - \text{Int}\left(\frac{i-1}{N}\right) \cdot N - \frac{1}{2} \right] D_{BL} \end{cases} \quad (7.9.1)$$

式中， $\text{Int}(\cdot)$  表示取整函数， $D_{AL}$  和  $D_{BL}$  表示将问题空间在  $A$  轴和  $B$  轴上的映射进行  $N$  等分后所得的单维区间长度，即

$$D_{AL} = \frac{\text{End } A - \text{Start } A}{N}, \quad D_{BL} = \frac{\text{End } B - \text{Start } B}{N} \quad (7.9.2)$$

这些区间在  $A$  轴和  $B$  轴上映射的左边界  $A_{iL}$ 、 $B_{iL}$  和右边界  $A_{iR}$ 、 $B_{iR}$  的取值分别为

$$\begin{cases} A_{iL} = A_{Si} - D_{AL}/2, & B_{iL} = B_{Si} - D_{BL}/2 \\ A_{iR} = A_{Si} + D_{AL}/2, & B_{iR} = B_{Si} + D_{BL}/2 \end{cases} \quad (7.9.3)$$

这样，每个单蚁就处于将参数空间的  $A$  轴分量和  $B$  轴分量在参数变化范围内均进行  $N$  等分后所得到的  $N^2$  个矩形子空间的中心。这里每个单蚁还带有一个随自己坐标位置变化的移动矩形子空间，而自己处于该移动的矩形子空间的中心。移动矩形子空间的长度和宽度分别为  $D_{AL}$  和  $D_{BL}$ 。当各单蚁处于各矩形子空间的中心时，定义此时各子空间内的蚁数为 1。而当各单蚁移动时，根据其所带移动矩形子空间与相邻子空间的重叠程度变化，定义相邻子空间内的实际蚁数变化。例如，如果单蚁  $i$  从  $(A_{Si}, B_{Si})$  移动至  $(B_{Si} + \Delta A, B_{Si} + \Delta B)$ ，则移动后与此单蚁所带移动子空间相交的  $A$  轴和  $B$  轴各子空间内的蚁数就相应变化  $\Delta n_A = \frac{\Delta A}{D_{AL}}$ ,  $\Delta n_B = \frac{\Delta B}{D_{BL}}$ 。

(2) 根据蚁群所处解空间位置的优劣，决定当前蚁群的信息量分布。这里蚁群信息量分布函数的定义要与各单蚁当前所处解空间位置  $(A_{Si}, B_{Si})$  针对参数辨识正确性的优劣相关。在二维空间内，可定义各单蚁所对应的信息量分布函数为

$$T_i(A_s, B_s) = \frac{M_i e^{-k_i \sqrt{(A_s - A_{Si})^2 + (B_s - B_{Si})^2}}}{\left[ 1 + e^{-k_i \sqrt{(A_s - A_{Si})^2 + (B_s - B_{Si})^2}} \right]^2} \quad (7.9.4)$$

式中， $k_i$  为根据实际问题所定义的压缩系数，而其峰值  $M_i$  的定义要与各单蚁当前所处的解空间位置  $(A_{Si}, B_{Si})$  的优劣相关，即

$$M_i = \frac{C_4 - [\dot{x} - A_{Si}x - B_{Si}u]^2}{C_5} \quad (7.9.5)$$

式中,  $x$ 、 $\dot{x}$ 、 $u$  分别表示被辨识系统当前检测的状态量、状态量变化率及系统的外加输入。显然当  $(A_{Si}, B_{Si})$  接近于  $(A_p, B_p)$  时, 以上所定义的信息量分布函数峰值  $M$  取值较高。

(3) 根据当前蚁群散布的总信息量分布情况和上一循环过程中信息量的遗留和挥发情况, 决定各子区间内应有的蚁数分布。先针对  $A$  轴进行以上操作。首先, 求得当前蚁群散布的总信息量分布函数在各  $A$  轴子区间内的积分值  $IN_{iA}$  ( $i=1 \sim N$ ), 即有

$$IN_{iA} = \int_{StartB}^{EndB} \int_{A_{iL}}^{A_{iR}} \sum_{s=1}^{N^2} T_i(A_s, B_s) dA_s dB_s \quad (7.9.6)$$

各  $A$  轴子区间的实际总信息量  $I_{iA}$  应为当前蚁群在该  $A$  轴子区间内散布的信息量 ( $IN_{iA}$ ) 加上上一次总信息量的遗留部分 ( $\eta I_{iALast}$ ,  $\eta$  为信息量留存系数), 再与所设定的信息量挥发常量  $E_V$  相减所得的结果, 即

$$I_{iA} = IN_{iA} + \eta I_{iALast} - E_V \quad (7.9.7)$$

然后, 按照下式求取实际总信息量在整个  $A$  轴问题区间的总积分值

$$I_{\Sigma A} = \sum_{i=1}^{N^2} I_{iA} \quad (7.9.8)$$

根据各  $A$  轴子区间实际总信息量  $I_{iA}$  占总积分值  $I_{\Sigma A}$  的比例, 可求得当前蚁群分布条件下决定的各  $A$  轴子区间内应有蚁数, 即有

$$N_{iMA} = \frac{I_{iA}}{I_{\Sigma A}} \cdot N^2 (i = 1 \sim N^2) \quad (7.9.9)$$

在实际的编程运算中, 由于所取的信息量分布函数  $T_i(A_s, B_s)$  为可积函数, 且函数曲面是连续平滑的。这样, 对积分值  $IN_{iA}$  的求取可得到一定程度的简化。

针对  $B$  轴各子空间的运算操作与  $A$  轴类似 (其中在针对  $B$  轴各子区间的积分值求取时,  $A$  轴分量的积分边界取  $[StartA, EndA]$ )。

(4) 根据各子区间内应有的蚁群分布状况和当前蚁群分布状况之间的差别, 决定蚁群的移动方向, 并加以移动。

同样, 先针对  $A$  轴进行以上操作。

首先, 根据已求得的  $A$  轴各子区间内的应有蚁数  $N_{iMA}$ , 以所考察之蚁当前所处的  $A$  轴区间为界进行求和操作, 求出被考察之蚁所处  $A$  轴区间  $i$  之左的应有蚁数之和  $N_{iRLA} = \sum_{j=1}^{i-1} N_{jRA}$ , 和所处  $A$  轴区间  $i$  之右的实际蚁数之和  $N_{iRRA} = \sum_{j=i+1}^N N_{jRA}$ 。

然后, 根据被考察之蚁所处  $A$  轴子区间及其左右的实际蚁数和应有蚁数之间的差别, 决定该蚁的运动方向, 并将该蚁的  $A$  轴坐标值变化  $\Delta A_s$ 。其他情况

下被考察之蚁的 A 轴坐标均不作变动。

为避免整个蚁群处于同一子区间时可能出现的停滞状态，这里规定，决定蚁群在 A 轴方向的移动，不仅仅以各单蚁 A 轴坐标所处子区间为准进行考察，而是同时在与各单蚁所在移动子区间相交的两个 A 轴子区间内进行考察，按照考察结果决定被考察之蚁的移动方向。蚁群 B 轴分量的变化寻优过程与 A 轴分量类似。

在蚁群做完一次整体移动之后，又可回到第（2）步，进行相应的信息量分布、考察和蚁群移动操作，如此循环往复，直到产生最优解为止。

### 7.9.1.2 仿真算例

令被辨识系统为  $\dot{x} = A_p x + B_p u$ ,  $A_p$  和  $B_p$  为单参数，其中  $A_p = B_p = 2$ ,  $u$  取阶跃式输入（幅值为 8）。假设系统的状态变量  $x$  及其变化率  $\dot{x}$ ，以及外界输入  $u$  的实际值以  $\Delta t = 0.00001\text{s}$  的时间间隔被准确采样，作为蚁群辨识系统的输入（在第五次辨识寻优过程中，外界输入  $u$  的幅值为 40，采样时间间隔为  $0.0005\text{s}$ ）。

这里，为描述整个蚁群的辨识寻优动态，令整个蚁群的辨识总误差为

$$E_{\Sigma} = E_{\Sigma A} + E_{\Sigma B} = \sum_{i=1}^{N^2} |A_s - A_p| + \sum_{i=1}^{N^2} |B_s - B_p| \quad (7.9.10)$$

在五次典型辨识过程中的蚁群算法参数设置及辨识结果分布如表 7.21 所示。其中第五次辨识寻优时，由于寻优区间大大缩小，则对蚁群系统各参数的调整幅度也大大增加。蚁群数目也改为  $N^2 = 3^2 = 9$ 。这里以第一次和第五次辨识寻优过程的总误差变化动态（分别如图 7.42（a）和图 7.42（b）所示）为例，说明蚁群算法辨识的有效性。

表 7.21 典型辨识过程中的蚁群算法参数设置及辨识结果分布

辨识次数	初始分布区域	$\eta$	$k_i$	dAdB	$C_4$	$C_5$	$E_V$	$E_{\Sigma}$	$N$	辨识结果分布区间
1	A : [0,5] B : [0,5]	0.01	1	0.01	1000	5	228	14.43	5500	A : [1.265, 3.265] B : [1.425, 3.205]
2	A : [1,3.5] B : [1,3.5]	0.01	1	0.01	1000	5	228	6.37	9250	A : [1.76, 2.30] B : [1.99, 2.68]
3	A : [1.5,2.8] B : [1.5,2.8]	0.01	1.2	0.01	1000	5	228	2.75	9200	A : [1.91, 2.26] B : [1.84, 2.20]
4	A : [1.7,2.4] B : [1.7,2.4]	0.01	1.4	0.01	1500	5	128	1.445	11000	A : [1.948, 2.218] B : [1.918, 2.098]
5	A : [1.8,2.2] B : [1.8,2.2]	0.51	12	0.01	8000	5	78	0.1186	6400	A : [1.9912, 2.0037] B : [1.9824, 2.0000]

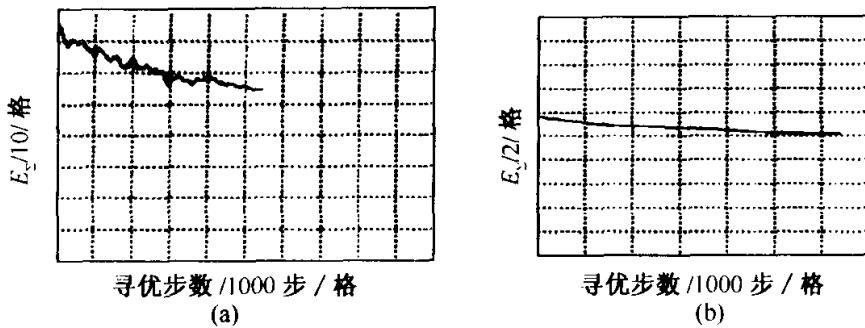


图 7.42 辨识寻优误差变化动态

由图 7.42 可见，在第五次辨识过程之后，整个蚁群的平均参数辨识误差已减至 0.33%。

## 7.9.2 蚁群算法与非线性系统参数辨识

### 7.9.2.1 问题描述

在高精度、超低速伺服系统中，由于非线性摩擦环节的存在，使系统的动态及静态性能受到很大影响，这主要表现为低速时出现爬行现象，稳态时有较大的静差或出现极限环振荡。因此，要提高系统的性能，必须采用适当的控制方法来消除摩擦力矩的影响。在基于摩擦模型的补偿方法中，选择一个合适的摩擦模型是非常重要的。实践表明，采用经典的库仑摩擦+黏性摩擦作为摩擦模型，并不能真实地反映摩擦现象的动态过程。Canudas de Wit C 等<sup>[134]</sup>在 1995 年提出的 LuGre 模型能够准确地描述摩擦过程复杂的动态、静态特性，如爬行、极限环振荡、滑前变形、摩擦记忆、变静摩擦及静态 Stribeck 曲线等，目前得到了广泛应用。

角位移伺服系统一般可用如下微分方程表示

$$u - F = m\ddot{\theta} \quad (7.9.11)$$

式中， $m$  表示转动惯量； $\theta$  表示转角； $u$  表示控制力矩； $F$  表示摩擦力矩。设状态变量  $z$  表示接触面鬃毛的平均变形<sup>[135]</sup>，则  $F$  可由如下 LuGre 模型来描述

$$F = \sigma_0 z + \sigma_1 \dot{z} + \alpha \dot{\theta} \quad (7.9.12)$$

$$\dot{z} = \dot{\theta} - \frac{\sigma_0 |\dot{\theta}|}{h(\dot{\theta})} z \quad (7.9.13)$$

式中， $\sigma_0$  表示刚度系数； $\sigma_1$  表示黏性阻尼系数； $\alpha$  表示黏性摩擦系数；而非线性摩擦特性函数  $h(\dot{\theta})$  为描述不同摩擦效应的有界正函数，其描述 Stribeck 效应的具体形式如下

$$h(\dot{\theta}) = f_c + (f_s - f_c) e^{-(\frac{\dot{\theta}}{f_s})^2} \quad (7.9.14)$$

式中,  $f_c$  表示 Coulomb 摩擦力矩;  $f_s$  表示最大静摩擦力矩;  $\dot{\theta}_s$  表示 Stribeck 速度。令  $\dot{z}=0$ , 将公式 (7.9.13) 和公式 (7.9.14) 代入公式 (7.9.12), 可得摩擦力矩与转速之间的稳态对应关系为

$$\begin{aligned} F_s &= \sigma_0 z + \alpha \dot{\theta} = \frac{h(\dot{\theta})\dot{\theta}}{|\dot{\theta}|} + \alpha \dot{\theta} \\ &= h(\dot{\theta}) \operatorname{sgn}(\dot{\theta}) + \alpha \dot{\theta} \\ &= [f_c + (f_s - f_c)e^{-(\frac{\dot{\theta}}{\dot{\theta}_s})^2}] \operatorname{sgn}(\dot{\theta}) + \alpha \dot{\theta} \end{aligned} \quad (7.9.15)$$

公式 (7.9.15) 所确定的转速-摩擦力矩曲线称为 Stribeck 曲线。

对于模型的动态、静态参数, 可分两步对其进行辨识: 首先辨识出静态参数  $f_c$ 、 $f_s$ 、 $\alpha$  和  $\dot{\theta}_s$ , 然后用得到的静态参数的估计值替代实际值, 进一步辨识出动态参数  $\sigma_0$  和  $\sigma_1$ 。

### 7.9.2.2 算法设计

#### 1. 静态参数辨识

令闭环伺服系统以一组恒定的转速  $\{\dot{\theta}\}_{i=1}^N$  运动, 得到相应的控制力矩序列  $\{u\}_{i=1}^N$ 。由公式 (7.9.11), 当  $\dot{\theta}=0$ , 有  $F=u$ 。因此,  $\{\dot{\theta}\}_{i=1}^N$  和  $\{u\}_{i=1}^N$  这两个序列确定了摩擦力矩与转速之间的稳态对应关系<sup>[136, 137]</sup>。设待辨识的参数向量为  $x_s$  为

$$x_s = [\hat{W}_1, \hat{W}_2]^T \quad (7.9.16)$$

式中,  $\hat{W}_1 = \{\hat{f}_c, \hat{f}_s, \hat{\alpha}, \hat{V}_s\} \cup \{\theta > 0\}$ ,  $\hat{W}_2 = \{\hat{f}_c, \hat{f}_s, \hat{\alpha}, \hat{V}_s\} \cup \{\theta < 0\}$ 。由此, 可定义静态辨识误差为

$$e(x_s, \dot{\theta}_i) = u_i - F_s(x_s, \dot{\theta}_i) \quad (7.9.17)$$

按下式取目标函数

$$J_s = \frac{1}{2} \sum_{i=1}^N e^2(x_s, \dot{\theta}_i) \quad (7.9.18)$$

设蚂蚁总数为  $m$ , 对于每只蚂蚁  $l$ , 定义其评价函数值为  $i$  点的目标函数  $J_{si}$  和相邻为  $j$  点的目标函数  $J_{sj}$  的差值, 并记

$$\Delta J_{sj} = J_{si} - J_{sj}, \quad \forall i, j \quad (7.9.19)$$

定义蚂蚁  $l$  的转移概率为

$$P_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\Delta J_{sj}]^\beta}{\sum_{r \in \text{allowed}_l} [\tau_{ir}]^\alpha [\Delta J_{sr}]^\beta}, & \text{若 } j \in \text{allowed}_l \\ 0, & \text{否则} \end{cases} \quad (7.9.20)$$

寻优时, 将蚂蚁按照随机原则散布在空间网格点上, 并记录具有最好评价函数值的精灵蚂蚁。然后, 按照公式 (7.9.20) 所给的空间状态转移概率移动各只

蚂蚁。在搜索过程中嵌入了邻近搜索机制，即当  $\Delta J_{Sj} > 0$  时，蚂蚁  $l$  按概率  $P_{ij}$  从其邻域  $i$  移动至邻域  $j$ ；当  $\Delta J_{Sj} \leq 0$  时，蚂蚁  $l$  进行自身的邻域搜索，以寻找更优的解。

一次循环结束时，蚂蚁所移动路径上的信息量按照下式做相应调整

$$\begin{cases} \tau_{ij}^{\text{New}} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \\ \Delta\tau_{ij} = \sum_{l=1}^m \Delta\tau_{ij}^l \end{cases} \quad (7.9.21)$$

$$\Delta\tau_{ij}^l = \begin{cases} \frac{Q}{J_s^l}, & \text{若 } l \text{ 只蚂蚁在本次循环中经过路径 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (7.9.22)$$

式中， $J_s^l$  表示第  $l$  只蚂蚁在本次循环中的目标函数计算值。上述过程持续进行，直到满足一定的结束条件为止。

## 2. 动态参数辨识

基于 PID 控制和 LuGre 摩擦补偿的闭环伺服系统结构如图 7.43 所示<sup>[122]</sup>。

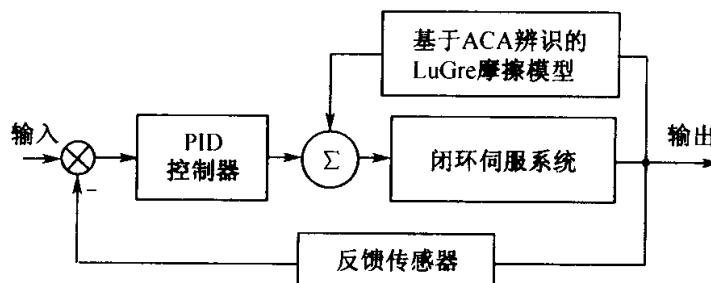


图 7.43 基于 PID 控制和 LuGre 摩擦补偿的闭环伺服系统结构

公式 (7.9.11) 给出的控制量  $u$  可通过 PID 控制器定义为

$$u = -k_p(\theta - \theta_d) - k_d\dot{\theta} - k_i \int (\theta - \theta_d) dt \quad (7.9.23)$$

设待辨识的参数向量为  $x_d = [\hat{\sigma}_0, \hat{\sigma}_1]^T$ ，同样地，可定义动态辨识误差为

$$e(x_d, t) = \theta(t) - \theta_1(x_d, t) \quad (7.9.24)$$

式中， $\theta(t)$  表示  $t$  时刻闭环伺服系统的转角输出值； $\theta_1(x_d, t)$  表示  $t$  时刻辨识参数所组成的模型系统输出值。同样地，取目标函数

$$J_D = c_1 \sum_{i=1}^N e^2(x_d, t) + c_2 \max\{|e(x_d, t)|\} \quad (7.9.25)$$

式中， $c_1, c_2$  均为权系数。定义其评价函数值为  $i$  点的目标函数  $J_{Di}$  和相邻为  $j$  点的目标函数  $J_{Dj}$  的差值，并记

$$\Delta J_{Dij} = J_{Di} - J_{Dj}, \quad \forall i, j \quad (7.9.26)$$

定义蚂蚁  $l$  的转移概率为

$$P_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\Delta J_{Dij}]^\beta}{\sum_{r \in \text{allowed}_i} [\tau_{ir}]^\alpha [\Delta J_{Dir}]^\beta}, & \text{若 } j \in \text{allowed}_i \\ 0, & \text{否则} \end{cases} \quad (7.9.27)$$

其他步骤与静态摩擦参数辨识类似。

### 7.9.2.3 仿真算例

以某型闭环伺服系统为例进行了 LuGre 摩擦参数辨识，初始化参数为： $k_p = 0.3$ ,  $k_i = 0.5$ ,  $k_d = 0.8$ ,  $m = 50$ ,  $F = 200$ ,  $\alpha = 1.2$ ,  $\beta = 2.5$ ,  $\rho = 0.6$ ，表 7.22 为蚁群算法迭代 491 次后的 LuGre 模型静态摩擦参数辨识结果。

表 7.22 某型闭环伺服系统的 LuGre 模型静态摩擦参数辨识结果

静态摩擦参数		$f_c$ (Nm)	$f_s$ (Nm)	$\alpha$ (Nms/rad)	$\beta_s$ (rad/s)
实际值	$\beta > 0$	0.32	0.47	0.05	0.03
辨识值	$\beta < 0$	0.30	0.46	0.06	0.04
实际值	$\beta > 0$	0.323	0.468	0.051	0.028
辨识值	$\beta < 0$	0.296	0.451	0.060	0.041

同样地，在静态摩擦参数辨识的基础上，可得到迭代 387 次后 LuGre 模型的动态摩擦参数辨识结果如表 7.23 所示。

表 7.23 某型闭环伺服系统的 LuGre 模型动态摩擦参数辨识结果

动态摩擦参数		$\sigma_0$ (Nm)	$\sigma_1$ (Nms/rad)
实际值		380.0	4.1
辨识值		402.62	4.58

由表 7.22 和表 7.23 可见，基于蚁群算法的静态摩擦参数和动态摩擦参数的辨识值都非常接近其实际值，从而验证了用蚁群算法进行非线性参数辨识的可行性和有效性。

## 7.10 聚类分析

聚类，又称分割，是对数据集进行分组，使类间相似性最小化，而使类内相似性最大化。虽然人类对聚类的研究始于 20 世纪 60 年代，但是聚类是一个古老的问题，它伴随着人类社会的产生和发展而不断深化，人类要认识世界，就必须区别不同的事物并认识事物间的相似性。

经典的聚类分析方法包括分层算法、 $K$  均值算法、模糊  $C$  均值算法、图论聚类法、神经网络法以及基于统计的方法等。近几年蚁群算法在聚类分析领域的应用也取得了很大的研究进展<sup>[138~145, 151~153]</sup>。将蚁群算法用于聚类分析，灵感源于蚂蚁堆积它们的尸体和分类它们的幼体。Deneubourg J L 等<sup>[146, 147]</sup>基于蚁群聚类现象建立了一种基本模型，Lumer E 和 Faieta B 将该模型推广到数据分析范畴<sup>[148]</sup>，其主要思想是将待聚类数据初始随机地散布在一个二维平面内，然后在该平面上产生一些虚拟蚂蚁对其进行聚类分析。

本节首先对一种单蚁群聚类算法做了改进；然后模仿多蚁群的协作性能，将运动速度各异的多个蚁群独立且并行地进行聚类分析，并将其聚类结果组合为超图，然后再用蚁群算法对超图进行二次划分。

## 7.10.1 算法设计

### 7.10.1.1 改进的单蚁群聚类算法

基于文献 [141]、[146]、[148] 及 [149] 中的一些研究成果，这里对单蚁群聚类算法做了一些改进。首先将数据对象随机地投影到一个平面，然后每只蚂蚁随机地选择一个数据对象，根据该对象在局部区域的相似性而得到的概率，决定蚂蚁是否“拾起”、“移动”或“放下”该对象。经过有限次迭代，平面上的数据对象按其相似性而聚集，最后得到聚类结果和聚类数目。

#### 1. 平均相似性

假设在时刻  $t$  某只蚂蚁在地点  $r$  发现一个数据对象  $o_i$ ，则可将对象  $o_i$  与其邻域对象  $o_j$  的平均相似性定义为

$$f(o_i) = \max \left\{ 0, \frac{1}{s^2} \sum_{o_j \in \text{Neigh}_{s \times s}(r)} \left[ 1 - \frac{d(o_i, o_j)}{\alpha(1 + (v-1)/v_{\max})} \right] \right\} \quad (7.10.1)$$

式中， $\alpha$  为相似性参数； $v$  表示蚂蚁运动的速度； $v_{\max}$  为最大速度； $\text{Neigh}_{s \times s}(r)$  表示地点  $r$  周围的以  $s$  为边长的正方形局部区域； $d(o_i, o_j)$  表示对象  $o_i$  和  $o_j$  在属性空间中的距离。

通常采用 Euclidean 或余弦距离函数，其定义为

$$d(o_i, o_j) = \sqrt{\sum_{k=1}^m (o_{ik} - o_{jk})^2} \quad (7.10.2)$$

式中， $m$  表示属性个数。余弦距离定义为

$$d(o_i, o_j) = 1 - \text{sim}(o_i, o_j) \quad (7.10.3)$$

式中

$$\text{sim}(o_i, o_j) = \frac{\sum_{k=1}^m (o_{ik} \cdot o_{jk})}{\sqrt{\sum_{k=1}^m (o_{ik})^2 \cdot \sum_{k=1}^m (o_{jk})^2}} \quad (7.10.4)$$

$\text{sim}(o_i, o_j)$  称作余弦相似函数，即二矢量之间的夹角的余弦。当数据对象变得越相似， $\text{sim}(o_i, o_j)$  趋近于 1；反之，则趋近于 0。

这里采用了上述两种距离函数的线性组合，原因是它们能够相互补偿。例如，当两矢量在一条直线上而又不完全相等时，余弦距离为 0，不能区分这两矢量；而 Euclidean 距离不为 0，能很好地区分它们。

在公式 (7.10.1) 中， $\alpha$  为调节数据对象间相似性的参数，同时它也决定了聚类数目和收敛速度。 $\alpha$  越大时，对象间相似程度  $f(o_i)$  越大，也许使不太相同的对象归为一类，其聚类数目越少，收敛速度也越快。反之， $\alpha$  越小，对象间相似程度越小，在极端情况下可能将一个大类分成了许多小类。同时随着聚类数目的增多，收敛速度将变慢。

蚂蚁的运动速度影响聚类的效果，运动速度快的蚂蚁能很快将对象粗略地分为大的几类，而慢速运动的蚂蚁能更精确地细分对象。因此按照蚂蚁运动速度的不同，这里设计了三种不同蚁群：

- (1)  $v$  为常数：所有蚂蚁在任何时刻以同样速度运动。
- (2)  $v$  为随机数：蚂蚁的速度为一个从 1 到  $v_{\max}$  范围内的随机数。
- (3)  $v$  为递减随机数：蚂蚁刚开始运动时的速度较快，以便迅速聚类；然后其值以随机的方式逐渐减小，以使聚类结果更为精细。

## 2. 概率转换函数

概率转换函数是  $f(o_i)$  的一个函数，它将数据对象的平均相似性转化为“拾起”概率或“放下”概率。其转换原则为：数据对象与其邻域的平均相似性越小，说明该数据对象属于此邻域的可能性越小，则“拾起”概率越高，“放下”概率越低；反之亦然。依据这一原则，选取对称式 Sigmoid 函数作为概率转换函数。Sigmoid 函数具有非线性，运算中只需调整一个参数，比 LF 算法<sup>[148]</sup>中的二次函数有更快的收敛性。

将随机运动的无负载蚂蚁“拾起”一个物体的“拾起”概率定义为

$$P_p = 1 - \text{Sigmoid}(f(o_i)) \quad (7.10.5)$$

类似地，将随机运动的有负载蚂蚁“放下”一个物体的“放下”概率定义为

$$P_d = \text{Sigmoid}(f(o_i)) \quad (7.10.6)$$

式中

$$\text{Sigmoid}(x) = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}} \quad (7.10.7)$$

为自然指数形式，参数  $c$  越大，曲线饱和越快，算法收敛速度也越快。值得注意的是，在聚类的过程中，有一些被称作孤立点的对象与其他对象均不相似，蚂蚁“拾起”它们后，很难尽快“放下”它们，从而影响算法的收敛速度。这里采用了在算法后期增大  $c$  值，尽快“放下”孤立点的策略。

### 3. 算法描述

综上所述，对单蚁群聚类算法及其改进部分做了较详细的介绍，其伪码描述如下。

1. 初始化蚁群中蚂蚁个数  $n\_ant$ , 最大迭代次数  $M$ , 局部区域边长  $s$ , 参数  $a$ ,  $c$  等。
2. 将数据对象投影到一个平面, 即给每个对象随机地分配一对坐标值  $(x, y)$ 。
3. 每只蚂蚁初始化为有负载, 并随机地选择一个对象。
4. 参数  $v$  取三种类型值之一: 常数、随机数或递减随机数。
5. For  $i=1, 2, \dots, M$   
    For  $j=1, 2, \dots, n\_ant$ 
  5. 1 根据公式(7.10.1)计算对象的平均相似性;
  5. 2 如果蚂蚁无负载, 根据公式(7.10.5)计算“拾起”概率  $P_p$ 。若  $P_p$  大于某一随机概率, 而同时该对象未被其他蚂蚁“拾起”, 则蚂蚁“拾起”该对象, 随机移往别处, 并标记自己有负载; 否则, 蚂蚁拒绝“拾起”该对象, 而随机选择其他对象;
  5. 3 如果蚂蚁为负载状态, 根据公式(7.10.6)计算“放下”概率  $P_d$ 。若  $P_d$  大于某一随机概率, 则蚂蚁“放下”该对象, 并标记自己无负载, 再重新选择一个新对象。
6. For  $i=1, 2, \dots, n$  //对所有对象而言
  6. 1 如果某个对象是孤立的或其邻域对象个数小于某一常数, 则标记该对象为孤立点;
  6. 2 否则给该对象分配一个聚类序列号, 并递归地将其邻域对象标记为同样的序列号。

#### 7.10.1.2 多蚁群聚类组合算法

##### 1. 系统结构图

图 7.44 给出了多蚁群聚类组合算法的系统结构图。

图 7.44 中的第一层为三个速度类型不同的蚁群模块, 每个模块采用改进的单蚁群聚类组合算法并行地进行聚类, 得到初步结果; 中间层为聚类组合模块, 它将初步聚类结果组合成为超图; 最后一层为图划分模块, 它采用基于蚁群算法的图划分算法对超图进行二次划分, 得到最终聚类结果。

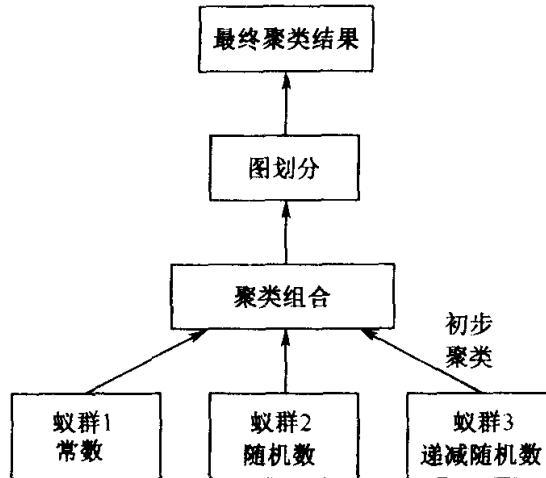


图 7.44 多蚁群聚类组合算法系统结构图

## 2. 聚类组合

Strehl A 和 Ghosh J 提出的超图模型可将已知的一组聚类结果表示成一个超图<sup>[154]</sup>。假设  $O = \{o_1, o_2, \dots, o_n\}$  表示一个对象集，这  $n$  个对象被分成  $k$  类，可表示成标记矢量  $\lambda \in I^n$ 。在  $r$  组聚类结果中，已知第  $q$  个聚类  $\lambda^{(q)}$  被分为  $k^{(q)}$  类，则可得到一个二进制成员矩阵  $H^{(q)} \in I^{n \times k^{(q)}}$ ，在该矩阵中，每个聚类被表示成一条超边（对应矩阵的列）。将这些成员矩阵组合起来，于是得到一个具有  $n$  个顶点  $\sum_{q=1}^r k^{(q)}$  条超边的超图邻接矩阵

$$H = (H(1), H(2), \dots, H(r)) \quad (7.10.8)$$

矩阵  $H$  中的每一行表示一个顶点（对象），每一列表示一条超边，属于同一超边的顶点取值为 1，否则为 0。至此，一组聚类结果已被映射成超图的邻接矩阵。

下面通过一个简单的例子来说明上述概念。表 7.24 给出了有 7 个对象  $o_i (i=1, 2, \dots, 7)$  的 3 个聚类标记矢量，其中聚类 1 和 2 逻辑上是一致的，聚

表 7.24 聚类标记矢量

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$
$o_1$	1	2	1
$o_2$	1	2	1
$o_3$	1	2	2
$o_4$	2	3	2
$o_5$	2	3	3
$o_6$	3	1	3
$o_7$	3	1	3

类3在对象3和5的分类上有些争议。将此映射成超图的邻接矩阵 $H$ 如表7.25所示，其中 $r=3$ ,  $k^{(1,2,3)}=3$ ，超图的7个顶点 $v_i$  ( $i=1, 2, \dots, 7$ ) 对应7个对象，每个聚类被表示成一条超边，共有9条超边。

表 7.25 超图之邻接矩阵

$H$	$H^{(1)}$			$H^{(2)}$			$H^{(3)}$		
$v_1$	1	0	0	0	1	0	1	0	0
$v_2$	1	0	0	0	1	0	1	0	0
$v_3$	1	0	0	0	1	0	0	1	0
$v_4$	0	1	0	0	0	1	0	1	0
$v_5$	0	1	0	0	0	1	0	0	1
$v_6$	0	0	1	1	0	0	0	0	1
$v_7$	0	0	1	1	0	0	0	0	1

接下来，可通过下式将有 $n$ 个顶点 $\sum_{q=1}^r k^{(q)}$ 条超边的超图邻接矩阵 $H$ 转换成 $n \times n$ 的对称邻接矩阵 $S$

$$S = HH^T \quad (7.10.9)$$

式中， $H^T$ 是 $H$ 的转置矩阵。 $S$ 的每一行及每一列均对应超图中的一个顶点，非对角线上值反映超边的加权值。若两个顶点属于同一超边的次数越多，则超边的加权值越大。

### 7.10.1.3 基于蚁群算法的图划分算法

对超图进行二次划分，用KLS算法尝试将基于蚁群算法的聚类LF算法<sup>[148]</sup>运用于图划分。其主要思想是将LF模型中对数据对象 $o_i$ 的操作改为对顶点 $v_i$ 的操作，公式(7.10.1)中对象间的距离 $d(o_i, o_j)$ 改为顶点间的距离 $d(v_i, v_j)$ 。由此，基于蚁群算法的图划分算法实质是通过顶点在平面上运动而动态聚类。

假设 $G(V, E)$ 表示一个有向图， $V=\{v_i, i=1, \dots, n\}$ 为顶点的集合， $E$ 为边的集合。有向图的邻接矩阵 $A=[a_{ij}]$ ，其中

$$\begin{cases} a_{ij} \neq 0, & \text{当且仅当 } (v_i, v_j) \in E \\ a_{ij} = 0, & \text{当且仅当 } (v_i, v_j) \notin E \end{cases} \quad (7.10.10)$$

可将有向图中任意两顶点间的距离 $d(v_i, v_j)$ 定义为

$$d(v_i, v_j) = \frac{|D(\rho(v_i), \rho(v_j))|}{|\rho(v_i)| + |\rho(v_j)|} \quad (7.10.11)$$

式中， $\rho(v_i)=\{v_j \in V; a_{ij} \neq 0\} \cup \{v_i\}$ 表示与顶点 $v_i$ 相邻接的所有顶点的集合，包

括  $v_i$  本身； $D$  表示两个集合间的对称差， $D(A, B) = (A \cup B) - (A \cap B)$ 。

如果两顶点  $v_i$  和  $v_j$  有大量共同相邻接的节点，即  $(\rho(v_i) \cup \rho(v_j)) - (\rho(v_i) \cap \rho(v_j))$  是一个小集合，则  $d(v_i, v_j)$  较小，亦即  $v_i$  和  $v_j$  最终将聚在一起，归为一类；反之，如果两顶点  $v_i$  和  $v_j$  间只有少量或没有相邻接的边，即  $(\rho(v_i) \cup \rho(v_j)) - (\rho(v_i) \cap \rho(v_j))$  是一个大集合，则  $d(v_i, v_j)$  较大，换言之， $v_i$  和  $v_j$  最终将相距甚远，归为不同类。

如果用有向图的邻接矩阵表示，则可将公式 (7.10.11) 简化为

$$d(v_i, v_j) = \frac{\sum_{k=1}^n |a_{ik} - a_{jk}|}{\sum_{k=1}^n |a_{ik}| + \sum_{k=1}^n |a_{jk}|} \quad (7.10.12)$$

式中， $d(v_i, v_j)$  的值在 0~1 之间变化。按照公式 (7.10.1)、公式 (7.10.5) 及公式 (7.10.6)，可计算出顶点  $v_i$  的平均相似性  $f(v_i)$ 、“拾起”概率  $P_p$  和“放下”概率  $P_d$ 。按照单蚁群聚类组合算法步骤对超图顶点进行聚类，即可得到最终的聚类结果。

## 7.10.2 仿真算例

用表 7.26 中的 4 个数据库对改进的单蚁群聚类组合算法和多蚁群聚类组合算法进行了测试。这些数据库有它们自己的分类表，可用于最终评价聚类的性能，但在聚类过程中不能应用。

表 7.26 数据库描述

领域	Iris	Zoo	Wine	Image Segmentation
数据大小	150	101	178	210
分类数	3	7	7	3
属性个数	4	17	19	13

任何聚类算法的结果都应该采用一种客观公正的质量评价方法来进行评价。一般而言，质量评价方法分为外部和内部两种，其依据是有无关于数据集的先验知识。这里介绍一种常用的外部评价方法：F-measure。

F-measure 组合了信息检索中查准率 (precision) 与查全率 (recall) 的思想。一个聚类  $j$  及与此相关的分类  $i$  的 precision 与 recall 定义为

$$P = \text{precision}(i, j) = \frac{N_{ij}}{N_i} \quad (7.10.13)$$

$$R = \text{recall}(i, j) = \frac{N_{ij}}{N_j} \quad (7.10.14)$$

式中,  $N_{ij}$  表示在聚类  $j$  中分类  $i$  的数目;  $N_j$  表示聚类  $j$  中所有对象的数目;  $N_i$  表示分类  $i$  中所有对象的数目。由此, 可将分类  $i$  的 F-measure 定义为

$$F(i) = \frac{2PR}{P+R} \quad (7.10.15)$$

对分类  $i$  而言, 哪个聚类的 F-measure 值高, 就认为该聚类代表分类  $i$  的映射, 换言之, F-measure 可看成分类  $i$  的评判分值。对于聚类结果  $K$  而言, 其总 F-measure 可由每个分类  $i$  的 F-measure 加权平均得到

$$F_\lambda = \frac{\sum_i (|i| \cdot F(i))}{\sum_i |i|} \quad (7.10.16)$$

式中,  $|i|$  表示分类  $i$  中所有对象的数目。

表 7.27 列出了对表 7.26 中 4 个数据库进行 100 次测试后的平均总 F-measure 结果。其中“蚁群 1、2、3”分别对应改进的单蚁群聚类算法速度取值为常数、随机数及递减随机数时的结果;“三蚁群平均”对应上述 3 项的平均值;“三蚁群组合”对应多蚁群聚类组合算法。

表 7.27 多蚁群聚类组合前后的平均总 F-measure

领域	Iris	Zoo	Wine	Image Segmentation
蚁群 1	0.918	0.801	0.728	0.522
蚁群 2	0.910	0.793	0.743	0.508
蚁群 3	0.915	0.809	0.737	0.493
三蚁群平均	0.914	0.801	0.736	0.508
三蚁群组合	0.927	0.802	0.742	0.501

上述实验结果表明, 一般来说, 多蚁群聚类组合性能优于单蚁群聚类算法的平均性能, 有时甚至超过单蚁群聚类算法中任意一个。例如, 对 Iris 数据集聚类的总 F-measure 来说, 在组合前分别为 0.918、0.910 和 0.915, 而组合后达到 0.927, 其组合结果超过任何一个单蚁群聚类结果。

## 7.11 数 据 挖 掘

数据挖掘是在海量的数据中寻找模式或规则的过程。随着信息分布越来越广泛以及信息量的迅猛增加, 传统的数据挖掘方法也难以胜任, 因此分布式数据挖掘技术成为了人工智能与数据库领域的研究热点。目前, 一些学者按照“局部学习, 总体结合”的思路研究了多种分布式数据挖掘算法, 其不足之处在于各数据库之间缺乏相互联系, 从而导致分析结果不太准确。近年来, 在一些文献中提出了基于蚂蚁智能体的分布式数据挖掘方法<sup>[155~157]</sup>, 即利用移动的、分布的、互相协作的蚂蚁智能体对分布式数据库进行挖掘。

本节研究了一种基于蚁群算法的分布式数据挖掘方法，该方法模拟自然界中由简单蚂蚁个体组成的群体行为，以完成分布式数据挖掘任务。

### 7.11.1 算法设计

#### 7.11.1.1 系统目标

在文献 [155] 中首次提出了基于蚁群优化的分类规则发现算法，实验结果表明：与决策数归纳分类方法相比，可以提高分类的准确率，产生更为简便的分类规则。其针对的是单一数据库，产生的分类规则形式为

$$\text{IF } <\text{条件项组合}> \text{ THEN } <\text{类型}> \quad (7.11.1)$$

其中，条件项组合用 $<\text{属性}, \text{操作符}, \text{值}>$ 表示。

例如，某一数据库的记录属性用 $A_1, A_2, \dots, A_n$ 表示，其中一条规则为

$$\text{IF } A_1 = V_{13} \text{ AND } A_2 = V_{24} \text{ THEN CLASS}_1 \quad (7.11.2)$$

这条规则中有两个条件项， $A_1 = V_{13}$  表示  $A_1$  等于其第 3 个候选数据值， $A_2 = V_{24}$  表示  $A_2$  等于其第 4 个候选数据值。在分布式环境下，分类系统的目标仍然是产生以上形式的分类规则。

#### 7.11.1.2 单一数据库挖掘的基本思路

在建立一条规则中，关键是产生某一类型的条件项。文献 [155] 的主要思想是：将条件项当作蚂蚁智能体的候选路径，按照由信息素和启发函数值所决定的可能性大小，确定选择的条件项，并将其添加到当前正在构建的规则中，其主要处理过程可参见文献 [155]。

#### 7.11.1.3 分布式群智能分类方法

分布式数据库是一组数据集，逻辑上它们属于同一系统，而物理上它们分散在用计算机网络连接的多个场地上。由于其分布性，对它们的管理及处理难以集中控制，分散与集中是其主要管理特色<sup>[158]</sup>。

在文献 [155] 所提出的算法中，数据源只有一个，蚂蚁智能体是一个接一个地构建一条规则的。但在分布式环境下，存在多个数据源，因此，需采用不同的处理过程，其系统结构如图 7.45 所示。

假设存在  $n$  个结构相同的数据库，分布在  $n$  个场地。按照此结构，提出了如下蚁群优化分布式分类算法：

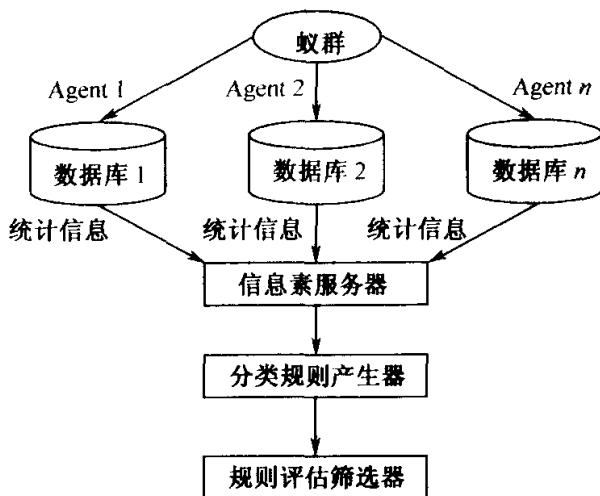


图 7.45 分布式分类挖掘系统结构

```

训练集 =  $n$  个场地数据库实例;
While (存在任一训练集中的实例数目  $> \text{max\_uncovered\_cases}$ )
    初始化信息素;
    For  $n$  agents in parallel
        统计各场地有关信息;
    End for
    计算启发函数值;
     $i = 0$ ;
    Repeat
         $i = i + n$ ;
        For  $n$  agents in parallel
            计算相关函数值;
            构建一条规则;
            更新刚建立的规则;
            修改 agents 所爬路径的信息素
        End for
        Until ( $i \geq \text{No\_of\_agents}$ ) or (Agenti 与 No_Rules_Converg-1 agent 构建的规则相同)
            从 agents 所构建的规则中选择一条质量最好的;
            从各场地训练集中移走满足此条新规则条件的实例
    End while

```

上述算法中存在如下参数：

- (1) max\_uncovered\_case: 终止条件, 即训练集中最多剩下的实例数。
- (2) No\_of\_Agents: 代理的数目。

(3) No\_Rules\_Converg: 判断收敛到同一条规则的代理数。

### 1. 各数据库统计的信息

- (1)  $k$ : 所有实例的类型数目。
- (2)  $\text{term}_{ij}$ , 即  $A_i = V_{ij}$ 。
- (3)  $|T_{ij}|$ : 满足  $\text{term}_{ij}$  (即  $A_i = V_{ij}$ ) 的实例数。
- (4)  $\text{freq}T_{ij}^w$ : 满足  $\text{term}_{ij}$  (即  $A_i = V_{ij}$ ), 且类型为  $w$  的实例数。
- (5)  $a$ : 是数据库中属性 (不包括类属性) 总数。
- (6)  $b_i$ : 属性  $i$  所有可能取值的数目。
- (7)  $\tau_{ij}(t)$ : 表示条件项  $\text{term}_{ij}$  在时间  $t$  的信息素总量, 其初始值为

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^a b_i} \quad (7.11.3)$$

### 2. 汇总各数据库统计信息

信息素服务器及相关计算信息素服务器汇总各数据库分别统计出的  $\text{freq}T_{ij}^w$ ,  $\tau_{ij}(t)$ , 并计算如下信息:

- (1)  $\eta_{ij}$  表示条件项  $\text{term}_{ij}$  基于密度的启发函数值, 可按下式进行计算

$$\eta_{ij} = \frac{\max(\sum_{i=1}^n \text{freq}T_{ij}^1, \sum_{i=1}^n \text{freq}T_{ij}^2, \dots, \sum_{i=1}^n \text{freq}T_{ij}^k)}{\sum_{i=1}^n |T_{ij}|} \quad (7.11.4)$$

- (2)  $R_{ij}(t)$  表示条件项  $\text{term}_{ij}$  在时间  $t$  的相关函数值, 其值按下式计算

$$R_{ij}(t) = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{i=1}^a \sum_{j=1}^{b_i} \tau_{ij}(t) \cdot \eta_{ij}}, \quad \forall i \in I \quad (7.11.5)$$

式中,  $I$  表示当前规则中尚未出现的属性。

- (3) 平均密度按下式确定

$$\theta = \frac{\sum_{i=1}^a \sum_{j=1}^{b_i} \eta_{ij}}{\sum_{i=1}^a b_i} \quad (7.11.6)$$

式中,  $\theta$  表示蚂蚁智能体选择条件项的阈值。

### 3. 分类规则的构建

在建立一条规则中, 关键是产生某一类型的条件项。一个条件项  $\text{term}_{ij}$  添加到当前规则中的可能性由下式决定

$$P_{ij}(t) = \frac{R_{ij}(t)^2}{R_{ij}(t)^2 + \theta^2} \quad (7.11.7)$$

#### 4. 信息素的修改

当  $n$  只蚂蚁构建完一条规则后，可根据公式 (7.11.8) 更改此规则的条件项信息素，而其他不在此规则中出现的条件项信息素仅进行规范化处理（即除以所有条件项总和）。

$$\tau_{ij}(t) = (1 - \rho) \cdot \tau_{ij}(t-1) + \left(1 - \frac{1}{1+Q}\right) \cdot \tau_{ij}(t-1) \quad (7.11.8)$$

式中， $Q$  表示所构建的规则质量。

#### 5. 规则质量评估及筛选

衡量分类规则质量的标准如公式 (7.11.9) 所示。经过若干蚂蚁智能体对同一训练集的挖掘后，产生了若干条分类规则，从中挑选质量最好的加入到最终分类规则集中。

$$Q = \left( \frac{\text{TruePos}}{\text{TruePos} + \text{FalseNeg}} \right) \cdot \left( \frac{\text{TrueNeg}}{\text{FalsePos} + \text{TrueNeg}} \right) \quad (7.11.9)$$

式中， $\text{TruePos}$  是满足规则条件，并且与规则的预测类型相同的实例个数； $\text{FalsePos}$  是满足规则条件，并且与规则的预测类型不同的实例个数； $\text{FalseNeg}$  是不满足规则条件，并且与规则的预测类型相同的实例个数； $\text{TrueNeg}$  是不满足规则条件，并且与规则的预测类型不同的实例个数。

### 7.11.2 仿真算例

从 UCI 公共数据库中选取了两组数据集 (Wisconsin Breast Cancer 和 Tic\\_ tac\\_ toe Endgame)，将每个数据集分为 10 个部分，假设它们分别分布在 10 个不同的数据库中，选择其中一个部分作为测试集，而其余 9 个部分作为训练集。实验参数取值如下：

- $\text{max\_uncovered\_case}=10$ ;
- $\text{No\_of\_agents}=1800$ ;
- $\text{No\_Rules\_Converg}=10$ ;
- $\rho=0.1$ 。

模拟实验通过使用训练集产生的规则对测试集进行实例进行分类，Wisconsin Breast Cancer 数据集的分类准确率为 93%，Tic\\_ tac\\_ toe Endgame 数据集的分类准确率为 71%，与用文献 [155] 中算法对单一数据库进行分类的实验结果非常接近。

## 7.12 图像处理

图像处理是指应用一系列方法获取、校正、增强、变换或压缩可视图像的技术，其目的在于提高信息的相对质量，以便提取信息，图像处理是当今计算机科学领域的一个研究热点。图像分割是图像处理领域的一个重要研究内容，但由于背景的复杂性、目标特征的多样性以及噪声等影响，从而使图像分割成为图像处理中的一个技术难点。

传统图像分割方法（如阈值法、边缘检测法、数学形态学法、区域处理法等）是目前应用比较广泛的方法。但是针对不同的应用目的以及不同的图像特性，传统方法又表现出很大的局限性。例如阈值法具有较高的计算效率，但是对噪声很敏感；边缘检测法存在边界不连续或边界不准确的问题；数学形态学法在一定程度上降低了噪声对图像的影响，但是开、闭、腐蚀、膨胀等运算会导致图像的过度平滑，从而导致图像变形及细节丢失。

蚁群算法的离散性和并行性特点对于数字图像处理非常适用，近几年许多国内外研究者应用蚁群算法在图像分割、图像特征提取、图像匹配、影像纹理分类等领域取得了相当丰富的研究成果<sup>[159~170]</sup>。

本节设计了一种基于蚁群算法的图像分割方法，该方法将图像分割看作对具有不同特征的像素进行聚类的过程，然后综合考虑图像中每个像素的灰度、梯度和邻域等特征，利用改进蚁群算法的模糊聚类能力，求出像素分别对目标、边界、背景及噪声的隶属度。

### 7.12.1 算法设计

图像内容一般包括目标、背景、边界和噪声等，特征提取的目的是找出体现图像内容之间区别的特征量，这对于后继的分类过程至关重要。区别目标和背景的一个重要特征是像素灰度，因此通常选用像素的灰度值作为聚类的一个特征<sup>[169]</sup>。另外，边界点或噪声点往往是灰度发生突变的地方，而该点处的梯度体现出这种变化，是反映边界点与背景或目标区域内点区别的重要特征。对于梯度值较高的边界点和噪声点，可利用像素的  $3 \times 3$  邻域进行区分。在一幅图像中，与区域内点灰度值相近的  $3 \times 3$  邻域的像素个数一般为 8，与边界点灰度值相近的  $3 \times 3$  邻域像素个数一般大于或等于 6，而对于噪声点，该数值一般小于 4。邻域特征的提取方法为：将当前像素和邻域像素的灰度差与灰度差阈值  $T$  做比较，小于该阈值的邻域像素个数即所要提取的邻域特征。 $T$  的设置根据图像的特点而变化，对于细节较多的图像取值较大，平滑图像取值较小，一般取值范围为 50~90。

上述三个特征反映了图像处理中目标、背景、边界和噪声的特点，这样每只蚂蚁成为一个以灰度（gray value）、梯度（gratitude）和邻域（neighbor）为特征的三维向量。

给定原始图像  $X$ ，将每个像素  $X_j$  ( $j=1, 2, \dots, N$ ) 看做一只蚂蚁，则可根据上述方法进行特征提取。每只蚂蚁是以灰度、梯度和邻域为特征的三维向量，图像分割就是这些具有不同特征的蚂蚁搜索食物源的过程。任意像素  $X_i$  到  $X_j$  的距离为  $d_{ij}$ ，可采用 Euclidean 距离对其进行计算

$$d_{ij} = \sqrt{\sum_{k=1}^m p_k (x_{ik} - x_{jk})^2} \quad (7.12.1)$$

式中， $m$  为蚂蚁数目，这里取  $m$  为 3； $p$  为加权因子，其值根据像素各分量对聚类的影响程度设定。设  $r$  为聚类半径， $ph_{ij}$  为信息量，则有

$$ph_{ij} = \begin{cases} 1, & \text{若 } d_{ij} \leq r \\ 0, & \text{否则} \end{cases} \quad (7.12.2)$$

$X_i$  选择到  $X_j$  路径的概率  $p_{ij}$  为

$$p_{ij} = \begin{cases} \frac{ph_{ij}^\alpha(t) \eta_{ij}^\beta(t)}{\sum_{s \in S} ph_{is}^\alpha(t) \eta_{is}^\beta(t)}, & \text{若 } j \in S \\ 0, & \text{否则} \end{cases} \quad (7.12.3)$$

式中， $S = \{X_s \mid d_{si} \leq r, s=1, 2, \dots, N\}$  为可行路径集合。经过一次循环，各路径上信息量可根据下式进行调整

$$ph_{ij}(t+1) = (1 - \rho) ph_{ij}(t) + \Delta ph_{ij} \quad (7.12.4)$$

$$\Delta ph_{ij} = \sum_{k=1}^N \Delta ph_{ij}^k \quad (7.12.5)$$

蚁群算法中，蚂蚁行走是随机和盲目的，将图像的每个像素看做一只蚂蚁，假设图像大小为  $m \times n$ ，在循环搜索过程中，每个像素要和其余  $m \times n - 1$  个像素进行距离和路径选择概率计算，而且系统必须经过多次循环才能完成聚类过程，导致搜索时间长，整体计算量大。针对这一问题，这里根据图像分割特点给出初始聚类中心加以引导，以减少蚂蚁行走的盲目性，并将蚂蚁与聚类中心的相似度作为引导函数，这样可降低计算量，从而加快聚类进程。

### 7.12.1.1 初始聚类中心设置

对图像特点进行分析，找出这些内容所对应的大致灰度、梯度及邻域特征作为初始聚类中心的特征，可使蚂蚁直接与聚类中心进行距离和概率计算，减少大量无关计算，从而加快聚类进程。

#### 1. 初始聚类中心灰度计算

图像的灰度直方图体现了不同灰度级像素出现的频数，很大程度上反映了灰

度聚类的结果。以原始图像的灰度直方图为基础，选择灰度直方图的  $n$  个峰值点作为聚类中心的灰度特征，同时  $n$  也决定了初始聚类中心的个数。这样，可将所有像素之间的大量循环计算转化为像素与少数几个峰值点之间的比较，引导蚂蚁快速趋于聚类中心附近，减少搜索过程，极大地降低了计算量。聚类中心  $C$  的第一个特征向量  $V$  可以确定。

## 2. 初始聚类中心梯度计算

图像中背景和目标内部像素的梯度一般较小，而边界点和噪声点梯度较大；同时背景和目标内部像素占多数，边界点像素个数又远大于噪声点的像素个数。所以，根据原始图像的灰度直方图及梯度图像，在确定的  $n$  个聚类中心中，如果某些聚类中心的灰度特征对应的像素个数远大于其他像素个数，则该聚类中心极可能在背景或目标内部，设置该聚类中心梯度特征为零，而对其余的聚类中心，使其梯度值为梯度图像最大梯度列的均值。

## 3. 初始聚类中心邻域特征计算

根据图像中不同种类像素邻域特点，将梯度为零的聚类中心邻域特征设为 8。梯度值较高的聚类中心，如果灰度特征对应的像素个数较多，则该聚类中心可能为边界，邻域特征设为 6；如果灰度特征对应的像素个数较少，则该聚类中心可能为噪声，邻域特征设为 3。

这样，所选定初始聚类中心表示为  $C_j(V, G, N_e)$ ，其中  $j=1, 2, \dots, n$ ，大致代表了各个种类的特征。

### 7.12.1.2 引导函数设置

随着蚂蚁的移动，各路径上信息量发生变化，经过一次循环引导函数体现像素与聚类中心的相似度，用如下公式表示

$$\eta_{ij} = \frac{1}{d_{ij}} = \frac{r}{\sqrt{\sum_{k=1}^m p_k (x_{ik} - c_{jk})^2}} \quad (7.12.6)$$

式中， $r$  表示聚类半径，聚类半径越大，引导函数值越大，则选择该聚类中心的概率随之增大；反之亦然。

### 7.12.1.3 算法流程

- (1) 初始化  $\alpha$ 、 $\beta$ 、 $ph_{ij}$ 、 $r$ 、 $\lambda$  等参数。
- (2) 根据公式 (7.12.1)，计算像素  $X_i$  到不同食物源  $C_j$  的距离  $d_{ij}$ 。如果  $d_{ij}$  为零，则该像素到该类的隶属度为 1，否则，如果  $d_{ij} < r$ ，根据公式

(7.12.7) 计算引导函数，并根据公式 (7.12.4) 计算  $X_i$  到各路径的信息量。

(3) 根据公式 (7.12.3)，计算像素的隶属度，判断隶属度是否大于  $\lambda$ 。若大于  $\lambda$ ，则根据公式 (7.12.5)，计算信息量增量  $\Delta ph_{ij}$ ，更新信息量，并按下式更新第  $j$  类聚类中心，其中  $J$  表示  $C_j$  类中元素个数。

$$\bar{C}_j = \frac{1}{J} \sum_{k=1}^J X_k \quad (7.12.7)$$

否则，将该蚂蚁记录到 SS 集中，SS 为没有被归类的像素集合。

(4) 计算各类的类间距离，当类间距小于阈值  $\epsilon$  时，将两类合并为一类，更新聚类中心。

(5) 如果还有待分类像素，则跳转到第 (2) 步，否则结束计算并输出结果。

## 7.12.2 仿真算例

原始图像为 256 色图，大小为  $224 \times 323$ ，如图 7.46 所示；图 7.47 和图 7.48 分别为用 Sobel 算子和 Canny 算子进行边缘检测的结果；图 7.49 和图 7.50 分别是当参数  $\alpha=1$ ,  $\beta=1$ ,  $r=50$ ,  $\lambda=0.9$ ,  $T=80$  时用基本蚁群算法和改进蚁群算法进行分割的结果。



图 7.46 原始图像



图 7.47 Sobel 算子的边缘检测结果



图 7.48 Canny 算子的边缘检测结果



图 7.49 基本蚁群算法的图像分割结果

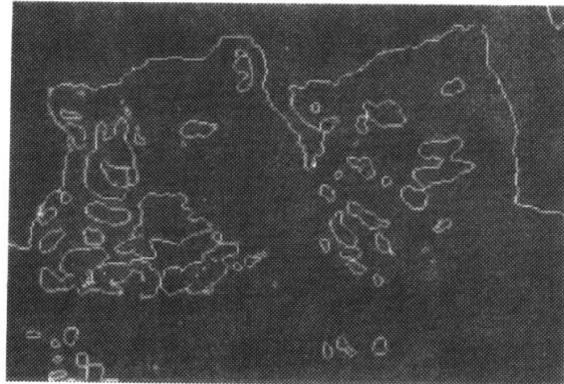


图 7.50 改进蚁群算法的图像分割结果

由图 7.46 可见，两只老虎右侧灰度较低，且图像纹理很多，因而检测存在很大的难度；由图 7.47 可见，Sobel 算子对于灰度较低的部分没有检测出来；由图 7.48 可见，Canny 算子效果很好，但是图像细节太多；而由图 7.49 和图 7.50 可以看出灰度较低部分被检测出来，分割结果比较准确。另外，基本蚁群算法完成聚类过程必须多次循环，耗时在 10 分钟以上，而改进蚁群算法仅用一次循环就完成了图像分割，用时仅 26.22s，与基本蚁群算法相比，极大降低了计算量，加速了进化进程。因此，改进蚁群算法是一种有效的图像分割方法。

## 7.13 航迹规划

航迹规划是指在特定的约束条件下，寻找运动体从初始点到目标点满足某种性能指标最优的运动轨迹<sup>[171]</sup>。而飞行器航迹规划就是指在综合考虑飞行器（如飞机、巡航导弹等）机动性能、突防概率、碰地概率和飞行时间等约束因素下，寻找一条从起始点到目标点的最优或可行的飞行轨迹。在防空技术日益先进、防空体系日益完善的现代战争中，航迹规划是提高飞行器作战效能、实施远程精确打击的有效手段。

在金飞虎等<sup>[82]</sup>对基于蚁群算法的自由飞行空间机器人路径规划研究的基础上，叶文等<sup>[172]</sup>提出了基于蚁群算法搜索最优（次优）飞机飞行航路的策略，该策略将蚁群算法进行了适当改进，使之适用于飞机低空突防航迹规划，经过蚁群的协同工作，最终找到一条优化航路；柳长安等<sup>[173, 174]</sup>提出了一种基于蚁群算法的无人机航迹规划方法，该方法能保证在航路制订时得到一条具有较小可探测概率及可接受航程的飞行航路。

本节以无人机为研究背景，研究了一种基于蚁群算法的航迹规划方法，并通过仿真实验验证了蚁群算法在该领域应用的可行性和有效性。

### 7.13.1 问题描述

如果把无人机进入敌方防御区域的方位作为起始点，而把战役/战术攻击目标所处方位定为目标点，通过对规划空间进行网格划分形成连接起始点和目标点的网络图，则寻求优化航路问题的本质就是路径优化问题。这种方法是一种确定性状态空间搜索方法，可以减小规划空间的规模，降低了航迹规划的难度<sup>[175]</sup>。假设无人机在执行任务过程中保持高度不变、速度不变，而且考虑敌方防御区处于平坦地域，那么无人机就无法利用地形因素进行威胁回避机动，航迹规划问题就可被简化成为一个二维航路（即水平航路）规划问题<sup>[176]</sup>。由于水平航迹规划仍需考虑无人机在执行作战任务过程中的生存性和有效性，并且考虑规划算法的实时性，所以是一个较为特殊的优化问题。

这里采用低于某一探测性指标，而且具有可接受航程的航路作为任务航路，按最短航路和最小可探测性航路加权方法计算代价函数作为描述航路的性能指标

$$\min W = \int_0^L [kw_t + (1-k)w_f] dt \quad (7.13.1)$$

式中， $L$  表示航路的长度； $W$  表示广义代价函数； $w_t$  表示航路的威胁代价； $w_f$  表示航路的油耗代价；系数  $k$  表示根据任务安排所做出的倾向性选择。油耗代价是航程的函数；威胁代价与无人机的可探测性指标相关联；而可探测性指标是根据无人机的雷达可探测概率计算的。

在对节点进行搜索过程中，依据公式 (7.13.1) 计算网络图中每条边的代价权值，以第  $i$  条边为例，有

$$w_i = kw_{t,i} + (1-k)w_{f,i} \quad (0 \leq k \leq 1) \quad (7.13.2)$$

式中， $w_i$  表示第  $i$  条边的广义代价； $w_{t,i}$  表示第  $i$  条边的威胁代价； $w_{f,i}$  表示第  $i$  条边的油耗代价。在计算中，系数  $k$  可取为 0.9。

这里可认为敌方防御区域内的各个雷达均相同且无相互联系，并对雷达威胁模型进行了简化处理，认为雷达信号正比于  $1/d^4$  ( $d$  是无人机到敌方雷达、导弹威胁阵地的距离)，故当无人机沿网络图的第  $i$  条边飞行时，两节点间的威胁代价可近似地认为正比于  $1/d^4$  沿这条边的积分，这里简单地把该条边划分为五段进行计算（见图 7.51），故有如下公式

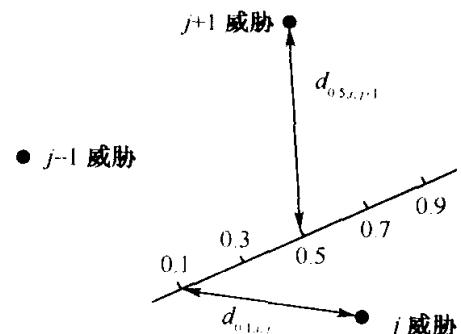


图 7.51 威胁代价的计算

$$w_{t,i} = L_i \sum_{j=1}^N \left( \frac{1}{d_{0,1,i,j}^4} + \frac{1}{d_{0,3,i,j}^4} + \frac{1}{d_{0,5,i,j}^4} + \frac{1}{d_{0,7,i,j}^4} + \frac{1}{d_{0,9,i,j}^4} \right) \quad (7.13.3)$$

式中,  $L_i$  表示第  $i$  条边的长度;  $N$  表示雷达、导弹等威胁阵地的数目;  $d_{0,1,i,j}$  等表示第  $i$  条边的  $1/10$  处距第  $j$  个威胁点的距离。另外, 在速度一定情况下, 可简单认为  $w_i = L_i$ , 则有  $w_{f,i} = L_i$ 。

### 7.13.2 算法设计

在模拟蚂蚁觅食行为来求解航迹规划问题时, 将  $m$  只蚂蚁定位于起始点, 每只蚂蚁使用一定的状态转换规则从一个状态转到另一个状态, 直到最终到达目标点, 完成一条候选航路(航迹规划问题的一个可行解)。当所有  $m$  只蚂蚁都完成了各自的候选航路选择后, 再利用信息素更新规则、当前  $m$  条候选航路以及历史上得到的一条代价最小的候选航路信息更新网络图中各条边的信息量。这一更新过程可引导蚂蚁搜索到航迹规划问题的最优解。

蚂蚁的状态转移规则类同于基本蚁群算法, 这里不再赘述。一旦所有蚂蚁完成了各自候选航路的选择过程(找到一条航迹规划问题的可行解), 必须对各边上的信息素做一次全面的更新, 其更新规则如下

$$\tau(r,s) = (1 - \rho)\tau(r,s) + \rho[\Delta\tau(r,s) + e \cdot \Delta\tau^e(r,s)] \quad (7.13.4)$$

$$\Delta\tau(r,s) = \sum_{k=1}^m \Delta\tau^k(r,s) \quad (7.13.5)$$

$$\Delta\tau^k(r,s) = \begin{cases} \frac{Q}{W_k}, & \text{若蚂蚁 } k \text{ 经过航路 } (r,s) \\ 0, & \text{否则} \end{cases} \quad (7.13.6)$$

$$\Delta\tau^k(r,s) = \begin{cases} \frac{Q}{W_e}, & \text{若航路 } (r,s) \text{ 属于最好航路} \\ 0, & \text{否则} \end{cases} \quad (7.13.7)$$

式中,  $W_k$  表示蚂蚁  $k$  选择的航路的广义代价; 而  $W_e$  表示当前最小的航路代价。

信息素更新的目的是分配更多的信息素到具有更小威胁代价航路的边上。

### 7.13.3 仿真算例

图 7.52 描述了无人机的任务示意图。

在图 7.52 中, 敌方阵地大小为  $60\text{km} \times 60\text{km}$ , 其中三角形表示无人机, 矩形方块表示目标点, 圆点表示雷达、导弹等威胁阵地, 其具体方位表 7.28 所示。

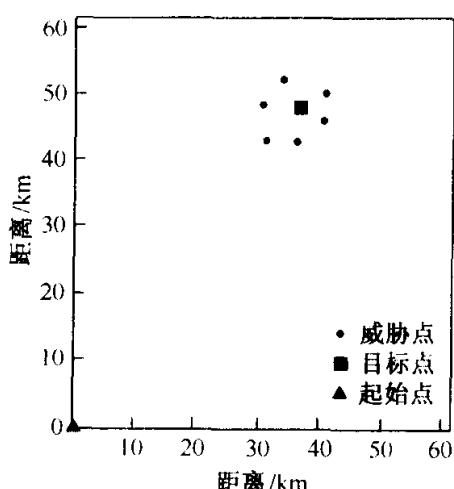


图 7.52 任务示意图

表 7.28 威胁点及起始点和目标点的坐标方位

起始点坐标	(0, 0)					
目标点坐标	(36, 48)					
威胁点坐标	编 号	坐 标	编 号	坐 标	编 号	坐 标
	1	(33.5, 52)	2	(31, 43)	3	(40.5, 50)
	4	(30, 48)	5	(40, 46)	6	(36, 43)

进入敌方防御区域后，无人机需要根据自身所处的威胁环境完成航路优化计算。为了制订、规划出一条连接起始点及目标点的航路，这里以起始点为原点进行网格划分，将敌方区域划分为  $2\text{km} \times 2\text{km}$  的正交网络图，则各节点即为无人机的可行途经节点，其坐标也随之确定。

为了确定无人机依次经过的可行节点，这里采用上述基于蚁群算法的航迹规划方法，利用仿真技术对无人机任务航路进行了规划，设置  $\alpha = 1.0$ ,  $\beta = 1.1$ ,  $e = 2.0$ ,  $\rho = 0.5$ ,  $Q = 15.0$ ,  $m = 50$ 。在其中一步转移过程中蚂蚁是这样搜索各自路线的：当前蚂蚁  $k$  所处节点坐标为  $(28, 42)$ ，如图 7.53 中星号所示。可供选择的可行节点有 4 个，分别为  $(28, 44)$ 、 $(30, 44)$ 、 $(30, 42)$  和  $(30, 40)$ ，如图 7.53 中加号所示。依据公式 (7.13.2) 可计算出连接这 4 个节点的边的代价分别是  $0.224$ 、 $0.302$ 、 $0.841$ 、 $0.481$ ，此时刻各条边的信息量为  $0.01$ 、 $0.015$ 、 $0.011$ 、 $0.013$ ，根据状态转移概率可计算出由该节点出发的各条边的选择概率分别是  $0.345$ 、 $0.373$ 、 $0.088$ 、 $0.194$ 。蚂蚁  $k$  会依据各条边的选择概率从这 4 个节点中选择该节点的后续节点，其坐标为  $(28, 44)$ ，如图 7.53 中圆圈所示。

依此类推，最终得到的优化航路如图 7.54 所示。

由图 7.54 可见，飞行航路尽可能远离威胁点，避开了敌方防空阵地。

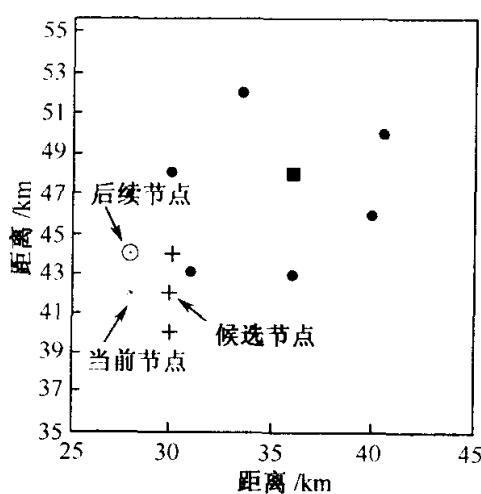


图 7.53 后续节点的选择

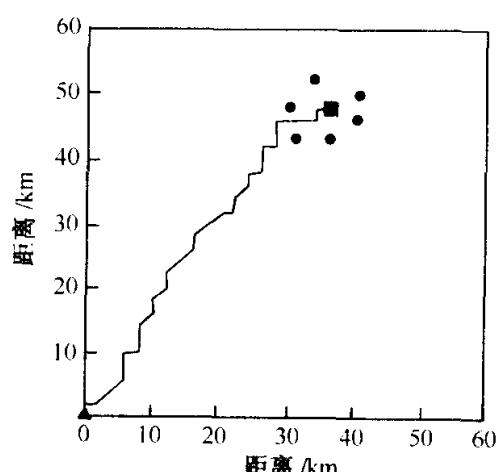


图 7.54 优化航路

## 7.14 空战决策

现代空战通常是指在超视距条件下机群与机群之间的对抗，而协同多目标攻击是机群之间进行攻击时采用的最重要的空战战术之一。要实施空战协同多目标攻击，必须进行协同多目标攻击的空战决策（以下简称空战决策），其目的是寻找一个友机对敌机的合适分配以达到最佳的攻击效果。空战决策问题是军事指挥决策理论中的一个重要研究内容，目前已成为现代战机实现超视距空战的关键技术之一，这在现代战争中十分重要，因此对其进行研究具有非常重要的战略意义。现已证明空战决策是 NP-C 问题<sup>[177]</sup>，至今仍有许多问题尚未解决。

本节提出了一种新型的启发式蚁群算法（heuristic ant colony algorithm, HACA），并将其成功应用于空战决策问题的求解。这里的启发式是指根据协同多目标攻击战术特定的启发知识而设计的一种局部搜索机制<sup>[178]</sup>，将其与蚁群算法融合可以提高蚁群算法对全局最优解的搜索速度。

### 7.14.1 问题描述

#### 7.14.1.1 空战态势

空战态势是空战决策的基础。设由  $M$  架蓝机组成的机群与由  $N$  架红机组成的机群进行空战，蓝机机群采用协同多目标攻击战术对红机机群进行攻击，所有蓝机与红机分别具有各自相同的性能，蓝机机群记为  $B$ ，且  $B = \{i, i=1, 2, \dots, M\}$ ；红机机群记为  $R$ ，且  $R = \{j, j=1, 2, \dots, N\}$ 。此处用  $B_i$  表示蓝机  $i$ ，用  $R_j$  表示红机  $j$ ，则蓝机  $i$  与红机  $j$  的相对态势如图 7.55 所示。

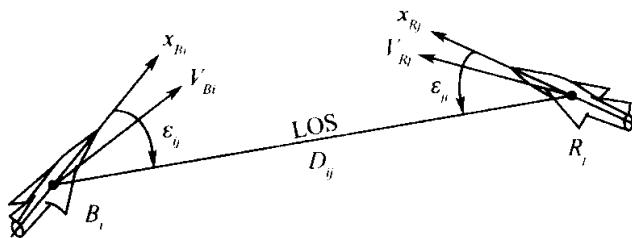


图 7.55 蓝机  $i$  与红机  $j$  的相对空战态势

图 7.55 中，蓝机  $i$  到红机  $j$  的连线 LOS 表示视线； $D_{ij}$  表示红机  $j$  相对蓝机  $i$  的距离； $x_{B_i}$  与  $V_{B_i}$  分别表示蓝机  $i$  的机体轴线和速度； $\epsilon_{ij}$  表示红机  $j$  相对蓝机  $i$  的离轴角； $x_{R_j}$ 、 $V_{R_j}$  和  $\epsilon_{ji}$  可用类似方法定义。

蓝机  $i$  对红机  $j$  的威胁可用如下组合函数表示<sup>[179, 180]</sup>

$$th_{ij} = w_1 th_{ij}^{D_{ij}} th_{ij}^{\epsilon_{ij}} + w_2 th_{ij}^{V_B} \quad (7.14.1)$$

$$th_{ij}^{D_{ij}} = \begin{cases} 1, & \text{若 } D_{ij} \leq Ra_B \\ 1 - \frac{D_{ij} - Ra_B}{Tr_B - Ra_B}, & \text{若 } Ra_B < D_{ij} \leq Tr_B \\ 0, & \text{若 } D_{ij} > Tr_B \end{cases} \quad (7.14.2)$$

$$th_{ij}^{\epsilon_{ij}} = e^{-\lambda_1 (\pi \epsilon_{ij} / 180)^{\lambda_2}} \quad (7.14.3)$$

$$th_{ij}^{V_B} = \begin{cases} 1, & \text{若 } V_{Rj} < 0.5 V_B \\ 1.5 - \frac{V_{Rj}}{V_B}, & \text{若 } 0.5 V_B \leq V_{Rj} \leq 1.4 V_B \\ 0.1, & \text{若 } V_{Rj} > 1.4 V_B \end{cases} \quad (7.14.4)$$

在公式 (7.14.1) ~ (7.14.4) 中,  $w_1$ 、 $w_2$  是非负的权重系数, 且满足  $w_1 + w_2 = 1$ ;  $th_{ij}^{D_{ij}}$  表示距离威胁因子;  $th_{ij}^{\epsilon_{ij}}$  表示角度威胁因子;  $th_{ij}^{V_B}$  表示速度威胁因子;  $Ra_B$  表示蓝机导弹的有效作用距离;  $Tr_B$  表示蓝机雷达的最大跟踪距离;  $\lambda_1$  与  $\lambda_2$  为正常数。

由公式 (7.14.2) ~ (7.14.4) 可见,  $th_{ij}^{D_{ij}} \in [0, 1]$ ,  $th_{ij}^{\epsilon_{ij}} \in (0, 1]$  及  $th_{ij}^{V_B} \in [0.1, 1]$ , 故有  $th_{ij} \in [0, 1]$ 。因此, 可将蓝机  $i$  对红机  $j$  的威胁看作蓝机  $i$  发射一枚导弹对红机  $j$  的摧毁概率。红机  $j$  对蓝机  $i$  的威胁  $th_{ji}$  用类似方法确定, 且有  $th_{ji} \in [0, 1]$ 。

### 7.14.1.2 协同多目标攻击空战决策模型

空战决策的过程是要寻找一个合适的  $M$  架蓝机对  $N$  架红机的分配方案, 以达到最佳攻击效果。设每架蓝机携带  $L$  枚导弹, 并具有多目标攻击能力, 即每架蓝机可一次发射其所携带的  $L$  枚导弹同时攻击  $L$  个不同的目标。由此可得机群  $B$  的导弹总数  $Z$  为

$$Z = M \cdot L \quad (7.14.5)$$

由于空战中机群  $B$  的导弹数量所限, 设其总数满足

$$N \leq Z \leq 2N \quad (7.14.6)$$

设机群  $B$  的  $Z$  枚导弹构成导弹集合  $G$ , 且  $G = \{r, r=1, 2, \dots, Z\}$ , 定义  $G$  的第  $r$  枚导弹为蓝机  $i$  的第  $h$  枚导弹, 则其对应关系为

$$r = (i-1) \cdot L + h, \quad h = 1, 2, \dots, L \quad (7.14.7)$$

在空战决策中, 假设:

(1) 一个目标分配 1~2 枚导弹, 对威胁较大的目标允许分配两枚导弹同时进行攻击。

(2)  $Z$  枚导弹均须分配目标攻击, 用  $th_{nj}$  表示导弹  $r$  攻击红机  $j$  的杀伤率。若将导弹  $r$  分配给红机  $j$ , 则红机  $j$  以概率  $1 - th_{nj}$  生存。在进行一次协同攻击

后, 红机  $j$  对蓝机  $i$  的期望剩余威胁为  $th_{ji} \cdot \prod_{r=1}^Z (1 - th_{ri})^{X_{rj}}$ 。

空战决策问题首先要寻找合适的导弹目标分配方案  $\pi$ , 以使如下机群  $R$  的总期望剩余威胁评估函数值最小

$$E(\pi) = \sum_{j=1}^N \sum_{i=1}^M \left[ th_{ji} \cdot \left( \prod_{r=1}^Z (1 - th_{ri})^{X_{rj}} \right) \right] \quad (7.14.8)$$

式中,  $X_{rj}$  的值为 1 或 0,  $X_{rj}=1$  表示导弹  $r$  攻击红机  $j$ 。考虑上述两条假设, 有

$$\sum_{j=1}^N X_{rj} = 1, \quad r = 1, 2, \dots, Z \quad (7.14.9)$$

$$\sum_{r=1}^Z X_{rj} = 1 \text{ 或 } 2, \quad j = 1, 2, \dots, N \quad (7.14.10)$$

定义  $\pi$  是一个可行的导弹目标分配矢量, 可将其表示为  $\pi = (T_1, T_2, \dots, T_r, \dots, T_Z)$ 。 $\pi(r) = T_r$  表示导弹  $r$  攻击目标  $T_r$ ,  $T_r \in R$ ,  $r=1, 2, \dots, Z$ 。由公式 (7.14.9) 和公式 (7.14.10) 可知在一个方案  $\pi$  中, 所有目标最少出现一次, 同一目标最多出现两次。

在求得最优导弹目标分配方案  $\pi$  后, 则可根据公式 (7.14.7) 唯一地确定空战决策方案。

#### 7.14.1.3 协同多目标攻击战术特点

在协同多目标攻击时, 机群  $B$  的形势应作为一个整体来考虑。在战术上要求机群  $B$  依据优先攻击分配规则<sup>[181]</sup>, 即一方面蓝机  $i$  应在其相对于其他蓝机更有利于攻击的目标中按其攻击能力分配一定数量的目标; 另一方面蓝机  $i$  在已分配的目标中, 应首先选择攻击对机群  $B$  威胁最大的目标。这里以图 7.56 所示的实例为例作进一步说明。

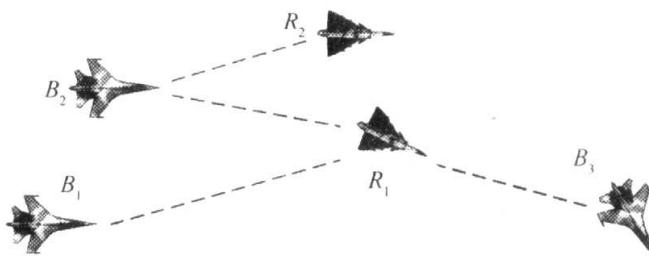


图 7.56 协同空战形势

在图 7.56 中, 机群  $B$  有蓝机  $B_1$ 、 $B_2$  及  $B_3$ , 而机群  $R$  有红机  $R_1$ 、 $R_2$ 。机群  $B$  采用协同多目标攻击战术与机群  $R$  空战。不难看出,  $R_1$ 、 $R_2$  均为  $B_2$  相对于  $B_1$ 、 $B_3$  更有利于攻击的目标。因而,  $R_1$ 、 $R_2$  应分配给  $B_2$  攻击。对于  $R_1$ 、 $R_2$ , 显然,  $R_1$  对机群  $B$  (尤其对  $B_3$ ) 威胁较大, 故  $B_2$  应优先攻击  $R_1$ 。

根据优先攻击分配规则，定义蓝机  $i$  对红机  $j$  的分配值为

$$ASM(i, j) = th_{ij} \cdot \sum_{i=1}^M th_{ji} \quad (7.14.11)$$

式中， $\sum_{i=1}^M th_{ji}$  表示红机  $j$  对机群  $B$  的威胁。显然，蓝机  $i$  应分配给具有尽可能大的分配值的目标。

如果红机  $j$  对机群  $B$  的威胁较大，为减小威胁，就需要分配两枚导弹，如导弹  $r$  与  $l$ ，同时对其攻击。定义这样的两枚导弹为一个导弹对  $(r, l, j)$ ，并定义导弹对  $(r, l, j)$  的分配差值为

$$DIS(r, l, j) = |ASM(r, j) - ASM(l, j)| \quad (7.14.12)$$

对于一个导弹对，希望这两枚导弹对目标的分配值都应尽量大，同时，希望其分配差值尽可能小。

## 7.14.2 算法设计

针对导弹目标分配问题，设蚁群中的蚂蚁数为  $m$ ，蚂蚁  $k$  ( $k = 1, 2, \dots, m$ ) 构建各自的导弹目标分配方案  $\pi_k$ 。在构建过程中，蚂蚁  $k$  从第 1 枚导弹开始在所有目标中选择一个目标分配给该枚导弹，随后对第 2 枚导弹，蚂蚁  $k$  在当前允许分配的目标集合中选择一个目标分配给该枚导弹，依此顺序进行分配，直到完成对第  $Z$  枚导弹的目标分配。在此过程中，蚂蚁  $k$  依据以下伪随机规则<sup>[182, 183]</sup> 选择目标  $j$  分配给导弹  $r$

$$p_k(r, j) = \begin{cases} \arg \max_{u \in \text{allowed}_k} \{\tau(r, u) \cdot \eta^\beta(r, u)\}, & \text{若 } q \leq q_0 \\ J, & \text{否则} \end{cases} \quad (7.14.13)$$

$$J = \begin{cases} \frac{\tau(r, j) \cdot \eta^\beta(r, j)}{\sum_{u \in \text{allowed}_k} \tau(r, ju) \cdot \eta^\beta(r, ju)}, & \text{若 } j \in \text{allowed}_k \\ 0, & \text{否则} \end{cases} \quad (7.14.14)$$

由于导弹  $r$  对目标  $j$  的分配值越大，蚂蚁将导弹  $r$  分配给目标  $j$  的可能性就越大，因此定义将目标  $j$  分配给导弹  $r$  的启发信息  $\eta(r, j)$  为  $ASM(r, j)$ 。在蚂蚁  $k$  将目标  $j$  分配给导弹  $r$  后，构建  $\pi_k$  中的第  $r$  个元素  $\pi_k(r) = j$ 。 $\text{tabu}_k$  则用于记录蚂蚁  $k$  当前不能再分配的目标，它依据当前  $\pi_k$  中已记录的目标和限制条件公式 (7.14.9)、公式 (7.14.10) 调整，则下一枚导弹可分配的目标集为  $\text{allowed}_k = \{R - \text{tabu}_k\}$ 。

蚂蚁  $k$  在构建可行解  $\pi_k$  的过程中，按如下局部更新规则对其信息量进行调整

$$\tau(r, j) = (1 - \zeta) \cdot \tau(r, j) + \zeta \cdot \tau_0 \quad (7.14.15)$$

式中,  $\zeta(0 \leq \zeta \leq 1)$  为常数,  $\tau_0 = 1/(m \cdot E_m)$  表示初始信息量,  $E_m$  为在不考虑信息量分量的情况下, 运行以上蚁群算法一次构建的解的评估值或根据最临近启发规则产生的解的评估值。

当  $m$  只蚂蚁均完成本次搜索后, HACA 用于对  $\pi_k(k=1, 2, \dots, m)$  进行局部搜索以寻求更好的临近解。HACA 是根据对协同多目标攻击战术的分析而获得的特定知识设计的局部搜索算法<sup>[184]</sup>, 其具体步骤为:

- (1) 找出  $\pi_k$  中仅分配了一枚导弹的目标, 组成目标集  $C$ , 其个数为  $n$ 。
- (2) 在  $\pi_k$  的所有导弹对中找出具有最大分配差值的导弹对  $(r, l, j)$ , 比较该导弹对中导弹  $r$  与  $l$  对目标  $j$  的分配值, 设其中导弹  $l$  的分配值较小。
- (3) 导弹  $l$  随机地选择  $C$  中的一个目标  $s$ , 若导弹  $l$  对目标  $s$  大于对当前目标  $j$  的分配期望值, 则导弹  $l$  改为攻击目标  $s$ ; 否则, 保持不变。
- (4) 将目标  $s$  从目标集  $C$  中去除。
- (5) 重复第 (3)、(4) 步  $n_1$  次,  $n_1(n_1 < n)$  为随机产生的一个正整数。找出搜索过程中的当前最优解  $\pi_{\text{elitist}}$ , 按如下全局更新规则对其信息量进行调整

$$\tau(r, j) = (1 - \rho) \cdot \tau(r, j) + \frac{\rho}{E(\pi_{\text{elitist}})} \quad (7.14.16)$$

上述搜索过程不断重复, 直到满足 HACA 的结束条件为止。

### 7.14.3 算例分析

在超视距空战中, 设由蓝机组成的机群  $B$  与由红机组成的机群  $R$  进行空战。对机群  $B$ , 设置  $M=4$ ,  $L=4$ ,  $Tr_B=120\text{km}$ ,  $Ra_B=70\text{km}$ ,  $V_B=350\text{m/s}$ ,  $i=1, 2, \dots, M$ 。因此, 机群  $B$  的导弹总数  $Z=16$ 。对机群  $R$ , 设置  $N=14$ ,  $V_R=300\text{m/s}$ ,  $j=1, 2, \dots, N$ 。蓝机  $Tr_B$  与  $Ra_B$  值设置均为红机对应参数值的 1.5 倍。设在某一时刻, 机群  $B$  与机群  $R$  在同一高度与不同的队形迎头飞行<sup>[185]</sup>, 所有目标均在机群  $B$  的联合攻击区内, 机群  $B$  与机群  $R$  的空战态势如图 7.57 所示。

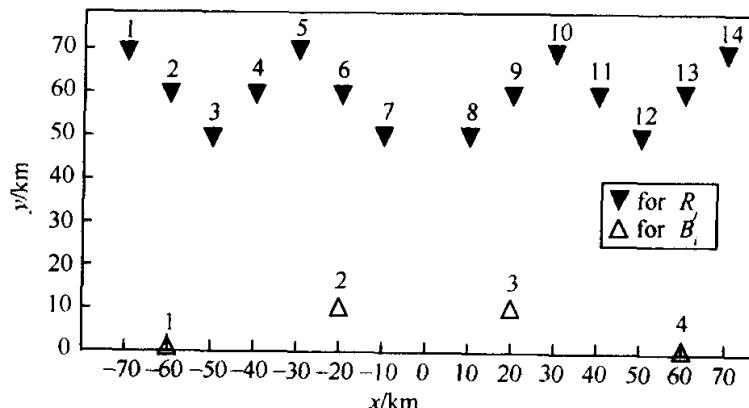


图 7.57 机群  $B$  与机群  $R$  的空战态势

设置  $m=3$ ,  $q_0=0.8$ ,  $\beta=2$ ,  $\zeta=0.1$ ,  $\rho=0.08$ 。采用 HACA 和基本蚁群算法分别对本例进行 10 次仿真实验, 算法终止条件是迭代 200 次终止。实验结果表明 HACA 均在循环 20 次以内获得最优解, 而采用基本蚁群算法最好情况需循环 100 次左右才能获得最优解。两种算法的最佳实验结果如图 7.58 所示。

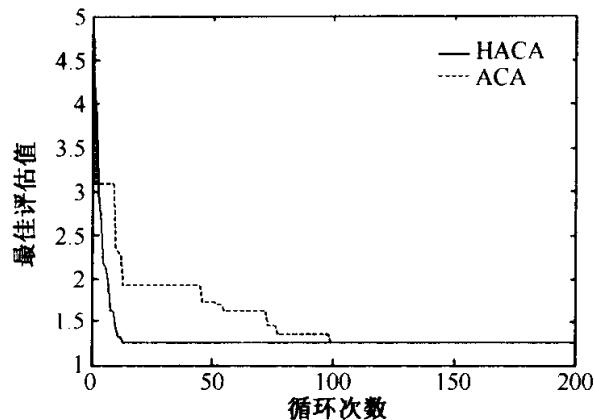


图 7.58 两种算法的最佳实验结果比较

在图 7.58 中, HACA 在迭代 14 次后获得了机群  $B$  对所有红机的最优攻击分配方案, 其最优攻击分配情况如表 7.29 所示。

表 7.29 最优攻击分配情况

导弹	蓝机 1				蓝机 2				蓝机 3				蓝机 4			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
红机	2	3	4	1	7	6	5	7	9	10	8	8	13	11	12	14

由表 7.29 可见, 第 1 枚导弹攻击红机 2, 第 2 枚导弹攻击红机 3, 直到第 16 枚导弹攻击红机 14。并可得到最终空战决策为: 蓝机 1 攻击 1, 2, 3 和 4 号红机; 蓝机 2 攻击 5, 6, 7 号红机, 并对 7 号红机分配两枚导弹攻击; 蓝机 3 攻击 8, 9, 10 号红机, 并对 8 号红机分配 2 枚导弹攻击; 蓝机 4 攻击 11, 12, 13 和 14 号红机。

## 7.15 岩土工程

岩土工程中的许多优化问题通常是一些非常复杂或无明确表达式的数学规划问题, 传统的优化方法一般较难获得满意的结果。近几年来许多学者尝试用仿生优化算法对其进行求解, 而利用蚁群算法解决岩土工程中的问题则是一个新的尝试。文献 [186] 将改进后的蚁群算法应用于硐群施工顺序优化; 文献 [187] 构建了一种新的自适应蚁群算法, 并将其成功地应用于边坡非圆弧临界滑动面搜

索；文献 [188] 将蚁群算法与土坡稳定性有限元分析方法相结合，提出了土坡任意形状临界滑动面的蚁群算法搜索技术；文献 [189] 提出了一种具有混沌扰动算子的启发式蚁群算法，并将其应用于边坡稳定性分析；文献 [190] 和 [191] 利用蚁群算法分别对岩土工程领域中的高填石路堤稳定性问题和边坡稳定性问题进行了深入分析。

本节研究了一种基于自适应蚁群算法的边坡非圆弧临界滑动面搜索技术，并通过算例验证了该算法可有效地防止停滞或过早收敛，且总能搜索到问题的全局最优解。

### 7.15.1 问题描述

将边坡体离散成如图 7.59 所示的格式，蚂蚁从 START 点出发，逐级经过条分线上的节点，最后到达 END 点，完成一次循环并形成一个滑动面<sup>[187]</sup>。图 7.59 中的虚线只用做引导蚂蚁搜索滑动面，不参与边坡安全系数计算。

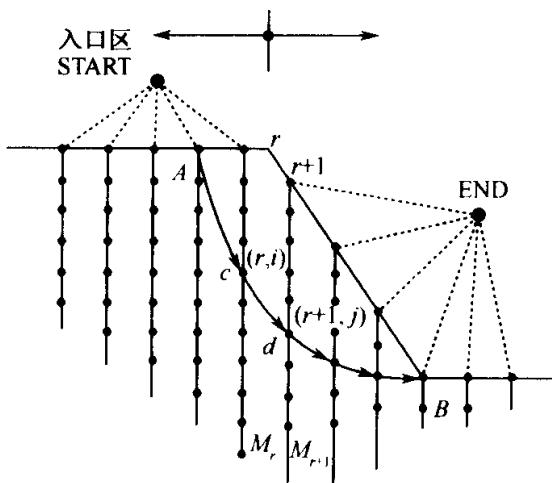


图 7.59 边坡体离散及滑动面搜索

### 7.15.2 算法设计

这里设  $(r, i)$  表示第  $r$  条分线上的第  $i$  个节点； $(r+1, j)$  表示第  $r+1$  条分线上的第  $j$  个节点； $[(r, i), (r+1, j)]$  表示节点  $(r, i)$  到节点  $(r+1, j)$  的连线，蚂蚁数目为  $m$ 。在运动过程中，蚂蚁  $k$  ( $k=1, 2, \dots, m$ ) 根据各条路径上的信息素量决定转移方向，在时刻  $t$  蚂蚁  $k$  由位置  $(r, i)$  转移到位置  $(r+1, j)$  的概率为

$$p_{[(r,i),(r+1,j)]}^k(t) = \frac{[\tau_{[(r,i),(r+1,j)]}(t)]^\alpha [\eta_{[(r,i),(r+1,j)]}(t)]^\beta}{\sum_{j=1}^{M_{r+1}} [\tau_{[(r,i),(r+1,j)]}(t)]^\alpha [\eta_{[(r,i),(r+1,j)]}(t)]^\beta} \quad (7.15.1)$$

蚂蚁  $k$  在寻找有效滑动面时，总有指向出口区 END 点的趋势。因此，可认为蚂蚁  $k$  由条分线  $r$  转移到条分线  $r+1$  时，条分线  $r+1$  上各点被选中的概率分布如图 7.60 所示，即有

$$[\eta_{[(r,i),(r+1,j)]}(t)]^\beta = \begin{cases} \frac{M_{r+1} + 1 - f_{r+1}}{M_{r+1} + 1}, & \text{若 } 1 \leq j \leq f_{r+1} \\ \frac{(M_{r+1} + 1 - f_{r+1})f_{r+1} + \sum_{s=f_{r+1}+1}^{M_{r+1}+1} (M_{r+1} + 1 - s)}{(M_{r+1} + 1 - f_{r+1})f_{r+1} + \sum_{s=f_{r+1}+1}^{M_{r+1}+1} (M_{r+1} + 1 - s)}, & \text{若 } f_{r+1} < j \leq M_{r+1} \end{cases} \quad (7.15.2)$$

式中， $f_{r+1}$  取图 7.60 中  $e$  点以下紧靠  $e$  点的节点编号（ $e$  点为上一段滑动面延长线与条分线  $r+1$  的交点）。

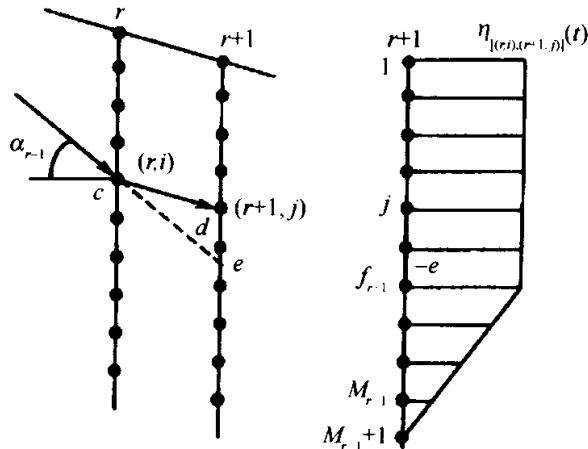


图 7.60 点  $(r, i)$  到点  $(r+1, j)$  的转移概率

当图 7.60 中  $e$  点越过  $r+1$  最下面的节点，以及蚂蚁  $k$  由 START 点出发选择  $A$  点的位置和从  $A$  点转移到下一阶段时（如图 7.59 所示），则以等概率选择条分线  $r+1$  上的各点。

随着时间的推移，在蚂蚁经过的路径上所留下的信息量将会逐渐减少。如果  $m$  只蚂蚁按照公式 (7.15.1) 在时间  $\Delta t$  内均各自找到了可行解（完成了一次循环），则每条路径  $[(r, i), (r+1, j)]$  上的信息量按下式调整

$$\begin{cases} \tau_{[(r,i),(r+1,j)]}(t+\Delta t) = (1-\rho)\tau_{[(r,i),(r+1,j)]}(t+\Delta t) + \Delta\tau_{[(r,i),(r+1,j)]} \\ \Delta\tau_{[(r,i),(r+1,j)]} = \sum_{k=1}^m \Delta\tau_{[(r,i),(r+1,j)]}^k \end{cases} \quad (7.15.3)$$

$$\Delta\tau_{[(r,i),(r+1,j)]}^k = \begin{cases} \frac{Q}{F_s^k}, & \text{若路径}[(r,i),(r+1,j)] \text{处在蚂蚁 } k \text{ 本次搜索到的滑动面上} \\ 0, & \text{否则} \end{cases} \quad (7.15.4)$$

式中,  $F_s^k$  为第  $k$  只蚂蚁本次搜索到的滑动面的安全系数值, 可采用各种边坡稳定性分析方法计算。

在边坡临界滑动面搜索中, 引入公式 (7.15.2) 可有效地提高搜索效率, 但仍然未能解决停滞问题。这里引入了一个自适应调整算子, 在经过数次循环后调整蚂蚁的选择策略, 加大随机选择概率, 以利于解空间的更完全搜索。该自适应调整算子规定, 当蚂蚁  $k$  从条分线  $r$  上第  $i$  个节点转移到条分线  $r+1$  上第  $j$  个节点时,  $j$  按下式确定

$$j = \begin{cases} \arg \max \{\eta_{[(r,i),(r+1,u)]}(t), u=1, 2, \dots, M_{r+1}\}, & \text{若 } r_0 < p_0 \\ \arg \max \{p_{[(r,i),(r+1,u)]}^k(t), u=1, 2, \dots, M_{r+1}\}, & \text{否则} \end{cases} \quad (7.15.5)$$

式中,  $p_{[(r,i),(r+1,u)]}^k(t)$  和  $\eta_{[(r,i),(r+1,u)]}(t)$  分别按公式 (7.15.1) 和公式 (7.15.2) 计算;  $r_0 \in (0, 1)$  为均匀分布随机数;  $p_0 \in [0, 1]$  为调整参数, 若  $p_0 = 0$ , 则完全按公式 (7.15.1) 确定  $j$ , 若  $p_0 = 1$ , 则完全按公式 (7.15.2) 确定  $j$ 。于是, 用于边坡稳定性分析及最危险滑动面搜索的自适应蚁群算法可描述如下:

- (1)  $N_c = 0$ , 置  $\tau_{[(r,i),(r+1,j)]}(0) = \epsilon$  (较小正数),  $\Delta\tau_{[(r,i),(r+1,j)]} = 0$ 。
- (2)  $m$  只蚂蚁从 START 点出发, 独立地按公式 (7.15.1) 和公式 (7.15.2) 计算转移概率并追踪生成有效的滑动面, 但每经过若干次迭代, 则换用公式 (7.15.5) 确定路径并搜索滑动面。
- (3) 按选定的边坡稳定性分析方法, 计算各有效滑动面的安全系数值  $F_s^k$ ; 保存其中最小者, 并将其记为当前最好解。
- (4) 对各蚂蚁及路径  $[(r, i), (r+1, j)]$ , 按公式 (7.15.3) 和公式 (7.15.4) 计算  $\Delta\tau_{[(r,i),(r+1,j)]}^k$  和  $\tau_{[(r,i),(r+1,j)]}(t)$ 。
- (5)  $N_c = N_c + 1$ ; 若  $N_c \geq N_{c_{\max}}$ , 则输出最佳路径 (即最危险滑动面) 和最小安全系数值; 否则跳转到第 (2) 步。

### 7.15.3 算例分析

某露天矿上盘边坡 S47 剖面, 设计台阶高 20m、台阶坡角  $70^\circ$ , 总体边坡高 99.3m、坡角  $53^\circ$ 。岩体的物理力学指标值如表 7.30 所示, 爆破地震系数  $K_c = 0.08$ 。

表 7.30 计算参数取值

岩石种类	$\gamma$ (kPa)	C (MPa)	$\varphi$ (°)
钠质正长岩	28.2	0.2656	41.256
复杂砂卡岩	28.0	0.3530	46.179
安山岩	27.4	0.2073	39.141
铁矿石	31.0	0.3530	46.179

分别采用自适应蚁群算法和文献 [192] 基本蚁群算法对上述边坡临界滑动面进行搜索，取相同的计算参数，得仿真计算结果如表 7.31 和图 7.61 所示。

表 7.31 计算结果比较

计算方法	计算结果 $F_s$	演化代数 $t$
文献 [192] 算法	1.913	97
自适应蚁群算法	1.913	65

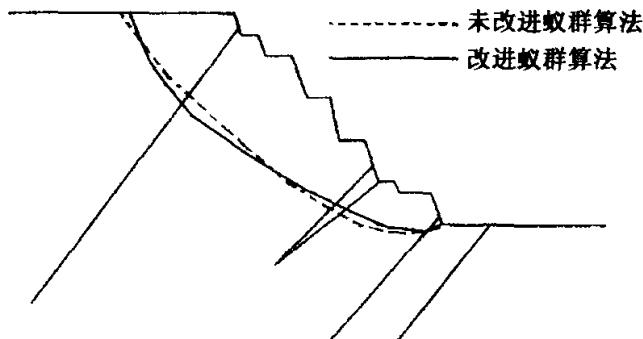


图 7.61 两种算法的仿真结果

由表 7.31 和图 7.61 可见，在相同的计算精度情况下，本节所提出的自适应蚁群算法具有更高的求解效率，计算所耗机时比基本蚁群算法减少了 29.3%。

## 7.16 化学工业

在许多领域中，每一种新算法的诞生都会带动新一轮研究热潮的掀起，并极大地推动着这个领域许多学科的发展。化学工业领域也不例外，自从丁亚平等<sup>[193, 194]</sup>首次将蚁群算法应用到化学计量学科的光谱解析之后，Shelokar P S 等<sup>[195]</sup>、贺益君等<sup>[196]</sup>在用蚁群算法求解化工领域的多种优化问题方面进行了许多研究工作。

超临界水氧化法对治理有机污染方面极具发展潜力，有机物在超临界水中氧

化的反应动力学机理是一项非常重要的研究内容。化工动力学参数估计是常见的优化问题，形式看似简单，其误差响应曲面往往相当复杂，具有很多的局部极值，常规优化算法易陷于局部极值区，而准确地估计相关的反应动力学参数，对超临界水氧化技术的机理解释和产业化尤为重要。本节构建了一种杂交蚁群系统(hybrid ant colony system, HACS) 算法，并将其应用于 2-氯苯酚在超临界水中氧化反应动力学参数估计问题。

### 7.16.1 问题描述

应用超临界水氧化技术处理有机废水，去除率为其最重要的指标。2-氯苯酚是一种较有代表性的有机废水，其全局反应动力学方程为

$$-r_A = A \exp\left(-\frac{E_a}{RT}\right)[2CP]^a[O_2]^b[H_2O]^c \quad (7.16.1)$$

式中， $A$ 、 $E_a$ 、 $a$ 、 $b$  和  $c$  为待定动力学参数，可由样本数据估计。所测定的自变量为  $[2CP]$ 、 $[O_2]$ 、 $[H_2O]$ 、 $T$ 、压力与停留时间，因变量为 2-氯苯酚去除率。设样本容量为  $M$ ，第  $i$  个样本的 2-氯苯酚去除率的实测值为  $r_i$ ，估算值为  $\hat{r}_i$ 。参数估计的原则是使下式所示的目标函数 EQS 达到最小

$$EQS = \sum_{i=1}^M (\hat{r}_i - r_i)^2 \quad (7.16.2)$$

### 7.16.2 算法设计

HACS 的解被视为解空间的点或个体，蚂蚁以多种方式引领个体在解空间中搜索，并在局部区间留下信息，实现互激励，以加速全局寻优<sup>[196]</sup>。这里可将待优化问题表述如下

$$\begin{aligned} \min f(x) &= f(x_1, x_2, \dots, x_n) \\ \text{s. t. } &x_i^L \leq x_i \leq x_i^U, \quad i = 1, 2, \dots, m \end{aligned} \quad (7.16.3)$$

#### 7.16.2.1 全局蚂蚁及其操作

定义全局蚂蚁数为  $ga\_num$ ，常为种群规模的 20%~30%。全局蚂蚁将通过选择引领一个个体，对它们实施交叉和变异，并执行种群控制操作。

##### 1. 选择操作

由种群适应度较高的个体，组成优良集，个数为  $num-ga\_num$ ，并以适应度升序排列。优良集被保留至下一代，并以随机遍历抽样法从中选出  $ga\_num$

一个个体，第一指针为区间 $[0, 1/ga\_num]$ 中的随机数，选择指针的间距为 $1/ga\_num$ ，则选择概率如下式所示

$$p_j(t) = \frac{1}{num - ga\_num} \left[ \eta^+ - (\eta^+ - \eta^-) \frac{j-1}{num - ga\_num} \right] \quad (7.16.4)$$

式中， $j$  表示个体序号； $\eta^+$ 、 $\eta^-$  为常数，且  $1 \leq \eta^+ \leq 2$ ， $\eta^- = 2 - \eta^+$ 。适应度较高，则被选中的概率较小。

## 2. 交叉操作

交叉算子如下式所示

$$\begin{cases} child1 = \eta \cdot parent1 + (1 - \eta) \cdot parent2 \\ child2 = \eta \cdot parent2 + (1 - \eta) \cdot parent1 \end{cases} \quad (7.16.5)$$

式中， $\eta$  表示交叉因子，其值为 $[0, 1.0]$ 中的随机数。 $parent1$  和  $parent2$  表示父代个体， $child1$  和  $child2$  表示子代个体。若子代个体为所在区域的唯一个体，将区域的位置移至该个体处，并按公式 (7.16.6) 更新其信息素，否则执行种群控制操作。

$$\begin{cases} \tau(child1, t) = \eta \cdot \tau(parent1, t) + (1 - \eta) \cdot \tau(parent2, t) \\ \tau(child2, t) = \eta \cdot \tau(parent2, t) + (1 - \eta) \cdot \tau(parent1, t) \end{cases} \quad (7.16.6)$$

## 3. 变异操作

对交叉操作产生的个体，执行如下式所示的变异操作

$$x'_i = x_i \pm 0.5L\Delta \quad (7.16.7)$$

式中， $\Delta = \sum_{i=0}^m \frac{a(i)}{2^i}$ ， $a(i)$  以概率  $1/m$  取值 1，否则为 0； $L$  为分量  $x_i$  的取值范围  $x_i^U - x_i^L$ ，随机选定正负号。若变异后的子代个体为所在区域的唯一个体，将区域的位置移至该个体处，并按公式 (7.16.8) 更新其信息量，否则执行种群控制操作。

$$\tau(x, t) = \tau(x, t) + \gamma [\text{fitness}(\text{child}) - \text{fitness}(\text{parent})] \quad (7.16.8)$$

式中， $parent$  表示变异前的父代个体； $child$  表示变异后的子代个体； $\gamma$  为常数因子。

## 4. 种群控制操作

若交叉或变异生成的个体  $x_2$  所在区域原先已有个体  $x_1$ ，则应调用种群控制操作，以确保各区域内最多只有一个个体。它比较  $\text{fitness}(x_1)$  与  $\text{fitness}(x_2)$ ，保留适应度大的个体，并将区域的当前位置移至该个体处，其信息素按公式 (7.16.8) 更新，式中的  $parent$  取为  $x_1$ ， $child$  取为  $x_2$ 。为维持种群规模不变，本操作还须再产生一个个体，可从当前无个体的区域中随机选一个，以其中心作为

新个体，其信息量取为 $(\tau_{\max} + \tau_{\min})/2$ 。

### 7.16.2.2 局部蚂蚁及其操作

定义局部蚂蚁数为 la\_num，常为种群规模的 5%~10%。局部蚂蚁将对全局蚂蚁探索后生成的种群进行挖掘。位置为  $x$  的区域被局部蚂蚁选中的概率如下式所示

$$p(x, t) = \frac{[\tau(x, t)]^\alpha [\text{fitness}(x)]^\beta}{\sum [\tau(x, t)]^\alpha [\text{fitness}(x)]^\beta} \quad (7.16.9)$$

对选中的区域，局部蚂蚁将执行局部寻优操作，其步骤如下：

- (1) 以区域位置为初始点，采用 Powell 算子在区域内寻优。
- (2) 若搜索点适应度高，将其设置为区域位置，并按公式 (7.16.8) 更新区域信息素；否则，区域位置保持不变，并按下式消减区域信息素

$$\tau(x, t) = \xi_1 \tau(x, t) \quad (7.16.10)$$

式中， $\xi_1$  表示厌恶率，且  $0 < \xi_1 < 1$ 。

### 7.16.2.3 信息素挥发

HACS 进行周期性寻优，由于信息素会随时间的推移而挥发，HACS 可用下式体现这一思想

$$\tau(x, t+1) = \xi_2 \tau(x, t) \quad (7.16.11)$$

式中， $\xi_2$  表示挥发率，且  $0 < \xi_2 < 1$ 。

### 7.16.2.4 HACS 的算法步骤

HACS 的主要步骤如下：

- (1) 设定各参数，种群初始化并计算个体适应度，进行排序，且记录最好解。
- (2) 发出全局蚂蚁执行选择、交叉、变异和种群控制操作。
- (3) 发出局部蚂蚁执行局部寻优操作。
- (4) 计算所有个体的适应度并排序，记录最好解。
- (5) 对所有区域执行信息素挥发操作。
- (6) 检查是否已达到终止条件，若是，输出最好解，并结束算法；否则，跳转到第 (2) 步。

### 7.16.3 算例分析

应用 HACS 估计公式 (7.16.1) 所示的动力学参数, 样本共 62 组, 包含温度、压力、停留时间、 $[2\text{CP}]$ 、 $[\text{O}_2]$ 、 $[\text{H}_2\text{O}]$  和 22 氯苯酚去除率, 设置  $\text{num}=50$ ,  $\text{ga\_num}=10$ ,  $\text{la\_num}=5$ ,  $\alpha=1$ ,  $\beta=2$ ,  $\gamma=1$ ,  $\xi_1=0.95$ ,  $\xi_2=0.9$ ,  $N_{c_{\max}}=50$ 。HACS 独立运行 20 次, 每次所得的最优值均为  $\text{EQS}=0.2177$ 。基于同一样本, 文献 [197] 采用混沌遗传算法 (chaos genetic algorithm, CGA), 文献 [198] 采用经典非线性回归 (nonlinear regression, NLR) 算法对其进行参数估计, 表 7.32 列出了这三种算法所估计的参数与 EQS 值。

表 7.32 2-氯苯酚超临界水氧化反应的动力学参数估计

算法	A	$E_a$	a	b	c	EQS
HACS	63.5	45625.9	0.8081	0.4444	0.3239	0.2177
CGA	70.4	45153.2	0.8181	0.4750	0.3267	0.2225
NLR	100.0	46200.0	0.8800	0.4100	0.3400	0.2494

由表 7.32 可见, HACS 与 CGA 所得的参数值比较接近。对于所优化的 EQS 值而言, HACS 的结果比文献 [198] 的结果降低了 12.7%, 下降幅度明显, 比 CGA 的结果降低了 2.2%。

对超临界水氧化动力学参数的估计, NLR 方法的效果不佳反映出常规方法容易陷于局部极值的困难, 而具有较强全局优化能力的 HACS 可有效地解决这一“瓶颈”性问题。

## 7.17 生命科学

蛋白质是生命活动的重要承担者, 蛋白质的生物功能由其空间结构决定, 在天然蛋白质中, 蛋白质的最终构象是由氨基酸序列唯一确定的。蛋白质折叠问题是生物信息学中的一个重要问题, 它研究蛋白质从一维序列到三维结构的过程。由于这一过程极其复杂, 人们往往利用简化的模型来对其进行研究, 亲-疏水格点 (hydrophobic and polarmonomers, HP) 模型就是 Dill K A 等<sup>[199]</sup> 提出的一种最为重要的简化模型。HP 模型认为疏水性作用是蛋白质折叠问题中最主要的相互作用, 构成蛋白质的 20 种氨基酸按照其残基的疏水性不同被分成两大类, 即疏水性 (hydrophobic, H) 基团和极性 (polar, P) 基团。在格点模型中, 这些基团组成的序列在网格空间形成了一条自回避路径, 而蛋白质的折叠过程则是在网格空间中建立一个自由能最低的构象的过程。目前已经证明蛋白质折叠问题是

一个 NP-C 问题<sup>[200]</sup>。

对 HP 模型，一般采用的方法大致可以分为进化算法<sup>[201]</sup>和蒙特卡罗 (Monte Carlo, MC) 仿真<sup>[202]</sup>两类，这两类算法各有利弊。最近有学者提出了将蚁群算法用于二维 HP 模型的新思路<sup>[203~210]</sup>，并取得了良好的应用效果。本节研究了一种蚁群算法和 MC 仿真的混合策略，并将其应用于 HP 模型的测试序列求解，该策略通过引入克隆和淘汰操作来提高改进后蚁群算法的求解效率。

### 7.17.1 算法设计

在改进后的蚁群算法中，可将蚁群构造问题的可能解这一过程分为内部循环和外部循环两部分。把内部循环看做对一代蚂蚁的操作，而外部循环则是迭代次数的循环。这样，对每一代蚂蚁，可以采用常用的 MC 仿真来计算；而在两代蚂蚁之间，则可以借鉴进化计算的思想，上一代的计算结果以某种形式影响下一代的计算<sup>[207]</sup>。

蚂蚁为给定的序列产生的候选构象，采用相对折叠方向（左转、前行、右转）来表示（如图 7.62 所示）。在前进第  $i$  步时，蚂蚁从当前的位置选择一个相对方向前进一步，到达一个新的位置。如果这个新位置从未到达过，则成功前进了一步，蚂蚁将把序列上第  $i+1$  个字符（H 或 P）放入这个新位置，并开始下一步行走。反之，如果这个新的位置以前曾经到达过，则构造过程失败。

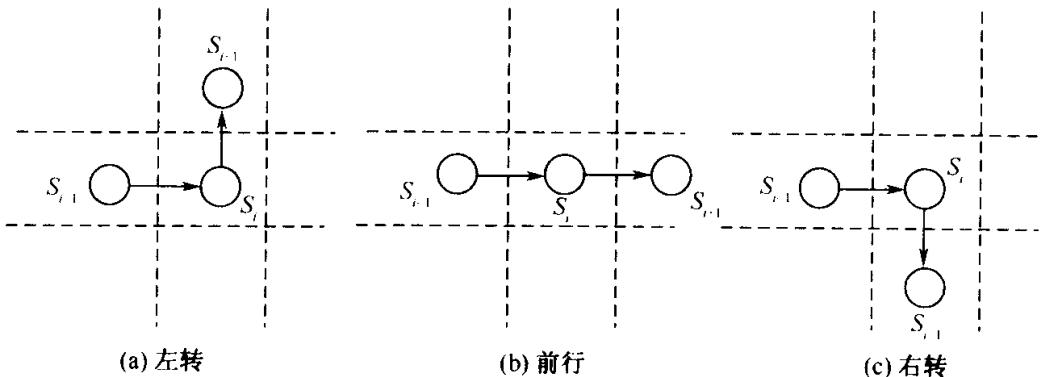


图 7.62 相对折叠方向

在图 7.62 中，从位置  $S_i$  出发，下一个位置有三种选择，以上一步前进的方向为基准，这三种选择可以分别称为向左、向前和向右，而不管其绝对方向如何。第一步没有相对方向，规定其绝对方向为向上，这一规定不影响计算结果。

在构造过程中，每个位置的三种折叠方向上都有一定的初始信息素，蚂蚁在该位置前进的时候，对可供选择的每个相对方向 ( $L$ 、 $S$  或  $R$ ) 都能够获得两种信息：信息量 ( $\tau_{i,d}$ ,  $d \in \{L, S, R\}$ ) 和启发式信息  $\eta_{i,d}$ 。这里可定义启发式信息

$\eta_{i,d}$  为新到达位置  $i+1$  对整个系统自由能的贡献函数，在一般的 MC 仿真中，该函数通常具有 Boltzmann 分布的形式，即

$$\eta_{i,d} = \exp\left(-\frac{\Delta E_{i+1}}{k_B T}\right) \quad (7.17.1)$$

式中， $k_B$  表示 Boltzmann 常数； $T$  表示温度。在本节的改进蚁群算法中也采用这一形式，信息量  $\tau_{i,d}$  存在两种操作：耗散过程和增强过程。在计算过程中，信息量会以速率  $\rho$  ( $0 < \rho < 1$ ) 挥发，此即耗散过程，它表示系统记忆的消退。而当蚂蚁获得了一条成功的路径时，就会对所经过的每一条边的信息素进行增强。信息素更新的公式为

$$\tau_{i,d} = (1 - \rho)\tau_{i,d} + \Delta_{i,d,c} \quad (7.17.2)$$

式中， $\Delta_{i,d,c}$  是一个固定的增强常数值，它是该路径的自由能与理论上最低的自由能之间的比值。从而，蚂蚁在位置  $i$  选择下一方向的概率可由下式确定

$$p_{i,d} = \frac{\tau_{i,d}\eta_{i,d}}{\sum_{e \in \{L,S,R\}} \tau_{i,e}\eta_{i,e}} \quad (7.17.3)$$

由于蚂蚁构造候选构象的过程是一个自回避随机行走过程，因此，当构象比较紧密且序列较长时，必然会出现行走到“死路”的情况，从而使得构造无效，这将导致计算效率的极大降低。因此，这里提出了克隆和淘汰的操作，即在第  $i$  只蚂蚁构造构象的过程中，如果在某一步构造时出现无效的构象，它将会被淘汰。同时，由于蚂蚁构造候选构象的操作是并行进行的，因此，此处可按照某种概率选取第  $j$  ( $0 < j < i$ ) 只蚂蚁进行克隆操作，以这只克隆出来的蚂蚁代替第  $i$  只蚂蚁，继续运行。由于第  $j$  只蚂蚁在这一步构造中是有效的构象，这一操作能够有效地去除无效的构象。在实际操作中，如果  $i=1$ ，上述操作将会失效，因此，这里还记录了前一步构造中具有最佳自由能的蚂蚁  $k$ ，它也以一定的概率参与克隆操作，而且对  $i=1$  的情况，直接对蚂蚁  $k$  进行克隆。

本节所提出改进蚁群算法的具体步骤如下：

- (1) 初始化路径上的信息素。
- (2) 外部循环（循环，直到满足停止条件）。
  - ① 初始化所有的蚂蚁；
  - ② 内部循环，若没有到达序列的结束位置，则做如下操作：
    - 对每只蚂蚁，前进一步，并计算其当前的自由能；
    - 获得当前具有最佳自由能的蚂蚁；
    - 对每只蚂蚁，若有必要，执行克隆和淘汰操作；
  - ③ 内部循环结束；
  - ④ 路径上的信息素耗散；
  - ⑤ 对获得较佳的自由能的蚂蚁，进行路径信息素增强操作；
  - ⑥ 记录获得的最好结果。

(3) 外部循环结束。

(4) 算法结束。

## 7.17.2 仿真算例

采用文献中通常使用的测试序列对改进的蚁群算法进行了测试。测试序列(包括其已知最佳构象的能量值)如表 7.33 所示。

表 7.33 测试序列

序号	长度	能量	序列
1	20	-9	(HP)2PH2PHP2HPH2P2HPH
2	24	-9	H2P2(HP2)6H2
3	25	-9	P2HP2H2P4H2P4H2P4H2
4	36	-14	P3H2P2H2P5H7P2H2P4H2P2HP2
5	48	-23	P2HP2H2P2H2P5H10P6H2P2H2P2HP2H5
6	50	-21	H2(PH)3PH4PHP3HP3HP4HP3H P3H PH4(PH)3PH2
7	60	-36	P2H3PH8P3H10PHP3H12P4H6PH2PHP
8	64	-42	H12(PH)2(P2H2)2P2H(P2H2)2P2H(P2H2)2P2HPH12
9	85	-53	H4P4H12P6(H12P3)3HP2(H2P2)2HPH

蚂蚁的规模选为 500 (这一规模的选择对算法的效率有一定的影响,但是不会影响本算例结论的成立),  $\rho=0.4$ , 计算  $\Delta_{i,d,c}$  过程中用到的自由能的理论最小值由序列中的 H 残基的数目计算出来 (序列内部的每个 H 残基被放入网格中时, 自由能最多降低 2 个单位, 两端的 H 残基最多导致自由能降低 3 个单位)。测试的结果如表 7.34 所示, 为了便于对比, 这里还列出了其他文献中给出的对

表 7.34 测试序列的计算结果

序号	长度	能量	GA		EMC		文献[210]蚁群算法		改进蚁群算法	
			最优解	时间 (s)	最优解	时间 (s)	最优解	时间 (s)	最优解	时间 (s)
1	20	-9	-9	30.492	-9	9.374	-9	3.33	-9	0.25
2	24	-9	-9	30.491	-9	6.929	-9	2.52	-9	1.71
3	25	-9	-8	20.400	-8	7.202	-8	10.62	-8	50.00
4	36	-14	-14	301.339	-14	12.447	-14	11.81	-14	12.72
5	48	-23	-23	126.547	-23	165.91	-23	405.79	-23	51.11
6	50	-21	-21	592.887	-21	74.613	-21	4.953	-21	23.87
7	60	-36	-34	208.781	-35	203.729	-36	62.471	-36	2.898
8	64	-42	-37	187.393	-39	564.809	-42	5.845	-42	4.588
9	85	-53	...	...	-52	44.029	-51	21.901	-51	11.237

相同序列的测试结果，包括遗传算法（GA）、进化 MC（evolution Monte Carlo, EMC）仿真以及文献 [210] 所给出的蚁群算法结果。

由表 7.34 可见，与 GA 和 EMC 仿真相比，蚁群算法在大部分情况下能够获得更好的运行效果。另外，文献 [210] 所给出的蚁群算法通过引入局部搜索操作来提高算法的效率，由于这种操作的代价比较大，因而对效率的提高是有限的。本节中则采用了克隆和淘汰操作，这两种操作本身的代价是相当小的，但是它们能够有效地克服算法中出现的效率瓶颈。两者的测试结果对比表明，本节改进后的蚁群算法对较长的序列获得了高于文献 [210] 中蚁群算法的效率。

## 7.18 布局优化

布局优化设计在工程上属于方案设计问题，而在数学上则属于具有组合最优化性质的 NP-C 问题，布局优化设计既存在计算复杂性的组合爆炸，又存在着工程复杂性。

卫星舱的布局优化设计是布局优化领域中非常具有代表性的问题。近几年来，国内外许多学者对卫星舱的复杂布局设计进行了多方面探讨<sup>[211~214]</sup>，较实用方法之一是利用启发式算法，再加上人机交互。早在 1990 年，钱学森给出了人机结合（human-computer cooperation）的思想<sup>[215]</sup>，类似的还有戴汝为提出的大成智慧<sup>[216]</sup>和 Lenat D B 等<sup>[217]</sup>提出的人机合作（man-machine synergy）思想，但在工程中如何实现，目前尚未得到很好的解决。Sun Z G 等<sup>[218, 219]</sup>在对蚁群算法改进的基础上，提出了一种魔方启发式技术，用于求解国际卫星（Intelsat-III）舱的布局优化设计问题，初步验证了蚁群算法求解这类问题的可行性和有效性，但是没有充分利用工程师的经验和智慧。基于此，霍军周等<sup>[220]</sup>提出了基于人-蚁群/遗传算法的人机结合算法，称为人机结合蚁群/遗传算法（human-computer cooperative ant colony/genetic algorithm, HCAGA），并将其成功地应用于卫星舱的布局优化设计。

### 7.18.1 问题描述

如图 7.63 所示的截圆锥体简化卫星舱以一定的角速度  $\omega$  旋转，设置有垂直于舱的中心轴线 ( $x_3$  轴) 的圆形承载隔板，需要布置的仪器、设备称为待布物。要求将  $n$  个给定的待布物最优地配置于承载隔板上下两侧，使得全舱总体布局满足：

- 待布物之间、待布物与容器之间不干涉；
- 系统动平衡误差小于允许值；
- 提高舱体空间利用率；等等。

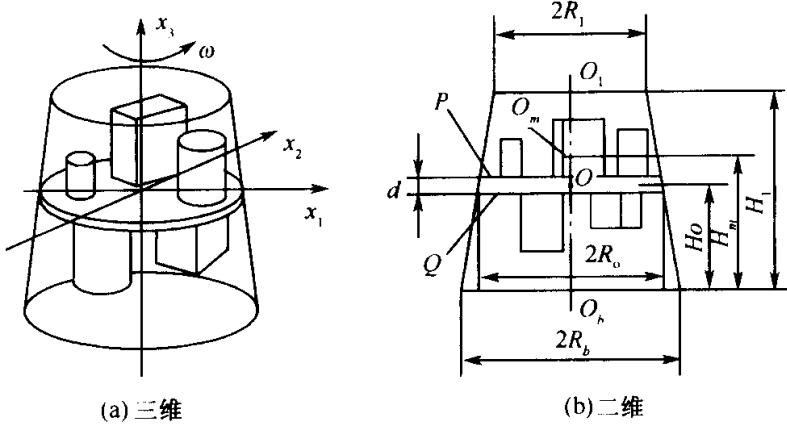


图 7.63 卫星舱布局示意图

从而，所求问题归结为待布物的基准点位置和放置方向。

图 7.63 所示的卫星舱布局设计可简化为二维半问题处理，首先在二维平面内布局，然后在三维空间内检查待布物是否与舱体干涉。设  $\theta \in [0, \pi]$  为待布物  $i$  在承载面上的方向角（逆时针为正）；设待布物  $i$  基准点（一般取质心）坐标为  $(x_{i1}, x_{i2}, x_{i3})^T \in R^3$ 。卫星舱布局优化数学模型可描述如下<sup>[214]</sup>：

求布局方案  $LS \in R^{4n}$ ，使其满足

$$\min f(x) = \lambda_1 F + \lambda_2 M \quad (7.18.1)$$

$$F = \sqrt{\left(\sum_{i=1}^N m_i \omega^2 x_{i1}\right)^2 + \left(\sum_{i=1}^N m_i \omega^2 x_{i2}\right)^2} \quad (7.18.2)$$

$$M = \sqrt{\left[\sum_{i=1}^N m_i \omega^2 x_{i1} (d/2 + x_{i3}) \beta_i\right]^2 + \left[\sum_{i=1}^N m_i \omega^2 x_{i2} (d/2 + x_{i3}) \beta_i\right]^2} \quad (7.18.3)$$

$$\text{s. t. } \sum_{i=1}^{n-1} \sum_{j=i+1}^n \Delta S_{ij} \leqslant 0 \quad (7.18.4)$$

$$R \leqslant R_i \quad (7.18.5)$$

$$R_i = \begin{cases} \frac{H_t - H_o - 2x_{i3} - 1.5d}{H_t - H_o} (R_o - R_t) + R_t, & \text{若 } \beta_i = 1 \\ \frac{(R_b - R_o)d}{2H_o} + R_o, & \text{若 } \beta_i = -1 \end{cases} \quad (7.18.6)$$

式中， $\lambda_1$ 、 $\lambda_2$  表示目标函数加权系数； $F$  表示系统动不平衡力； $M$  表示系统动不平衡力矩； $R$  表示各待布物外包络圆的半径； $R_i$  表示待布物  $i$  顶端与舱体壁相交所形成的圆平面的半径； $\Delta S_{ij}$  表示待布物  $i$ 、 $j$  的干涉面积， $i=0$  表示舱体壁及舱内所有固定件； $m_i$  表示待布物  $i$  的质量； $\omega$  表示卫星舱角速度； $d$  表示承载板厚度； $H_o$  表示承载板距舱体底面的高度； $H_t$  表示舱体总体高度； $R_o$  表示承载板半径； $R_t$  表示舱体顶端圆半径； $R_b$  表示舱体底面半径； $\beta_i$  表示待布物所处

基面参数，在上基面为  $\beta_i=1$ ，在下基面为  $\beta_i=-1$ 。

### 7.18.2 算法设计

将蚁群算法和遗传算法相结合构成蚁群/遗传混合算法，便于将人工解与算法解都转化为遗传算法编码串的统一表达形式，构成解群，共同参与算法操作，以实现人工个体与算法个体在基因层面上的结合，并给出人工和自动加入人工方案的方式。

人与算法如何结合关键在于将人工设计的方案（人工解）和算法解相结合，这里蚁群/遗传算法的解（个体）是用编码串的形式表示的。首先应将人工方案转化为编码串形式构成人工个体（人工解），然后将人工个体加入到算法的解（个体，以编码串形式表达）中构成解群，共同参与蚁群/遗传算法操作，人工个体与算法个体的结合如图 7.64 所示<sup>[221]</sup>。

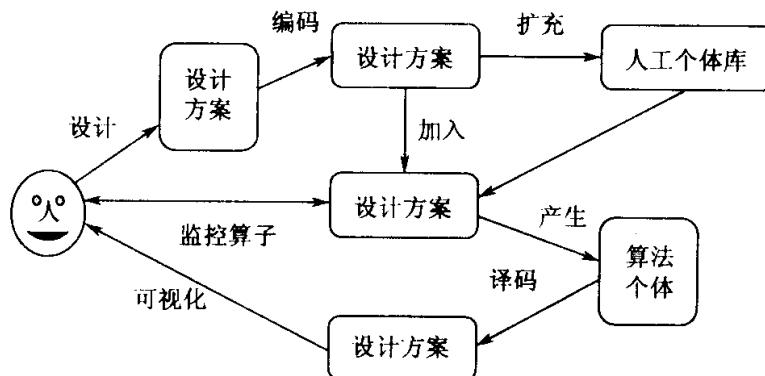


图 7.64 人工个体与算法个体结合图

人工个体添加的时机也非常重要，关系到算法的收敛速度和是否早熟等重要问题。通过定义规则库和基于事例推理，或借助人机界面来决定人工个体添加的时机。规则库定义如下：

- (1) 算法运行初期、中期，某子群中超过一定数量的个体与最优个体的解分布空间相邻，即此群体的多样性较低。
- (2) 算法运行初期，连续进化  $k$  代，最优个体的值停滞不变。
- (3) 算法运行中后期，最优个体值与结束准则的误差在一定范围内。

当规则库中条件(1)或(2)为真，可增加新人工方案（人工个体）以提高群体多样性；当规则库中条件(3)为真，应增加与最优个体值相近的人工个体，以加快算法收敛，即借助人机界面微调最优个体的方案以构造与其相近的人工个体。

以上规则的判断，可由人通过人机界面和经验判断，也可由计算机根据算法

迭代过程中的有关数据来判断，从而实现了人工和自动两种输入方式。

HCAGA 的程序结构流程如图 7.65 所示。

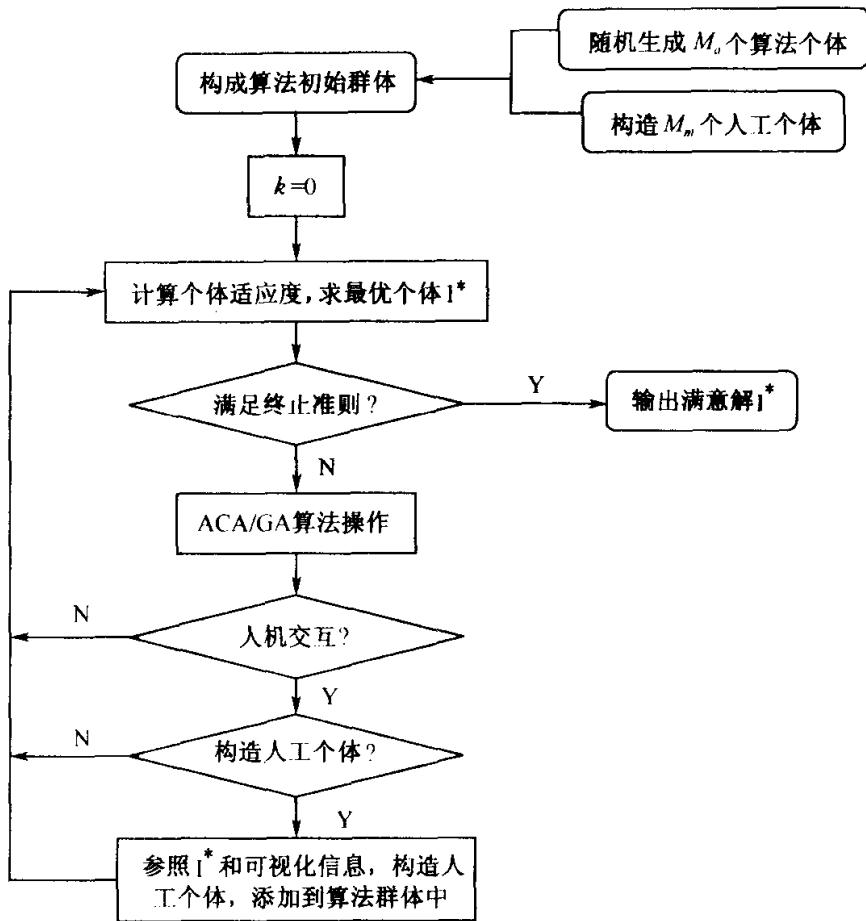


图 7.65 HCAGA 的程序结构流程

对于二维问题，设计人工个体的操作较为简单，但通过人机交互界面用鼠标移动待布物形心位置，计算机便可自动记录待布物位置（形心等）坐标值，形成人工解。一般而言，人工设计方案无法实时地输入到算法中，所以可设计人工个体库将人工个体存储起来，以便加快后续的人机交互过程。用所加人工个体替换子群体中适应度较差的算法个体，以加快收敛，或者替换适应度相近的一些算法个体，增加群体多样性，以探索新的解空间。

### 7.18.3 算例分析

人造卫星舱的简化模型如图 7.63 所示，卫星舱旋转角速度  $\omega = 40\text{r}/\text{min}$ ,  $R_b = 1000\text{mm}$ ,  $R_t = 600\text{mm}$ ,  $H_t = 1400\text{mm}$ ,  $H_o = 800\text{mm}$ ,  $d = 70\text{mm}$ 。要在承重板上、下基面上总计安装 19 个待布物，长方体数为 12，圆方体数为 1，长方体各边平行于坐标轴。利用 HCAGA、参考文献 [218] 的蚁群算法和并行遗传算法

(parallel genetic algorithm, PGA), 求解得到的待布物布局方案的各项性能指标、耗用相同时间三种算法优化的性能指标及三种算法求得相近的布局方案所耗用的时间分别如表 7.35~表 7.37 所示, 表中的总时间  $t$  包括交互时间  $t_1$ 。

表 7.35 布局方案的各项性能指标

算 法	外包络圆半径 $R$ (mm)	动不平衡力 $F$ (N)	动不平衡力矩 $M$ (Nm)	干涉量 $\Delta S$ ( $\text{mm}^2$ )	交互时间 $t_1$ (s)	总时间 $t$ (s)
PGA	643.02	12.31	14.04	0	0	5634.25
ACA	637.77	4.27	8.61	0	0	3836.42
HCAGA	614.56	1.85	3.93	0	584.80	2034.65

表 7.36 耗用相同时间三种算法优化的性能指标

算 法	外包络圆半径 $R$ (mm)	动不平衡力 $F$ (N)	动不平衡力矩 $M$ (Nm)	干涉量 $\Delta S$ ( $\text{mm}^2$ )	交互时间 $t_1$ (s)	总时间 $t$ (s)
PGA	669.50	14.53	17.64	2356.40	0	2034.65
ACA	645.60	6.53	10.29	1825.30	0	2034.65
HCAGA	614.56	1.85	3.93	0	584.80	2034.65

表 7.37 三种算法求得相近的布局方案所耗用的时间

算 法	外包络圆半径 $R$ (mm)	动不平衡力 $F$ (N)	动不平衡力矩 $M$ (Nm)	干涉量 $\Delta S$ ( $\text{mm}^2$ )	交互时间 $t_1$ (s)	总时间 $t$ (s)
PGA	643.02	12.31	14.04	0	0	5634.25
ACA	642.80	12.08	14.13	0	0	2956.40
HCAGA	640.05	12.25	13.98	0	470.60	856.80

由表 7.35~表 7.37 可见, 基于 HCAGA 所设计人造卫星舱布局方案的各项性能指标均优于 ACA 和 PGA, 从而验证了 HCAGA 可有效地发挥人机各自特长, 对卫星舱的布局方案实现了较好的优化设计。

## 7.19 本 章 小 结

本章主要对蚁群算法在车间作业调度问题、网络路由问题、车辆路径问题、机器人领域、电力系统、故障诊断、控制参数优化、参数辨识、聚类分析、数据挖掘、图像处理、航迹规划、空战决策、岩土工程、化学工业、生命科学、布局优化等若干领域中的典型应用情况做了系统而深入的介绍。从目前的应用情况来看, 蚁群算法在智能优化应用领域具有极强的适应性和生命力, 已经显示出极为广阔的发展前景。

## 参 考 文 献

- 1 Rodammer F A, White K P. Recent survey of production scheduling. *IEEE Transactions on System, Man, and Cybernetics*, 1998, 18(6): 841~851
- 2 Jackek B, Wolfgang D, Erwin P. The job shop scheduling problem: conventional and new solution techniques. *European Journal of Operational Research*, 1996, 93(1): 1~33
- 3 Yu I K, Chou C S, Song Y H. Application of the ant colony search algorithm to short-term generation scheduling problem of thermal units. *Proceedings of the 1998 International Conference on Power System Technology*, 1998, 1: 552~556
- 4 Liang Y C. Ant colony optimization approach to combinatorial problems. Ph. D. Thesis, Alabama: Auburn University, 2001
- 5 Cicirello V A, Smith S F. Ant colony control for autonomous decentralized shop floor routing. *Proceedings of the 5th International Symposium on Autonomous Decentralized Systems*, 2001, 383~390
- 6 Fournier O, Lopez P, Lan Sun Luk J D. Cyclic scheduling following the social behavior of ant colonies. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2002, 3: 450~454
- 7 Wang X R, Wu T J. Ant colony optimization for intelligent scheduling. *Proceedings of the 4th World Congress on Intelligent Control and Automation*, 2002, 1: 66~70
- 8 Pereira J, Bautista J. Ant algorithm for assembly line balancing. *Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science*, 2002, 2463: 65~75
- 9 Merkle D, Middendorf M, Schmeck H. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 2002, 6(4): 333~346
- 10 Zhou P, Li X P, Zhang H F. An ant colony algorithm for job shop scheduling problem. *Proceedings of the 5th World Congress on Intelligent Control and Automation*, 2004, 4: 2899~2903
- 11 Corry P, Kozan E. Ant colony optimization for machine layout problems. *Computational Optimization and Applications*, 2004, 28(3): 287~310
- 12 Ying K C, Liao C J. An ant colony system for permutation flow-shop sequencing. *Computers and Operations Research*, 2004, 31(5): 791~801
- 13 Rajendran C, Ziegler H. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, 2004, 155(2): 426~438
- 14 Solimanpur M, Vrat P, Shankar R. An ant algorithm for the single row layout problem in flexible manufacturing systems. *Computers & Operations Research*, 2005, 32(3): 583~598
- 15 Wang J F, Liu J H, Zhong Y F. A novel ant colony algorithm for assembly sequence planning. *International Journal of Advanced Manufacturing Technology*, 2005, 25(11~12): 1137~1143
- 16 李莉, 乔非, 吴启迪. 半导体生产线群体智能调度模型研究. *中国机械工程*, 2004, 15(22): 2006~2009, 2067
- 17 孙新宇<sup>\*</sup>·万筱宁<sup>\*</sup>孙林岩. 蚁群算法在混流装配线调度问题中的应用. *信息与控制*, 2002, 31(6): 486~490
- 18 赵伟, 韩文秀, 罗永泰. 准时生产方式下混流装配线的调度问题. *管理科学学报*, 2000, 3(4): 23~28
- 19 邹庆路, 罗欣, 杨叔子. 基于蚂蚁算法的混流车间动态调度研究. *计算机集成制造系统-CIMS*, 2003, 9(6): 456~459, 475
- 20 Gao Q L, Luo X, Yang S Z. Stigmergic cooperation mechanism for shop floor control system. *International Journal of Advanced Manufacturing Technology*, 2005, 25(7~8): 743~753

- 21 王常青, 操云甫, 戴国忠. 用双向收敛蚁群算法解作业车间调度问题. 计算机集成制造系统, 2004, 10(7): 820~824
- 22 Dorigo M, Stüenzle T. An experimental study of the simple ant colony optimization algorithm. Proceedings of the WSES International Conference on Evolutionary Computation, 2001, 253~258
- 23 Wang Z, Crowcroft J. Quality of service routing for supporting multimedia applications. IEEE Journal on Selected Areas in Communications, 1996, 14(7): 1228~1234
- 24 Chu C H, Gu J H, Hou X D, et al. A heuristic ant algorithm for solving QoS multicast routing problem. Proceedings of the 2002 Congress on Evolutionary Computation, 2002, 2: 1630~1635
- 25 Hussein O, Saadawi T. Ant routing algorithm for mobile ad-hoc networks (ARAMA), Proceedings of the 2003 IEEE International Conference on Performance, Computing and Communications Conference, 2003, 281~290
- 26 Lu G Y, Liu Z M. Multicast routing based on ant-algorithm with delay and delay variation constraints. Proceedings of the 2000 IEEE Asia-Pacific Conference on Circuits and Systems, 2000, 243~246
- 27 Lu G Y, Liu Z M, Zhou Z. Multicast routing based on ant algorithm for delay-bounded and load-balancing traffic. Proceedings of the 25th Annual IEEE Conference on Local Computer Networks, 2000, 362~368
- 28 Zhang S B, Liu Z M. A QoS routing algorithm based on ant algorithm. Proceedings of the IEEE International Conference on Communications, 2001, 5: 1581~1585
- 29 Zhang S B, Liu Z M. A QoS routing algorithm based on ant algorithm. Proceedings of the 25th Annual IEEE Conference on Local Computer Networks, 2000, 574~578
- 30 Lu G Y, Zhang S B, Liu Z M. Distributed dynamic routing using ant algorithm for telecommunication networks. Proceedings of the International Conference on Communication Technology, 2000, 2: 1607~1612
- 31 Sim K M, Sun W H. Multiple ant-colony optimization for network routing. Proceedings of the 1st International Symposium on Cyber Worlds, 2002, 277~281
- 32 Li L Y, Liu Z M, Zhou Z. A new dynamic distributed routing algorithm on telecommunication networks. Proceedings of the 2000 International Conference on Communication Technology, 2000, 1: 849~852
- 33 LIU Y, Wu J P, Xu K, et al. The degree-constrained multicasting algorithm using ant algorithm. Proceedings of the 10th International Conference on Telecommunications, 2003, 1: 370~374
- 34 Lu Y, Zhao G Z, Su F J. Adaptive ant-based dynamic routing algorithm. Proceedings of the 5th World Congress on Intelligent Control and Automation, 2004, 3: 2694~2697
- 35 Gunes M, Sorges U, Bouazizi I. ARA-the ant-colony based routing algorithm for MANETs. Proceedings of the International Conference on Parallel Processing Workshops, 2002, 79~85
- 36 Hsiao Y T, Chuang C L, Chien C C. Computer network load-balancing and routing by ant colony optimization. Proceedings of the 12th IEEE International Conference on Networks, 2004, 1: 313~318
- 37 Schoonderwoerd R, Holland O, Bruton J. Ant-based load balancing in telecommunications networks. Adaptive Behavior, 1996, 5(2): 169~207
- 38 Gianni Di Caro, Dorigo M. Ant-Net: distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 1998, 9: 317~365
- 39 纪竹亮, 戴连奎. 一种基于时延信息的多 QoS 快速自适应路由算法. 电路与系统学报, 2004, 9(4): 142~145
- 40 张素兵, 吕国英, 刘泽民. 基于蚂蚁算法的 QoS 路由调度方法. 电路与系统学报, 2000, 5(1): 1~5
- 41 张素兵, 刘泽民. 基于蚂蚁算法的分级 QoS 路由调度方法. 北京邮电大学学报, 2000, 23(4): 11~15
- 42 高坚. 基于自适应蚁群算法的多受限网络 QoS 路由优化. 计算机工程, 2003, 29(19): 40~41, 67
- 43 林国辉, 马正新, 王勇前等. 基于蚂蚁算法的拥塞规避路由算法. 清华大学学报, 2003, 43(1): 1~4

- 44 Schoonderwoerd R, Holland O, Bruton J, et al. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 1996, 5(2): 169~207
- 45 Bonabeau E, Theraulaz G. Swarm smarts. *Scientific American*, 2000, 6(2): 74~79
- 46 Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1993, 1(4): 397~413
- 47 Panta L. Modeling Transportation Problems using concepts of swarm intelligence and soft computing. Ph. D. Thesis, Virginia Polytechnic Institute and State University, 2002
- 48 Matos A C, Oliveira R C. An experimental study of the ant colony system. *Proceedings of the 4th International Workshop on Ant Algorithms/ANTS2004*, Lecture Notes in Computer Science, 2004, 3172: 286~293
- 49 Tatomir B, Kroon R, Rothkrantz I. Dynamic routing in traffic networks using ant net. *Proceedings of the 4th International Workshop on Ant Algorithms/ANTS2004*, Lecture Notes in Computer Science, 2004, 3172: 424~425
- 50 Bell J E, McMullen P R. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 2004, 18(1): 41~48
- 51 Donati A V, Montemanni R, Gambardella L M, et al. Integration of a robust shortest path algorithm with a time dependent vehicle routing model and applications. *Proceedings of the 2003 IEEE International Symposium on Computational Intelligence for Measurement Systems and Applications*, 2003, 26~31
- 52 Leith C, Takahara G. A control framework for ant-based routing algorithms. *Proceedings of the 2003 Congress on Evolutionary Computation*, 2003, 3: 1788~1795
- 53 Teodorovic D, Lucic P. Schedule synchronization in public transit using the fuzzy ant system. *Transportation Planning and Technology*, 2005, 28(1): 47~76
- 54 Wen Y, Wu T J. Regional signal coordinated control system based on an ant algorithm. *Proceedings of the 5th World Congress on Intelligent Control and Automation*, 2004, 6: 5222~5226
- 55 Hu J M, Yang Z S, Jian F. Study on the optimization methods of transit network based on ant algorithm. *Proceedings of the IEEE International Vehicle Electronics Conference*, 2001, 215~219
- 56 Kuo R J, Chiu C Y, Lin Y J. Integration of fuzzy theory and ant algorithm for vehicle routing problem with time window. *Proceedings of the IEEE Annual Meeting of the Fuzzy Information*, 2004, 2: 925~930
- 57 Ellabib I, Basir O A, Calamai P. An experimental study of a simple ant colony system for the vehicle routing problem with time windows. *Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002*, Lecture Notes in Computer Science, 2002, 2463: 53~64
- 58 Reimann M, Doerner K, Hartl R F. Insertion based ants for vehicle routing problems with backhauls and time windows. *Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002*, Lecture Notes in Computer Science, 2002, 2463: 135~148
- 59 Tian Y, Song J Y, Yao D Y, et al. Dynamic vehicle routing problem using hybrid ant system. *Proceedings of the 2003 IEEE Intelligent Transportation Systems*, 2003, 2: 970~974
- 60 马良, 姚俭, 范炳全. 蚂蚁算法在交通配流中的应用. *科技通报*, 2003, 19(5): 377~380
- 61 崔雪丽, 马良, 范炳全. 车辆路径问题的蚂蚁搜索算法. *系统工程学报*, 2004, 19(4): 418~422
- 62 刘志硕, 申金升, 柴跃廷. 基于自适应蚁群算法的车辆路径问题研究. *控制与决策*, 2005, 20(5): 562~566
- 63 Clark G, Wright J W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 1964, 12(4): 568~581

- 64 邱峰, 陈学广, 迟嘉昱. 一种改进的蚂蚁算法在多目标配路中的应用. 华中科技大学学报, 2003, 31(4): 39~41
- 65 万旭, 林健良, 杨晓伟. 改进的最大-最小蚂蚁算法在有时间窗车辆路径问题中的应用. 计算机集成制造系统, 2005, 11(4): 572~576
- 66 Stützle T, Hoos H H. Improving the ant system: a detailed report on the MAX-MIN ant system. Technical Report AIDA-96-11, FG Intellektik, TH Darmstadt, 1996
- 67 Stützle T, Hoos H. MAX-MIN ant system and local search for the traveling salesman problem. Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, 1997, 309~314
- 68 Thomas S, Holger H H. MAX-MIN ant system. Future Generation Computer Systems, 2000, 16(8): 889~914
- 69 Reimann M, Doerner K, Richard F H. D-Ants: savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research, 2004, 31(4): 563~591
- 70 Bullnheimer B, Hartl R, Strauss C. An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research, 1999, 89(13): 319~328
- 71 Brysy O, Dullaert W. A fast evolutionary metaheuristic for the vehicle routing problem with time windows. International Journal of Artificial Intelligence Tools, 2002, 12(2): 143~157
- 72 Marius M, Solomon M. Algorithms for vehicle routing and scheduling problems with time window constraints. Operations Research, 1987, 35(2): 763~781
- 73 Michael J B K, Jean-Bernard B, Laurent K. Ant-like task and recruitment in cooperative robots. Nature, 2000, 406(31): 992~995
- 74 Konishi M, Nishi T, Nakano K, et al. Evolutionary routing method for multi mobile robots in transportation. Proceedings of the 2002 IEEE International Symposium on Intelligent Control, 2002, 490~495
- 75 Hu Y, Jing T, Hong X L, et al. An-OARSMAN: obstacle-avoiding routing tree construction with good length performance. Proceedings of the Asia and South Pacific Design Automation Conference, 2005, 1: 7~12
- 76 Mucientes M, Casillas J. Obtaining a fuzzy controller with high interpretability in mobile robots navigation. Proceedings of the 2004 IEEE International Conference on Fuzzy Systems, 2004, 3: 1637~1642
- 77 Liu G Q, Li T J, Peng Y Q, et al. The Ant Algorithm for Solving Robot Path Planning Problem. Proceedings of the 3rd International Conference on Information Technology and Applications, 2005, 2: 25~27
- 78 Parker C A C, Zhang H. Biologically inspired decision making for collective robotic systems. Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, 1: 375~380
- 79 Hsiao Y T, Chuang C L, Chien C C. Ant colony optimization for best path planning. Proceedings of the IEEE International Symposium on Communications and Information Technology, 2004, 1: 109~113
- 80 Fan X P, Luo X, Yi S, et al. Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003, 1: 131~136
- 81 樊晓平, 罗熊, 易晟等. 复杂环境下基于蚁群优化算法的机器人路径规划. 控制与决策, 2004, 19(2): 166~170
- 82 金飞虎, 洪炳熔, 高庆吉. 基于蚁群算法的自由飞行空间机器人路径规划. 机器人, 2002, 24(6): 526~529
- 83 胡小兵, 黄席樾. 基于蚁群算法的三维空间机器人路径规划. 重庆大学学报, 2004, 27(8): 132~135

- 84 朱庆保, 张玉兰. 基于栅格法的机器人路径规划蚁群算法. 机器人, 2005, 27(2): 132~136
- 85 丁滢颖, 何衍, 蒋静坪. 基于蚁群算法的多机器人协作策略. 机器人, 2003, 25(5): 414~418
- 86 Ding Y Y, He Y, Jiang J P. Multi-robot cooperation method based on the ant algorithm. Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, 14~18
- 87 Brooks R A. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, 1986, RA-2(1): 14~23
- 88 Miranda V, Ranito J V, Proenco L M. Genetic algorithms in optimal multistage distribution network planning. IEEE Transactions on Power System, 1994, 9(11): 1927~1933
- 89 Suresh K K, Lawrence C L. Power distribution planning: a review of models and issues. IEEE Transactions on Power System, 1997, 12(3): 1151~1159
- 90 Dai H W, Yu Y X, Huang C H, et al. Optimal planning of distribution substation locations and sizes-model and algorithm. International Journal of Electrical Power and Energy System, 1996, 18(6): 353~357
- 91 Ruzic S, Rajakovic N. A new approach for solving extended unit commit problem. IEEE Transactions on Power System, 1991, 6(2): 269~277
- 92 Cheng C P, Liu C W, Liu C C. Unit commitment by Lagrangian relaxation and genetic algorithm. IEEE Transactions on Power System, 2000, 15(2): 707~714
- 93 Ouyang Z, Shahidehpour S M. A multi-stage intelligent system for unit commitment. IEEE Transactions on Power Systems, 1992, 7(2): 639~646
- 94 Zhuang F, Galiana F D. Unit commitment by simulated annealing. IEEE Transactions on Power Systems, 1990, 5(1): 311~318
- 95 Gomez J F, Khodr H M, De Oliveira P M, et al. Ant colony system algorithm for the planning of primary distribution circuits. IEEE Transactions on Power Systems, 2004, 19(2): 996~1004
- 96 Huang S J. Enhancement of hydroelectric generation scheduling using ant colony system based optimization approaches. IEEE Transactions on Energy Conversion, 2001, 16(3): 296~301
- 97 Pahwa A, Chavali S, Das S. Intelligent computational methods for power systems optimization problems. Proceedings of the 2003 IEEE Power Engineering Society General Meeting, 2003, 1: 135~138
- 98 Ippolito M G, Sanseverino E R, Vuinovich F. Multi-objective ant colony search algorithm optimal electrical distribution system planning. Proceedings of the 2004 Congress on Evolutionary Computation, 2004, 2: 1924~1931
- 99 侯云鹤, 熊信良, 吴耀武等. 基于广义蚁群算法的电力系统经济负荷分配. 中国电机工程学报, 2003, 23(3): 59~64
- 100 高炜欣, 罗先觉. 基于蚂蚁算法的配电网网络规划. 中国电机工程学报, 2004, 24(9): 110~114
- 101 翟海保, 程浩忠, 吕干云等. 多阶段输电网络最优规划的并行蚁群算法. 电力系统自动化, 2004, 28(20): 37~42
- 102 郝晋, 石立宝, 周家启等. 基于蚁群优化算法的机组最优投入. 电网技术, 2002, 26(11): 26~31
- 103 陈皓勇, 王锡凡. 机组组合问题的优化方法综述. 电力系统自动化, 1997, 23(4): 71~75
- 104 Chang C S, Tian L, Wen F S. A new approach to fault section estimation in power systems using ant system. Electric Power Systems Research, 1999, 49(1): 63~70
- 105 孙京浩, 李秋艳, 杨欣斌等. 基于蚁群算法的故障识别. 华东理工大学学报, 2004, 30(2): 194~198
- 106 覃方君, 田蔚风, 李安等. 基于蚁群算法的复杂系统多故障状态的决策. 中国惯性技术学报, 2004, 12(4): 12~15

- 107 樊友平, 陈允平, 黄席樾等. 运载火箭控制系统漏电故障诊断研究. 宇航学报, 2004, 25(5): 507~513
- 108 边肇祺, 张学工. 模式识别. 北京: 清华大学出版社, 2000
- 109 Cameron F, Seborg D E. A self tuning controller with a PID structure. International Journal of Control, 1983, 38(2): 831~835
- 110 Tor S T. A method for closed loop automatic tuning of PID controllers. Automatica, 1992, 28(3): 587~591
- 111 Majhi S, Litz L. On-line tuning of PID controllers. Proceedings of the 2003 American Control Conference, 2003, 6: 5003~5004
- 112 Astrom K J, Hagglund T. PID controllers: theory, design and tuning. Instrument Society of America Press, Research Triangle Park, NC, Second edition, 1995
- 113 Hang C C, Astrom K J, Ho W K. Refinements of the Ziegler-Nichols tuning formula. IEE Proceedings of Control Theory Application (Series D), 1991, 138: 111~118
- 114 Varol H A, Bingul Z. A new PID tuning technique using ant algorithm. Proceedings of the 2004 American Control Conference, 2004, 3: 2154~2159
- 115 Hsiao Y T, Chuang C L, Chien C C. Ant colony optimization for designing of PID controllers. Proceedings of the 2004 IEEE International Symposium on Computer Aided Control Systems Design, 2004, 321~326
- 116 Tan G Z, Zeng Q D, Li W B. Intelligent PID controller based on ant system algorithm and fuzzy inference and its application to bionic artificial leg. Journal of Central South University of Technology (English Edition), 2004, 11(3): 316~322
- 117 谭冠政, 李文斌. 基于蚁群算法的智能人工腿最优 PID 控制器设计. 中南大学学报, 2004, 35(1): 91~96
- 118 韩京清. 非线性 PID 控制器. 自动化学报, 1994, 20(4): 487~490
- 119 韩京清, 王伟. 非线性跟踪微分器. 系统科学与数学, 1994, 14(2): 177~183
- 120 Huang H P, Han J Q. Nonlinear PID controller and its applications in power plants. Proceedings of the 2002 International Conference on Power System Technology, 2002, 3: 1513~1517
- 121 段海滨, 王道波, 尧放哉. 一种大负载三轴电动仿真测试转台设计与实现. 南京航空航天大学学报, 2004, 3(3): 358~363
- 122 段海滨, 王道波, 于秀芬等. 基于蚁群算法的 PID 参数优化. 武汉大学学报, 2004, 37(5): 97~100
- 123 段海滨, 王道波, 于秀芬. 基于改进蚁群算法的飞行仿真转台 NLPID 控制参数优化研究. 航空学报, 待发表
- 124 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 125 Zheng Y L, Ma L H, Zhang L Y, et al. Robust PID controller design using particle swarm optimizer. Proceedings of the 2003 IEEE International Symposium on Intelligent Control, 2003, 974~979
- 126 Gaing Z L. A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE Transactions on Energy Conversion, 2004, 19(2): 384~391
- 127 Liu Y J, Zhang J M, Wang S Q. Optimization design based on PSO algorithm for PID controller. Proceedings of the 5th World Congress on Intelligent Control and Automation, 2004, 3: 2419~2422
- 128 Graham D, Lathrop L C. The synthesis of optimal transient response: criteria and standard forms. Transactions on AIEE-Part II, 1953, 72: 273~288
- 129 王颖, 谢剑英. 一种自适应蚁群算法及其仿真研究. 系统仿真学报, 2002, 14(1): 31~33

- 130 Abbaspour K C, Schulin R, van Genuchten M Th. Estimating unsaturated soil hydraulic parameters using ant colony optimization. *Advances in Water Resources*, 2001, 24(8): 827~841
- 131 Ramos V, Abraham A. Swarms on continuous data. *Proceedings of the 2003 Congress on Evolutionary Computation*, 2003, 2: 1370~1375
- 132 Wang L, Wu Q D. Linear system parameters identification based on ant system algorithm. *Proceedings of the IEEE Conference on Control Applications*, 2001, 401~406
- 133 汪镭, 吴启迪. 蚁群算法在系统辨识中的应用. *自动化学报*, 2003, 29(1): 102~109
- 134 Canudas de Wit C, Olsson H, Astrom K J, et al. A new model for control of systems with friction. *IEEE Transactions on Automatic Control*, 1995, 40(3): 419~425
- 135 Canudas de Wit C, Lischinsky P. Adaptive friction compensation with partially known dynamic friction model. *International Journal of Adaptive Control and Signal Processing*, 1997, 11: 65~80
- 136 Duan H B, Wang D B, Zhu J Q, et al. Parameter identification of LuGre friction model for flight simulation servo system based on ant colony algorithm. *Transactions of Nanjing University of Aeronautics & Astronautics*, 2004, 21(3): 179~183
- 137 Duan H B, Wang D B, Zhu J Q. A novel method based on ant colony optimization algorithm for solving ill-conditioned linear systems of equations. *Journal of System Engineering and Electronic*, 2005, 16 (3): 606~610
- 138 Schockaert S, Cock M D, Cornelis C. Fuzzy ant based clustering. *Proceedings of the 4th International Workshop on Ant Algorithms/ANTS2004*, Lecture Notes in Computer Science, 2004, 3172: 342~349
- 139 Shelokar P S, Jayaraman V K, Kulkarni B D. An ant colony approach for clustering. *Analytica Chimica Acta*, 2004, 509(2): 187~195
- 140 Yang X B, Sun J G, Huang D. A new clustering method based on ant colony algorithm. *Proceedings of the 4th World Congress on Intelligent Control and Automation*, 2002, 3: 2222~2226
- 141 Yang Y, Kamel M. Clustering ensemble using swarm intelligence. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 2003, 65~71
- 142 Chen L, Xu X H, Chen Y X, et al. A novel ant clustering algorithm based on cellular automata. *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2004, 148~154
- 143 Xiao J T. Applying the ant colony optimization algorithm to the spatial cluster scheduling problem. *Proceedings of the 2004 International Conference on Machine Learning and Cybernetics*, 2004, 3: 1341~1346
- 144 丁建立, 陈增强, 袁著祉. 基于动态聚类邻域分区的并行蚁群优化算法. *系统工程理论与实践*, 2003, 23(9): 105~110
- 145 陈卓, 孟庆春, 魏振钢. 基于群体智能理论的聚类模型及优化算法. *计算机工程*, 2005, 31(4): 34~36
- 146 Deneubourg J L, Goss S, Franks N, et al. The dynamics of collective sorting: robot-like ants and ant-like robots. *Proceedings of the 1st International Conference on Simulation of Adaptive Behavior: From Animals to Animals*, 1991
- 147 Dorigo M, Bonabeaub E, Theraulaz G. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 2000, 16(8): 851~871
- 148 Lumer E, Faieta B. Diversity and adaptation in populations of clustering ants. *Proceedings of the 3rd international conference on simulation of adaptive behavior: from animals to animals*, 1994, 499~508
- 149 Wu B, Shi Z Z. A clustering algorithm based on swarm intelligence. *Proceedings of the IEEE international conferences on info-tech and info-net proceeding*, 2001, 3: 58~66

- 150 Bonabeau E, Dorigo M, Theraulaz G. *Swarm intelligence—from natural to artificial system*. New York: Oxford University Press, 1999
- 151 Ramos V, Merelo J J. Self-organized stigmergic document maps: environment as a mechanism for context learning. Proceedings of the AEB'2002-1st Spanish conference on evolutionary and bio-inspired algorithms, 2002, 284~293
- 152 高坚. 基于并行多种群自适应蚁群算法的聚类分析. 计算机工程与应用, 2003, 39(25): 78~79, 82
- 153 杨燕, 斯蕃, Mohamed K. 一种基于蚁群算法的聚类组合方法. 铁道学报, 2004, 26(4): 64~69
- 154 Strehl A, Ghosh J. Cluster ensembles—a knowledge reuse framework for combining partitionings. Proceedings of Artificial Intelligence, 2002, 93~98
- 155 Parpinelli R S, Lopes H S, Freitas A A. Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 321~332
- 156 Tsai C F, Tsai C W, Wu H C, et al. ACODF: a novel data clustering approach for data mining in large databases. Journal of Systems and Software, 2004, 73(1): 133~145
- 157 Admane L, Benatchba K, Koudil M, et al. Using ant colonies to solve data-mining problems. Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, 2004, 4: 3151~3157
- 158 刘波, 潘久辉. 基于群体智能的分布式数据挖掘方法. 计算机工程, 2005, 31(8): 145~147
- 159 Ge Y, Meng Q C, Yan C J, et al. A hybrid ant colony algorithm for global optimization of continuous multi-extreme functions. Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, 2004, 4: 2427~2432
- 160 Zheng H, Wong A, Nahavandi S. Hybrid ant colony algorithm for texture classification. Proceedings of the 2003 Congress on Evolutionary Computation, 2003, 4: 2648~2652
- 161 Zheng H, Zheng Z B, Xiang Y. The application of ant colony system to image texture classification. Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, 2003, 3: 1491~1495
- 162 Ouadfel S, Batouche M. Ant colony system with local search for Markov random field image segmentation. Proceedings of the 2003 International Conference on Image Processing, 2003, 1: 133~136
- 163 Meshoul S, Batouche M. Ant colony system with extremal dynamics for point matching and pose estimation. Proceedings of the 16th International Conference on Pattern Recognition, 2002, 3: 823~826
- 164 Li X, Luo X H, Zhang J H. Modeling of vector quantization image coding in an ant colony system. Chinese Journal of Electronics, 2004, 13(2): 305~307
- 165 Li X, Luo X H, Zhang J H. Codebook design by a hybridization of ant colony with improved LBG algorithm. Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003, 1: 469~472
- 166 Zhuang X. Image feature extraction with the perceptual graph based on the ant colony system. Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, 2004, 7: 6354~6359
- 167 Zhuang X. Edge feature extraction in digital images with the ant colony system. Proceedings of the 2004 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2004, 133~136
- 168 燕忠, 袁春伟. 基于蚁群智能和支持向量机的人脸性别分类方法. 电子与信息学报, 2004, 26(8): 1177~1182
- 169 韩彦芳, 施鹏飞. 基于蚁群算法的图像分割方法. 计算机工程与应用, 2004, 40(18): 5~7

- 170 郑肇葆, 叶志伟. 基于蚁群行为仿真的影像纹理分类. 武汉大学学报, 2004, 29(8): 669~673
- 171 杜萍, 杨春. 飞行器航迹规划算法综述. 飞行力学, 2005, 23(2): 10~14
- 172 叶文, 范洪达. 基于改进蚁群算法的飞机低空突防航路规划. 飞行力学, 2004, 22(3): 35~38
- 173 柳长安, 李为吉, 王和平. 基于蚁群算法的无人机航路规划. 空军工程大学学报, 2004, 5(2): 9~12
- 174 王和平, 柳长安, 李为吉. 基于蚁群算法的无人机任务规划. 西北工业大学学报, 2005, 23(1): 98~101
- 175 Ma T, Abbott B. Optimal route re-planning for mobile robots: a massively parallel incremental algorithm. Proceedings of the IEEE Conference on Robotics and Automation, 1997, 2727~2733
- 176 Timothy W M, Randal W B. Trajectory planning for coordinated rendezvous of unmanned air vehicles. AIAA-2000-4339-CP, 2000
- 177 Abbattista F, Abbattista N, Caponetti L. An evolutionary and cooperative agents model for optimization. Proceedings of the IEEE International Conference on Evolutionary Computation, 1995, 2: 668~671
- 178 Luo D L, Duan H B. Research on air combat decision-making for cooperative multiple target attack using heuristic ant colony algorithm. Chinese Journal of Astronautics, to appear
- 179 Huynh H T, Costes P, Aumasson C. Numerical optimization of air combat maneuvers. AIAA-87-2392, 1987
- 180 Austin F. Game theory for automated maneuvering during air-to-air combat. Guidance, 1990, 13(6), 1143~1147
- 181 Lazarus E. The application of value-driven decision-making in air combat simulation. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics & Computational Cybernetics and Simulation, 1997, 3: 2302~2307
- 182 Dorigo M, Caro G D. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66
- 183 段海滨, 王道波. 一种快速全局优化的改进蚁群算法及仿真. 信息与控制, 2004, 33(2): 241~244
- 184 Lee Z J, Lee C Y, Su S F. An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem. Applied Soft Computing, 2002, 2(1): 39~47
- 185 Mulgund S, Harper K, Krishnakumar K, et al. Air combat tactics optimization using stochastic genetic algorithms. Proceedings of the IEEE International Conference on System, Man and Cybernetics, 1998, 4: 3136~3141
- 186 高玮, 郑颖人. 蚁群算法及其在硐群施工优化中的应用. 岩石力学与工程学报, 2002, 21(4): 471~474
- 187 陈昌富, 龚晓南, 王贻荪. 自适应蚁群算法及其在边坡工程中的应用. 浙江大学学报, 2003, 37(5): 566~569
- 188 王成华, 夏续勇, 李广信. 基于应力场的土坡临界滑动面的蚂蚁算法搜索技术. 岩石力学与工程学报, 2003, 22(5): 813~819
- 189 陈昌富, 龚晓南. 混沌扰动启发式蚁群算法及其在边坡非圆弧临界滑动面搜索中的应用. 岩石力学与工程学报, 2004, 23(20): 3450~3453
- 190 陈昌富, 龚晓南. 启发式蚁群算法及其在高填石路堤稳定性分析中的应用. 数学的实践与认识, 2004, 34(6): 89~92
- 191 李亮, 迟世春, 林皋. 基于蚁群算法的复合形法及其在边坡稳定分析中的应用. 岩土工程学报, 2004, 26(5): 691~696

- 192 陈昌富, 谢学斌. 露天采场边坡临界滑动面搜索蚁群算法研究. 湘潭矿业学院学报, 2002, 17(1): 62~64
- 193 丁亚平, 刘平阳, 苏庆德等. 化学蚁群算法及其在光谱解析中的应用. 计算机与应用化学, 2002, 19(3): 326~328
- 194 丁亚平, 苏庆德, 吴庆生. 化学蚁群算法与多组分导数荧光光谱解析. 高等学校化学学报, 2002, 23(9): 1695~1697
- 195 Shelokar P S, Jayaraman V K, Kulkarni B D. An ant colony classifier system: application to some process engineering problems. Computers & Chemical Engineering, 2004, 28(9): 1577~1584
- 196 贺益君, 陈德钊, 吴晓华. 杂交蚁群系统的构建并用于反应动力学参数的估计. 化工学报, 2005, 56(3): 487~491
- 197 颜学峰, 陈德钊, 胡上序等. 混沌遗传算法估计反应动力学参数. 化工学报, 2002, 53(8): 810~814
- 198 Li R, Savage P E, Szmukler D. 2-chlorophenol oxidation in supercritical water: global kinetics and reaction products. AIChE Journal, 1993, 39(1): 178~187
- 199 Berger B, Leighton T. Protein folding in the Hydrophilic-Hydrophobic (HP) model is NP-Complete. Journal of Computational Biology, 1998, 5(1): 27~40
- 200 Hart W E, Istrail S. Robust proofs of NP-hardness for protein folding general lattices and energy potentials. Journal of Computational Biology, 1997, 4(1): 1~22
- 201 Krasnogor N, Hart W E, Smith J, et al. Protein structure prediction with evolutionary algorithms. Proceedings of the Genetic and Evolutionary Computation Conference, 1999, 1596~1601
- 202 Liang F, Wong W H. Evolutionary Monte Carlo for protein folding simulations. Journal of Chemistry Physics, 2001, 115(7): 3374~3380
- 203 Meksangsouy P, Chaiyaratana N. DNA fragment assembly using an ant colony system algorithm. Proceedings of the 2003 Congress on Evolutionary Computation, 2003, 3: 1756~1763
- 204 Chu D, Till M, Zomaya A. Parallel ant colony optimization for 3D protein structure prediction using the HP lattice model. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005, 1~7
- 205 Ando S, Iba H. Ant algorithm for construction of evolutionary tree. Proceedings of the 2002 Congress on Evolutionary Computation, 2002, 2: 1552~1557
- 206 梁栋, 霍红卫. 自适应蚁群算法在序列比对中的应用. 计算机仿真, 2005, 22(1): 100~102, 106
- 207 李冬冬, 王正志, 杜耀华等. 蚂蚁群落优化算法在蛋白质折叠二维亲-疏水格点模型中的应用. 生物物理学报, 2004, 20(5): 371~374
- 208 何莲莲, 石峰, 周怀北. 改进的蚁群算法在2DHP模型中的应用. 武汉大学学报, 2005, 51(1): 33~38
- 209 Shmygelska A, Aguine-Hernandez R, Hoos H H. An ant colony algorithm for the 2D HP protein folding problem. Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science, 2002, 2463: 40~52
- 210 Shmygelska A, Hoos H H. An improved ant colony optimization algorithm for the 2D HP protein folding problem. Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence, Lecture Notes in Computer Science, 2003, 2671: 400~417
- 211 Kamran D, Maziar A, Hossein S F. "FARAGAM" algorithm in satellite layout. Proceedings of the 6th Asia-Pacific Conference on Multilateral Cooperation in Space Technology and Application, 2001, 120~127

- 212 Braun R D, Moore A A, Kroo I M. Collaborative approach to launch vehicle design. *Journal of Spacecraft and Rockets*, 1997, 34(4): 478~486
- 213 Potts C, Catledge L. Collaborative conceptual design: a large software project case study. *Computer Supported Cooperative Work: an International Journal*, 1996, 5(4): 415~445
- 214 Teng H F, Sun S L, Liu D Q, *et al.* Layout optimization for the objects located within a rotating vessel: a three-dimensional packing problem with behavioral constraints. *Computer and Operations Research*, 2001, 28(6): 521~535
- 215 钱学森, 于景元, 戴汝为. 一个科学新领域—开放的复杂巨系统及其方法论. *自然杂志*, 1990, 13(1): 3~10
- 216 戴汝为, 王珏. 巨型智能系统的探讨. *自动化学报*, 1993, 19(6): 645~655
- 217 Lenat D B, Feigenbaum E A. On the thresholds of knowledge. *Proceedings of the IEEE International Workshop on Artificial Intelligence for Industrial Applications*, 1998, 291~300
- 218 Sun Z G, Teng H F. Optimal layout design of a satellite module. *Engineering Optimization*, 2003, 35(6): 513~529
- 219 Sun Z G, Teng H F. An ant colony optimization based layout optimization algorithm. *Proceedings of the 2002 IEEE Conference on Computers, Communications, Control and Power Engineering*, 2002, 1: 675~678
- 220 霍军周, 李广强, 腾弘飞等. 人机结合蚁群/遗传算法及其在卫星舱布局设计中的应用. *机械工程学报*, 2005, 41(3): 112~116
- 221 Qian Z Q, Teng H F, Xiong D L, *et al.* Human-computer cooperation genetic algorithm and its application to layout design. *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, 2002, 299~302

## 第 8 章 蚁群算法的硬件实现

### 8.1 引言

随着对蚁群算法研究的不断深入，人们已开始关注蚁群算法的硬件实现这一新的研究方向。蚁群算法的硬件实现是仿生硬件（bio-inspired hardware, BHW）领域内的一个分支，也是蚁群算法发展的高级阶段。若要用硬件实现类似蚁群这一表现出复杂群体行为的系统，首先就得构造具有单只蚂蚁功能的智能体。

Isaacs J C 等<sup>[1]</sup>将遗传算法和蚁群算法结合，提出了一种嵌入式硬件随机数据发生器设计的新思路，但是他们只是做了离线仿真，并没有在硬件上给予实现。忻斌健等<sup>[2]</sup>提出了一种将归类结构（subsumption architecture）方法与现场可编程门阵列（field programmable gate array, FPGA）设计相结合的蚂蚁智能体硬件实现设想，归类结构是一种行为控制法（behavior-control paradigm, BCP），它将自动控制的问题按任务而不是按功能进行分解，主张构造面向任务的专用模块，直接与传感器和执行器相连，其工作方式是并行的；Scheuermann B 等<sup>[3, 4]</sup>在深入分析了将蚁群算法映射到 FPGA 难点的基础上，提出了一种基于群体-蚁群优化（population-based ant colony optimization, P-ACO）算法的仿生硬件实现方案，并详细给出了 P-ACO 算法 FPGA 硬件总体结构中各主要模块的实现过程；Xiong Z H 等<sup>[5, 6]</sup>提出了遗传算法与蚁群算法动态融合的软硬件划分算法（dynamic combination of genetic algorithm and ant algorithm, DCG3A），其基本思想是利用遗传算法群体性、全局随机搜索等优势生成初始划分解，并将其转化为蚁群算法所需的初始信息量分布。实验表明 DCG3A 可非常有效地解决嵌入式系统的软硬件划分问题，且划分问题规模越大，其优势越明显。

蚁群算法的硬件实现是当今仿生硬件研究领域中一个新的研究内容<sup>[7]</sup>，也是一个研究难点。本章首先介绍了仿生硬件的一些基本概念、发展概况、基本原理以及主要特点，随后着重研究了基于 FPGA 的蚁群算法硬件实现问题和基于 DCG3A 的软硬件划分问题。

### 8.2 仿生硬件概述

本节将对仿生硬件的定义、发展概况、基本原理及其主要特点作简要阐述。

### 8.2.1 仿生硬件的定义

仿生硬件是一种能根据外部环境的变化而自主地、动态地改变自身的结构和行为以适应其生存环境的硬件电路，它可以像生物一样具有硬件自适应、自组织和自修复特性。仿生硬件早期也称为演化硬件（evolvable hardware, EHW）。狭义上，仿生硬件是通过进化机制来实现电子电路系统的实时自身重构；广义上，仿生硬件包括各种形式的硬件，从传感器到能够适应变化的环境，且在运行期间增强其性能的整个电子系统、微机电系统。除了自组织生成具有新功能的电路外，仿生硬件还可用于保持现有功能、获得生物性容错以及实现硬件的实时“康复”。

### 8.2.2 仿生硬件的发展概况

早在 20 世纪 60 年代，计算机之父 Neumann J Von 就提出了研制具有自繁殖与自修复能力通用机器的伟大构想<sup>[8]</sup>。1992 年，日本学者 Garis H de 和瑞士联邦工学院的科学家们同时提出了将 FPGA 的结构可重配置特性与进化算法相结合的方案<sup>[9]</sup>，标志着仿生硬件这一新兴研究领域的正式诞生。随后仿生硬件的研究得到了迅猛发展，很多国内外学者投身其中，并于 1995 年 10 月在瑞士洛桑召开了仿生硬件国际专题会议（Towards Evolvable Hardware），1996 年 10 月在日本召开了首届进化系统国际大会（ICES96），以后该会议每隔两年举行一次。同时，鉴于这一新兴研究领域可望在空间探索和国防应用中产生巨大影响，美国航空航天局（National Aeronautics and Space Administration, NASA）和国防部（Department of Defense, DoD）对仿生硬件也表现出极大的兴趣<sup>[10, 11]</sup>，并于 1999 年召开了首届 NASA/DoD 仿生硬件专题讨论会，之后每年都要举行一次类似的讨论会，以促进仿生硬件的理论研究和应用开发，其研究目标是研制用于航天飞机、宇宙飞船、空间探测器、人造卫星、战略飞机、核潜艇的自适应与自修复电子系统和测控系统。

### 8.2.3 仿生硬件的基本原理

仿生硬件模拟自然进化过程，将仿生优化算法的思想用于硬件物理结构的设计，特别是电子系统的设计。仿生优化算法的发展和快速可重配置硬件的出现极大地推动和促进了仿生硬件的发展<sup>[12~15]</sup>。所以有学者对仿生硬件提出了这样一个公式性的描述

$$\text{BHW} = \text{BAs} + \text{PLDs}$$

即

## 仿生硬件=仿生算法+可编程逻辑器件

仿生硬件的硬件基础是可编程逻辑器件 (programmable logic device, PLD)。由于仿生进化过程具有随机性且要进行很多次，因此要求相应的硬件能够被多次反复配置，所以更严格地说，可重配置硬件 (reconfigurable hardware) 是仿生硬件的硬件基础，而可无穷次重复配置的静态随机存储器 (static random access memory, SRAM) 型 FPGA 是比较理想的实现设备。

传统的基于 FPGA 的硬件设计是一种自顶向下的设计流程，这其中还要在各个不同层次进行仿真模拟，以保证设计的正确性。将 PLD 能够形成的所有功能电路的集合作为一个空间，利用仿生优化算法在这个空间内对问题进行求解，以找到满足预定功能的硬件结构，此即仿生硬件的基本原理。

对仿生硬件设计个体的评价往往是比较费时的，随着输入变量的增加，评价的输入所有的可能组合数是成指数增长的，另一方面由于不当的输入可能会造成硬件的物理损坏，所以这就要求硬件本身具有较强的容错性和鲁棒性，同时也要求从算法上保证生成的个体都是合理的。因此，可以将仿生硬件的实现条件归结为

- (1) 硬件支持运行时间的动态重配置，包括结构、连线等。
- (2) 硬件结构要公开，能够知道硬件内部结构的配置方法和配置位串的信息。
- (3) 硬件要有一定的容错性和鲁棒性。
- (4) 要有快速的硬件设计流程，包括工艺映射、布局、布线及下载等。

### 8.2.4 仿生硬件的主要特点<sup>[16, 17]</sup>

下面介绍仿生硬件的主要特点。

#### 1. 硬件自组织

基于仿生进化机理的仿生硬件无须人员干预，可实现硬件的自组织、自动化设计，能通过硬件自身的在线进化过程来获得具备预期功能的电路和系统结构。利用仿生硬件能实现硬件的标准化、可重用性、多用途以及通用化。

#### 2. 硬件自适应

仿生硬件能满足环境变化对硬件所要求的结构与功能自适应性，而目前传统的硬件电路无法实现这种结构自适应。仿生硬件通过电路结构与参数的在线自适应调整，可有效解决如因元件逐步老化、环境改变而引起电路性能变化的问题，也可组成高速并行信息处理的自适应硬件系统。

#### 3. 硬件自修复

目前电子系统的结构越来越复杂，系统集成化是必然趋势。如果高度集成化

的嵌入式电子系统（如片上系统（system-on-a-chip, SoC））在运行中出现故障，传统的容错与系统功能恢复方法（基于板级重构技术的故障检测、故障定位、硬件冗余）难以实现，而且这种板级部件冗余容错系统结构复杂，其重构算法繁琐、硬件体积较大、成本较高、容错能力与系统可靠性有限；而仿生硬件可以弥补这些不足。

#### 4. 执行速度快

仿生硬件自适应的结果是新的硬件结构自身，因此，与其他基于软件的自适应系统相比，仿生硬件能得到显著的加速，而这一优点正是实时系统所需要的。

### 8.3 基于 FPGA 的蚁群算法硬件实现

本节研究了一种基于 FPGA 的蚁群算法硬件实现方案。首先在介绍 FPGA 基本结构的基础上，分析了将蚁群算法映射到 FPGA 的难点；然后介绍了一种改进蚁群算法——P-ACO 算法<sup>[18]</sup>，并给出了 P-ACO 算法硬件实现的总体结构；紧接着详细阐述了 P-ACO 算法硬件总体结构中各主要模块的实现过程；随后将基于硬件实现的 P-ACO 算法同基于软件实现的 P-ACO 算法作了实验对比；最后在实验分析的基础上，对 P-ACO 算法的硬件实现做了一些改进，同时提出了降低 P-ACO 算法空间复杂度的策略<sup>[3]</sup>，并对如何使 P-ACO 算法更有效地利用启发式信息进行了探讨。

#### 8.3.1 FPGA 基本结构

通过对交替逻辑与路由结构之间的选择，可将 FPGA 视为软件与硬件系统之间联系的桥梁。利用目前流行的专业软件可快速地完成许多传统电路的设计，从而避免了传统电路设计中一些不必要的开支，并能快速纠正设计中所出现的一些错误。将电路配置到 FPGA 芯片上并切换到运行模式，而该模式的内在并行性可保证其高效的快速性能。FPGA 技术加速了系统的开发研制进程，这项技术不仅适合于快速成型，而且还可替代中小规模生产中的逻辑门阵列。图 8.1 描述了一个简化的 FPGA 结构。

由图 8.1 可见，FPGA 包括以下三个核心配置单元：

- (1) 配置逻辑模块 (configurable logic block, CLB)：由可完成逻辑运算、寄存等功能的基本器件组成。
- (2) 输入/输出模块 (input/output block, IOB)：形成路由网络和封装引脚的中介。
- (3) 路由网络：包括多个横向通道、纵向通道和容许逻辑模块介入的可配置换向器，从而构成一个复杂的计算结构。

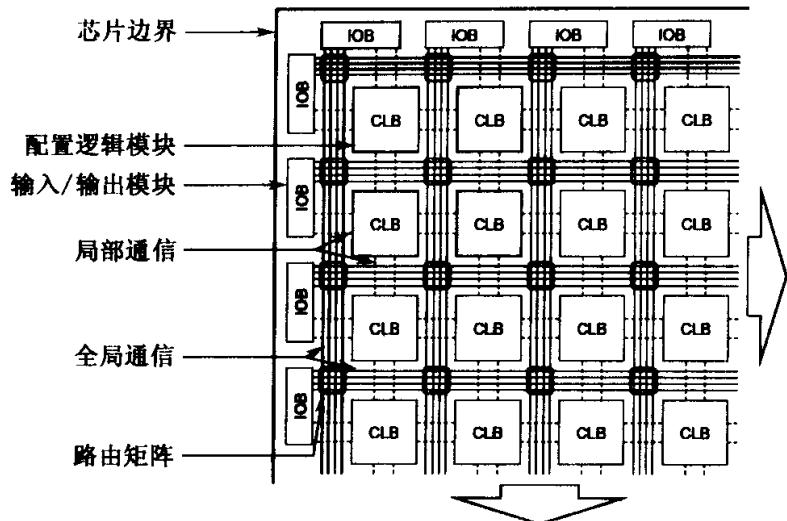


图 8.1 FPGA 的简化结构示意图 (FPGA 芯片的左上角)

### 8.3.2 将蚁群算法映射到 FPGA 的难点

蚁群算法的硬件实现必须考虑其目标结构的限制，这些限制主要侧重于数量、种类以及存储器、寄存器、I/O 资源的分布等方面。

一种比较直接的实现方式是把信息素矩阵映射到 FPGA 上。对于静态分布矩阵上的每一个元素而言，这种设计包含着所有的信息处理和存储资源<sup>[19]</sup>。每只蚂蚁都占据着矩阵中与其所作决策相对应的某一行，显然，这种实现方式会随着问题规模的增大而使其所需空间急剧增加。

在用蚁群算法设计 FPGA 时，其技术难点主要体现在如下三个方面：

(1) 所用的信息量和随机数需要用浮点表示，而这种表示并不是通过细粒度可编程逻辑来实现的。

(2) 启发式信息的挥发和聚集需要乘运算，然而 FPGA 上可行的计算资源并不能非常有效地支持乘运算电路。目前只有少量的 FPGA 有专门进行乘运算的模块，但这些模块的应用受到了其体积的限制。

(3) 由于蚁群算法中必须根据状态转移概率进行状态选择，即

$$p_{ij} = \frac{\tau_{ij}^\alpha \eta_j^\beta}{\sum_{z \in S} \tau_{iz}^\alpha \eta_z^\beta}, \quad \forall j \in S \quad (8.3.1)$$

这样就必须计算上式中分子项中的乘积，因此就需要有  $n$  个电路，而信息素矩阵中的每一行均对应着一个电路，因此大大增加了计算所需的空间复杂度和时间复杂度。

基于上述难点，这里引入了 P-ACO 算法。

### 8.3.3 P-ACO 算法

#### 8.3.3.1 P-ACO 算法描述

P-ACO 算法便于硬件实现的最明显之处在于只将本次迭代中少量且最重要的信息传递到下次迭代，而基本蚁群算法是把完整的信息素矩阵传递到下次迭代。

用一个  $n \times k$  维的群矩阵  $Q = [q_{ij}]$  存放 P-ACO 算法中的群体，群矩阵  $Q$  包含着前  $k$  次迭代中的最优解，且其中每一列均包含一个解。当采用精灵策略后， $j=0$  列包含着迄今为止所有迭代中所发现的最好解，而其余列包含着前  $k-1$  次迭代中所发现的相对最好解。

蚂蚁在某次迭代中完成解的构造之后，P-ACO 算法即更新群体，而非更新信息素矩阵。假如群  $P$  中已包含  $k$  个解，用当前迭代后的最优解替代  $P$  中最早的解，即在群体模式中保持  $|P| = k$ 。这意味着群矩阵  $Q$  中的某一列  $j \in \{1, 2, \dots, k-1\}$  将受到影响。若当前解优于最好解，列  $j=0$  将被新解替代。初始化时群矩阵为空，因此首个  $k-1$  次迭代后必须填满该群体，这样可在不改变早期解的前提下，将相应迭代中的最优解加入到该群体。

值得注意的是，群矩阵  $Q$  中的列  $j \in \{1, 2, \dots, k-1\}$  与 FIFO 序列极为相似。因此，序列中的每一个解直接影响到  $k-1$  次后续迭代中蚂蚁的决策<sup>[20]</sup>。

P-ACO 算法中，蚂蚁用于构造可行解的信息素矩阵  $\tau_{ij}$  由群矩阵决定，且有

$$\tau_{ij} = \tau_{init} + \zeta_{ij}\Delta, \quad \forall i, j \in [0, n-1] \quad (8.3.2)$$

式中， $\tau_{init}$  为初始值； $\zeta_{ij}$  表示解  $\pi \in P$  的数目，且有  $\pi(i) = j$ ，或根据群矩阵  $Q$ ，有  $\zeta_{ij} = |\{h : q_{ih} = j\}|$ 。

信息素更新可按照下述规则进行：

(1) 进入到群体的解  $\pi$  对应一个正的更新

$$\tau_{in}(i) = \tau_{in}(i) + \Delta, \quad \forall i, j \in [0, n-1] \quad (8.3.3)$$

(2) 离开群体的解  $\pi$  对应一个负的更新

$$\tau_{in}(i) = \tau_{in}(i) - \Delta, \quad \forall i, j \in [0, n-1] \quad (8.3.4)$$

上述信息素更新规则不同于基本蚁群算法的信息素更新规则，增量值  $\Delta > 0$ 、群体规模  $k$ 、每次迭代的蚂蚁数目  $m$  均为 P-ACO 算法的主要参数。

#### 8.3.3.2 P-ACO 算法影射到 FPGA 的优点

尽管蚁群算法需要用浮点来表示信息量，而在 P-ACO 算法中，信息素保存于群体之中，FPGA 的实现仅受限于管理一个  $n \times k$  维矩阵的解群，而这一点实

现起来并不困难。

对于基本蚁群算法中的每只蚂蚁而言，必须计算其状态转移概率。在序列机上，该运算需要乘、加运算的时间复杂度为  $O(n)$  步，但当通过采用硬件并行方式，其所需乘、加运算的时间复杂度降为  $O(\lg n)$  步。

在 P-ACO 算法中，蚂蚁是根据  $k+1$  个可能的离散信息量进行决策的，进一步而言，一行中至少有  $n-k$  个离散信息量是相同的，决策所需时间降为  $\Theta(k)$ ，文献 [21] 所给出的  $k$  的取值范围是  $1 \leq k \leq 8$ 。对于一个规模为  $n$  的问题而言，这意味着蚂蚁构造一个解需要  $\Theta(n \cdot k)$  个时间单位；相比较而言，在一个采用基本蚁群算法的顺序处理器中，每只蚂蚁需要  $O(n^2)$  个时间单位才能构造一个解。在 FPGA 上  $m$  只并行蚂蚁产生  $m$  个解的时间为  $\Theta(n \cdot k)$ ，而采用基本蚁群算法的顺序处理器所需要的时间为  $O(m \cdot n^2)$ 。

### 8.3.4 P-ACO 算法在 FPGA 上的实现

这一部分以单机总误时排序问题 (single machine total tardiness problem, SMTTP)<sup>[22]</sup> 为例，首先给出了 P-ACO 算法硬件实现的总体结构，然后详细阐述了其中各主要模块的实现过程。

#### 8.3.4.1 总体结构

基于 P-ACO 算法的 FPGA 设计结构如图 8.2 所示。

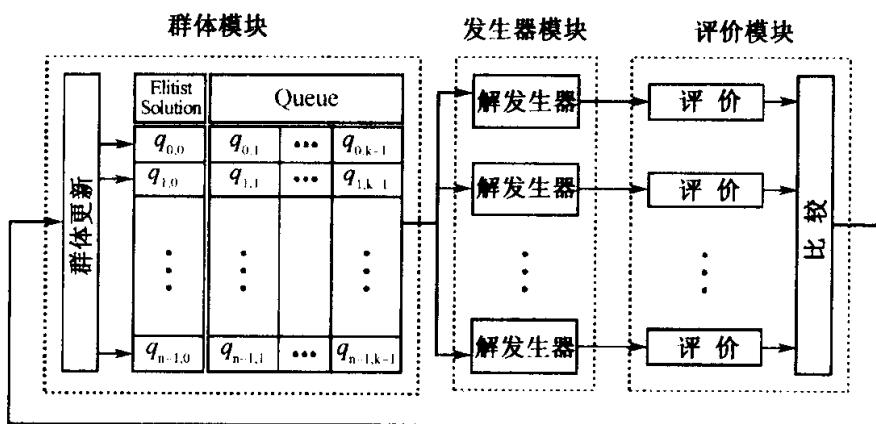


图 8.2 包含群体、发生器和评价模块的 P-ACO 算法设计结构

由图 8.2 可见，该结构主要包括 3 个基本模块，即群体模块、发生器模块和评价模块。为了清楚说明该设计结构，这里没有包括控制模块。群体模块包括群矩阵  $Q = [q_{ij}]_{n \times k}$ ，该矩阵由  $j=0$  列的精灵解和  $j \in \{1, 2, \dots, k-1\}$  列中的

FIFO-序列构成。对于 SMTTP 而言，每一个  $q_{ij}$  对应一个计划内的工作数目。群体模块负责将群矩阵中第  $i$  行的  $q_{ih}$  ( $h \in \{0, 1, \dots, k-1\}$ ) 传送到发生器模块，当迭代结束时，群体模块可从评价模块中接收最优解，然后将其加入到序列中。发生器模块可使  $m$  个解发生器同时工作，其中一个解发生器对应一只蚂蚁，所有的解均由解发生器传送到评价模块中的  $m$  个并行评价框内。 $m$  个解的评价结果将集中于比较框中，由此来确定当前迭代中的最好解。如果所得最好解优于当前最好解，则将当前最好解置换。

#### 8.3.4.2 发生器模块

发生器模块包含  $m$  个相同的解发生器，每一个解发生器模拟一只蚂蚁利用 P-ACO 算法机制构造解的行为。

解发生器（如图 8.3 所示）包含三个基本单元，即 S-阵列、匹配缓冲区和选择器。S-阵列用来存储和保持  $s_i \in S$  中的选择集  $\{S, \dots, n-1\}$ ，其中  $i \in \{0, \dots, c\}$ ，且  $c = |S| - 1$ 。同时，S-阵列可被其包含的内容所访问。当它从群体模块接收到传送项  $q_{ih}$  时，可将  $q_{ih}$  与 S-阵列中的剩余项进行比较。若存在某项  $s_l \in S$ ，且  $s_l = q_{ih}$ ，则 S-阵列返回地址  $a_l$ ，其中  $a_l$  是 S-阵列中  $s_l$  的地址，且将  $a_l$  保存于匹配缓冲区内。选择器用来建立状态转移概率分布并随机选择  $s_{l^*} \in S$ ，随后将其送至评价模块。

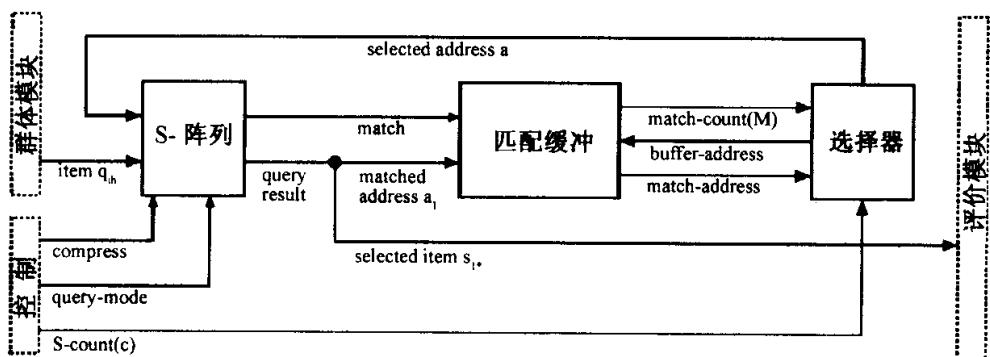


图 8.3 解发生器

解发生器的处理流程如图 8.4 所示。

由图 8.4 可见，处理流程开始后，解发生器使可选集  $S = \{0, \dots, n-1\}$ ，且初始化 S-计数器  $c = n-1$ 。该 S-计数器是由控制模块中的外围逻辑实现，对于所有的解发生器而言，当前的 S-计数器值都是可用的；匹配计数器  $M$  用来表示存于匹配缓冲区中匹配地址的数目。该循环初始化完成之后，群体  $i$  行中的所有项  $q_{ih}$  ( $h \in \{0, 1, \dots, k-1\}$ ) 都传送到 S-阵列，当 S-阵列中某项  $s_l$  被匹配后，即  $s_l = q_{ih}$ ，则将与其相对应的匹配地址保存于匹配缓冲区内。当该循环周期的最后

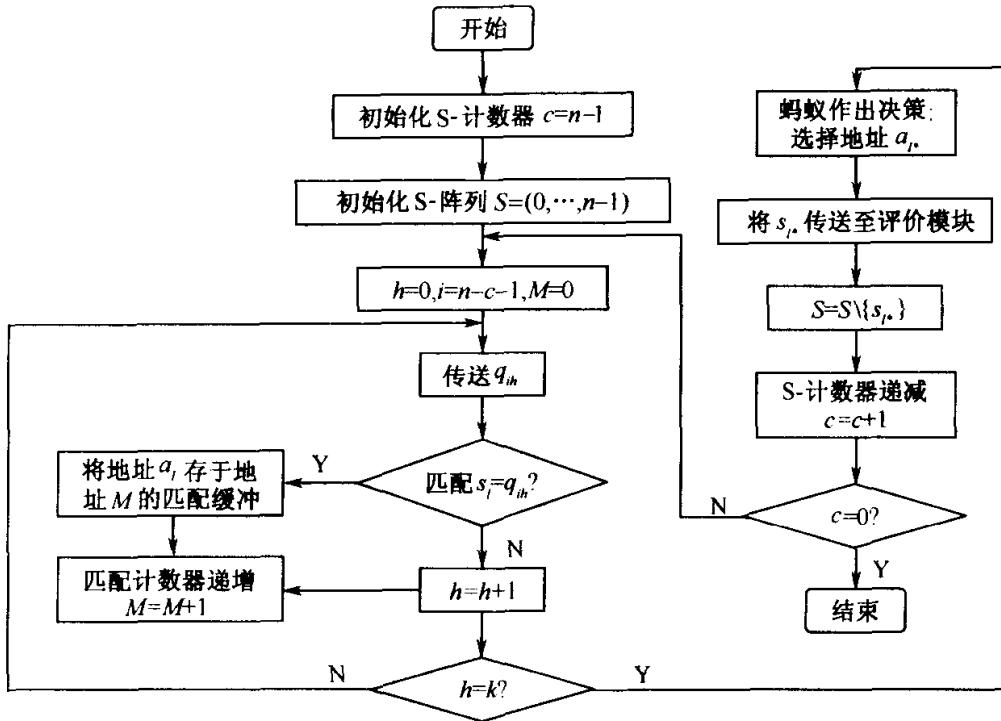


图 8.4 解发生器的处理流程

一次迭代结束后，匹配缓冲区将包含群体和可选集  $i$  行中的所有项，由此可得如下新的状态转移概率公式

$$p_{ij} = \frac{\tau_{ij}}{\sum_{z \in S} \tau_{iz}} = \frac{\tau_{init} + \zeta_{ij}\Delta}{\sum_{z \in S} (\tau_{init} + \zeta_{iz}\Delta)} = \frac{\zeta_{ij}\Delta + 1}{c + M\Delta + 1}, \quad \forall j \in S \quad (8.3.5)$$

根据上述公式，蚂蚁（解发生器）可作出解向量中第  $i$  个位置的决策，它随机选择地址  $a_{i^*}$ ，并将其传送到 S-阵列中，随后将被选项  $s_{i^*}$  存于地址  $a_{i^*}$  中，且将  $s_{i^*}$  传送到对应的评价模块中，然后将  $s_{i^*}$  从选择集 S 中移走，该过程不断对解向量中剩余的  $n-c$  个位置做出决策。

发生器模块中 S-阵列的组织结构如图 8.5 所示。

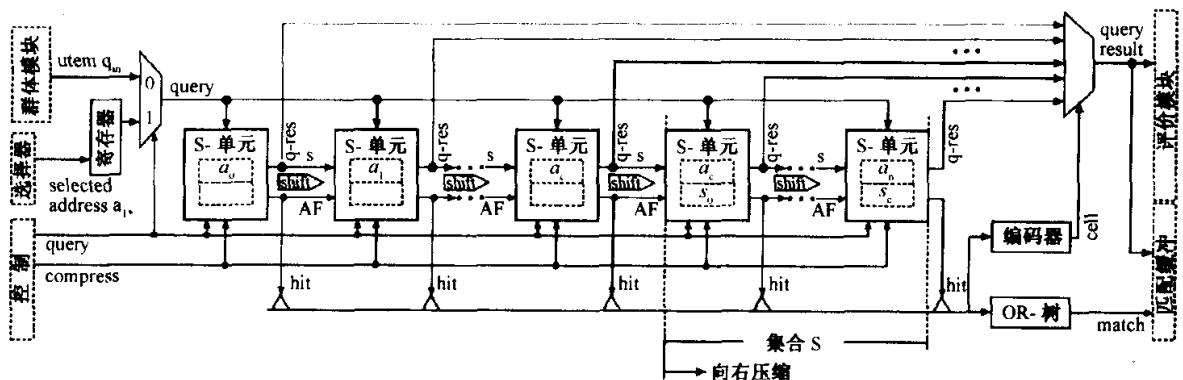


图 8.5 S-阵列

由图 8.5 可见, 解发生器包括  $n$  个由输出总线传送序列连接的 S-单元。每个 S-单元均与其后的 S-单元相连接, 以实现阵列压缩过程中的项右移 (移除集合 S 中的被选项), 这样可保证集合 S 中的各项均保存在地址为  $a_c, \dots, a_{n-1}$  的 S-单元中。只有活跃的 S-单元包含着集合 S 中的有效项, 并用活跃标志 (active flag, AF) 标识。OR-树可产生整个阵列的匹配测试位, 编码器用来产生 S-单元的地址, 该地址用以表示连接查询结果总线到 S-阵列上查询输出的多路复用选择信号。S-单元的总体结构如图 8.6 所示。

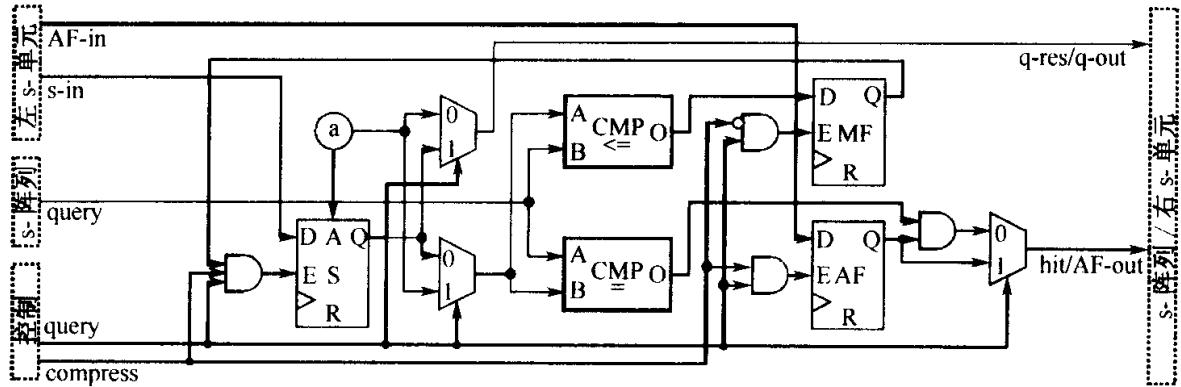


图 8.6 S-单元结构

由图 8.6 可见, S-单元包括两个比较器、一个用以缓冲所选地址  $a_l^*$  的寄存器和两个控制信号。

S-阵列的操作包含三个基本阶段, 即传送阶段、选择阶段和压缩阶段。在传送阶段, 群矩阵  $i$  行中的  $k$  个项被传送到 S-单元, 即  $\text{query} = q_{ik}$ , S-单元中的比较器把传送项与其已存有的  $s$  项进行比较。若相等, 则当各自的 S-单元活跃时 ( $AF=1$ ), S-单元将去除地址  $a$ 。

当选择器随机选择被选项  $s_l^*$  后, S-阵列即进入了选择阶段。将被选地址  $a_{l^*}$  传送到 S-单元, 每一个 S-单元用其比较器将传来的地址与其  $a$  地址进行比较。

S-单元  $l^*$  将其所存的  $s_{l^*}$  项作为查询结果传送到评价模块。每一个 S-单元  $j$  也会利用其自身的比较器检查是否为被选 S-单元的前继, 即  $a_j \leq a_{l^*}$ 。所有的前继 S-单元设置其匹配标志 (match flag, MF) 为  $MF = 1$ , 而其他的则设置  $MF = 0$ 。

在 S-阵列的压缩阶段, 每一个 S-单元将其活跃标志 (AF) 及其存储项送至后续 S-单元。如果选择阶段的匹配标志  $MF = 1$ , 则 S-单元锁存来自前继 S-单元的相应数据, 且逻辑数 0 被设置到第一个 S-单元的 AF 标志中。最终结果是被选项  $s_{l^*}$  被覆盖, 被选单元 (即  $a_j < l^*$ ) 的左边所有项均向右移动一个单元。一次迭代完成之后, 通过将 S-单元地址值调入到 s-寄存器的方式将 S-阵列中数据寄存器的原始值再次初始化。

匹配缓冲区是一个将后续匹配地址存储到  $k$  规模寄存器的简单电路，其结构如图 8.7 所示，而选择器结构如图 8.8 所示。

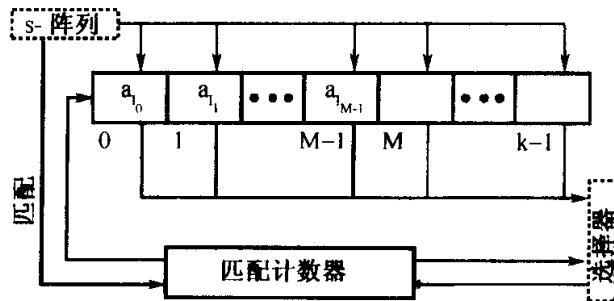


图 8.7 匹配缓冲区（阴影部分为空）

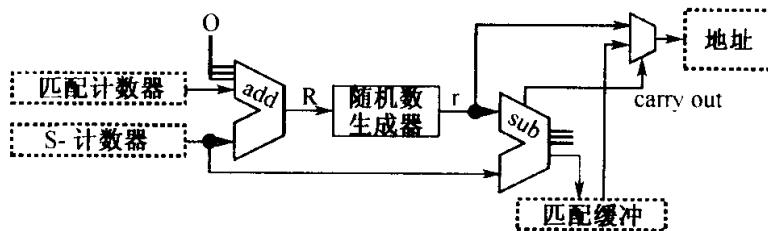


图 8.8 选择器

选择器用来接收来自匹配缓冲区内匹配项  $M$  的数目，将选择器设计成可根据公式 (8.3.5) 所示形式的对各项进行选择。此处，随机数生成器的上界可根据公式  $R=c+M\Delta$  计算，其中  $\Delta=2^y$ ，这样乘除运算只需通过简单地左移或右移操作即可实现；而随机数  $r$  是从间隔  $[0, R]$  中均一产生的。在没有较大规模乘除电路的情况下，从任意间隔中产生随机数比较困难。可产生随机位的硬件随机数生成器用于从间隔  $[0, 2^x-1]$  中产生随机数，然而由于  $R+1$  并不一定是 2 的幂，这里可不断提取随机数  $r \in [0, R']$ ，直至  $r \leq R$ ，其中  $R'=2^{\lfloor \lg(R) \rfloor} - 1$  是大于或等于  $R$  的 2 的最小幂。令  $p = \text{Prob}(r \leq R) = (R+1)/(R'+1)$  表示获得随机数  $r \in [0, R]$  的概率，其中  $1/2 < p \leq 1$ 。经过  $u-1$  次成功尝试后，获得随机数  $r \in [0, R]$  的概率为  $P(U=u) = p(1-p)^{u-1}$ ，由此，期望值  $E(U) = p \sum_{u=1}^{\infty} u(1-p)^{u-1} = 1/p$ 。一般而言，为了获得均一分布的随机数  $r \in [0, R]$ ，当  $1 \leq E(U) < 2$  时，将获得随机数  $E(U)$  次。文献 [23] 详细介绍了利用随机数生成器产生随机数时所需随机位的情况。

一旦随机数  $r$  产生之后，可将其与  $c$  相比较，以决定 S-阵列或匹配缓冲区内缓冲地址中的某项是否被选中。如果  $r \leq c$ ，则  $a_{l^*} = a_r$  是 S 中被选项的标志；否则， $a_{l^*} = a_{M^*}$ ，其中  $M^* = l_{(r-c) \gg y}$ 。当  $a_{l^*}$  被确定之后，则将 S-阵列中的对应项  $s_{l^*}$  送至评价模块。

### 8.3.4.3 评价模块

评价模块用来评价由解发生器所产生解的质量，该模块的核心即为比较模块。每一个需要用 P-ACO 算法解决的优化问题均需特定的评价模块，该评价模块包含着特定问题的评价参数，并可计算来自解发生器的目标函数值。

比较模块用来评价算法迭代中所产生的最好解和新的精灵解（如果已经发现），并将其存于群体中。这里所设计的 SMTTP 比较模块如图 8.9 所示。

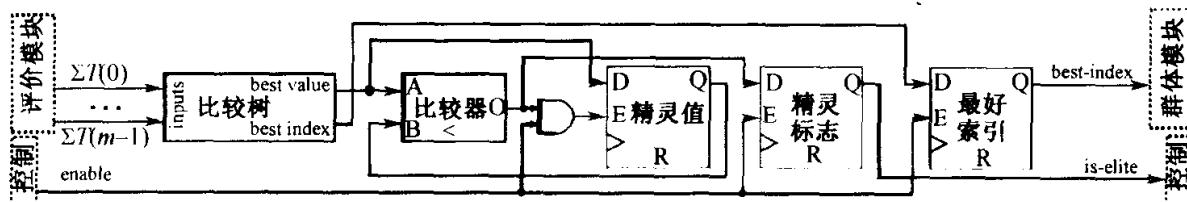


图 8.9 SMTTP 比较模块

在图 8.9 中，比较树可减少比较模块的逻辑延时，用来构造最好解的解发生器标志（best-index），并将其传送到群体模块。一个附加比较器将目前迭代中最好解与已发现最好解进行比较，如果当前最好解优于已发现的最好解，则设置精灵标志“is-elite”。

### 8.3.4.4 群体模块

群体模块结构如图 8.10 所示。

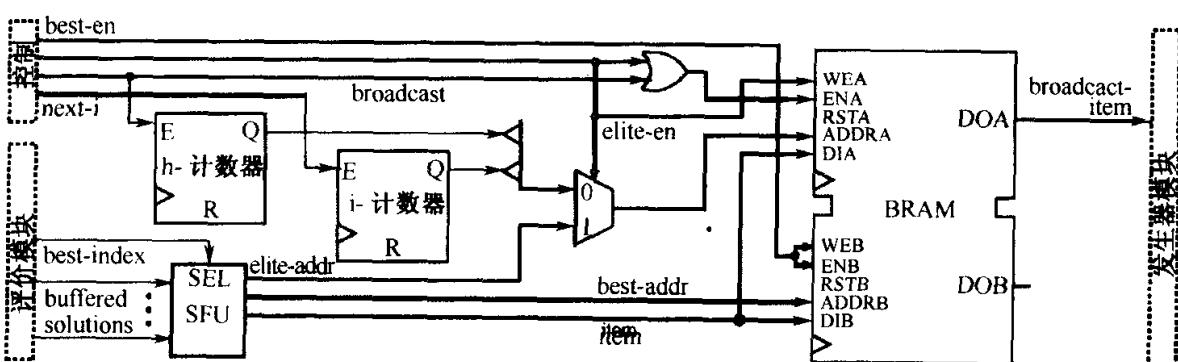


图 8.10 群体模块

由图 8.10 可见，群体模块为群矩阵提供存储功能。群体模块从评价模块中接收到最好解的解发生器标志（best-index），而解传送单元（solution forward-

ing unit, SFU) 负责将各自的最好解传送到 BRAM，同时计算所传送解的写地址，随后将传送解通过 BRAM 的端口 B 写入到 FIFO 序列（即群矩阵中的  $j \in \{1, \dots, k-1\}$  列）。如果该解是一个新的最优解，则它通过端口 A 写入到  $j=0$  列。在查询阶段，群体模块将存于群矩阵中第  $i$  行的所有  $k$  项  $q_{ih}$  传送到发生器模块，因此，它包含着用来表征  $h$  和  $i$  的两个计数器（如图 8.4 所描述），将这两个计数器的输出结合即形成可通过端口 A 读取的地址总线。

### 8.3.5 实验结果

为了验证本节所研究 P-ACO 算法 FPGA 硬件实现方案的可行性和有效性，这里将基于软件实现的 P-ACO 算法与基于硬件实现的 P-ACO 算法进行了实验对比，其结果如图 8.11 和图 8.12 所示。由于软件和硬件产生的是同一个解，其性能则通过每次迭代时间来进行比较。这里，定义软件实现中每次迭代时间和硬

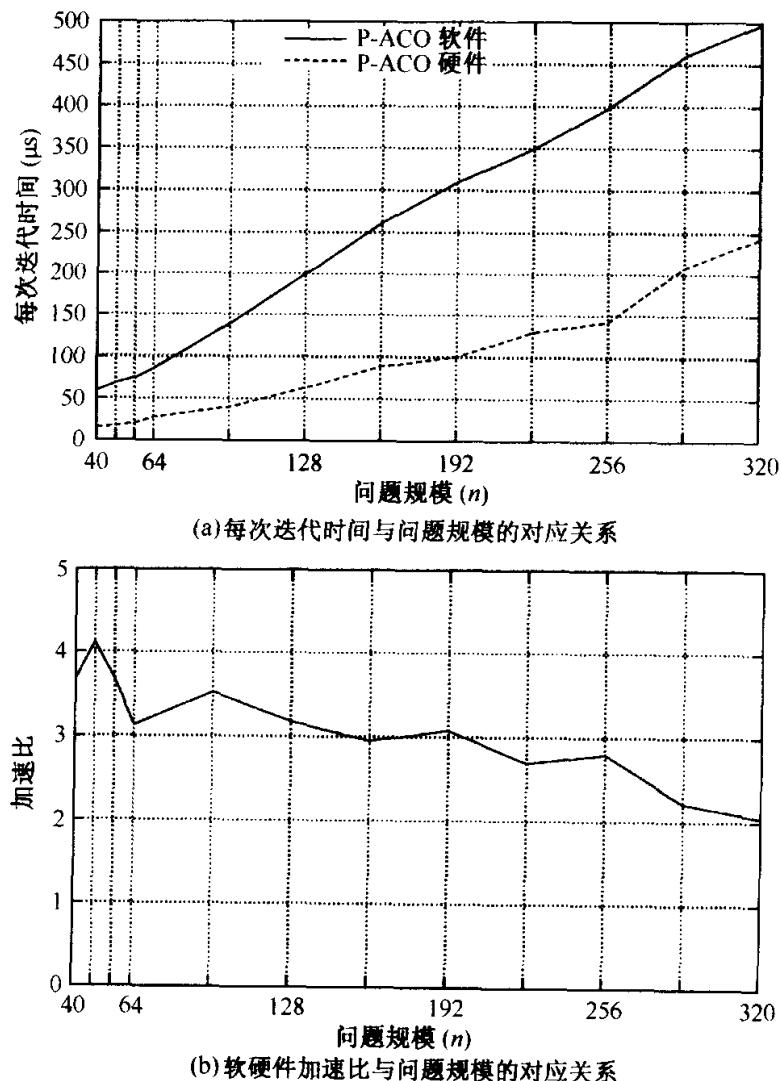


图 8.11 P-ACO 算法的硬件实现与软件实现比较 ( $m=8$ )

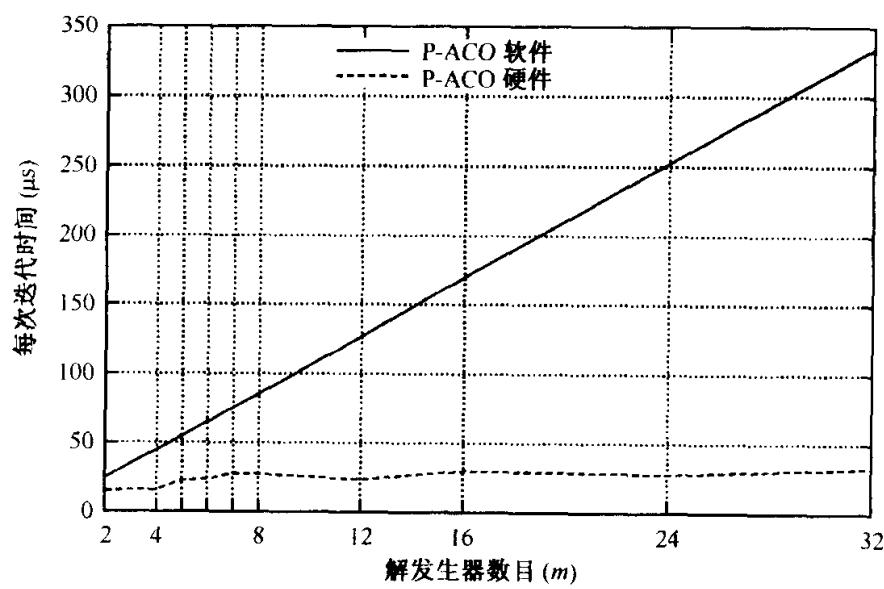
件实现中每次迭代时间的加速比为

$$\text{加速比} = \frac{\text{软件实现中每次迭代时间}}{\text{硬件实现中每次迭代时间}} \quad (8.3.6)$$

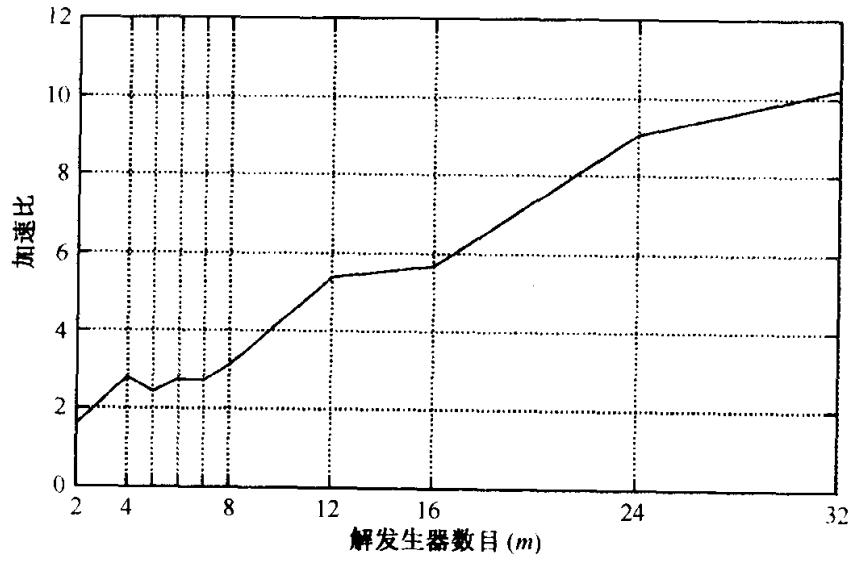
在触发器数目固定的情况下（如  $m=8$ ），P-ACO 算法的硬件实现与软件实现性能比较如图 8.11 所示。

由图 8.11 可见，每次迭代中软硬件运行时间随着问题规模  $n$  的增加而呈线性增长关系，然而就问题规模而言，硬件实现比软件实现相对要快一些。该实验中，加速比的变化范围为从  $n=320$  时的最小值 2.04 到  $n=48$  时的最大值 4.07。

在问题规模固定的情况下（如  $n=64$ ），基于软件实现的 P-ACO 算法与基于硬件实现的 P-ACO 算法性能比较如图 8.12 所示。



(a) 每次迭代时间与解发生器数目的对应关系



(b) 软硬件加速比与解发生器数目的对应关系

图 8.12 P-ACO 算法的硬件实现与软件实现比较 ( $n=64$ )

图 8.12 表明，随着解发生器数目的增多，基于软件实现的 P-ACO 算法每次迭代所需时间呈线性增长，而对于基于硬件实现的 P-ACO 算法而言，每次迭代所需时间大致保持在约  $28\mu s$  的常数值。该实验中，加速比的变化范围为从  $m=2$  时的最小值 1.9 到  $m=32$  时的最大值 10.22。

### 8.3.6 几点改进

这一部分将对前述 P-ACO 算法的硬件实现做一些改进，进而提出了一种降低 P-ACO 算法空间复杂度的方法，并对在 P-ACO 算法的硬件实现过程中如何更有效地利用启发式信息做了讨论。

#### 8.3.6.1 空间约束

P-ACO 算法的“高度”随着问题规模  $n$  的增加而呈指数增加，而其“宽度”却呈线性增加，这样会导致一个平面长方形结构的出现。如果问题规模  $n$  已知，可行资源会制约 FPGA 上的解发生器数目。假设在  $g$  个周期内只完成  $m' < m$  个解发生器运算，其中  $m = gm'$ （见图 8.13）。每一个周期  $m'$  中的解均并行产生，通过评价策略对这些解进行比较。每一循环结束后，新建立的解都要相互进行比较，且将当前循环中的最好解同以前循环中的最好解进行比较。比较要花费  $t_{cmp}$  个时间单位，在下一解的产生过程中也进行这一比较操作。因此，一个完整运算过程的执行时间为

$$t_{it} = \max\{gt_g + t_{cmp}, gt_{cmp} + t_g\} + t_u \quad (8.3.7)$$

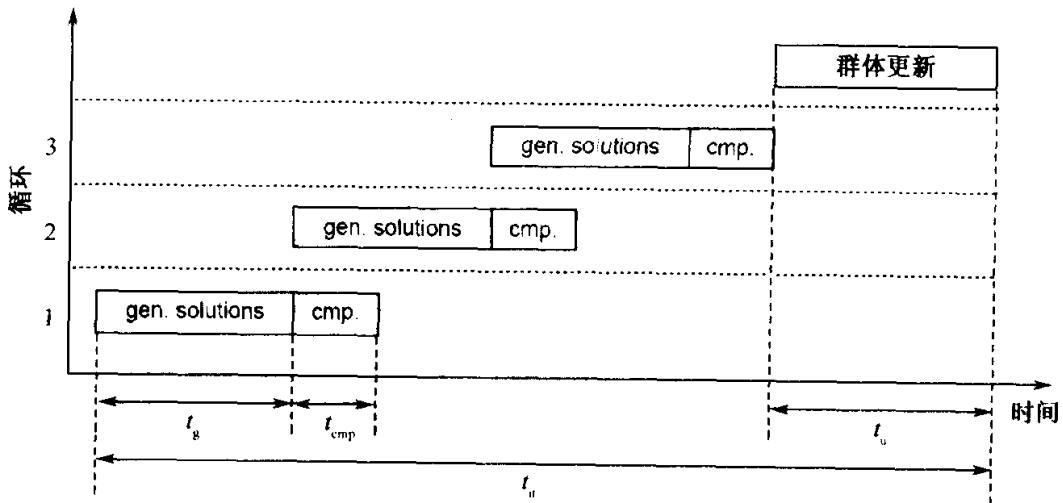


图 8.13 包含  $g=3$  个循环周期的完整迭代过程

更好利用可行空间的另外一种方法是从“可选集 S 会随时间的流逝而缩减”

这一情况中得到启发的。图 8.14 给出了对其改进的两个实例，图中的阴影区域表示可选集  $S$  中行的活跃部分。

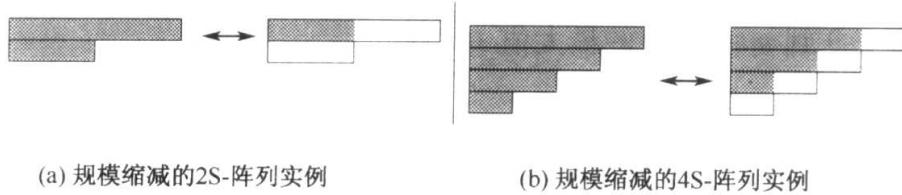


图 8.14 规模缩减的 S-阵列实例

图 8.14 (a) 和图 8.14 (b) 中的 2S-阵列和 4S-阵列配置通过让蚂蚁处于完成过程的不同阶段而分别节约了 25% 和 37.5% 的空间。在这两个配置中，左边表示启用新蚂蚁后的所有 S-阵列，右边表示新蚂蚁启用前的阵列情况。

### 8.3.6.2 启发式信息

将启发式信息集成到 P-ACO 硬件算法会带来两个问题：一是启发量为实数值；二是对于集合  $S$  中的所有项而非仅对具有  $O(k)$  规模的子集而言，启发量存在。因此，这里提出了将启发矩阵  $\vec{\eta}_i = (\eta_{i,0}, \dots, \eta_{i,n-1}) \in \mathbb{R}^n$  中每一行  $i$  的信息转化为  $l$  个启发向量  $\{\vec{h}_{i,0}, \dots, \vec{h}_{i,l-1}\}$  的集合，且  $\vec{h}_{i,u} = [0, n-1]^t$ ,  $u \in \{0, \dots, l-1\}$ 。每一向量  $\vec{h}_{i,u}$  都包含着较大的启发式信息值。

$$(1) \text{ 计算 } \delta_i = \frac{1}{tl} \sum_{j=0}^{n-1} n_{ij};$$

(2) 执行  $n$  次如下操作：

- ①确定  $j^*$ ，使得  $\eta_{j^*}^* = \max_{j=0, \dots, n-1} \eta_{ij}$ ；
- ②将  $j^*$  增加到用以构造启发式向量的数集中；
- ③更新  $n_{j^*} \leftarrow \eta_{j^*} - \delta_i$ 。

用上述方法所达到的近似质量依赖于  $\eta_{ij}$ 、 $t$  和  $l$  值，而决定将哪一项加入到给定向量是通过加入尽可能多向量的任意项来实现的，其中每个向量  $\vec{h}_{i,u}$  中各项的顺序是随机的。如果  $l > 1$ ，即对于某一给定的行，至少有一个启发向量存在，则可认为某个  $\vec{h}_{i,u^*}$  向量是活跃的，而其他  $l-1$  个启发向量是非活跃的，只有活跃的启发向量决定着蚂蚁的决策进程。当  $m$  只蚂蚁的一次迭代完成以后，活跃的启发向量  $\vec{h}_{i,u^*}$  被其他某个启发向量替代，且  $u \in [0, l-1] \setminus \{u^*\}$ 。值得注意的是，启发向量  $\vec{h}_{i,0}, \dots, \vec{h}_{i,l-1}$  的构造必须在算法编入到 FPGA 之前完成。因此，这种方法不能处理需要在线计算启发信息的问题。

在 P-ACO 算法中，用具有  $k$  个优良解的群体替代蚁群算法中的信息素矩阵，这样就必须将群矩阵  $Q$  中的第  $i$  行转换为公式 (8.3.2) 中的  $\tau_{ij}$ 。同样地，在对

第  $i$  行做出决策时，可按下式把任意启发向量  $\vec{h}_{i,u^*}$  转换成其所对应的用以描述第  $j$  项影响的启发值  $\hat{\eta}_{ij}$

$$\hat{\eta}_{ij} \mapsto \eta_{init} + \gamma_{ij}\Delta_H, \quad \forall i, j \in [0, n-1] \quad (8.3.8)$$

式中， $\eta_{init} \geq 0$  表示赋予给第  $j$  项的初始启发值； $\gamma_{ij}$  表示活跃启发向量中第  $j$  项的发生次数，即  $\gamma_{ij} = |\{v : h_{i,u^*v} = j, v=0, \dots, t-1\}|$ ； $\Delta_H$  表示权重系数。为了区分两种权重，用  $\Delta_P$  替代公式 (8.3.2) 中的  $\Delta$ 。由此，蚁群算法的状态转移概率变为

$$p_{ij} = \frac{\tau_{ij} \hat{\eta}_{ij}}{\sum_{z \in S} \tau_{iz} \hat{\eta}_{iz}} = \frac{(\tau_{init} + \varsigma_{ij}\Delta_P)(\eta_{init} + \gamma_{ij}\Delta_H)}{\sum_{z \in S} (\tau_{init} + \varsigma_{iz}\Delta_P)(\eta_{init} + \gamma_{iz}\Delta_H)}, \quad \forall j \in S \quad (8.3.9)$$

上式不再需要指数项  $\alpha$  和  $\beta$ ，信息素和启发值通过选择初始值  $\tau_{init}$  和  $\Delta$  来确定。公式 (8.3.9) 的数学运算更适合于 FPGA 结构所允许的资源。

令群向量  $(q_{i,0}, \dots, q_{i,k-1})$  为群体中的当前行，且  $\vec{h}_{i,u^*} = \{h_{i,u^*,0}, \dots, h_{i,u^*,t-1}\}$  是当前的启发向量，如图 8.15 所示。

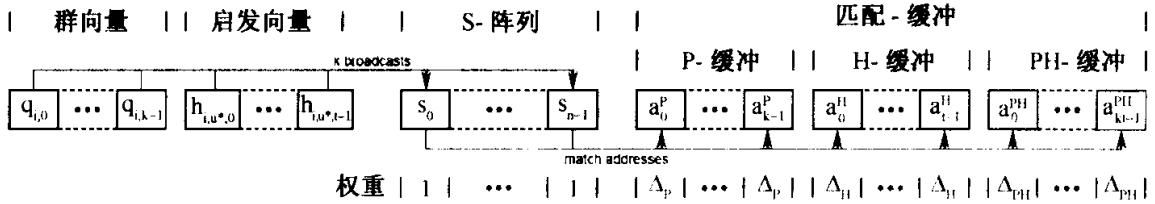


图 8.15 当前群中  $i$  行的各项和启发向量  $\vec{h}_{i,u^*}$  被传递的情况

两个向量中所有的  $k+t$  项都被传送到包含可选集  $S$  中各项的 S-阵列中。匹配项地址保存于三个不同的匹配缓冲区内，其中 P-缓冲区用来保存群向量和集合  $S$  中  $M_P \leq k$  项的地址，即 P-缓冲区等价于图 8.7、图 8.8 所示的匹配缓冲区。将启发向量和可选集  $S$  中  $M_H \leq t$  项的地址复制到 H-缓冲区的独立位置；由于蚁群算法中信息量和启发值的乘积关系，设计时需要一个附加的缓冲区，即 PH-缓冲区，该缓冲区用以保存启发向量、群向量以及可选集  $S$  中的各项。令  $\Delta_P = 2^{y^P}$  表示与群向量相关联的权重， $\Delta_H = 2^{y^H}$  表示启发向量的权重，则  $\Delta_{PH} = \Delta_P \Delta_H = 2^{y^P + y^H}$  表示地址存储于 PH-缓冲区  $\varsigma_{ij} \gamma_{ij}$  次的某项  $j$  的权重。当完成所有传送过程后，PH-缓冲区即包含  $M_{PH} \leq kt$  个地址。利用缓冲概念，并设置  $\tau_{init} = \eta_{init} = 1$ ，则公式 (8.3.9) 变为

$$p_{ij} = \frac{\tau_{ij} \hat{\eta}_{ij}}{\sum_{z \in S} \tau_{iz} \hat{\eta}_{iz}} = \frac{1 + \varsigma_{ij}\Delta_P + \gamma_{ij}\Delta_H + \phi_{ij}\Delta_{PH}}{|S| + M_P\Delta_P + M_H\Delta_H + M_{PH}\Delta_{PH}}, \quad \forall j \in S \quad (8.3.10)$$

为了使蚂蚁能按照上述状态转移概率作出决策，图 8.8 所示基本布局中的选择器也需做相应改进。FPGA 上匹配缓冲区和选择器并行完成一只蚂蚁决策所需

的有效时间为  $\Theta(k+t)$ ，其中， $k$  和  $t$  均为很小的常数。

## 8.4 基于蚁群算法和遗传算法动态融合的软硬件划分

### 8.4.1 软硬件划分问题

软硬件划分问题是软硬件协同设计系统中的一个关键环节，其基本任务是在满足某些约束的条件下，将系统功能行为最优化地分配到一定的软硬件系统结构上。根据目标系统结构的不同，可将软硬件划分问题分为双路划分（bi-partitioning）和多路划分（multi-way partitioning）两大类，其中双路划分应用最为广泛，也是软硬件划分问题的基础。在有些研究中，常把划分作为软硬件综合的一部分<sup>[24,25]</sup>。

软硬件协同设计的目标是完成由硬件和软件组合而成的系统的辅助设计。由于硬件和软件具有各自的特性和价值，传统的设计方法往往是手工进行软硬件划分后，再利用适当的工具分别进行开发，最后再组合成完整的系统。这种传统的设计方法在面对诸如 SoC 之类的复杂系统时，往往因为主观性很强的划分格局而造成系统设计期间针对实际情况调整系统结构、设计后期软硬件整合调试等多方面的困难，有时会严重影响设计周期和设计效率。针对这一问题，人们对软硬件协同设计方法开展了各种研究。

Gupta R K 等<sup>[25]</sup>设计了一种软硬件划分算法用于自动化设计空间探索过程。此后，人们研究了各具特色的划分算法，其中应用效果较好的典型算法有启发式算法<sup>[26]</sup>，如爬山法<sup>[27]</sup>、遗传算法<sup>[28, 29]</sup>、模拟退火算法<sup>[30]</sup>、禁忌搜索算法<sup>[31]</sup>等，这些算法通过定义启发信息指导搜索过程逐步向最优解收敛。另外，较有特色的 GCLP/IBS 算法<sup>[32]</sup>通过定义软硬件极端节点、软硬件排斥节点和普通节点来体现进程节点在硬件和软件上执行时所表现出的性能特征，但该算法对目标系统结构的依赖性较大。程国达等<sup>[33]</sup>提出的约束驱动与松弛时间消除相结合的软硬件划分算法在求解效率上改进了 GCLP/IBS 算法。

文献 [34] 提出了应用蚁群算法求解软硬件系统任务双路划分问题的方法，试图在满足面积约束的条件下优化时间性能。与前述方法相比，该方法在求解时间和寻优精度上取得了更好的结果。但单纯的蚁群算法在运行初期缺乏信息素，这限制了算法搜索效率的进一步提高。研究表明，蚁群算法中约占整个运算过程 65% 的时间被用于形成最优解上的信息素强度。

文献 [35] 提出了遗传算法与蚁群算法相融合的一般性优化问题求解策略；文献 [36] 将蚁群算法与遗传算法反复交叉，利用蚁群算法不断生成种群个体，进行同步时序电路的初始化，这种优化方法能够尽可能多地初始化触发器，但每次种群进化都要经历几代，运行效率较低。

针对面向嵌入式系统和 SoC 的软硬件双路划分问题，本节研究了 DCG3A，其基本思想如下：

(1) 先利用遗传算法快速随机的群体性全局搜索能力生成划分问题初始解，并将其转化为初始信息素分布，然后利用蚁群算法的正反馈、高效收敛等优势寻求最优划分解。

(2) 遗传算法迭代过程中统计子代群体的进化率，在给定的遗传迭代次数范围内，如果在连续的若干次迭代中，子代群体的进化率都低于预先设定的最小进化率，则终止遗传算法过程，进入蚁群算法，并确保遗传算法和蚁群算法在最佳时机融合。

## 8.4.2 遗传算法与蚁群算法动态融合

### 8.4.2.1 遗传算法与蚁群算法动态融合基本原理

通过对遗传算法与蚁群算法的研究与实验发现，它们在总体态势上呈现出如图 8.16 所示的速度-时间曲线。

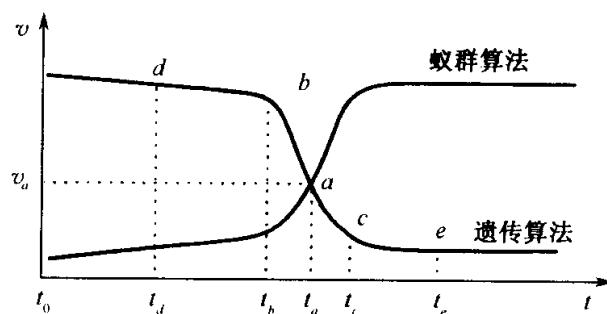


图 8.16 遗传算法与蚁群算法的速度-时间曲线

由图 8.16 可见，遗传算法在搜索的初期 ( $t_0 \sim t_a$  时间段) 具有较高向最优解收敛的速度，但  $t_a$  之后求最优解的效率显著降低。而蚁群算法在搜索的初期 ( $t_0 \sim t_a$  时间段) 由于缺乏信息素，使得搜索速度缓慢，但当信息素积累到一定强度之后 ( $t_a$  时刻之后)，向最优解收敛的速度迅速提高。遗传算法与蚁群算法动态融合的基本思想就是：在最佳点 (a 点) 之前采用遗传算法生成初始信息素分布，在最佳点之后采用蚁群算法求取最优解。

文献 [35] 提出了遗传算法与蚁群算法融合的一般性优化问题求解策略，在 TSP 应用实验中取得了良好的优化效果。该策略在遗传算法中设置了固定的迭代次数，这样会造成过早（如  $t_d$  时刻）或过晚（如  $t_e$  时刻）结束遗传算法过程，不能有效保证两者在最佳时机 ( $t_a$  时刻) 进行融合。

这里提出的动态融合策略可以确保遗传算法与蚁群算法在最佳时机融合，其

具体方法如下：

- (1) 设置最小遗传迭代次数  $Gene_{min}$  (如  $t_b$  时刻) 和最大遗传迭代次数  $Gene_{max}$  (如  $t_c$  时刻)。
- (2) 遗传算法迭代过程中统计子代群体的进化率，并以此设置子代群体的最小进化率  $Gene_{min-impro-ratio}$ 。
- (3) 在设定的迭代次数范围内，如果连续  $Gene_{die}$  代，子代群体的进化率都小于  $Gene_{min-impro-ratio}$ ，说明这时遗传算法的优化速度较低，因此，这时可终止遗传算法过程，进入蚁群算法。

#### 8.4.2.2 软硬件划分问题与双着色模型

**定义 8.4.1  $k$  路划分问题** 对于模块集合  $M = \{m_1, m_2, \dots, m_n\}$ ， $k$  路划分问题就是寻找簇的集合  $P = \{p_1, p_2, \dots, p_k\}$ ，使其满足

$$\begin{cases} p_i \subseteq M, & 1 \leq i \leq k \\ \bigcup_{i=1}^k p_i = M \\ p_i \cap p_j = \emptyset, & 1 \leq i, j \leq k, i \neq j \end{cases} \quad (8.4.1)$$

$k$  路划分问题的求解通常是在满足某些约束条件下优化目标函数。当  $k=2$  时，称为双路划分问题；当  $k>2$  时，称为多路划分问题。

软硬件划分是嵌入式系统设计中的关键问题之一，目前研究和应用较多的是双路划分，即考虑系统中只有一个处理器再加上 ASIC 或其他硬件部件的情况。本节考虑了嵌入式系统任务的软硬件双路划分，并将其简称为软硬件划分。

求解软硬件划分问题时，通常用有向无环图 (directed acyclic graph, DAG) 来描述任务图，记  $G=(T, E)$ ，其中  $T=(t_0, t_1, \dots, t_n)$  表示任务节点集， $E$  表示有向边集。节点表示功能行为 ( $t_0$  和  $t_n$  是虚拟任务节点，用于确保任务图中只有一个开始节点和一个结尾节点)，有向边表示节点之间的控制/数据依赖关系与数据通信代价。

可以用图的双着色模型<sup>[34]</sup>来表示软硬件双路划分问题，如图 8.17 所示。把划分到软件部分的功能用颜色  $c_1$  表示，划分到硬件部分的功能用颜色  $c_2$  表示。划分的目标是寻找任务图中所有节点的最优着色方案，使得在不超过给定硬件面积的条件下，系统的执行时间最短。由于  $t_0$  和  $t_n$  是虚拟任务节点，因此，最终优化方案与节点  $t_0$ 、 $t_n$ ，边  $e_{01}$  以及边  $e_{8n}$  的颜色无关。着色过程中将边  $e_{ij}$  的颜色设置为该边目标节点 (即  $t_j$ ) 的颜色，并且每条边  $e_{ij}$  对应于两个全局启发信息  $\tau_{ij}(k)$ ，分别表示任务  $t_j$  被着色为颜色  $c_k$  的概率，其中  $k=1, 2$ 。

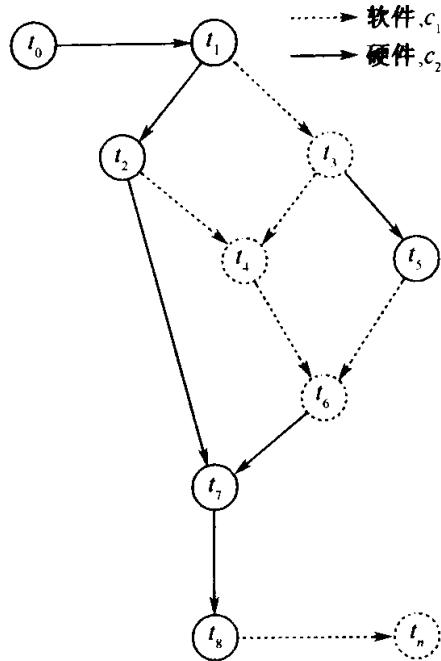


图 8.17 双着色模型

### 8.4.3 基于 DCG3A 的软硬件划分算法

下面以软硬件划分的双着色模型为基础来介绍基于 DCG3A 的软硬件划分算法。

#### 8.4.3.1 DCG3A 中的遗传算法规则

**遗传编码：**采用二进制编码方法<sup>[35]</sup>，0 表示颜色  $c_1$ （划分到软件），1 表示颜色  $c_2$ （划分到硬件）。

**目标函数与适应值函数：**这里讨论的软硬件划分是在硬件面积约束条件下使系统的运行时间最短，因此，将目标函数定义为  $s_{best} = \text{argmin}(\text{time}_s)$ ，适应值函数定义为  $\text{fitness}(s) = 1/\text{time}_s$ 。其中， $\text{time}_s$  表示着色（划分）方案  $s$  的运行时间。

**初始种群生成：**采用随机方法生成初始种群。

**选择算子：**采用遗传算法中应用最广的赌轮选择策略。

**交叉算子：**采用均匀杂交（uniform crossover）策略。

**变异算子：**以变异概率  $p_m$  将所选个体位取反。

**遗传控制的参数设置：**采用算子执行非重叠遗传过程，依据文献 [36] 给定的控制参数经验值范围，设定种群规模  $N = 50$ ，杂交概率  $p_c = 0.6$ ，变异概率  $p_m = 0.2$ 。

遗传算法的结束条件：遗传算法结束条件实际上就是判断遗传算法与蚁群算法的最佳融合时机。这里设置最小遗传迭代次数  $Gene_{min}=15$ ，最大遗传迭代次数  $Gene_{max}=50$ ，最小进化率  $Gene_{min-impro\_ratio}=3\%$ ， $Gene_{die}=3$ 。

#### 8.4.3.2 DCG3A 中的蚁群算法规则

目标函数与适应值函数：同遗传算法。

信息素设置与更新规则：以 Stützle T 等<sup>[37]</sup>在 MMAS 中所提出的信息素设置与更新策略为基础，此处  $\tau_{ij}(k)$  的更新方程为

$$\tau_{ij}(k) = \begin{cases} \tau_{ij}(k)_{max}, & \text{若 } \tau_{ij}(k) > \tau_{ij}(k)_{max} \\ (1 - \rho)\tau_{ij}(k) + \Delta\tau_{ij}(k)_{best}, & \text{若 } \tau_{ij}(k)_{min} \leq \tau_{ij}(k) \leq \tau_{ij}(k)_{max} \\ \tau_{ij}(k)_{min}, & \text{若 } \tau_{ij}(k) < \tau_{ij}(k)_{min} \end{cases} \quad (8.4.2)$$

式中， $\Delta\tau_{ij}(k)_{best}$  表示本次循环最佳蚂蚁对边  $e_{ij}$  上  $c_k$  信息素的增量，将其定义为

$$\Delta\tau_{ij}(k)_{best} = \begin{cases} Q/time_{s_{best}}, & \text{若 } s_{best} \text{ 中 } e_{ij} \text{ 着色为 } c_k \\ 0, & \text{否则} \end{cases} \quad (8.4.3)$$

式中， $Q$  是常数。

Ant-Cycle 模型各因子定义：对于除  $t_n$  以外的每个节点  $t_i$ ，蚂蚁试图确定  $t_i$  的每个后续  $t_j$  的颜色，它主要依据边  $e_{ij}$  上的全局启发信息  $\tau_{ij}(k)$  和节点  $t_j$  上的局部启发信息（运行时间与面积的综合代价）来完成这一工作。准确地说，节点  $t_i$  上的蚂蚁按照如下概率猜测后续节点  $t_j$  将被着色为  $c_k$ 。

$$p_{ij}(k) = \frac{[\tau_{ij}(k)]^\alpha [\eta_j(k)]^\beta}{\sum_{l=1,2} [\tau_{ij}(l)]^\alpha [\eta_j(l)]^\beta} \quad (8.4.4)$$

式中， $\eta_j(k)$  表示  $t_j$  被着色为  $c_k$  的局部启发信息，将其定义为

$$\eta_j(k) = \frac{1}{w_t \times time_j(k) + w_a \times area_j(k)} \quad (8.4.5)$$

式中， $w_t$  和  $w_a$  表示运行时间和所需硬件面积之间的归一化权重因子； $time_j(k)$  和  $area_j(k)$  分别表示节点  $t_j$  被着色为  $c_k$  时的运行时间和所需面积。

当蚂蚁进入节点  $t_i$  时，将综合考虑  $t_i$  所有前驱节点对  $t_i$  着色猜测的结果，猜测  $t_i$  在本次蚂蚁迭代中的颜色。蚂蚁  $k$  以如下式所示概率猜测  $t_i$  被着色为  $c_k$

$$p_i(k) = \frac{\text{猜测 } t_i \text{ 着色为 } c_k \text{ 的前驱个数}}{\text{节点 } t_i \text{ 的前驱个数}} \quad (8.4.6)$$

蚁群算法控制参数设置： $\alpha=\beta=1$ ， $w_t=1$ ， $w_a=2$ ， $\rho=0.2$ ， $Q=1000$ ，而蚂蚁数目  $m$  的设置有如下两种选择：

- $m$  等于任务图平均分支数；
- $m$  等于任务图最大分支数。

此处将  $m$  设置为任务图平均分支数。

蚁群算法结束条件：当满足以下条件之一时，蚁群算法终止：

- (1) 蚁群算法迭代代数达到  $\text{Ant}_{\max}$ ；
- (2) 迭代中连续  $\text{Ant}_{\text{die}}$  代，子代优化解改进率都小于  $\text{Ant}_{\min\text{-improv-ratio}}$ ；同时设置  $\text{Ant}_{\max} = 100$ ,  $\text{Ant}_{\text{die}} = 3$ ,  $\text{Ant}_{\min\text{-improv-ratio}} = 0.5\%$ 。

#### 8.4.3.3 DCG3A 中遗传算法与蚁群算法的衔接

DCG3A 前期执行遗传算法过程，后期执行蚁群算法过程，因此两者的衔接很重要。主要包括如下问题。

- (1) 动态融合时机设置：同前述“遗传算法结束条件”。
- (2) 蚁群算法信息量初值设置：MMAS 把各路径信息量初值设定为最大值  $\tau_{\max}$ ，文献 [34] 中所提算法把初值固定为  $\tau_0 = 100$ 。这里利用遗传算法得到初始信息素分布，将各边  $e_{ij}$  的信息量初值设定为

$$\tau_{ij}^S(k) = \tau_{ij}^C(k) + \tau_{ij}^G(k) \quad (8.4.7)$$

式中， $\tau_{ij}^C(k)$  表示  $e_{ij}$  上  $c_k$  信息素常数，相当于 MMAS 中的  $\tau_{\min}$ ； $\tau_{ij}^G(k)$  表示从遗传算法求解结果转换而来的  $e_{ij}$  上的  $c_k$  信息量。设置  $\tau_{ij}^C(k) = \tau_{ij}(k)_{\min} = 60$ ，其中  $0 \leq i, j \leq n, k = 1, 2$ 。

遗传算法求解结果向信息量转换：这里选取遗传算法终止时种群中适应值最好的前 10%（即  $0.1 \times N = 0.1 \times 50 = 5$ ）个体作为遗传优化解集合，记为  $S_{10\% \text{ better}}^{\text{gene}}$ 。开始时，设置  $\tau_{ij}^G(k)$  为 0,  $0 \leq i, j \leq n, k = 1, 2$ 。对于  $S_{10\% \text{ better}}^{\text{gene}}$  中的每个解  $s$ ，如果边  $e_{ij}$  被着色为  $c_k$ ，则  $\tau_{ij}^G(k)$  自加 20。

#### 8.4.3.4 基于 DCG3A 的软硬件划分算法过程

首先将问题描述为双着色模型，随后执行遗传算法过程，直到到达最佳融合时机为止，然后按公式 (8.4.7) 将遗传算法的求解结果转换为蚁群算法的信息量初值，最后执行蚁群算法直到结束。

下面详细描述 DCG3A 中遗传算法、蚁群算法以及两者衔接部分的执行过程。

##### 1. 遗传算法部分

- (1) 初始化遗传算法控制参数（种群规模  $N = 50$ ，杂交概率  $p_c = 0.6$ ，变异概率  $p_m = 0.2$ ）。
- (2) 设置遗传算法结束条件 ( $\text{Gene}_{\min} = 15$ ,  $\text{Gene}_{\max} = 50$ ,  $\text{Gene}_{\min\text{-impro-ratio}} = 3\%$ ,  $\text{Gene}_{\text{die}} = 3$ )。

(3) 随机生成初始种群  $P(0)$ ,  $g=0$  ( $g$  是遗传代数)。

(4) 计算  $P(0)$  中个体的适应值。

(5) 反复执行下列操作, 直到满足遗传算法的结束条件为止:

- ① 根据个体适应值及赌轮选择策略确定  $P(g)$  内每个个体的选择概率  $p_i$ ;
- ② For ( $k=0$ ;  $k < N$ ;  $k=k+2$ )  
 {  
 1) 根据概率  $p_i$  在  $P(g)$  内选择两个父体;  
 2)  $r=\text{random } [0, 1]$ ;  
 3) 若  $r \leq p_m$ , 对所选两个父体执行变异操作, 并将所得两个后代插入新群体  $P(g+1)$  中;  
 else if ( $r \leq p_m + p_c$ ), 执行杂交操作, 并将所得两个后代插入新群体  $P(g+1)$  中;  
 else 执行繁殖操作, 将两个父体不变地插入新群体  $P(g+1)$  中;  
 } //end for ( $k=0$ ;  $k < N$ ;  $k=k+2$ )
- ③ 计算  $P(g+1)$  中个体的适应值,  $g=g+1$ ;

## 2. 遗传算法与蚁群算法衔接部分

- (6) 从  $P(g)$  中选择适应能力强的前 10% 个体 (5 个), 放入集合  $S_{10\% \text{ better}}^{\text{gene}}$  中, 作为优化解集合。
- (7) 对于  $S_{10\% \text{ better}}^{\text{gene}}$  中的每个优化解  $s$ , 按前面描述的“遗传算法求解结果向信息量转换”策略、“蚁群算法信息量初值设置”策略以及公式 (8.4.7), 计算任务图中各边  $e_{ij}$  关于  $c_k$  的信息量初值  $\tau_{ij}^s(k)$ , 其中  $k=1, 2$ 。

## 3. 蚁群算法部分

- (8) 初始化蚁群算法控制参数 ( $\alpha=\beta=1$ ,  $w_t=1$ ,  $w_a=2$ ,  $\rho=0.2$ ,  $Q=1000$ )。
- (9) 设置蚁群算法结束条件 ( $\text{Ant}_{\max}=100$ ,  $\text{Ant}_{\text{die}}=3$ ,  $\text{Ant}_{\text{min-improv-ratio}}=0.5\%$ )。
- (10) 在节点  $t_0$  处放置  $m$  只蚂蚁 ( $t_0$  是唯一的起点,  $m$  取为任务图的平均分支数)。
- (11) 每只蚂蚁按任务图所对应 DAG 中边的方向遍历所有的边和节点, 最终到达节点  $t_n$ 。每只蚂蚁在遍历过程中, 都要根据公式 (8.4.4) 和公式 (8.4.6) 所计算的概率, 猜测所达节点的颜色, 并最终确定一个可行的着色方案  $s_i$ , 其中  $i=1, \dots, m$ ,  $t_n$  是唯一的终点。
- (12) 计算所得  $m$  种着色方案的适应值, 并从中选择最佳方案  $s_{\text{best}} = \text{argmin} (\text{time}_s)$  (即满足面积约束并且运行时间最小的着色方案)。
- (13) 根据公式 (8.4.2) 和公式 (8.4.3) 更新任务图中所有边的信息量。

(14) 若满足蚁群算法结束条件，则输出最佳着色方案  $s_{best}$  并结束；否则，跳转到步骤 10。

#### 8.4.3.5 算法分析与讨论

##### 1. 复杂性分析

DCG3A 的运行时间  $T_{DCG3A}$  主要由遗传算法部分运行时间  $T_{GA}$ 、蚁群算法部分运行时间  $T_{ACA}$  和衔接部分运行时间  $T_{JOINT}$  组成，可表示为  $T_{DCG3A} = T_{GA} + T_{ACA} + T_{JOINT}$ 。DCG3A 中的遗传算法与蚁群算法过程采用相同的适应值计算规则，因此，这里可使用一致的时间值  $T_{cal-fitness}$  来表示这两个算法中计算每个划分方案适应值所需的时间。下面分别分析  $T_{GA}$ 、 $T_{ACA}$  与  $T_{JOINT}$  的大小。

(1)  $T_{GA}$  包括遗传算法初始设置时间  $T_{GA-init}$  (步骤 1~步骤 3)、终止条件判断时间  $T_{GA-termi}$  和种群进化迭代时间 (步骤 4、步骤 5)。假设遗传算法进化了  $I_{GA}$  代，种群规模为  $N$ ，种群中每个个体完成一次遗传操作 (步骤 5 中的过程①、②) 所需的时间为  $T_{GA-operation}$ ，则种群迭代进化时间为  $I_{GA} \times N \times (T_{GA-operation} + T_{cal-fitness})$ 。这样，

$T_{GA} = T_{GA-init} + T_{GA-termi} + I_{GA} \times N \times (T_{GA-operation} + T_{cal-fitness})$ ，而  $T_{GA-init} + T_{GA-termi}$  远小于  $I_{GA} \times N \times (T_{GA-operation} + T_{cal-fitness})$ ，因此，有

$$T_{GA} \approx I_{GA} \times N \times (T_{GA-operation} + T_{cal-fitness}) \quad (8.4.8)$$

(2)  $T_{ACA}$  包括蚁群算法初始设置时间  $T_{ACA-init}$  (步骤 8、步骤 9)、终止条件判断时间  $T_{ACA-termi}$  和蚁群算法迭代时间 (步骤 10~步骤 14)。假设蚁群算法迭代了  $I_{ACA}$  次，每次迭代中使用了  $m$  只蚂蚁，每只蚂蚁完成一次任务图着色 (即求取一个解) 的时间为  $T_{ACA-operation}$ ，则蚁群算法的迭代时间为  $I_{ACA} \times m \times (T_{ACA-operation} + T_{cal-fitness})$ 。由此，可得

$$T_{ACA} = T_{ACA-init} + T_{ACA-termi} + I_{ACA} \times m \times (T_{ACA-operation} + T_{cal-fitness}) \quad (8.4.9)$$

而  $T_{ACA-init} + T_{ACA-termi}$  远小于  $I_{ACA} \times m \times (T_{ACA-operation} + T_{cal-fitness})$ ，因此，有

$$T_{ACA} \approx I_{ACA} \times m \times (T_{ACA-operation} + T_{cal-fitness}) \quad (8.4.10)$$

(3)  $T_{JOINT}$  包括构造优化解集合 (步骤 6) 所需的时间  $T_{JOINT-10\%better}$  与计算信息量初值 (步骤 7) 所需的时间  $T_{JOINT-initial-pheromone}$ 。假设采用逐个顺序比较的方法构造优化解集合，那么  $T_{JOINT-10\%better} \approx N \times N \times 10\% \times T_{float-compare}$  (其中， $T_{float-compare}$  为完成一次浮点数比较所需的时间；10% 表示选择了适应值高的 10% 个体)。进一步假设任务图节点数为  $K$ ，边数为  $H$ ，每次计算一条边上的信息量的时间为  $T_{JOINT-edge-pheromone}$ ，那么，信息量初值计算时间  $T_{JOINT-initial-pheromone}$  为  $N \times 10\% \times H \times T_{JOINT-edge-pheromone}$ 。这样，即有

$$T_{JOINT} \approx N \times N \times 10\% \times T_{float-compare} + N \times 10\% \times H \times T_{JOINT-edge-pheromone} \quad (8.4.11)$$

由上述分析可得

$$T_{DCG3A} \approx I_{GA} \times N \times (T_{GA-operation} + T_{cal-fitness}) + I_{ACA} \times m \times (T_{ACA-operation} + T_{cal-fitness}) \\ + N \times N \times 10\% \times T_{float-compare} + N \times 10\% \times H \times T_{JOINT-edge-pheromone} \quad (8.4.12)$$

实际运算中,发现  $T_{cal-fitness}$  的复杂度为  $O(K \times H)$ ,远大于  $T_{GA-operation}$ ,  $T_{ACA-operation}$  以及  $N \times N \times 10\% \times T_{float-compare} + N \times 10\% \times H \times T_{JOINT-edge-pheromone}$ 。因此,可以将  $T_{DCG3A}$  简化为

$$T_{DCG3A} \approx I_{GA} \times N \times T_{cal-fitness} + I_{ACA} \times m \times T_{cal-fitness} \\ = (I_{GA} \times N + I_{ACA} \times m) \times T_{cal-fitness} \quad (8.4.13)$$

在运行过程中,遗传算法部分所需的存储空间约为  $2 \times N \times O(K)$ ,蚁群算法部分所需的存储空间约为  $m \times O(K + H)$ ,算法衔接部分构造优化解集合所需的存储空间为  $10\% \times N \times O(K)$ 。当然,不同的具体实现对 DCG3A 的运算时间与存储空间也有一定的影响。

## 2. 算法运行效率探讨

上述复杂性分析表明,DCG3A 的运行时间  $T_{DCG3A}$  主要取决于  $I_{GA} \times N + I_{ACA} \times m$  与  $T_{cal-fitness}$  的乘积。对于特定的组合优化问题,  $T_{cal-fitness}$  本质上是评价问题空间中一个解所需的时间。因此,DCG3A 效率改进的关键在于最小化  $I_{GA} \times N + I_{ACA} \times m$ ,其中  $N, m$  是控制参数,分别表示遗传算法的种群规模和蚁群算法的蚂蚁数目,在算法中一般是固定的值。极端情况下,当遗传算法迭代次数  $I_{GA}$  为 0 时,DCG3A 就退化为基本蚁群算法;当 DCG3A 中的蚁群算法迭代次数  $I_{ACA}$  为 0 时,DCG3A 则退化为遗传算法。

DCG3A 在遗传算法运行过程中加入了子代优化解改进效率检测机制,使得当遗传算法对优化解改进效率降低到一定程度时,及时终止遗传过程,从而避免  $I_{GA}$  无谓的增长。此外,由于有了较为准确的初始信息素,使蚁群算法大大降低了用于形成最优解上信息素所需的迭代次数。这时利用蚁群算法正反馈、高效收敛的优势,可以在  $I_{ACA}$  很小的情况下迅速找到最优解。因此,DCG3A 通过抑制  $I_{GA}$  与  $I_{ACA}$  无谓的增长,从而最小化  $I_{GA} \times N + I_{ACA} \times m$  来达到提高搜索效率的目的。

## 8.4.4 实验

采用与文献 [34] 相似的实验模型和方法,与基于遗传算法<sup>[24]</sup>、基于蚁群算法<sup>[34]</sup>的软硬件划分进行比较。

### 8.4.4.1 目标系统结构

这里考虑软硬件系统任务双路划分问题,目标系统中有一个处理器和一个硬

件可编程部件（如 FPGA）。使用 Xilinx Virtex II Pro Platform FPGA 作为参考模型，它最多可包含 4 个处理器核，13404 个可编程逻辑块。实验中，可限制使用一个处理器和最多 1232 个可编程逻辑块。目标系统结构的抽象模型如图 8.18 所示。

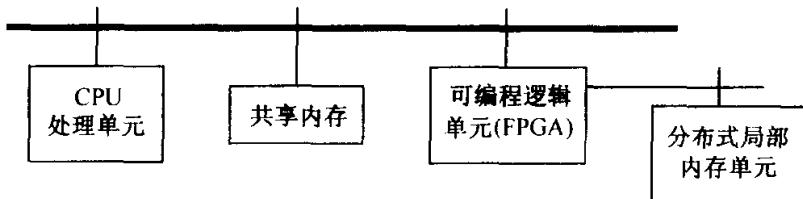


图 8.18 目标系统结构的抽象模型

#### 8.4.4.2 实验假设

为简化实验，这里提出了以下合理假设：

- (1) 任务在处理器和在可编程部件上实现所需的运行时间和占用的硬件面积是静态的，可预先计算出来。
- (2) 任务间通信开销不变并已经包含在任务的开销（时间、面积）中。
- (3) 不考虑硬件实现的并行性。

#### 8.4.4.3 实验环境与实例

目前，在嵌入式系统软硬件划分领域，国际上还没有公认的测试集<sup>[34]</sup>。常见做法是随机生成 DAG，并对其节点与边赋以属性。这里采用以下方法构造测试集：

- (1) 用 GVF 工具包随机生成指定节点数、指定平均分支数的若干个 DAG，以此作为任务图。
- (2) 将每个任务与一个函数（或算法过程）相关联，并以此函数的开销（运行时间、硬件面积、通信代价等）作为任务开销。
- (3) 从 MediaBench 基准程序包中选择与任务相关联的函数。

利用上述方法生成 7 组 DAG（分别记为  $DAG_{20}, DAG_{30}, \dots, DAG_{80}$ ），每组包含 30 个 DAG 样本。7 组 DAG 的节点数分别为 20, 30, …, 80，平均分支数依次为 5, 5, 7, 7, 9, 9, 11。以各组 30 个样本的性能平均值作为该组的最终性能结果。

实验环境如下：

- AMD Duron 700MHz 处理器，128MB 内存；
- Linux 7.3 操作系统；

- KDevelop 2.1 编程环境。

#### 8.4.4.4 实验结果与分析

表 8.1 给出了 DCG3A 与遗传算法<sup>[24]</sup>、蚁群算法<sup>[34]</sup>进行软硬件划分得到的实验数据。其中，遗传算法与蚁群算法中控制参数的设置与 DCG3A 中相应的设置相同。这里使这三种算法求得的最优解与理论最优解之间的误差基本一致，以保证公平性。

表 8.1 遗传算法、蚁群算法与 DCG3A 实验结果对比

DAG 节点规模	GA		ACA		DCG3A	
	时间 (s)	迭代次数	时间 (s)	迭代次数	时间 (s)	迭代次数
20	258	72.3	332	43.2	207	23.1+14.6
30	579	65.1	659	36.8	513	31.2+12.4
40	1 296	83.6	1 463	47.3	742	19.4+21.6
50	2 694	89.2	2 235	44.7	1 662	27.7+18.3
60	4 833	92.7	3 280	68.7	2 038	26.8+21.8
70	8 436	68.0	5 762	51.3	2 679	21.4+19.4
80	15 623	97.5	9 368	55.1	3 651	27.2+16.3

图 8.19 描述了表 8.1 中三种算法求解时间与节点规模之间的关系。

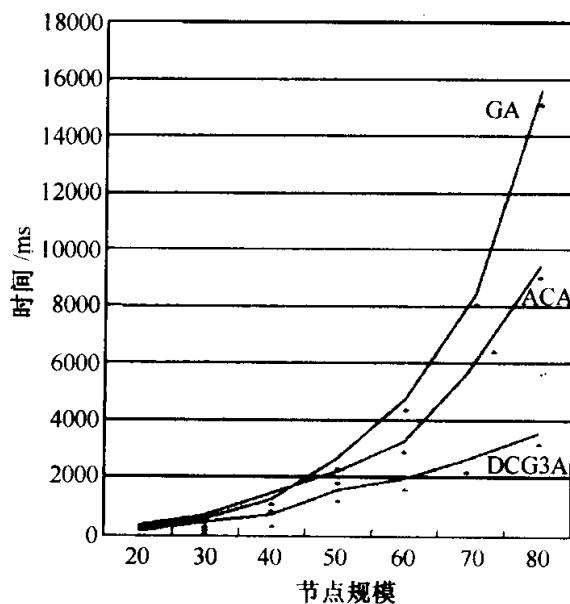


图 8.19 运算时间-节点规模曲线

当任务图节点数较少（20~30）时，DCG3A 略优于遗传算法和蚁群算法，但效果不很明显。而当节点数超过 40 时，DCG3A 明显优于遗传算法和蚁群算法。随着任务图规模的增大，性能改进的效果越显著。算法控制参数对实验结果也有一定的影响，本实验使用了文献 [24, 34] 中相同的控制参数，以增强可比性。对于特定的组合优化问题，可分别使用相应问题中已验证的优化控制参数，也可以使用典型控制参数设置值<sup>[38, 39]</sup>。

控制参数  $\text{Gene}_{\text{min-impro-ratio}}$  决定了遗传算法的终止时机，此值若过大，则遗传算法在更接近  $t_b$ （如图 8.16 所示）的时刻结束；若过小，遗传算法就将在靠近  $t_c$  的时刻结束。实验发现，当  $\text{Gene}_{\text{min-impro-ratio}} = 3\%$  时，遗传算法在  $t_b$  与  $t_c$  之间中间点位置（靠近最佳融合时机  $t_a$ ）结束，因此，本实验将  $\text{Gene}_{\text{min-impro-ratio}}$  设置为 3%。

在所使用的目标体系结构下，本实验在一定的硬件面积约束下优化系统的运行时间。实际上，在嵌入式系统与 SoC 软硬件划分中，其他性能指标（如功耗）的约束与优化也是非常重要的方面，DCG3A 同样适用于求解这类划分问题。另外，对于具有强实时性需求的嵌入式系统与 SoC 软硬件划分问题，需要多个嵌入式处理器协同工作，这时除了运用 DCG3A 进行划分求解外，还需要在划分中考虑嵌入式处理器之间的任务分配与负载平衡。

实验表明，与已有的基于遗传操作的软硬件划分算法和基于蚁群优化的软硬件划分算法相比，DCG3A 的运行效率提高了约 1 倍，而且划分问题规模越大，改进效果越明显。

## 8.5 本章小结

仿生硬件是并行计算环境下的产物，蚁群算法的硬件实现是仿生硬件研究领域新出现的一个子分支。本章在对仿生硬件定义、发展概况、基本原理以及主要特点进行简要阐述的基础上，研究了一种基于 FPGA 的蚁群算法硬件实现方案，并详细给出了 P-ACO 算法 FPGA 硬件总体结构中各主要模块的实现过程，同时对如何更有效利用 P-ACO 算法的启发式信息进行了探讨；随后研究了一种基于 DCG3A 的软硬件划分策略。实验表明，DCG3A 在嵌入式系统和 SoC 软硬件划分中取得了很好的效果。

虽然蚁群算法的硬件实现技术已经取得了明显进展，但是对其可靠性、可测试性、普适性与鲁棒性等问题还需进一步研究<sup>[40~42]</sup>。此外，开发可批量工程应用的蚁群算法仿生硬件芯片、研制蚁群算法仿生硬件的开发平台、建立蚁群算法仿生硬件产品的技术规范等都是今后蚁群算法硬件研究领域的一些重要研究内容。

## 参 考 文 献

- 1 Isaacs J C, Watkins R K, Foo S Y. Evolving ant colony systems in hardware for random number generation. Proceedings of the 2002 Congress on Evolutionary Computation, 2002, 1450~1455
- 2 忻斌健, 汪镭, 吴启迪. 蚁群算法的研究现状和应用及蚂蚁智能体的硬件实现. 同济大学学报, 2002, 30(1): 82~87
- 3 Scheuermann B, So K, Guntzsch M, *et al.* FPGA implementation of population-based ant colony optimization. Applied Soft Computing, 2004, 4(4): 303~322
- 4 Scheuermann B, Guntzsch M, Middendorf M, *et al.* Time-scattered heuristic for the hardware implementation of population-based ACO. Proceedings of the 4th International Workshop on Ant Algorithms/ANTS2004, Lecture Notes in Computer Science, 2004, 3172: 250~261
- 5 Xiong Z H, Chen J H, Li S K. Hardware/software partitioning for platform-based design method. Proceedings of the 2005 Asia and South Pacific Design Automation Conference, 2005, 2: 691~696
- 6 熊志辉, 李思昆, 陈吉华. 遗传算法与蚂蚁算法动态融合的软硬件划分. 软件学报, 2005, 16(4): 503~512
- 7 段海滨, 王道波, 朱家强等. 蚁群算法理论及应用研究的进展. 控制与决策, 2004, 19(12): 1321~1326, 1340
- 8 Neumann J Von. Theory of self-reproducing automata. Illinois: University of Illinois Press, 1966
- 9 Garis H de. Evolvable hardware: genetic programming of Darwin machine. Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, 1993, 441~449
- 10 Levi D, Guccione S A. Genetic FPGA: Evolving stable circuits on mainstream FPGA devices. Proceedings of the 1st NASA/DOD Workshop on Evolvable Hardware, 1999, 12~17
- 11 Langeheine J, Becker J, Folling S, *et al.* A CMOS FPTA chip for intrinsic hardware evolution of analog electronic circuits. Proceedings of the 3rd NASA/ DOD Workshop on Evolvable Hardware, 2001, 172~175
- 12 Back T, Hammel U, Schwefel H P. Evolutionary computation: comments on the history and current state. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 3~17
- 13 Yao X, Higuchi T. Promises and challenges of evolvable hardware. IEEE Transactions on Systems, Man, and Cybernetics-Part C, 1999, 29(1): 87~97
- 14 康立山, 何魏. 用函数型可编程器件实现演化硬件. 计算机学报, 1999, 22(7): 781~784
- 15 赵曙光, 杨万海. 基于函数级 FPGA 原型的硬件内部进化. 计算机学报, 2002, 25(6): 666~669
- 16 Stephen L S, David P C. Evolving image processing operations for an evolvable hardware environment. Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware, 2003, 332~343
- 17 Andy T, Eduardo S, Dario F, *et al.* POEtic tissue; an integrated architecture for bio-inspired hardware. Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware, 2003, 129~140
- 18 Guntzsch M, Middendorf M, Scheuermann B, *et al.* Population based ant colony optimization on FPGA. Proceedings of the 2002 IEEE International Conference on Field- Programmable Technology, 2002, 125~132
- 19 Merkle D, Middendorf M. Fast ant colony optimization on runtime reconfigurable processor arrays. Genetic Programming and Evolvable Machines, 2002, 3(4): 345~361
- 20 Guntzsch M, Middendorf M. Applying population based ACO to dynamic optimization problems. Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science, 2002, 2463: 111~122

- 21 Guntsch M, Middendorf M. A population based approach for ACO. Proceedings of the Applications of Evolutionary Computing-Evolutionary Workshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN, Lecture Notes in Computer Science, 2002, 2279: 72~81
- 22 Du J Z, Leung Joseph Y T. Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research, 1990, 15(3): 483~485
- 23 Ackermann J, Tangen U, Bödekker B, *et al.* Parallel random number generator for inexpensive configurable hardware cells. Computer Physics Communications, 2001, 140(3): 293~302
- 24 张鲁峰. 软硬件协同综合及虚拟微处理器技术研究. 长沙: 国防科学技术大学博士学位论文, 2002
- 25 Gupta R K, Micheli G D. System-level synthesis using re-programmable components. Proceedings of the European Conference on Design Automation, 1992, 2~7
- 26 Kastner R. Synthesis techniques and optimizations for reconfigurable systems. Ph. D. Thesis, Los Angeles: University of California, 2002
- 27 Ernst R, Henkel J, Benner T. Hardware-software cosynthesis for microcontrollers. IEEE Design & Test of Computers, 1993, 10(4): 64~75
- 28 Saha D, Mitra R S, Basu A. Hardware software partitioning using genetic algorithm. Proceedings of the 10th International Conference on VLSI Design, 1997, 155~160
- 29 郭晓东, 刘积仁, 文晖. 一种基于遗传算法的硬件/软件划分方法. 计算机辅助设计与图形学报, 2001, 13(1): 24~27
- 30 Peng Z, Kuchcinski K. An algorithm for partitioning of application specific systems. Proceedings of the European Conference on Design Automation, 1993, 316~321
- 31 Else P, Peng Z, Kuchcinski K, Doboli A. System level hardware/software partitioning based on simulated annealing and tabu search. Design Automation of Embedded Systems, 1997, 2(1): 5~32
- 32 Kalavade A, Lee E A. The extended partitioning problem: hardware/software mapping, scheduling, and implementation-bin selection. Design Automation of Embedded Systems, 1997, 2(1): 125~163
- 33 程国达, 彭澄廉. 约束驱动与松弛时间消除相结合的硬/软件划分算法. 计算机研究与发展, 2003, 40(6): 889~896
- 34 Wang G, Gong W R, Kastner R. A new approach for task level computational resource bi-partitioning. Proceedings of the International Conference on Parallel and Distributed Computing and Systems, 2003, 434~444
- 35 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法的融合. 计算机研究与发展, 2003, 40(9): 1351~1356
- 36 李智, 许川佩, 莫玮等. 基于蚂蚁算法和遗传算法的同步时序电路初始化. 电子学报, 2003, 31(8): 1276~1280
- 37 Stützle T, Hoos H H. MAX-MIN ant system. Future Generation Computer System, 2000, 16(8): 889~914
- 38 潘正君, 康立山, 陈毓屏. 演化计算. 北京: 清华大学出版社, 1998
- 39 Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- 40 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 41 段海滨, 王道波, 于秀芬. 蚁群算法硬件实现的研究进展. 控制与决策, 待发表
- 42 Mostafa A E B, Sait S M, Sarif B A B, *et al.* A modified ant colony algorithm for evolutionary design of digital circuits. Proceedings of the 2003 Congress on Evolutionary Computation, 2003, 1: 708~715

# 第9章 蚁群算法同其他仿生优化算法的比较与融合

## 9.1 引言

地球上的生物物种在漫长的演化过程中形成了丰富的行为特性，并且一直在不断地完善和发展，以更好地适应其所生存的环境。随着计算机的诞生，人们期望借助计算机程序的形式创造出一些新型的智能体，于是对人类的大脑活动、生物物种的社会行为以及生物界的进化过程进行了模拟研究。

自从 20 世纪 40 年代以来，人们一直在利用来自于生物系统的灵感来解决许多实际问题，并构造和设计出许多仿生优化算法，如人工神经网络、遗传算法、人工免疫算法、蚁群算法、微粒群算法、人工鱼群算法、混合蛙跳算法等，它们都属于一类模拟自然界生物系统行为或过程的最优化仿生智能算法。

本章首先简要介绍几种目前比较流行的仿生优化算法：遗传算法、人工神经网络、微粒群算法、人工免疫算法以及人工鱼群算法；然后分析这些算法的异同之处；随后给出蚁群算法与这些仿生优化算法的融合策略，并通过仿真实例对这些融合策略的有效性作了验证。

## 9.2 其他几种仿生优化算法的基本原理

### 9.2.1 遗传算法

遗传算法（GA）是一类借鉴生物界中自然选择和自然遗传机制的随机化自适应搜索算法，最先由美国 Michigan 大学的 Holland J H 教授于 1975 年在《Adaptation in Natural and Artificial Systems》一书中系统提出<sup>[1]</sup>。GA 主要由选择、交叉和变异算子组成，分别模仿达尔文进化过程中的自然选择和群体遗传过程中发生的交配和突变等现象。GA 应用于实际问题时，要对优化问题进行编码，称之为个体。个体的集合称为群体，每一个体都表示所求问题的一个潜在解。GA 以随机产生的一群初始解（初始群体）为开始，通过遗传操作一代一代地向最优解进化。

选择机制基于适者生存理论，它应用于在群体中的每一个个体按照其适应度值进行选择，并进化到下一代群体的过程中<sup>[2]</sup>。选择的具体方法有很多种形式，一般总是保证当前群体中具有较高适应度值的个体以较大的概率通过复制和繁衍生成新的群体。

GA 利用交叉和变异算子对解空间进行搜索，交叉算子是二者中的主要算子，交叉算子组合了父辈个体的特征，是产生新个体的主要途径，它和选择算子相结合，构成 GA 中交换信息的重要方法，同时也加强了 GA 的全局搜索能力。

GA 中的适应度函数是根据不同的问题和目标而制定的目标函数，当其值越大或者越小时，个体被选中复制的概率就越大。在这一过程中，选择、交叉和变异是 GA 的三个主要操作算子，它们构成了所谓的遗传操作。GA 的基本步骤如下：

- (1) 问题的染色体表示。
- (2) 初始解组（种群）的生成。
- (3) 计算解组中各个解的适应度函数（代价函数）。
- (4) 从解组中随机抽取两个解作为父代。
- (5) 对父代实行交叉、变异等遗传操作，以产生一个后代解。
- (6) 按照某种规则，用该后代解来替换原解组中的某个解。
- (7) 若当前解符合 GA 终止条件，则算法结束，否则跳转到第 (4) 步。

## 9.2.2 人工神经网络

人工神经网络 (artificial neural network, ANN) 是由大量处理单元 (神经元) 广泛互连而成的网络。人类在对其大脑工作机理认识的基础上，以人脑的组织结构和活动规律为背景进行研究，反映了人脑的某些基本特征。

1943 年，心理学家 McCulloch W 和数理逻辑学家 Pitts W 在研究生物神经元的基础上，首先在数学生物物理学会刊《Bulletin of Mathematical Biophysics》上发表论文，该论文利用数理逻辑提出了第一个简单的 ANN 模型，简称 M-P 模型，从此开创了 ANN 研究的先河，并为以后的研究提供了依据。1949 年，心理学家 Hebb D O 发表了论著《The Organization of Behavior》<sup>[3]</sup>，首次提出了调整 ANN 连接权的学习规则，这就是著名的 Hebb 学习律，至今仍在 ANN 的研究中发挥着重要作用。1957 年，Rosenblatt F 首次提出并设计了著名的感知机模型，第一次使得 ANN 从理论研究转入工程实现阶段，掀起了 ANN 研究的高潮。美国加州理工学院生物物理学家 Hopfield J J 博士先后于 1984 年和 1986 年发表了两篇十分重要的论文<sup>[4, 5]</sup>，在其所提出的 Hopfield 网络模型中首次引入了网络能量的概念，并给出了网络稳定性的判定依据。Hopfield 网络不仅在理论分析与综合上达到了相当的深度，最有意义的是，该网络可以用集成电路来实现。

ANN 的学习也称为训练，指的是 ANN 在外界环境的刺激作用下调整网络自由参数，并以新的方式来响应外部环境的过程。能够从环境中学习并在学习中提高自身性能是 ANN 最为显著的性质，ANN 的学习方法主要分为以下三类。

### 1. 监督学习（有教师学习）

在学习时由教师提供期望输出，通常 ANN 对于周围的环境未知，而教师具有周围环境的知识。输入学习样本，教师可根据自身的知识为训练样本提供最佳逼近结果，ANN 的自由参数在误差信号的影响下进行调整，其最终目的是让 ANN 模拟教师。

### 2. 非监督学习（无教师学习）

非监督学习也称为自组织学习，系统在学习过程中，没有外部教师信号，而是提供一个关于网络学习性质的度量，它独立于学习任务，并以此尺度来逐步优化 ANN<sup>[6]</sup>。

### 3. 强化学习（激励学习）

在强化学习系统中，对输入输出映射的学习是通过与外部环境的不断交互作用来完成的，其目的是网络标量函数值最小。

ANN 针对学习问题修改网络自由参数的过程称为学习规则，设计学习规则的目的是训练网络来完成某些任务，没有一个独特的学习规则可以完成所有的学习任务。ANN 有五个基本的学习规则：①误差-修正学习；②基于记忆的学习；③Hebb 学习；④竞争学习；⑤随机学习。

## 9.2.3 微粒群算法

微粒群（particle swarm optimization, PSO）算法是由美国的 Kennedy J 和 Eberhart R C 受鸟群觅食行为的启发于 1995 年提出的一种仿生优化算法<sup>[7, 8]</sup>。最初的设想是仿真简单的社会系统，研究并解释复杂的社会行为，后来发现 PSO 算法可用于复杂优化问题的求解。

PSO 算法的思想来源于人工生命和进化计算理论。生物学家对鸟群飞行的研究发现，鸟仅仅追踪有限数量的邻居，但最终的整体结果是整个鸟群好像在一个中心的控制之下。PSO 算法即源于对鸟群捕食行为的研究，一群鸟在随机搜寻食物，如果某区域内只有一块食物，那么找到该食物的最简单有效的策略就是搜寻目前离食物最近的鸟的周围区域。PSO 算法就是从这种模型中得到启示而产生的，并用于解决组合优化问题。

PSO 算法求解优化问题时，问题的解对应于搜索空间中鸟的实际位置，称这些鸟为“粒子”或“主体”。每个粒子都有自己的位置和速度（决定飞行的方向和距离），还有一个由被优化函数决定的适应度值。各个粒子记忆、追随当前的最优粒子，在解空间中搜索。每次迭代的过程不是完全随机的，如果找到较好解，将会以此为依据来寻找下一个解。PSO 算法的基本步骤如下。

(1) 初始化：初始搜索点的位置及其速度通常是在允许范围内随机产生的，每个粒子的个体极值点坐标设置为其当前位置，且计算出其相应的个体极值（即个体极值点的适应度值）。而全局极值（即全局极值点的适应度值）就是个体极值中最好的，记录该最好值的粒子序号，并将全局极值点设置为最好粒子的当前位置。

(2) 评价每一个粒子：计算粒子的适应度值，如果好于该粒子当前的个体极值，则将个体极值点设置为该粒子的位置，且更新个体极值。如果所有粒子的个体极值中的最好值好于当前的全局极值，则将全局极值点设置为该粒子的位置，记录该粒子的序号，且更新全局极值。

(3) 粒子的更新：用速度和位置的更新方程对每一个粒子的速度和位置进行更新。

(4) 检验是否符合结束条件：如果满足结束条件，则停止迭代，输出最优解；否则跳转到第(2)步。

#### 9.2.4 人工免疫算法

免疫系统是哺乳动物抵御外来有害物质侵害的防御系统，动物一生始终处于复杂多变、充满伤害的自然环境中，但它们能够平安无事地进行正常的生命活动，其原因是免疫系统在其中起着重要作用。人工免疫算法 (artificial immune algorithm, AIA) 正是基于生物免疫抗体 (antibody) 产生记忆系统的学习机理的产物。这方面的研究最初从 20 世纪 80 年代中期的免疫学研究发展而来，20 世纪 90 年代初，Bersini H 和 Varela F 首次使用 AIA 来解决实际问题<sup>[9]</sup>。

AIA 依据的主要免疫学原理包括免疫网络理论、克隆选择原理、免疫学习机制等。免疫系统由淋巴细胞把自身细胞与抗原识别出来，并产生抗体对付入侵的抗原，以达到消灭抗原的目的。免疫系统对自身也具有免疫能力，它能抑制过多抗体的产生。免疫系统有能力产生很多种抗体，但实际上系统只需根据抗原的种类和数目产生适量的相关抗体。根据免疫网络理论，每个细胞系的 B 型淋巴细胞识别出感受器的遗传类型后，相互之间便构成了链接的网络，从而只产生所需数量的抗体，免疫系统的控制机制可完成这一调节功能。如果抗原的刺激激活了某细胞系中的细胞并开始繁殖，则其他能识别这种基因类型的细胞系也被激活并开始繁殖。如果这一过程连续地进行，就构成了对自身的免疫，并通过所有淋巴细胞的作用实现了内部调节机制<sup>[10~12]</sup>。AIA 的基本步骤如下。

(1) 抗原输入并识别抗原：免疫系统确认抗原入侵。

(2) 产生初始抗体群体：激活记忆细胞产生抗体，清除以前出现过的抗原，从包含最优抗体（最优解）的数据库中用随机的方法来选择抗体。

(3) 计算亲和力 (affinity)：分别计算抗原与抗体之间的亲和力以及抗体与

抗体之间的亲和力。

(4) 记忆细胞分化：与抗原有最大亲和力的抗体加给记忆细胞。由于记忆细胞数目有限，新产生的与抗原具有更高亲和力的抗体替换较低亲和力的抗体。

(5) 抗体促进和抑制：高亲和力抗体受到促进，高密度抗体受到抑制。通常通过计算抗体存活的期望值来实施。

(6) 产生新抗体：一般而言，与抗原亲合性高的抗体和低密度的抗体生存机率较大，但为了有利于优化过程的进行，某些与抗原有较高亲和力的抗体也必须受到抑制，从而体现了抗体克隆控制机制的多样性，随后可再随机产生新的抗体，也可通过变异和交叉产生进入下一代的抗体。

(7) 终止条件：通常采用限定迭代次数或在连续几次迭代中的最好解都无法改善，以及二者的混合形式作为终止条件。

在使用 AIA 解决问题时，一般各个步骤有其对应形式：抗原对应于所求问题的数据输入；抗体对应于所求问题的最优解；亲和力对应于对解的评估和对结合强度的评估；记忆细胞分化对应于保留优化解；抗体促进和抑制对应于优化解的促进及非优化解的删除；抗体产生对应于优化解的出现；等等。

## 9.2.5 人工鱼群算法

人工鱼群算法 (artificial fish-swarm algorithm, AFA) 是李晓磊等<sup>[13]</sup>在前人对群体智能行为研究的基础上提出的一种新型仿生优化算法，该算法根据“水域中鱼生存数目最多的地方一般就是该水域中富含营养物质最多的地方”这一特点来模拟鱼群的觅食行为来实现全局寻优<sup>[14~17]</sup>。

### 9.2.5.1 人工鱼模型 (artificial fish, AF)

人工鱼的模型可用如下一个类来描述：

```
class Artificial_fish
{
    float AF-X[n]; //人工鱼的位置
    float AF-step; //人工鱼移动的移动步长
    float AF-visual; //人工鱼的感知距离
    float AF-food consistence(); //人工鱼当前位置的食物浓度
    void AF-move(); //人工鱼移动到下一位置
    float AF-follow(); //追尾行为
    float AF-prey(); //觅食行为
    float AF-swarm(); //聚群行为
    int AF-evaluate(); //评价和选择行为
}
```

```

void AF-init( ); //人工鱼初始化
Artificial-fish( );
Virtual~Artificial-fish( );
}

```

## 1. 符号定义

为便于对 AFA 进行描述，这里对算法中一些符号的含义做如下说明：

人工鱼个体的状态可表示为向量  $X=(x_1, x_2, \dots, x_n)$ ，其中  $x_i (i=1, \dots, n)$  表示欲寻优的变量；人工鱼当前所在位置的食物浓度表示为  $Y=f(X)$ ，其中  $Y$  为目标函数值；人工鱼个体之间的距离表示为  $d_{i,j} = \|X_i - X_j\|$ ；Visual 表示人工鱼的感知距离；Step 表示人工鱼移动的步长； $\delta$  表示拥挤度因子；try-number 表示人工鱼每次移动时最大的试探次数。

## 2. 鱼群行为描述

### (1) 觅食行为。

设人工鱼当前状态为  $X_i$ ，在其感知范围内随机选择一个状态  $X_j$ ，如果在求极大问题中， $Y_i < Y_j$ ，则向该方向前进一步；反之，再重新随机选择状态  $X_j$ ，判断是否满足前进条件；反复几次后，如果仍不满足前进条件，则随机移动一步。其伪代码描述如下：

```

float Artificial-fish::AF-prey( )
{
    For ( i=0; i<try-number; i ++ )
    {
         $X_j = X_i + \text{Random(Visual)}$ ;
        If ( $Y_i < Y_j$ )
             $X_{i|\text{next}} = X_i + \text{Random(Step)}(X_j - X_i) / \|X_j - X_i\|$ 
        else
             $X_{i|\text{next}} = X_i + \text{Random(Step)}$ ;
    }
    Return AF-foodconsistence( $X_{i|\text{next}}$ );
}

```

### (2) 聚群行为。

鱼在游动过程中会自然地聚集成群，这也是为了保证群体的生存和躲避危害而形成的一种生活习性。鱼群的形成是一种突现的生动示例，Reynolds C W 认

为鸟类和鱼类群集行为的形成并不需要一个领头者<sup>[18]</sup>，只需每只鸟或每条鱼遵循一些局部的相互作用规则即可，然后群集行为作为整体模式从个体局部的相互作用中突现出来。Reynolds C W 所采用的规则有三条，AFA 中基本沿用了这三条规则，具体如下：

- ① 分隔规则：尽量避免与临近伙伴过于拥挤；
- ② 对准规则：尽量与临近伙伴的平均方向一致；
- ③ 内聚规则：尽量朝临近伙伴的中心移动。

设人工鱼当前状态为  $X_i$ ，探索当前邻域内（即  $d_{i,j} < \text{Visual}$ ）的伙伴数目  $n_f$  及中心位置  $X_c$ ，如果  $Y_c/n_f > \delta Y_i$ ，表明伙伴中心有较多的食物并且不太拥挤，则朝伙伴的中心位置方向前进一步；否则执行觅食行为。其伪代码描述如下：

```

float Artificial-fish::AF-swarm( )
{
    n_f = 0; X_c = 0;
    For (j=0; j<friend-number; j++)
        If (d_{i,j} < Visual)
        {
            n_f++;
            X_c += X_j;
        }
    X_c = X_c / n_f;
    If (Y_c / n_f > δY_i)
        X_{i,next} = X_i + Random(Step)(X_c - X_i) / || X_c - X_i ||
    else
        AF-prey();
    Return AF-foodconsistence(X_{i,next});
}

```

### (3) 追尾行为。

在鱼群的游动过程中，当其中一条或几条发现食物时，其临近的伙伴会尾随其快速到达食物点。设人工鱼当前状态为  $X_i$ ，探索当前邻域内（即  $d_{i,j} < \text{Visual}$ ）的伙伴中  $Y_j$  为最大的伙伴  $X_j$ ，如果  $Y_j/n_f > \delta Y_i$ ，表明伙伴  $X_j$  的状态具有较高的食物浓度并且其周围不太拥挤，则朝伙伴  $X_j$  的方向前进一步；否则执行觅食行为。其伪代码描述如下：

```

float Artificial-fish:: AF-follow( )
{
    Y_max = -∞;
    For (j=0; j<friend-number; j++)

```

```

If ( $d_{i,j} < \text{Visual} \& \& Y_j > Y_{\max}$ )
{
     $Y_{\max} = Y_j$  ;
     $X_{\max} = X_j$ 
}
 $n = 0$  ;
For ( $j = 0$ ;  $j < \text{friend-number}$ ;  $j++$ )
If ( $d_{\max, j} < \text{Visual}$ )
{
     $n_f++$  ;
}
If ( $Y_{\max}/n_f > \delta Y_i$ )
     $X_{i+1} = X_i + \text{Random(Step)}(X_{\max} - X_i) / \|X_{\max} - X_i\|$  ;
else
    AF-prey();
Return AF-foodconsistence( $X_{i+1}$ );
}

```

#### (4) 约束行为。

在寻优过程中，由于聚群行为、随机行为等操作的作用，容易使人工鱼的状态变得不可行，这时就需要加入相应的约束来对其进行规整化，使它们由无效状态或不可行状态转变成可行状态。

#### (5) 公告板。

公告板用来记录最优人工鱼个体的状态。各人工鱼个体在寻优过程中，每次行动完毕就检验自身状态与公告板的状态，如果自身状态优于公告板状态，就将公告板的状态改写为自身状态，这样就使公告板记录下了历史最优的状态。

#### (6) 移动策略。

根据所要解决的问题性质，对人工鱼当前所处的环境进行评价，从而选择一种合适的行为策略。

由此，有了底层的人工鱼模型，算法的展开就在一群人工鱼的自主活动中开始了。整个算法没有高层的指挥者，也不需要关于所求问题的先验知识的启发，每条人工鱼就按照自己的规则游动着，所求问题的满意解就这样在公告板上显示出来。AFA 整体表现出快速向极值域收敛的特性，随机移动行为的存在使得寻优活动更加全面地展开，便于 AFA 全局收敛。

### 9.2.5.2 算法描述

鉴于以上描述的人工鱼模型及其行为，每个人工鱼探索其当前所处的环境状况和伙伴状况。其实伙伴状况相对于其自身应该也归属于环境状况，从而选择一种行为，最终人工鱼聚集在几个局部极值的周围。一般而言，在讨论求极大值问题时，拥有较大的 AF-Foodconsistence 值的人工鱼往往处于值较大的极值域周围，这有助于获取全局极值域，而值较大的极值域周围一般能聚集较多的人工鱼，这有助于判断并获取全局极值。AFA 的基本描述如下：

```
Procedure Artificial _ Fish-swarm _ Algorithm
    ::AF init( );
    While 不满足算法结束条件 do
        Switch ( ::AF _ evaluate( ))
            case value 1:
                ::AF _ follow( )
            case value2:
                ::AF _ swarm( );
            default:
                :: AF _ prey( );
        End switch
        ::AF_move( );
        get_result( );
    End while
```

## 9.3 蚁群算法与其他仿生优化算法的异同比较

### 9.3.1 相同点

蚁群算法同目前流行的遗传算法、人工神经网络、微粒群算法、人工免疫算法、人工鱼群算法等都属于仿生优化算法，它们都属于一类模拟自然界生物系统、完全依赖生物体自身本能、通过无意识寻优行为来优化其生存状态以适应环境需要的最优化智能算法<sup>[19~27]</sup>。因此，这些仿生优化算法有许多相同的特点。

#### 1. 都是一类不确定的算法

不确定性体现了自然界生物的生理机制，并且在求解某些特定问题方面优于确定性算法。仿生优化算法的不确定性是伴随其随机性而来的，其主要步骤含有

随机因素，从而在算法的迭代过程中，事件发生与否有很大的不确定性。

2. 都是一类概率型的全局优化算法

非确定算法的优点在于算法能有更多的机会求得全局最优解。

3. 都不依赖于优化问题本身的严格数学性质

在优化过程中都不依赖于优化问题本身的严格数学性质（如连续性、可导性）以及目标函数和约束条件的精确数学描述。

4. 都是一种基于多个智能体的仿生优化算法

仿生优化算法中的各个智能体之间通过相互协作来更好地适应环境，表现出与环境交互的能力。

5. 都具有本质并行性

仿生优化算法的本质并行性表现在两个方面：一是仿生优化计算的内在并行性 (inherent parallelism)，即仿生优化算法本身非常适合大规模并行；二是仿生优化计算的内含并行性 (implicit parallelism)，这使得仿生优化算法能以较少的计算获得较大的收益。

6. 都具有突现性

仿生优化算法总目标的完成是在多个智能体个体行为的运动过程中突现出来的。

7. 都具有自组织性和进化性

在不确定的复杂时变环境中，仿生优化算法可通过自学习不断提高算法中个体的适应性。

8. 都具有稳健性

仿生优化算法的稳健性是指在不同条件和环境下算法的适用性和有效性。由于仿生优化算法不依赖于优化问题本身的严格数学性质和所求解问题本身的结构特征，因此用仿生优化算法求解许多不同问题时，只需要设计相应的评价函数（代价函数），而基本上无需修改算法的其他部分。

### 9.3.2 不同点

虽然蚁群算法同目前流行的遗传算法、人工神经网络、微粒群算法、人工免疫算法、人工鱼群算法等都属于仿生优化算法，但是它们在算法机理、实现形式等方面存在许多不同之处，具体如下。

### 9.3.2.1 蚁群算法

- ① 蚁群算法采用了正反馈机制，这是不同于其他仿生优化算法最为显著的一个特点。
- ② 基本蚁群算法一般需要较长的搜索时间<sup>[28, 29]</sup>，且容易出现停滞现象。
- ③ 蚁群算法的收敛性能对初始化参数的设置比较敏感。

### 9.3.2.2 遗传算法

① 遗传算法以决策变量的编码作为运算对象，在优化过程中借鉴了生物学中的染色体和基因等概念，模拟自然界中生物的遗传和进化等机理，应用遗传操作求解无数值概念或很难有数值概念的优化问题<sup>[30]</sup>，这一点恰恰是遗传算法的本质特色。

② 遗传算法是基于个体适应度来进行概率选择操作的，从而使搜索过程表现出较大的灵活性。

③ 遗传算法中的个体重组技术采用了交叉算子，而交叉算子是遗传算法所强调的关键技术，它是遗传算法中产生新个体的主要方法，也是遗传算法区别于其他仿生优化算法的又一个主要不同之处。

### 9.3.2.3 人工神经网络

① 人工神经网络系统是一个高度复杂的非线性动力学系统，不但具有一般非线性系统的共性，更主要的是它还具有高维性和神经元之间的广泛互连性。

② 人工神经网络能广泛地进行知识索引，且对带噪声、不完整或不一致数据具有很强的处理能力，使人工神经网络成为多变量经验建模的有效工具。

③ 未改进的人工神经网络收敛速度较慢，且目前尚无成熟的理论依据确定其隐含层数和隐含层节点数。

### 9.3.2.4 微粒群算法

① 微粒群算法是一种原理相当简单的启发式算法，与其他仿生优化算法相比，它所需的代码和参数较少。

② 微粒群算法受所求问题维数的影响较小。

③ 微粒群算法的数学基础相对较为薄弱，目前还缺乏深刻且具有普遍意义的理论分析。

### 9.3.2.5 人工免疫算法

① 在人工免疫算法中，模拟了人体免疫系统所特有的自适应性和人工免疫这一加强人体免疫系统的手段，并采用了基于浓度的选择更新策略，从而有效地防止了其他仿生优化算法中“早熟”的问题，将搜索过程引向全局最优。

② 人工免疫算法实现的是多样性搜索，其搜索目标具有一定的分散性和独立性。

③ 人工免疫算法一般建立在精确的数学模型或进化计算的基础上，其数学模型固然简单，易于实现，但功能不强，且结果容易失真，其智能度也没有其他几种仿生优化算法高，改进起来较其他仿生优化算法麻烦。

### 9.3.2.6 人工鱼群算法

① 人工鱼群算法具有快速跟踪极值点漂移的能力，而且也具有较强的跳出局部极值点的能力。

② 人工鱼群算法获取的是系统的满意解域，对于精确解的获取，还需对其模型进行适当改进。

③ 当人工鱼个体的数目较少时，人工鱼群算法便不能体现其快速有效的集群性优势。

④ 人工鱼群算法的数学基础比较薄弱，目前还缺乏具有普遍意义的理论分析。

## 9.4 蚁群算法与遗传算法的融合

Abbattista F 等<sup>[31]</sup>最早提出了将遗传算法（GA）和蚁群算法相融合的改进策略，并在 Oliver30TSP 和 Eilon50TSP 的仿真实验中得到了较好的结果；随后，White T<sup>[32]</sup>、Li M<sup>[33]</sup>、Pilat M L<sup>[34]</sup>、Acan A<sup>[35]</sup>、Gong D X<sup>[36]</sup>、孙力娟<sup>[37]</sup>、丁建立<sup>[38]</sup>、邵晓巍<sup>[39]</sup>、陈宏建<sup>[40]</sup>、Kumar G M<sup>[41]</sup>等将蚁群算法与 GA 相融合来解决离散域和连续域中的多种优化问题，并取得了较好的应用效果。

蚁群算法与 GA 相融合是蚁群算法与仿生优化算法相互融合方面研究最早且应用最广的一个尝试，本书前面许多章节中对此已经有所阐述。本节将分别对离散域和连续域中蚁群算法与 GA 的典型融合策略作进一步介绍。

### 9.4.1 离散域蚁群遗传算法

Pilat M L 等<sup>[34]</sup>采用 GA 对蚁群算法中的 3 个参数  $\beta$ 、 $\rho$ 、 $q_0$  进行优化，并在

TSP 中进行了仿真验证。由于文献 [34] 中对  $\alpha$  取默认值,  $\beta$ 、 $\rho$ 、 $q_0$  的取值是相对于  $\alpha$  的一个比值, 所以无法实现对蚁群算法 4 个组合参数  $\alpha$ 、 $\beta$ 、 $\rho$ 、 $q_0$  的全局寻优, 也就很难求得 TSP 的全局最优解。这里研究了一种求解离散域优化问题的蚁群遗传算法 (ant colony algorithm genetic algorithm, ACAGA), 其核心是应用 GA 对蚁群算法的 4 个参数 ( $\alpha$ 、 $\beta$ 、 $\rho$ 、 $q_0$ ) 进行优化<sup>[40]</sup>, 并运用 MMAS<sup>[42]</sup>改进蚁群算法的寻径行为, 以实现对搜索空间高效、快速地全局寻优。仿真实验表明, 改进后的蚁群算法具有很强的全局搜索能力。

求解离散域优化问题的 ACAGA 算法的伪代码如下:

```

For iteration = 0 to Generation do
    对参加进化的每个个体的变量  $\alpha, \beta, \rho, q_0$  进行随机编码;
    从个体中随机地选择 4 个;
    根据给出的 4 个变量的值, 求适应度函数值, 即 4 个个体分别进行 TSP 寻径;
    ① 初始化
         $t=0$ ; //  $t$  是计时器
         $N_c=0$ ; //  $N_c$  是循环计数器
        对所有的边  $(i, j)$  上的信息素赋初值  $\tau_{ij}(t) = \text{const}$ ,  $\Delta\tau_{ij} = 0$ ;
        将 4 只蚂蚁随机地放置在  $n$  个节点上;
    ②  $s=1$ ; // 变量  $s$  是蚂蚁  $k$  在一次寻径过程中走的步数
        For  $k=1$  to 4 do
            将第  $k$  只蚂蚁的初始节点放入数组  $\text{tabu}_k(s)$ ;
             $J_k(s) = \{1, 2, 3, \dots, n\} - \text{tabu}_k(s)$ ; //  $J_k(s)$  是蚂蚁  $k$  在第  $s$  步时尚未经过的节点集合
        ③  $s=s+1$ ;
            For  $k=1$  to 4 do
                根据状态转移概率公式选择蚂蚁  $k$  的下一跳节点  $j$ ; // 蚂蚁  $k$  在  $t$  时刻处在节点
                 $i=\text{tabu}_k(s-1)$ 
                将蚂蚁  $k$  放置于节点  $j$ , 并将节点  $j$  插入数组  $\text{tabu}_k(s)$ ;
                 $\text{tabu}_k(s)=j$ ;  $J_k(s)=J_k(s-1)-j$ ;
                对链路进行局部信息素更新;
                重复③, 直到  $s=n$ 
        ④ For  $k=1$  to 4 do
            将蚂蚁  $k$  从节点  $\text{tabu}_k(n)$  移到  $\text{tabu}_k(1)$ ; // 蚂蚁回到初始节点, 完成一次回路寻径
            计算蚂蚁  $k$  的路由长度  $L_k$ , 并比较其大小;
            求得最短长度  $L_{\text{best}}$  及其对应的  $k_{\text{best}}$  和  $\text{tabu}_{k_{\text{best}}}$ ;
            对链路进行全局信息素更新;
            得到任一条边的信息素浓度值  $\tau_{ij}(t+n)$ ;
    ⑤  $t=t+n$ ;
         $N_c=N_c+1$ ;
        If ( $N_c < N_{c_{\text{max}}}$ ) and 没有停滞行为

```

```

Then {清空所有 tabu 列表;
跳转到②;}
else
    将寻径后的最短路径长度作为适应度函数;
    从被选的 4 个个体中选出 2 个最优个体;
    进行交叉和变异操作生成 2 个子个体;
    替代 4 个个体中最差的 2 个, 放入待进化的个体中;
End for
输出最优结果

```

上述伪代码流程中, 针对参加遗传运算的每只蚂蚁而言, 对蚁群算法中的 4 个参数  $\alpha$ 、 $\beta$ 、 $\rho$ 、 $q_0$  进行了 28bit 编码, 其中每一参数占用 7bit。

### 9.4.2 连续域蚁群遗传算法

改进后的连续域蚁群遗传算法将空间进行均匀分割, 基于这些子空间选取初始种群, 并定义每个子空间的初始信息量, 遗传操作中根据信息量的留存情况来控制个体选择<sup>[39]</sup>。由于初始种群均匀地分散在解空间, 降低了发生过早收敛的可能性; 而采用蚁群算法中“信息量留存”的思想, 可保证算法能够快速收敛到具有最优(次优)解的子空间。

#### 9.4.2.1 算法设计

可将全局优化问题定义如下

$$\begin{aligned} &\text{Maximize } f(x) \\ &\text{s. t. } l \leqslant x \leqslant u \end{aligned} \quad (9.4.1)$$

式中,  $x = \{x_1, x_2, \dots, x_n\}$ , 变量  $x_i$  的定义域为  $[l_i, u_i]$ ;  $l = \{l_1, l_2, \dots, l_n\}$ ;  $u = \{u_1, u_2, \dots, u_n\}$ ;  $f(x)$  表示目标函数,  $n$  为维数。

在求解优化问题之前, 通常不会有全局最优点在解空间位置分布的信息, 因此希望算法的搜索种群能够均匀地分散在解空间, 这有利于算法在后续操作中较均匀地扫描解空间。蚁群遗传算法首先将解空间均匀地分解为若干个子区域, 分别从这些子区域中产生初始种群; 然后采用信息量标定每一个子区域, 在各个子区域信息量变化的约束下进行遗传操作搜索整个解空间; 最后在结束条件控制下停止搜索。

改进后的连续域蚁群遗传算法的主要步骤如下。

(1) 将解空间按一定的原则分解成若干子空间。

针对全局最优问题的维数和每一维定义域的大小对解空间加以分解。这里以

二维优化问题为例，设  $x = \{x_1, x_2\}$ ，其定义域分别为  $[l_1, u_1]$  和  $[l_2, u_2]$ ，定义域均匀地分解为  $M \times N$  个子空间  $E_{ij}$ ，其中  $i=1, 2, \dots, M$ ,  $j=1, 2, \dots, N$ ，且子区域的区间长度为

$$\begin{cases} D_{1L} = \frac{u_1 - l_1}{M} \\ D_{2L} = \frac{u_2 - l_2}{N} \end{cases} \quad (9.4.2)$$

$E_{ij}$  的左右边界分别为  $x_{1iL}, x_{2jL}$  和  $x_{1iR}, x_{2jR}$ ，即有

$$\begin{cases} x_{1iL} = l_1 + (i-1)D_{1L} \\ x_{2jL} = l_2 + (j-1)D_{2L} \\ x_{1iR} = l_1 + iD_{1L} \\ x_{2jR} = l_2 + jD_{2L} \end{cases} \quad (9.4.3)$$

(2) 确定初始种群并标定各个子空间。

① 初始种群的产生：在第（1）步中确定的每个子空间中随机产生一个个体，所有个体组成初始种群。与子空间  $E_{ij}$  相对应的个体定义为  $A_{ij}(1)$ ，括号中的 1 表示是整个算法的第一代个体，则初始种群可表示为  $\{A_{ij}(1)\}$ ，其中  $i=1, 2, \dots, M$ ,  $j=1, 2, \dots, N$ ，种群规模为  $M \times N$ 。

② 子空间的初始标定：每个子空间  $E_{ij}$  都由一个信息量  $Ph_{ij}$  标定，各个子空间信息量的初始值由其中产生的初始个体的适应度值确定。当  $f(A_{ij}(1)) > 0$  时，定义  $Ph_{ij}(1) = C_1 f(A_{ij}(1))$ ，其中  $C_1$  为根据问题而设定的正常数；当  $f(A_{ij}(1)) < 0$  时，定义  $Ph_{ij}(1) = \frac{C_3}{C_2 + f(A_{ij}(1))}$ ， $f(A_{ij})$  为个体  $A_{ij}$  的适应度值。 $C_2$  和  $C_3$  的设定同  $C_1$ 。

(3) 对种群进行蚁群遗传操作。

① 选择操作：选择操作的主要目的是为了避免基因缺失，以提高全局收敛性和计算效率。蚁群遗传算法选择操作的基本思想是：第  $k$  代中的个体  $l$  被选中的概率是其个体适应度值和所处子空间信息量的函数。群体规模为  $M \times N$ ，第  $k$  代中个体  $l$  的适应度值为  $f(A_l(k))$ ，个体  $l$  所处子空间的上一代中标定的信息量为  $Ph_l(k-1)$ ，则个体  $l$  被选中的概率为

$$P_l(k) = \frac{Ph_l^{*}(k-1) f^{\beta}(A_l(k))}{\sum_{i=1}^{M \times N} Ph_i^{*}(k-1) f^{\beta}(A_i(k))} \quad (9.4.4)$$

② 交叉操作：交叉操作是蚁群遗传算法中产生新个体的主要方法，可以针对具体问题，根据编码方法的不同选择各种常用交叉操作和交叉概率。

③ 变异操作：变异操作是产生新个体的辅助方法，同时也决定了蚁群遗传算法的局部搜索能力。与交叉操作相似，可根据具体问题选取具体的变异方法和变异概率。

④ 子空间信息素更新：随着种群一代代进化，各子空间的信息量也不断积累，在积累过程中必须对残留的信息量按照公式（9.4.5）进行更新处理，并根据第（2）步确定第一代中各子空间的信息量。

$$Ph_{ij}(k) = (1 - \rho)Ph_{ij}(k-1) + Ph'_{ij}(k) \quad (9.4.5)$$

式中， $Ph_{ij}(k)$  表示第  $k$  代子空间  $ij$  上的信息量； $Ph_{ij}(k-1)$  表示第  $k-1$  代子空间  $ij$  上的信息量； $Ph'_{ij}(k)$  表示第  $k$  代遗传操作后子空间  $ij$  上加入的信息量。不妨设在子空间  $ij$  上，第  $k$  代以前具有最大适应度值的个体为  $A_{ij\max}$ ，第  $k$  代选择、交叉和变异操作后，具有最大适应度值的个体为  $A_{ij\max}(k)$ 。如果  $f(A_{ij\max}(k)) > f(A_{ij\max})$ ，则  $A_{ij\max} = A_{ij\max}(k)$ ；否则， $A_{ij\max}$  不变。当  $f(A_{ij\max}) > 0$  时，定义

$$Ph'_{ij}(k) = C_1 f(A_{ij\max}) \quad (9.4.6)$$

当  $f(A_{ij\max}) < 0$  时，定义

$$Ph'_{ij}(k) = \frac{C_3}{C_2 - f(A_{ij\max})} \quad (9.4.7)$$

式中， $C_1$ 、 $C_2$ 、 $C_3$  为根据问题而设定的正常数，且  $C_2 > C_3$ ； $f(A_{ij\max})$  为个体  $A_{ij\max}$  的适应度值。如果  $k$  代中某一子空间没有个体，则  $A_{ij\max}$  保持不变。

子空间  $ij$  中具有最大适应度值的个体  $A_{ij\max}$  是随着蚁群遗传操作一代代保留下来的，所以不必每一代都比较  $ij$  中所有个体的适应度值，只需同该代中交叉和变异产生的新个体的适应度值进行比较即可。结合各子空间内的初始信息量值，利用公式（9.4.5）可保证各子空间内的信息量随遗传进化而不断积累和更新。

（4）当群体中的最优个体满足一定要求或总代数达到一定数量时，结束进化操作。

#### 9.4.2.2 仿真算例

这里采用文献 [43] 给出的 Michalewicz 函数和 Schaffer 的  $F_6$  函数，分别采用蚁群遗传算法和简单 GA 对其进行仿真计算。

##### 1. Michalewicz 函数

$$f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2) \quad (9.4.8)$$

自变量范围： $-3.0 \leq x_1 \leq 12.1$ ， $4.1 \leq x_2 \leq 5.8$ 。该函数包含多个局部最优点，因此容易出现过早收敛现象。

这里采用二进制编码形式，要求精确到小数点后 5 位， $M=10$ ， $N=5$ ，故初始种群为 50；采用单点交叉，交叉概率  $P_c=0.4$ ；采用简单变异，变异概率  $P_m=0.1$ ；选择操作中， $\Delta=0.1$ ， $E=2$ ；信息量更新时， $\rho=0.8$ ， $C_1=1$ ；遗传终

止代数 500。

为使仿真具有可比性，简单 GA 也采用二进制编码形式，初始种群取 50；单点交叉， $P_c=0.4$ ；简单变异， $P_m=0.1$ ；终止代数为 500。

## 2. Schaffer 的 $F_6$ 函数

$$f(x, y) = 0.5 - \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{(1 + 0.001(x^2 + y^2))^2} \quad (9.4.9)$$

自变量范围： $-4 < x, y < 4$ 。函数有无数个局部极大点，其中只有一个(0, 0)为全局最大，最大值为 1。此函数的最大值峰周围有一圈脊，其取值均为 0.99028，因此很容易停滞在该局部极大点。

同样采用二进制编码形式，要求精确到小数点后 5 位， $M=8, N=8$ ，故初始种群为 64；采用单点交叉，交叉概率  $P_c=0.4$ ；采用简单变异，变异概率  $P_m=0.08$ ；选择操作中， $\Delta=0.1, E=2$ ；信息量更新时， $\rho=0.9, C_1=1$ ；终止代数为 500。

简单 GA 也采用二进制编码形式，初始种群取 64；单点交叉， $P_c=0.5$ ；简单变异， $P_m=0.06$ ；终止代数为 500。

经过 100 次重复仿真，统计结果如表 9.1 所示。

表 9.1 测试函数仿真结果

算 法	Michalewicz 函数		Schaffer 的 $F_6$ 函数	
	平均收敛代数	过早收敛次数	平均收敛代数	过早收敛次数
蚁群遗传算法	141	7	153	11
简单 GA	207	21	245	34

由表 9.1 可见，与简单 GA 相比，蚁群遗传算法不论是收敛速度还是抗早熟特性均有明显的优势。

## 9.5 蚁群算法与人工神经网络的融合

将蚁群算法和人工神经网络（ANN）相融合，可兼有 ANN 的广泛映射能力和蚁群算法的全局收敛以及启发式学习等特点，从而可在某种程度上避免 ANN 收敛速度慢、易于陷入局部极小点等问题。

Li S H 等<sup>[44]</sup>首先将蚁群算法与 ANN 的融合策略应用于解决异步传输模式（asynchronous transfer mode, ATM）网中的呼叫接纳控制（call admission control, CAC）问题，并取得了很好的仿真效果；Zhang S B 等<sup>[45]</sup>在 Li S H 等研究的基础上，将蚁群算法与 ANN 的融合策略应用于解决 ATM 网中的 CAC 问题

和用法参数控制 (usage parameter control, UPC) 问题; 洪炳熔等<sup>[46]</sup>提出了用蚁群算法学习反向传播 (back propagation, BP) 神经网络的权值, 并将其用于求解非线性模型的辨识问题及倒立摆的控制问题, 随后又提出了用蚁群算法学习 Hopfield 神经网络的权值策略, 并将其应用于 TSP 的求解<sup>[47]</sup>; 邹政达等<sup>[48]</sup>在洪炳熔等研究的基础上, 提出了一种基于蚁群算法的递归神经网络 (recurrent neural network, RNN), 并将其应用于电力系统的短期负荷预测 (short-term load forecasting, STLF)。

### 9.5.1 基于蚁群算法的多层前馈神经网络

ANN 具有复杂的非线性映射能力、函数逼近及大规模并行分布处理能力, ANN 中应用最普遍的是多层前馈网络模型, 其中 BP 神经网络是一种在 ANN 中应用十分广泛的多层前馈神经网络。但由于 BP 神经网络采用的是沿梯度下降算法, 所以, 训练通常需要很长时间才能收敛, 而且不可避免地会出现局部极小的问题。

#### 9.5.1.1 基于蚁群算法的人工神经网络训练

蚁群算法是一种全局优化的启发式算法, 因此用它来训练 ANN 的权值, 可避免 BP 神经网络的缺陷。算法融合的基本思想是: 假定网络中有  $m$  个参数, 它包括所有权值和阈值。首先, 将神经网络参数  $p_i$  ( $1 \leq i \leq m$ ) 设置为  $N$  个随机非零值, 形成集合  $I_{p_i}$ 。蚁群中的每只蚂蚁在集合  $I_{p_i}$  中选择一个权值, 在全部集合中选择一组神经网络权值。蚂蚁的数目为  $h$ ,  $\tau_j(I_{p_i})$  表示集合  $I_{p_i}$  ( $1 \leq i \leq m$ ) 中第  $j$  个元素  $p_j(I_{p_i})$  的信息量。蚂蚁搜索时, 不同的蚂蚁选择元素是相互独立的。每只蚂蚁从集合  $I_{p_i}$  出发, 根据集合中每个元素的信息量和状态转移概率从每个集合  $I_{p_i}$  中选择一个元素。当蚂蚁在所有集合中完成选择元素后, 它就到达食物源, 然后调节集合中元素的信息量。这一过程反复进行, 直到进化趋势不明显或达到给定的迭代次数时为止。

改进后蚁群算法的具体步骤如下:

(1) 初始化: 令时间  $t$  和循环次数  $N_c$  为零, 设置最大循环次数  $N_{c_{\max}}$ , 令每个集合中的每个元素的信息量  $\tau_j(I_{p_i}) = C$ , 且  $\Delta\tau_j(I_{p_i}) = 0$ , 将全部蚂蚁置于蚁巢。

(2) 启动所有蚂蚁, 针对集合  $I_{p_i}$ , 蚂蚁  $k$  ( $k = 1, 2, \dots, h$ ) 根据下式计算状态转移概率

$$\Pr(\tau_j^k(I_{p_i})) = \frac{\tau_j^k(I_{p_i})}{\sum_{g=1}^N \tau_g(I_{p_i})} \quad (9.5.1)$$

(3) 重复第(2)步, 直到蚁群全部到达食物源。

(4) 令  $t \leftarrow t+m$ ;  $N_c \leftarrow N_c+1$ ; 利用各蚂蚁选择的权值计算神经网络的输出值和误差, 记录当前最优解。经过  $m$  个时间单位, 蚂蚁从蚁巢到达食物源, 各路径上的信息量则根据下式更新

$$\tau_j(I_{p_i})(t+m) = (1-\rho)\tau_j(I_{p_i})(t) + \Delta\tau_j(I_{p_i}) \quad (9.5.2)$$

$$\Delta\tau_j(I_{p_i}) = \sum_{k=1}^h \Delta\tau_j^k(I_{p_i}) \quad (9.5.3)$$

$$\Delta\tau_j^k(I_{p_i}) = \begin{cases} \frac{Q}{e^k}, & \text{若第 } k \text{ 只蚂蚁在本次循环中选择元素 } p_j(I_{p_i}) \\ 0, & \text{否则} \end{cases} \quad (9.5.4)$$

式中,  $e^k$  将第  $k$  个蚂蚁选择的一组权值作为神经网络权值的输出误差, 其定义如下

$$e^k = |O - O_q| \quad (9.5.5)$$

式中,  $O$  和  $O_q$  分别表示神经网络的实际输出和期望输出。可见, 误差  $e^k$  越小, 相应信息量的增加就越多。

(5) 如果蚁群全部收敛到一条路径或循环次数  $N_c \geq N_{c_{max}}$ , 则循环结束, 并输出计算结果; 否则跳转到第(2)步。

### 9.5.1.2 仿真算例

为了验证利用蚁群算法训练 ANN 的可行性, 这里分别以非线性系统辨识问题和倒立摆控制问题为例进行了仿真实验。

#### 1. 非线性系统辨识

这里通过用蚁群算法训练三层 BP 神经网络来逼近非线性函数  $e^{-x} \sin(2\pi x)$ 。输入样本区间取为  $[0, 1]$ , 步长为 0.1, 则共有 11 个输入输出样本对。BP 神经网络结构中输入、输出层均为 1 个节点, 隐含层为 4 个节点, 隐含层和输出层的激发函数均为非线性 Sigmoid 函数。该网络的连接权数目  $m=13$ 。蚁群算法参数选为  $\rho=0.7$ ,  $h=20$ ,  $Q=10$ ; BP 神经网络参数  $p_i$  为  $-2 \sim 2$  之间的随机数,  $N$  取 20。蚁群经过 130 次学习, 得到比较理想的结果, 其训练结果如图 9.1 所示。

由图 9.1 可见, BP 神经网络的实际输出与样本输出基本吻合。

#### 2. 倒立摆控制

倒立摆是一种非线性、不稳定的典型被控对象, 本例由 BP 神经网络控制器

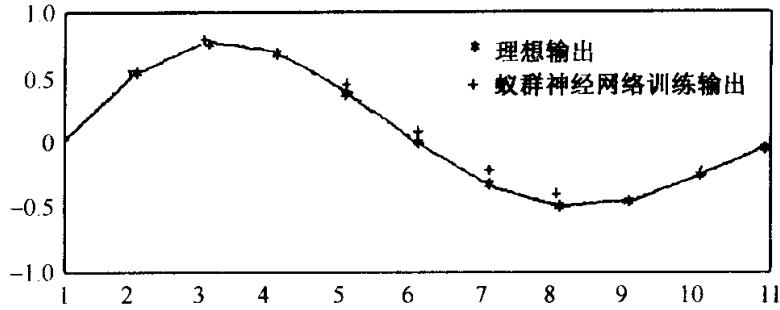


图 9.1 蚁群算法对 BP 神经网络的训练结果

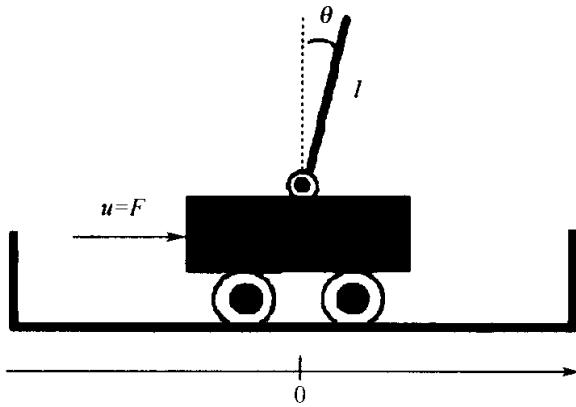


图 9.2 倒立摆系统

实现对倒立摆的控制，用蚁群算法来学习 BP 神经网络的权系。如图 9.2 所示，有一个带轮的小车，其顶端铰链系一刚性倒立摆，小车可沿一笔直的有界轨道向左或向右运动，同时摆可在垂直平面内自由运动。在任意时刻，该系统的状态由 4 个变量描述：小车位置  $X$ ，小车平移速度  $\dot{X}$ ，摆偏离垂直方向的角度  $\theta$  以及摆的角速度为  $\ddot{\theta}$ 。由此，可得倒立摆的运动方程为

$$\ddot{\theta} = \frac{(M+m)g\sin\theta - \cos\theta(F + ml\dot{\theta}^2)\sin\theta}{\frac{4}{3} \times (M+m)l - ml\cos^2\theta} \quad (9.5.6)$$

$$\ddot{X} = \frac{F + ml[\dot{\theta}\sin\theta - \ddot{\theta}\cos\theta]}{M+m} \quad (9.5.7)$$

式中，小车的质量  $M=0.5\text{kg}$ ，倒立摆的质量  $m=0.2\text{kg}$ ，摆长  $l=0.6\text{m}$ ，重力加速度  $g=9.8\text{m/s}^2$ ； $F$  为 BP 神经网络控制器输出的控制力（忽略小车对地的摩擦及倒立摆对小车的摩擦）。

BP 神经网络控制器输入层为 4 个节点，输出层为 1 个节点，隐含层为 5 个节点。输入层 4 个节点分别接受倒立摆系统的 4 个状态值，隐含层的激发函数取为非对称 Sigmoid 函数，输出层的激发函数为线性函数。该网络的连接权数目

$m=31$ , BP 神经网络参数  $p_i$  为  $-500 \sim 500$  之间的随机数,  $N$  取 60。蚁群算法参数取为  $\rho=0.65$ ,  $h=60$ ,  $Q=15$ 。

首先用蚁群算法离线学习 BP 神经网络控制器, 经过 600 次的迭代, 得到了较理想的控制器, 然后用这个控制器对倒立摆系统进行控制。倒立摆初始状态给定后, 由 BP 神经网络控制器给小车底座施加一个向左或向右的力  $F$ , 在尽可能长的时间 (如 100s) 内, 将小车保持在预先定义的轨迹界限内, 且不让摆倒下。

倒立摆系统的初值为  $[X, \theta, \dot{X}, \dot{\theta}] = [-1, -0.2, 0, 0]$ , 采样周期取  $T=0.02\text{s}$ 。仿真结果如图 9.3、图 9.4 所示。

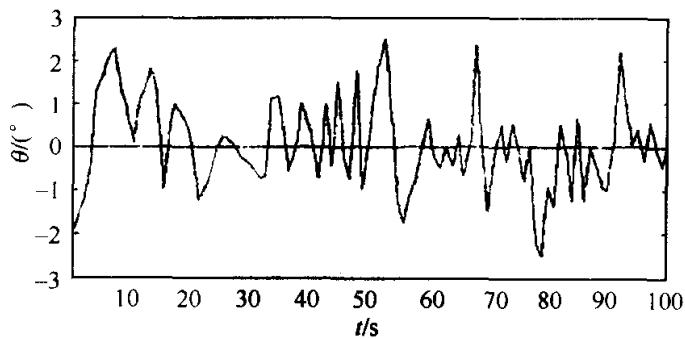


图 9.3 倒立摆摆角的变化曲线

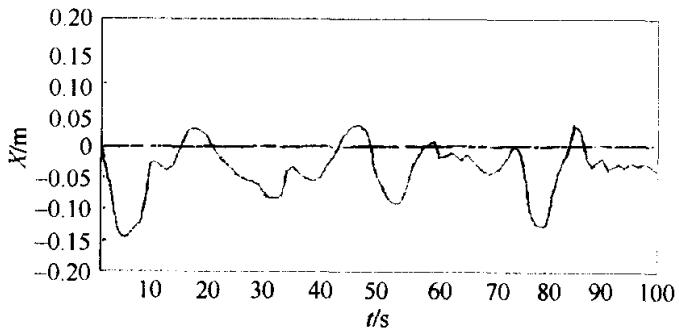


图 9.4 倒立摆位置的变化曲线

由图 9.3 和图 9.4 可见, 倒立摆能较好地控制在摆动范围不大于  $3^\circ$  的状态, 小车位置偏离中心点不超过  $0.2\text{m}$ , 满足控制要求。

表 9.2 是蚁群神经网络算法与 BP 神经网络的比较, 表中数据是在相同运行环境下, 蚁群神经网络程序和 BP 神经网络程序分别运行 10 次的平均结果。

由表 9.2 所示的实验结果可见, 蚁群神经网络的学习次数远小于 BP 神经网络, 从而验证了蚁群神经网络融合策略的有效性。

表 9.2 BP 神经网络和蚁群神经网络算法的比较

控制问题	算法	学习周期(迭代次数)	运行时间(s)
非线性辨识	BP 神经网络	1893	83
	蚁群神经网络	128	53
倒立摆	BP 神经网络	7298	583
	蚁群神经网络	578	372

## 9.5.2 基于蚁群算法的 RNN

邹政达等<sup>[48]</sup>在洪炳熔等研究的基础上, 提出了一种基于蚁群算法的 RNN, 并将其应用于电力系统的 STLF。不同于本章第 9.5.1 节, 基于蚁群算法的 RNN (ACA-RNN) 将第  $k$  个蚂蚁的输出误差  $e^k$  定义为训练样本集后的最大输出误差, 即

$$e^k = \max_{p=1}^s |O - O_p| \quad (9.5.8)$$

### 9.5.2.1 基于蚁群算法的 RNN 模型构造

考虑到 STLF 模型中预测点的动态性, 这里采用单步预测输出, 并反馈用作下一步预测输出的输入元素。输入矢量维数  $m=7$ , 即以预测时刻前 7 个点的负荷作为预测模型的输入维。根据训练样本集, 用穷举法确定 ACA-RNN 模型隐含层的节点数为 8, 预测神经网络模型的结构为 7-8-1, 其输出表达式为

$$y(t + \tau) = F[x(t), x(t - \tau), x(t - 2\tau), \dots, x(t - (m - 1)\tau)] \quad (9.5.9)$$

采用 RNN 的结构如图 9.5 所示。这里主要研究蚁群算法对预测精度性能的影响, 仅用蚁群算法形成优化 RNN。训练样本是按预测日前 14 天的历史负荷数据形成对应于 24 个预测点的固定训练样本集。

在 ACA-RNN 训练和仿真测试中, 发现学习样本过多会导致网络的学习过度, 使预测精度下降, 这里取实际负荷系统的训练样本集数为 10~15 个(天)时, 预测效果较好。在对预测日进行负荷预测时, 各预测点的测试样本按公式 (9.5.9) 的预测点步进动态移动获得。

### 9.5.2.2 仿真算例

电力系统 STLF 的准确性将直接影响电力市场运营的有效性, 因此预测的准确性对电力系统的运行和经济性都具有重要意义。

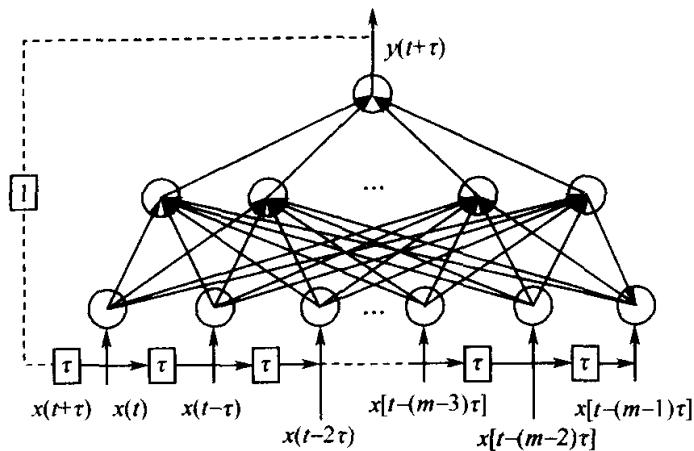


图 9.5 RNN 预测模型结构

### 1. 预测模型基本数据

这里通过 ACA-RNN 与改进的 BP-RNN 这两个预测模型对某实际地区电网负荷系统中的日负荷预测仿真进行了测试。为突出两个模型的算法对预测性能的影响，模型均采用 7-8-1 结构的 RNN 模型。

### 2. 两种模型日负荷预测精度及其性能分析

用某实际地区电网 2001 年 3 月 30 日至 2001 年 4 月 12 日数据对两个模型分别进行训练。对 2001 年 4 月 13 日进行小时/日（24 点）负荷预测，其预测误差曲线如图 9.6 所示。

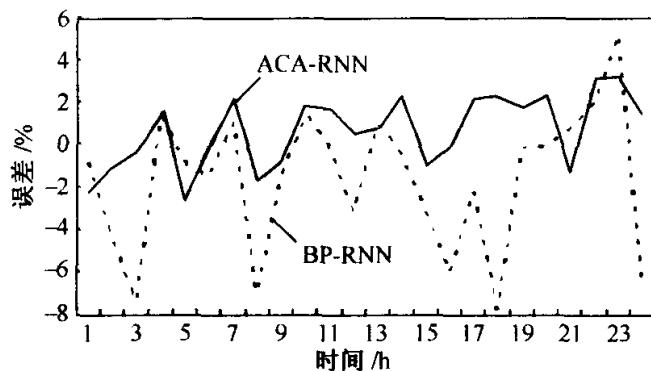


图 9.6 实际地区网日负荷的预测误差及其比较

由图 9.6 所示曲线数据可知，ACA-RNN 和改进 BP-RNN 的最大相对误差分别为 3.173% 和 -7.782%；日绝对平均误差分别为 1.60% 和 2.93%。可见，ACA-RNN 模型的预测性能明显优于 BP-RNN。

### 3. ACA-RNN 模型的预测稳定性和适应性分析

仅根据对某一天的预测结果不足以对 ACA-RNN 模型做出评价，通过两种模型的非高温周的预测性能来对 ACA-RNN 模型的预测精度、稳定性和适应性能力进行评价，即可分析模型对工作日和双休日的适应能力。该实际地区电网非高温周的仿真测试结果如表 9.3 所示。

表 9.3 实际地区电网非高温周的预测误差

星期	日最高~最低温度(℃)	ACA-RNN		BP-RNN	
		平均误差(%)	最大相对误差(%)	平均误差(%)	最大相对误差(%)
一	15~7	1.64	3.12	3.19	7.35
二	12~4	1.73	3.17	3.28	7.95
三	16~5	1.60	3.17	2.74	7.78
四	21~7	1.65	3.51	3.29	7.54
五	17~6	1.60	3.37	2.93	7.89
六	24~10	1.68	3.54	3.78	8.19
日	28~10	1.67	3.66	3.51	8.45
一周平均		1.65	3.36	3.19	7.88

由表 9.3 可见，ACA-RNN 的预测误差比 BP-RNN 小一半以上，而且表现出很好的预测稳定性和预测适应性，即不管是工作日还是双休日都具有适应能力。

用该实际地区 2001 年 6 月 25 日到 2001 年 7 月 8 日的数据进行训练，连续进行高温周的预测测试数据如表 9.4 所示，两个误差性能指标要比表 9.3 的数据都增加几乎 1 倍左右。表 9.3 中的日最高温度的变化为 15~28℃，相差 13 度；而表 9.4 中的最高温度变化为 32~38℃，仅相差 6 度，说明预测模型不能适应外部高温引起的负荷敏感性变化。

表 9.4 实际地区电网高温周的预测误差

星期	日最高~最低温度 (℃)	ACA-RNN	
		平均误差 (%)	最大相对误差 (%)
一	32~24	3.42	6.41
二	32~24	3.74	6.10
三	35~24	3.83	6.37
四	38~26	3.66	6.58
五	36~25	3.76	6.68
六	33~25	3.64	6.68
日	33~24	3.25	6.55
一周平均		3.64	6.44

在实际负荷预测系统中的测试结果表明，基于 ACA-RNN 的 STLF 模型可在一定程度上有效地提高预测精度，其预测性能明显优于 BP-RNN。

## 9.6 蚁群算法与微粒群算法的融合

侯云鹤等<sup>[49]</sup>将微粒群（PSO）算法引入一类用于经济负荷分配（economic dispatch, ED）的广义蚁群算法（GACA）的局部搜索中，采用 GACA 进行全局搜索，确定最优解存在的邻域。通过 PSO 算法实现局部搜索，由于 PSO 算法充分利用以往的信息，又不依赖于梯度，可实现非凸空间的高效搜索，在实际计算中比 Monte Carlo 仿真<sup>[50]</sup>具有更高的精度和更快的速度。GACA 与 PSO 算法相融合可保证在算法全局收敛的前提下，提高了优化的效率和计算精度。

### 9.6.1 GACA-PSO 融合算法

GACA 的基本原理在第 3 章的第 3.10.1 节中已做介绍，这里不再赘述，本节重点介绍 GACA 与 PSO 算法的融合策略，具体如下：

采用 PSO 算法对 GACA 步骤的第（2）～（4）步进行邻域搜索，假设已搜索到  $X$  点，其邻域为  $S_X$ ，PSO 算法迭代次数上限为  $T_p$ 。其优化步骤如下。

(1) 初始化。

当前迭代次数  $K=0$ ，在  $S_X$  选取  $N_p-1$  组服从均匀分布的  $n$  维随机矢量，与  $X$  共同确定  $N_p$  个粒子初始位置，形成初始微粒群： $[X_1(0), X_2(0), \dots, X_N(0)]^T$ ，其中  $X_i(0) \in S_X$ ，且  $X_i(0) = [x_{i1}(0), x_{i2}(0), \dots, x_{in}(0)]^T$ 。

在  $R^n$  上的闭矩阵  $[0, v_{1,\max}] \times [0, v_{2,\max}] \times \dots \times [0, v_{n,\max}]$  ( $v_{j,\max}$  为第  $j$  维速度的上限) 中选取  $N_p$  组服从均匀分布的  $n$  维随机矢量，第  $i$  个粒子的初始速度为

$$v_i(0) = [v_{i1}(0), v_{i2}(0), \dots, v_{in}(0)]^T \quad (9.6.1)$$

(2) 搜索策略。

假设  $K$  次迭代后搜索到的最优解为

$$X_g(K) = [x_{g1}(K), x_{g2}(K), \dots, x_{gn}(K)]^T \quad (9.6.2)$$

粒子  $i$  经历过的最优解为

$$X_{pi}(K) = [x_{p1i}(K), x_{p2i}(K), \dots, x_{pin}(K)]^T \quad (9.6.3)$$

每个粒子在每个方向上的运行速度为

$$\begin{aligned} \bar{v}_{ij}(K+1) &= \omega v_{ij}(K) + \lambda_1[x_{gi}(K) - x_{ij}(K)] + \lambda_2[x_{pij}(K) - x_{ij}(K)] \\ &\quad (9.6.4) \end{aligned}$$

$$v_{ij}(K+1) = \text{sgn}[\bar{v}_{ij}(K+1)] \cdot \min[|\bar{v}_{ij}(K+1)|, v_{i,\max}] \quad (9.6.5)$$

式中,  $i=1, 2, \dots, N$ ,  $j=1, 2, \dots, n$ ,  $\lambda_1, \lambda_2$  分别表示在  $[0, \lambda_{1\max}]$  和  $[0, \lambda_{2\max}]$  上均匀分布的随机变量;  $\omega$  表示正参数;  $\text{sgn}()$  表示符号函数, 其定义如下

$$\text{sgn}(x) = \begin{cases} 1, & \text{若 } x > 0 \\ -1, & \text{若 } x < 0 \\ 0, & \text{若 } x = 0 \end{cases} \quad (9.6.6)$$

更新每个粒子在每个方向上的坐标  $x_{ij}(K+1)$ , 先计算

$$\bar{x}_{ij}(K+1) = x_{ij}(K) + v_{ij}(K+1), \quad i = 1, 2, \dots, N_p; j = 1, 2, \dots, n \quad (9.6.7)$$

形成矢量  $\bar{X}_i(K+1) = [x_{i0}(K+1), x_{i1}(K+1), \dots, x_{in}(K+1)]^T$ , 若  $\bar{X}_i(K+1) \in S_x$ , 则  $X_i(K+1) = \bar{X}_i(K+1)$ ; 若  $\bar{X}_i(K+1) \notin S_x$ , 则  $X_i(K+1) = \{\bar{X}_i(K+1) | X_i(K)\} \cap \Omega_{S_x}$ , 其中,  $\{\bar{X}_i(K+1) | X_i(K)\}$  为  $\bar{X}_i(K+1)$  与  $X_i(K)$  之间线段上的点集,  $\Omega_{S_x}$  表示局部搜索域的边界。

计算每个粒子对应的目标函数  $F(X_i(K+1))$ , 与 PSO 算法中最优解及个体最优解比较, 并进行相应的更新。

(3)  $K=K+1$ 。

若  $K < T_p$ , 则跳转到第 (2) 步; 若  $K \geq T_p$ , 比较本次迭代中的最优值  $F[X_{\text{Best}}(T_p)]$  与已搜索到的最优值  $F[X_g(T_p)]$ : 如果  $F[X_{\text{Best}}(T_p)] < F[X_g(T_p)]$ , 则输出  $X_{\text{Best}}(T_p)$ ; 否则输出  $X_g(T_p)$ 。

## 9.6.2 算例分析

ED 是一类电力系统规划和运行调度中的典型优化问题。由于火电机组阀点效应、系统运行约束、系统稳定性等条件的制约, 使 ED 问题具有非凸、高维数、非线性和不可导等特性。

### 9.6.2.1 算例 A

以文献 [50]、[51]、[52] 的 3 机 6 母线电力系统为例, 原始数据参见文献 [51], 考虑耗量曲线的阀点效应和系统网损, 采用 B 系数法计算网损。发电机承担的总负荷为  $P_D = 500\text{MW}$ 。GACA 的参数与文献 [50] 一致, PSO 算法中的粒子个数  $N_p = 10$ , 迭代次数  $T_p = 15$ ,  $\lambda_{1\max} = \lambda_{2\max} = 2.05$ ,  $\omega = 1.05$ ; 粒子速度在每个方向上的上限值  $v_{i,\max} = 0.8 \times$  局部搜索邻域宽度, 其优化结果如表 9.5 所示。

文献 [52] 采用遗传算法得出的结果为:  $P_{G1} = 300.01\text{MW}$ ,  $P_{G2} = 170.20\text{MW}$ ,  $P_{G3} = 100.10\text{MW}$ , 网损为  $70.99\text{MW}$ , 总费用为  $5745.11\text{USD}$ ; 采

表 9.5 算例 A 的结果

可行解	$P_{G1}$ (MW)	$P_{G2}$ (MW)	$P_{G3}$ (MW)	$\sum P_{Gi}$ (MW)	网损 (MW)	总费用 (USD)
1	199.73	174.80	177.74	552.27	52.32	5740.63
2	297.36	173.88	99.79	571.03	70.98	5737.30
3	299.34	172.04	99.80	571.18	71.20	5735.85
最优解	299.46	171.89	99.87	571.22	71.22	5735.78

用混沌优化得出结果为： $P_{G1}=299.46\text{MW}$ ， $P_{G2}=172.00\text{MW}$ ， $P_{G3}=98.84\text{MW}$ ，网损为 70.24MW，总费用为 5735.93USD。文献 [50] 采用 GACA 计算的最优结果为 5735.74USD。

对算例 A 独立计算 20 次，目标函数值小于 5742.0USD 时算法收敛。采用 GACA-PSO 融合算法，收敛迭代次数平均值为 233.75，样本标准差为 63.978；文献 [52] 对与本节相同的算例，采用遗传算法计算目标函数 5000 次，混沌优化计算目标函数 3000 次；文献 [50] 采用 GACA，收敛迭代次数平均值为 261.75，样本标准差为 59.725。采用不同算法，各独立计算 20 次，其平均收敛曲线如图 9.7 所示。

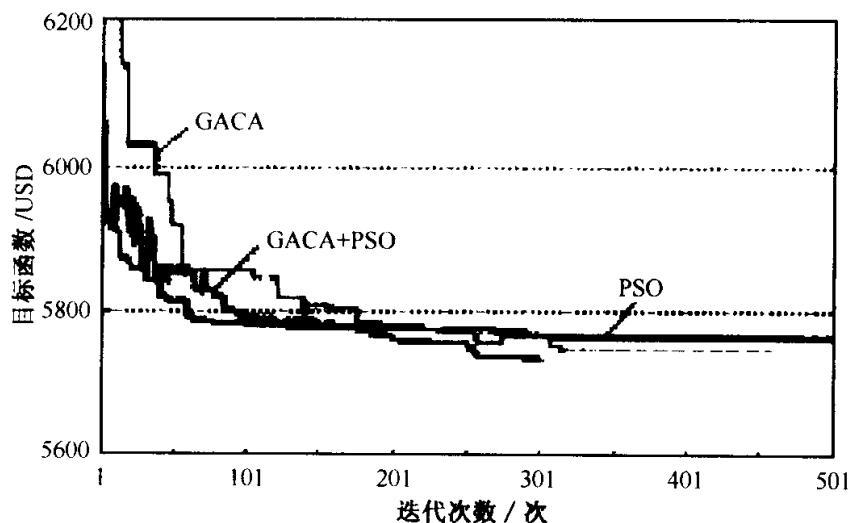


图 9.7 算例 A 中不同算法平均收敛曲线比较

由图 9.7 可见，GACA-PSO 融合算法的计算量相对较小。

### 9.6.2.2 算例 B

以文献 [53] 的 13 机电力系统为例，考虑耗量特性的阀点效应，同时忽略

网损，原始数据见文献 [53]。算法参数与算例 A 一致，其计算结果如表 9.6 所示。

表 9.6 算例 B 的结果

	次优解 1	次优解 2	次优解 3	最优解
$P_{G1}$ (MW)	538.61	628.36	628.28	628.24
$P_{G2}$ (MW)	301.52	224.13	223.73	298.91
$P_{G3}$ (MW)	360.00	297.87	299.16	223.40
$P_{G4}$ (MW)	60.00	60.00	60.00	60.00
$P_{G5}$ (MW)	60.00	60.00	60.00	60.00
$P_{G6}$ (MW)	109.86	60.00	60.00	60.00
$P_{G7}$ (MW)	60.00	60.00	158.82	60.00
$P_{G8}$ (MW)	60.00	159.64	60.00	60.00
$P_{G9}$ (MW)	60.00	60.00	60.00	159.44
$P_{G10}$ (MW)	40.00	40.00	40.00	40.00
$P_{G11}$ (MW)	40.00	40.00	40.00	40.00
$P_{G12}$ (MW)	55.00	55.00	55.00	55.00
$P_{G13}$ (MW)	55.00	55.00	55.00	55.00
总费用(USD)	18096.90	17989.77	17989.61	17989.37

分别采用 PSO 算法、GACA-PSO 融合算法各独立计算 100 次，迭代次数上限取 300，计算结果如表 9.7 所示。文献 [53] 采用多种改进形式的进化规划算法求解，得到的最优结果在表 9.7 中记为 IFEP（独立计算 50 次）。

表 9.7 算例 A 最优结果分布比较

目标函数值(USD)	最优结果分布(%)		
	IFEP <sup>[53]</sup>	PSO	GACA-PSO
>18200	38	39	21
18150~18100	26	22	18
18100~18050	22	18	27
18050~18000	12	19	19
<18000	2	2	15

由表 9.7 可见，GACA-PSO 融合算法与 PSO 算法及改进进化规划算法相比，其收敛精度更高，解的离散度更小。

## 9.7 蚁群算法与人工免疫算法的融合

人工免疫算法（AIA）具有快速随机的全局搜索能力，但对于系统中的反馈信息利用不足，当求解到一定范围时往往做大量无为的冗余迭代，求解效率低；而蚁群算法具有分布式并行全局搜索能力，通过信息量的积累和更新收敛于最优路径上，但其初期信息素匮乏，求解速度较慢。

Lee Z J 等<sup>[54]</sup>借鉴人体免疫系统的适应能力和蚁群算法的全局寻优能力，提出了一种新的融合算法即免疫-蚁群算法，并将其成功应用于求解 WTAP；Feng Y J 等<sup>[55]</sup>将基于人工免疫的蚁群算法用于求解连续空间的多模型函数优化问题；胡纯德等<sup>[56]</sup>提出了一种利用 AIA 生成信息素分布的人工免疫-蚁群混合算法；蒋加伏等<sup>[57]</sup>提出了一种基于免疫-蚁群算法的多约束 QoS 路由选择算法，也取得了很好的应用效果。

### 9.7.1 基于 AIA 和蚁群算法的混合算法

#### 9.7.1.1 算法的设计思想

AIA 和蚁群算法相融合的基本思想是：算法的前过程充分利用 AIA 的快速性和全局收敛性，寻找较优的可行解，而算法的后过程采用蚁群算法，即利用前过程中 AIA 获得的较优可行解，产生初始信息素分布，然后充分利用蚁群算法的正反馈性，以提高求解效率。

#### 9.7.1.2 人工免疫算子的构造

人工免疫算子有字符换位算子、字符串移位算子、字符串逆转算子和优质字符串的保留等几种。对于 TSP，AIA 中人工免疫算子的构造过程如下。

(1) 字符换位算子，可分为单对字符换位算子和多对字符换位算子。

单对字符换位操作是随机取两个正整数  $i, j(1 < i, j \leq n, i \neq j)$ ，以一定的概率  $p_c$  交换抗体  $A = (c_1, c_2, \dots, c_n)$  中的一对字符  $c_i, c_j$  的位置；多对字符换位操作是预先确定一个正整数  $u_c$ ，在抗体  $A = (c_1, c_2, \dots, c_n)$  中随机取  $r(1 < r \leq u_c)$  对字符作字符串换位操作。

(2) 字符串移位算子，可分为单个字符串移位算子和多个字符串移位算子。

单个字符串移位操作是随机取两个正整数  $i, j(1 < i, j \leq n, i \neq j)$ ，在抗体  $A = (c_1, c_2, \dots, c_n)$  中取一个字符串子串  $A_1 = (c_i, c_{i+1}, \dots, c_{j-1}, c_j)$ ，以一定的概率  $p_s$  依次向右移动  $A_1$  中的各个字符，最右边的一个字符则移动到最左边的位置；

多个字符串移位操作是预先确定一个正整数  $u_i$ ，在抗体  $A = (c_1, c_2, \dots, c_n)$  中随机取  $r(1 < r \leq u_i)$  个字符子串作字符串移位操作。

(3) 字符串逆转算子，可分为单个字符串逆转算子和多个字符串逆转算子。

单个字符串逆转操作是随机取两个正整数  $i, j(1 < i, j \leq n, i \neq j)$ ，在抗体  $A = (c_1, c_2, \dots, c_n)$  中取一个字符子串  $A_1 = (c_i, c_{i+1}, \dots, c_{j-1}, c_j)$ ，以一定的概率  $p_i$  使  $A_1$  中的各个字符首尾倒置；多个字符串逆转操作是预先确定一个正整数  $u_i$ ，在抗体  $A = (c_1, c_2, \dots, c_n)$  中随机取  $r(1 < r \leq u_i)$  个字符子串作字符串逆转操作。

(4) 优质字符串的保留。

如果若干个抗体与抗原之间的亲和力都很大，且这些抗体中包含了一个相同的字符子串，则称该字符子串为优质字符串。如果抗体中存在优质字符串，则在抗体产生过程中以概率  $p_o$  使该优质字符串不受破坏。

### 9.7.1.3 算法的基本步骤

根据上述设计思想，其基本步骤可描述如下。

(1) 输入问题和确定抗体的编码表示。

一般输入问题的目标函数和约束条件作为 AIA 的抗原。AIA 的抗体采用自然数编码方式，一个字符串代表一个候选解。对于  $n$  个城市的 TSP，设其城市编码分别为  $1, 2, \dots, n$ ，并把商人出发的城市编为第 1 号，其他城市可随意编号。把这  $n$  个城市的编号任意排列成一个长度为  $n$  的字符串都可以形成一个抗体，因此抗体空间包含  $n!$  个抗体。为了缩小抗体空间，提高搜索效率，将每个人工抗体（字符串）的第一个字符固定为出发城市的编号 1。这样，每个抗体只有  $n-1$  个字符可任意排列，抗体空间就只包含  $(n-1)!$  个抗体。

(2) 产生初始抗体并进行预处理。

在一般情况下，可按上述抗体编码方式，在解空间中随机产生  $N$  个抗体作为初始抗体，构成初始抗体群，其中  $N$  为抗体群中抗体的数目。考虑到 TSP 的任何一条路径都是闭合路径，从任一城市出发，要到达的下一城市选择为未到过的城市中距该城市最近的一个。为了提高搜索效率，先对每一个初始抗体进行预处理，然后才开始算法的迭代计算。设初始抗体  $A = (c_1, c_2, \dots, c_n)$ ，其中  $c_1 = 1$ ，对  $A$  进行预处理的具体操作如下：

① 随机取正整数  $r(1 \leq r \leq n)$ 。若  $r = c_1 = 1$ ，则令  $A' = (c'_1, c'_2, \dots, c'_n)$ ，跳转到③；若  $r = c_k \neq 1$ ，则跳转到②。

② 对初始抗体  $A$  中的各个字符依次循环左移位  $k-1$  次，每次移位时，使  $c_{i+1}$  移到  $c_i(i=1, 2, \dots, i-1)$  的位置，且使  $c_k$  移到  $c_1$  的位置。设移位以后初始抗体  $A$  变为  $A'$ ，则  $A' = (c_k, c_{k+1}, \dots, c_n, c_1, c_2, \dots, c_{k-1})$ 。令  $A' = (c'_1,$

$c'_1, \dots, c'_n$ ), 其中  $c'_1 = c_k$ , 其余依此类推。

③ 随机取  $c'_m = c'_1$ , 令  $C = (c'_2, c'_3, \dots, c'_n)$ , 若对  $C$  中任一个元素  $c'_k$ , 都有  $d(c'_m, c'_k) \leq d(c'_m, c'_l)$ ,  $c'_l \in C$ , 则把  $c'_l$  置于  $A'$  中  $c'_{m+1}$  的位置, 此时  $A'$  变为  $A''$ , 令  $A'' = (c''_1, c''_2, \dots, c''_n)$ 。然后从  $C$  中删除元素  $c'_l$ , 再取  $c''_m = c''_1$ , 重复上述步骤, 直到  $C$  中的元素全部被删除为止。设这一步完成以后, 初始抗体  $A$  变为  $B$ 。

④ 对初始抗体  $B$  中的各个字符依次循环右移位若干次, 直到抗体中第一个字符为 1 为止, 移位方法与第②步中的类似, 但方向相反。

(3) 计算亲和力和排斥力。

对于 TSP, 可定义抗体  $B$  与抗原  $G$  之间的  $\text{affinityApp}(B) = 1/(T_B - T_G)$ , 其中  $T_B$ ,  $T_G$  分别为抗体  $B$  与抗原  $G$  对应的旅行路线的总长度,  $T_G$  也是所求的最短路线的总长度。因在计算结束之前并不知道  $T_G$  的大小, 可用一适当大的正数  $T$  ( $T < T_G$ ) 代替  $T_G$ 。又因计算  $\text{App}(B)$  需作除法运算, 可定义  $\text{App}(B) = T_M - T_B$ 。其中  $T_M$  为较大的正数, 且要求  $T_M$  大于任意抗体对应的旅行路线的总长度, 则可避免在计算  $\text{App}(B)$  时作除法运算。因此, 构造抗体  $B$  与抗原  $G$  之间的  $\text{affinityApp}(B) = T_M - T_B$ , 并计算  $\text{App}(B)$ 。抗体与抗原之间的亲和力反映抗体与抗原之间的匹配程度,  $\text{App}(B)$  越大, 说明抗体  $B$  与抗原  $G$  之间的匹配越好。

对于 TSP, 可定义抗体  $B_1$  与抗体  $B_2$  之间的排斥力  $\text{Rep}(B_1, B_2) = |T_{B1} - T_{B2}|$ , 并计算  $\text{Rep}(B_1, B_2)$ , 其中  $T_{B1}$ ,  $T_{B2}$  分别表示抗体  $B_1$  与抗体  $B_2$  所对应旅行路线的总长度。抗体与抗体之间的排斥力反映抗体与抗体之间的差距,  $\text{Rep}(B_1, B_2)$  越大, 说明抗体  $B_1$  与抗体  $B_2$  之间的差距越大。计算抗体群中所有抗体与当前最佳抗体之间的排斥力。

(4) 产生新抗体并计算其亲和力和排斥力。

构造适当的人工免疫算子, 通过人工免疫算子的作用概率  $p_c$ ,  $p_s$ ,  $p_i$ ,  $p_o$  和预先确定的正整数  $u_c$ ,  $u_s$ ,  $u_i$  产生新抗体; 同时, 计算各个新抗体的  $\text{affinityApp}(B)$  和新抗体间的排斥力  $\text{Rep}(B_1, B_2)$ 。若新抗体中有与抗原相匹配的抗体, 从而获得较优的可行解; 否则跳转到第(5)步。

(5) 抗体选择。

按照“优胜劣汰”的自然选择机制, 在新产生的若干个抗体中, 选出与抗原匹配得较好的抗体构成新的抗体群, 转第(4)步。

(6) 初始化参数  $\tau_C$ ,  $\tau_G$ ,  $m$ ,  $\alpha$ ,  $\beta$ ,  $\rho$ ,  $Q$ , 根据第(4)步获得的较优可行解, 生成信息素初始分布, 将  $m$  只蚂蚁置于  $n$  个节点。

这里,  $\tau_C$  表示根据问题规模而给定的信息量常值,  $\tau_G$  表示 AIA 求解结果转换的信息量。在初始时刻, 根据下式设置信息量初值

$$\tau_{ij}(0) = \begin{cases} \tau_C + \tau_G, & \text{若边 } (i, j) \text{ 在第(4)步获得的较优可行路径上} \\ \tau_G, & \text{否则} \end{cases} \quad (9.7.1)$$

(7) 计算每只蚂蚁的选择概率, 根据状态转移概率移动每只蚂蚁到下一点。

(8) 经过  $n$  个时刻,  $m$  只蚂蚁遍历所有节点, 即完成一次循环。此时, 对各个路径上的信息素进行更新。

(9) 进行迭代循环, 直到满足算法的停止条件为止。

#### 9.7.1.4 仿真算例

为了验证本小节所提出的人工免疫-蚁群混合算法的可行性, 这里以 Eil51TSP 为例进行仿真实验。

AIA 中, 每次抗体选择时保留与抗原匹配得最好的抗体, 并对随机产生的初始抗体进行预处理, 迭代 10000 次。蚁群算法中, 设置  $\tau_C=100$ ,  $\tau_G=4$ 。分别采用 AIA、基本蚁群算法和人工免疫-蚁群混合算法进行实验比较, 重复运行 20 次, 其计算结果如表 9.8 所示。

表 9.8 不同算法求解 Eil51TSP 的比较结果

算法	最优解	最差解	平均解
AIA	426	452	431.12
基本蚁群算法	426	—	428.06
人工免疫-蚁群混合算法	426	442	427.52

由表 9.8 可见, 人工免疫-蚁群混合算法在求解性能上明显优于基本蚁群算法或 AIA。

#### 9.7.2 基于免疫-蚁群算法的 QoS 路由算法

##### 9.7.2.1 免疫-蚁群算法实现中的若干问题

免疫算法在 QoS 路由问题中具体实现时必须解决如下几个关键问题。

(1) 抗体的编码形式如下:

$L(f, n_1, n_2, \dots, n_k, a)$	cost	delay	loss	jitter
---------------------------------	------	-------	------	--------

记为  $A(L(f, n_1, n_2, \dots, n_k, a), \text{cost}, \text{delay}, \text{loss}, \text{jitter})$ , 其中  $L(f, n_1, n_2, \dots, n_k, a)$ 、cost、delay、loss、jitter 分别表示路径、抗体的实际费用、时

延、丢失率和时延抖动。

(2) 抗原的编码形式如下：

from	aim	bandwidth	delay	loss	jitter
------	-----	-----------	-------	------	--------

记为  $R(\text{from}, \text{aim}, \text{bandwidth}, \text{delay}, \text{loss}, \text{jitter})$ , 其中 from、aim、bandwidth、delay、loss、jitter 分别表示路由请求的源节点、目的节点、所需带宽、允许的最大延时、丢失率和时延抖动。

(3) 抗原与抗体的匹配规则：如果某个抗体的  $L(f, n_1, n_2, \dots, n_k, a)$  是以抗原的 from 为起点、aim 为终点的不间断路径，并且满足第 7 章 7.3.1.1 小节所述的 4 个约束条件，则认为匹配；否则认为不匹配。

(4) 抗体的评价方法：对于多个抗体（候选解），通常根据抗体和抗原间 affinity 的大小来判断哪一个是最优抗体（最优解），这里将 affinity 定义为

$$\text{affinity} = \begin{cases} 2 \max \text{const} - c \cdot \cos t - d \cdot \text{delay} - l \cdot \text{loss} - j \cdot \text{jitter}, & \text{若满足 4 个约束} \\ \max \text{const} - c \cdot \text{cost} - d \cdot \text{delay} - l \cdot \text{loss} - j \cdot \text{jitter}, & \text{否则} \end{cases} \quad (9.7.2)$$

式中，maxconst 是一个比较大的常数， $c$ 、 $d$ 、 $l$ 、 $j$  均为常数，在算法中根据 cost、delay、loss 和 jitter 的相对重要性来选定其值。

(5) 新抗体的产生和新群体的形成规则：利用蚁群算法产生一批新抗体，再把它们和局部记忆库内的现有抗体一起进行配对、交叉，产生一批新抗体，最后从新产生的抗体和局部记忆库内的抗体中选取 affinity 值大的一批作为新抗体群。

(6) 记忆库的设计和使用：算法中采用两个记忆库，分别为局部记忆库和全局记忆库。局部记忆库存储求解某个抗原过程中的局部最优抗体，其规模为 10；全局记忆库存储针对某个抗原求得的最优抗体，其规模为 5。

(7) 抗体交叉规则：多个抗体随机配对后，在一对抗体中寻找相同节点（源节点和目的节点除外），若有多个相同节点，则随机选择其中一个作为交叉点。

此处的蚁群算法涉及到如下三种主要规则：

① 状态转移规则：蚂蚁在节点  $f$  处按照下式选择节点  $t$  的概率

$$p(f, t) = \frac{\text{pheromone}(f, t)}{\sum_i \text{pheromone}(f, i)} \quad (9.7.3)$$

式中， $i$  表示所有与  $f$  存在直接边的节点。

## ② 信息素局部更新规则

$$\text{pheromone}(f, t) = \begin{cases} (1 - a_0)\text{pheromone}(f, t) + a_0 \cdot \text{const}, & \text{若 } f \text{ 和 } t \\ & \text{是新选路径上的两个相邻节点} \\ (1 - a_0)\text{pheromone}(f, t) + b_0 \cdot a_0 \cdot \text{const}, & \text{否则} \end{cases} \quad (9.7.4)$$

式中,  $a_0$  表示信息素挥发程度的常数, 且  $0 < a_0 < 1$ ;  $b_0$  是为了避免不同边上的信息素两极分化, 从而减少陷入局部极小点的一个常数, 且  $0 < b_0 < 1$ ;  $\text{const}$  是常数。

## ③ 信息素全局更新规则

$$\text{pheromone}(f, t) = \begin{cases} (1 - a_1)\text{pheromone}(f, t) + a_1 \cdot \text{affinity}, & \text{若 } f \\ & \text{和 } t \text{ 是新选路径上的两个相邻节点} \\ (1 - a_1)\text{pheromone}(f, t) + b_1 \cdot a_1 \cdot \text{affinity}, & \text{否则} \end{cases} \quad (9.7.5)$$

式中,  $a_1$  表示信息素挥发程度的常数, 且  $0 < a_1 < 1$ ;  $b_1$  是为了避免不同边上的信息素的浓度两极分化, 从而减少陷入局部极小点的一个常数, 且  $0 < b_1 < 1$ ;  $\text{affinity}$  表示最优抗体和抗原的亲和力。

此处同时采用②、③两种更新规则, 可以加速全局优化过程。

### 9.7.2.2 基于免疫-蚁群算法的多约束 QoS 路由选择算法

为了提高效率, 在程序开始前拒绝所有时延抖动不能满足的请求, 并过滤掉那些带宽小于需求带宽的边, 具体实现中是令这些边的信息量为零。在针对某个抗原寻找抗体的过程中先不考虑时延、丢失率的限制, 而是待找到抗体后, 再判断新求得的抗体是否满足要求。根据上述分析, 基于免疫-蚁群算法的多约束 QoS 路由选择算法如下 (算法流程见图 9.8)。

(1) 输入请求: 输入请求  $R(\text{from}, \text{aim}, \text{bandwidth}, \text{delay}, \text{loss}, \text{jitter})$ , 即输入路由请求的起点、终点、所要求的最小带宽、所允许的最大延时、丢失率和时延抖动。

(2) 否定选择: 即检查源节点与目的节点间是否存在直接通路, 如果是, 则将其保存到局部记忆库中。

(3) 查询记忆库: 检查全局记忆库内有无符合该请求的记忆细胞 (抗体), 若有, 则将其保存到局部记忆库中。

(4) 产生初始抗体: 采用蚁群算法针对具体的路由请求 (抗原), 按照状态转移规则即公式 (9.7.3) 产生一批初始抗体, 作为初始抗体种群, 按公式 (9.7.4) 进行信息素局部更新。

(5) 抗体更新: 利用蚁群算法对具体的路由请求 (抗原体) 产生一批新抗

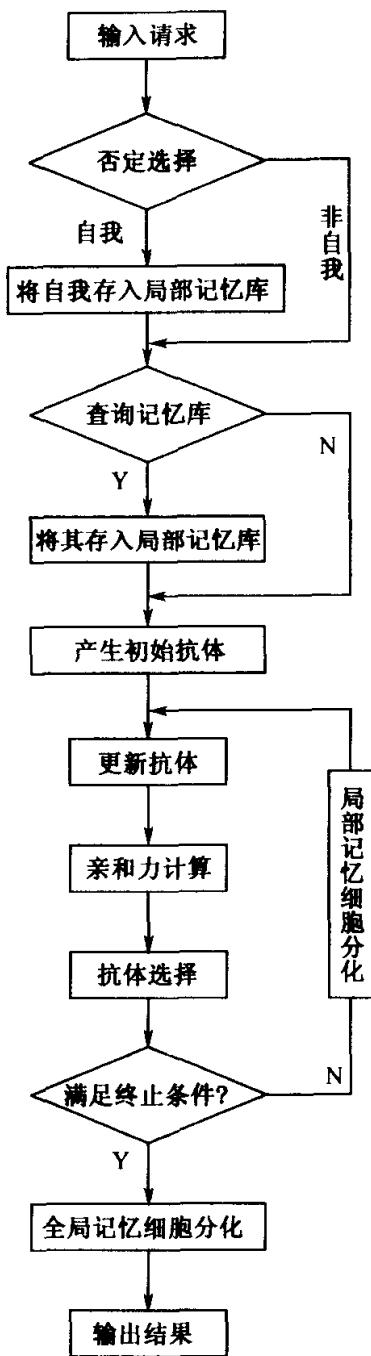


图 9.8 免疫-蚁群算法流程图

体，并将其和局部记忆库内的现有抗体一起进行配对、交叉，产生新抗体。

(6) 计算 affinity：按公式 (9.7.2) 计算各个抗体与抗原（路由请求）间的 affinity。

(7) 抗体选择：从本次循环所获得的抗体中，选出具有最大 affinity 的抗体 A。

(8) 判断终止条件：若在获得某个候选全局抗体 A 后，连续 T 次迭代中获

得的局部最优抗体的 affinity 都不大于 A 的 affinity，则跳转到第（10）步；否则跳转到第（9）步。

(9) 局部记忆细胞分化：从本次新产生的抗体中，根据 affinity 的大小，选取一批较优的抗体存入局部记忆库中；根据本次新加入局部记忆库中的那些抗体（局部最优抗体），按公式（9.7.5）进行信息素全局更新，跳转到第（5）步。

(10) 全局记忆细胞分化：用全局最优抗体取代全局记忆库中最少被选为初始抗体的那个记忆细胞，并且输出结果，结束循环。

最终目标是通过上述算法找一条链路，不仅满足所有的约束条件，而且该链路传输的综合代价最小，也就是抗体与抗原的 affinity 最大。

### 9.7.2.3 仿真实验

本实验采用了第 7 章中图 7.8 所示的网络拓扑图及其参数。实验时，设置  $a_0 = 0.069$ ,  $a_1 = 2.2 \times 10^{-7}$ ,  $b_0 = b_1 = 0.05$ ,  $\text{maxconst} = 100000$ ,  $c = 1000$ ,  $d = 100$ ,  $l = 10$ ,  $j = 1$ 。每条边的信息量初始值取为 100，每次迭代采用 10 只蚂蚁。算法的终止条件为在获得某个候选全局抗体 A 后，连续 T 次迭代中获得局部最优抗体的亲和力都不大于 A 的亲和力，仿真实验中设定  $T = 4$ 。运行后，所获得的全局优化结果如表 9.9 所示。

表 9.9 免疫-蚁群算法的全局优化结果

路由请求						QoS 路由结果					
源节点	目的节点	带宽	时延	丢失率	时延抖动	所选路径	费用	时延	丢失率	时延抖动	迭代次数
1	6	70	7	0.001	3	1→2→4→6	8	7	1.1E-05	2	11
2	6	70	7	0.001	3	2→4→6	6	3	1.0E-05	2	11
3	8	70	7	0.001	3	3→4→2→8	7	5	1.1E-05	0	6
1	7	70	10	0.001	3	1→2→8→7	6	10	2.0E-06	1	3
1	7	70	12	0.001	3	1→5→6→7	3	12	1.0E-06	1	14

由表 9.9 可见，在保证其他参数与文献 [58, 59, 60] 中算法相同的条件下，这里所提出的免疫-蚁群算法平均迭代 10 次就能得到最优解，小于基于基本蚁群算法的 QoS 路由选择的迭代次数（134 次）<sup>[60]</sup>和基于遗传算法的 QoS 路由选择的迭代次数（152 次）<sup>[58]</sup>，更远远小于基于 Hopfield 神经网络的 QoS 路由选择的迭代次数（10941 次）<sup>[59]</sup>。由表 9.9 的最后两行可见，随着约束条件的不同，即使请求（抗原）具有相同的源节点、目的节点，免疫-蚁群算法也会给出不同的解（抗体），这说明了公式（9.7.2）中的各个参数是相互关联的。

## 9.8 本章小结

本章在阐述遗传算法、人工神经网络、微粒群算法、人工免疫算法、人工鱼群算法等几种代表性仿生优化算法基本原理的基础上，深入分析了这些仿生优化算法的相同点和不同点，并给出了蚁群算法与遗传算法、人工神经网络、微粒群算法、人工免疫算法的融合策略。

目前，蚁群算法与仿生优化算法融合策略的研究还处于初始阶段，在融合模型上还需要进一步改进，其应用范围也有待于进一步拓宽。此外，蚁群算法与人工鱼群算法、混合蛙跳算法<sup>[61]</sup>等新兴仿生优化算法的融合到目前还是一个研究空白，今后将蚁群算法与一些新兴仿生优化算法进行融合方面还存在着相当丰富 的研究内容。

## 参 考 文 献

- 1 Holland J H. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press, 1975; MIT Press, 1992
- 2 Goldberg D E. Genetic algorithm in search optimization and machine learning. New York: Addison-Wesley Publishing Company, 1989
- 3 Hebb D O. The organization of behavior. New York: Wiley Publishing Press, 1949
- 4 Hopfield J J. Neurons with graded response have collective computational properties like those of two state neurons. Proceedings of the Natl. Acad. Sci., 1984, 81: 3088~3092
- 5 Hopfield J J, Tank D W. Computing with neural circuits: a model. Science, 1986, 233: 625~633
- 6 Becker S. Unsupervised learning procedures for neural networks. International Journal of Neural Systems, 1991, 2: 17~33
- 7 Kennedy J, Eberhart R C. Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks, 1995, 1942~1948
- 8 Eberhart R, Kennedy J. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micro-Machine and Human Science, 1995, 39~43
- 9 Bersini H, Varela F. Hints for adaptive problem solving leaned from immune network. Parallel Problem Solving from Nature, Berlin Heidelberg: Springer-Verlag, 1991
- 10 Yoshiteru I, Norihiko A. Active noise control by an immune algorithm: adaptation in immune system as an evolution. Proceedings of the International Conference on Evolutionary Computation, 1996, 150~153
- 11 Hong J, Lee W, Lee S, et al. An efficient production algorithm for *multihead* surface mounting machines using the biological immune algorithm. International Journal of Fuzzy Systems, 2000, 2(1): 45~53
- 12 Dasgupta D. Artificial neural networks and artificial immune systems: similarities and differences. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1997, 1: 873~878
- 13 李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法. 系统工程理论与实践, 2002, 22(11): 32~38

- 14 李晓磊, 钱积新. 基于分解协调的人工鱼群优化算法研究. 电路与系统学报, 2003, 8(1): 1~6
- 15 李晓磊, 路飞, 田国会等. 组合优化问题的人工鱼群算法应用. 山东大学学报, 2004, 34(5): 64~67
- 16 李晓磊, 冯少辉, 钱积新. 基于人工鱼群算法的鲁棒 PID 控制器参数整定方法研究. 信息与控制, 2004, 33(1): 112~115
- 17 李晓磊, 薛云灿, 路飞等. 基于人工鱼群算法的参数估计方法. 山东大学学报, 2004, 34(3): 84~87
- 18 Reynolds C W. Flocks, herds, schools: a distributed behavioral model. Computer Graphics, 1987, 21(4): 25~34
- 19 段海滨, 王道波, 朱家强等. 蚁群算法理论及应用研究的进展. 控制与决策, 2004, 19(12): 1321~1326, 1340
- 20 段海滨, 王道波, 于秀芬. 几种新型仿生优化算法的比较研究. 系统工程与电子技术, 待发表
- 21 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 22 Duan H B, Wang D B, Yu X F. Research on the optimum configuration strategy for the adjustable parameters in ant colony algorithm. Journal of Communication and Computer, 2005, 2(9): 32~35
- 23 Duan H B, Wang D B, Yu X F. Comparison of several kinds of intelligent bionic optimization algorithm: ACO, AFO, PSO, BCO, MA and SFL. Journal of Bionics Engineering, to appear
- 24 段海滨, 王道波. 一种快速全局优化的改进蚁群算法及仿真. 信息与控制, 2004, 33(2): 241~244
- 25 李艳君. 拟生态系统算法及其在工业过程控制中的应用. 杭州: 浙江大学博士学位论文, 2001
- 26 王凌. 智能优化算法及其应用. 北京: 清华大学出版社, 2001
- 27 徐宁, 李春光, 张健等. 几种现代优化算法的比较研究. 系统工程与电子技术, 2002, 24(12): 100~103
- 28 Colorni A, Dorigo M, Maniezzo V, et al. Distributed optimization by ant colonies. Proceedings of the 1st European Conference on Artificial Life, 1991, 134~142
- 29 Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- 30 周明, 孙树栋. 遗传算法原理及其应用. 北京: 国防工业出版社, 1996
- 31 Abbattista F, Abbattista N, Caponetti L. An evolutionary and cooperative agents model for optimization. Proceedings of the IEEE International Conference on Evolutionary Computation, 1995, 2: 668~671
- 32 White T, Pagurek B, Oppacher F. ASGA: improving the ant system by integration with genetic algorithms. Proceedings of the 3rd Annual Genetic Programming Conference, 1998, 610~617
- 33 Li M, Wang H, Li P. Tasks mapping in multi-core based system: hybrid ACO&GA approach. Proceedings of the 5th International Conference on ASIC, 2003, 1: 335~340
- 34 Pilat M L, White T. Using genetic algorithms to optimize ACS-TSP. Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science, 2002, 2463: 282~287
- 35 Acan A. GAACO: A GA+ACO hybrid for faster and better search capability. Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS2002, Lecture Notes in Computer Science, 2002, 2463: 300~301
- 36 Gong D X, Ruan X G. A hybrid approach of GA and ACO for TSP. Proceedings of the 5th World Congress on Intelligent Control and Automation, 2004, 3: 2068~2072
- 37 孙力娟, 王良俊, 王汝传. 改进的蚁群算法及其在 TSP 中的应用研究. 通信学报, 2004, 25(10): 111~116

- 38 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法的融合. 计算机研究与发展, 2003, 40(9): 1351~1356
- 39 邵晓巍, 邵长胜, 赵长安. 利用信息量留存的蚁群遗传算法. 控制与决策, 2004, 19(10): 1187~1189, 1193
- 40 陈宏建, 陈 岭, 徐晓华等. 改进的增强型蚁群算法. 计算机工程, 2005, 31(2): 176~178
- 41 Kumar G M, Haq A N. Hybrid genetic-ant colony algorithms for solving aggregate production plan. Journal of Advanced Manufacturing Systems, 2005, 4(1): 103~111
- 42 Stützle T, Hoos H H. MAX-MIN ant system. Future Generation Computer System, 2000, 16(8): 889~914
- 43 Michalewicz Z. Genetic algorithm + data structure = evolution programs. New York: Springer-Verlag, 1994
- 44 Li S H, Liu Z M. A general CAC approach using novel ant algorithm training based neural network. Proceedings of the International Joint Conference on Neural Networks, 1999, 3: 1885~1888
- 45 Zhang S B, Liu Z M. Neural network training using ant algorithm in ATM traffic control. Proceedings of the 2001 IEEE International Symposium on Circuits and Systems, 2001, 2: 157~160
- 46 洪炳熔, 金飞虎, 高庆吉. 基于蚁群算法的多层前馈神经网络. 哈尔滨工业大学学报, 2003, 35(7): 823~825
- 47 Hong B R, Jin F H, Guo Q. Hopfield neural network based on ant system. Journal of Harbin Institute of Technology (New Series), 2004, 11(3): 267~269
- 48 邹政达, 孙雅明, 张智晟. 基于蚁群优化算法递归神经网络的短期负荷预测. 电网技术, 2005, 29(3): 59~63
- 49 侯云鹤, 鲁丽娟, 熊信良等. 广义蚁群与粒子群结合算法在电力系统经济负荷分配中的应用. 电网技术, 2004, 28(21): 34~38
- 50 侯云鹤, 熊信良, 吴耀武等. 基于广义蚁群算法的电力系统经济负荷分配. 中国电机工程学报, 2003, 23(3): 59~64
- 51 Walters D C, Sheble G B. Genetic algorithm solution of economic dispatch with valve point loading. IEEE Transactions on Power Systems, 1993, 8(3): 1325~1332
- 52 唐巍, 李殿璞. 电力系统经济负荷分配的混沌优化方法. 中国电机工程学报, 2000, 20(10): 36~40
- 53 Sinha N, Chakrabarti R, Chattopadhyay P K. Evolutionary programming techniques for economic load dispatch. IEEE Transactions on Evolutionary Computation, 2003, 7(1): 83~94
- 54 Lee Z J, Lee C Y, Su S F. An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem. Applied Soft Computing, 2002, 2(1): 39~47
- 55 Feng Y J, Feng Z R. An immunity-based ant system for continuous space multi-modal function optimization. Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, 2004, 2: 1050~1054
- 56 胡纯德, 祝延军, 高随祥. 基于人工免疫算法和蚁群算法求解旅行商问题. 计算机工程与应用, 2004, 40(34): 60~63
- 57 蒋加伏, 陈荣元, 唐贤瑛等. 基于免疫-蚂蚁算法的多约束 QoS 路由选择. 通信学报, 2004, 25(8): 89~95
- 58 Feng X, Li J Z, Wang J V, et al. QoS routing based on genetic algorithm. Computer Communications, 1999, 22(15-16): 1392~1399
- 59 Chotpat P, Goutam C, Norio S. Neural network approach to multicast routing in real-time communication networks. Proceedings of the International Conference on Network Protocols, 1995, 332~339

- 60 Zhang S, Liu Z. A QoS routing algorithm based on ant algorithm. Proceedings of the IEEE International Conference on Communications, 2001, 5: 1581~1585
- 61 Eusuff M M, Lansey K E. Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management, 2003, 129(3): 210~225

# 第 10 章 展望

## 10.1 引言

蚁群算法创立十多年来，无论在算法理论还是在算法应用方面都取得了很多突破性研究进展。作为一个前沿性的热点研究领域，蚁群算法已引起越来越多国内外研究者的关注，近五年内其研究人员和研究成果均成几何级数增长。初步统计结果表明，2000 年蚁群算法的相关学术论文还不足 200 篇；而到了 2005 年 6 月，蚁群算法的相关学术论文已经突破了 1600 篇，其应用范围几乎涉及到各个优化领域，而且还出现了蚁群算法仿生硬件，可见这种新兴的仿生优化算法已经显示出强大的生命力和广阔的发展前景。

本章作为全书的最后一章，将集中讨论目前蚁群算法研究领域中所存在的主要问题，并重点从算法的模型改进、理论分析、并行实现、应用领域、硬件实现、智能融合等角度对蚁群算法在今后的研究方向进行探讨，希望能为有志于蚁群算法研究的科技工作者提供有益的借鉴和参考。

## 10.2 蚁群算法的模型改进

蚁群算法发展至今，人们已经针对不同的具体问题提出了许多不同类型的改进蚁群算法模型<sup>[1]</sup>，但这些改进的蚁群算法模型往往普适性不强，同时基本蚁群算法模型也不能直接应用于解决所有的优化问题。另外，自然界中的真实蚁群还有许多其他的智能行为，用逆向思维和发散思维开发不同的蚁群算法模型是一条新的发展思路。基于此，今后可以从以下几个方面对其模型进行改进。

### 1. 设计一种通用的蚁群算法普适性模型

蚁群算法在被首次提出时是用于解决 TSP 的，后来提出的一些改进蚁群算法基本上都是针对某一类问题的专用算法，各种算法之间的相似性较差。而现实中的问题千差万别，并不是每一个问题都能够很容易划归到这样一个拥有高度社会性的蚁群智能模型中，而且系统高层次的行为是通过低层次昆虫之间简单行为的交互涌现而产生的。单个个体控制的简单并不意味着整个系统设计的简单，因此必须将高层次的复杂行为（即系统所要执行的功能）映射到低层次简单个体的简单行为上面，而这两者之间存在较大差别<sup>[2~4]</sup>。而且在进行系统设计时，也要保证多个个体简单行为的交互，以能涌现出所期望的高层次复杂行为，这可以说

是群体仿生优化领域中一个极为困难的问题。因此，在算法的普适性研究方面，今后应进一步探索蚁群算法多种模型的高层次统一机制，这方面可结合复杂性系统展开研究，提出更普适的蚁群算法模型。目前来看，蚁群算法的正反馈机制就是一个很好的普适性模型。

## 2. 进一步研究自然蚁群行为，提出更加智能化的蚁群混合行为模型

从深度上说，虽然蚁群算法是基于蚂蚁觅食行为而发展起来的，但是自然界中的真实蚂蚁觅食机理并不那么简单，至今仍有许多借鉴之处，今后应该继续挖掘蚂蚁觅食中一些未知的行为，从而得到新的蚁群算法模型；从广度上说，蚂蚁等昆虫除了寻找食物之外，还有着任务分配、构造墓地、建造巢穴、繁衍后代、清除垃圾、保卫领地等多种复杂的社会行为<sup>[5]</sup>，今后可将蚁群的这些行为与其觅食行为进行融合、统一，进而提出更加智能化的蚁群算法混合行为模型。这方面还有待于今后从昆虫学、社会学、仿生学、信息学等综合角度对其进行深入研究。

## 3. 摆脱传统模拟框架，开发新的蚁群算法模型

由于蚁群算法是建立在对自然界中真实蚂蚁觅食行为进行模拟这一基础上的，模拟把蚁群算法与自然界中可能发生的过程紧密的联系在一起。模拟具有解释能力的同时，也有不利的方面，沿着模拟的道路走下去会阻碍那些不符合模拟所认可模式的发展。因此，在可供选择的框架下，有必要采用逆向思维和发散思维重新表示各种通用方法的原理和过程，进而开发出更加优良的蚁群算法模型，这能够摆脱给蚁群算法以启示的模拟参照的最初限制，从而开辟一条新的通向比较和应用的综合之路。

## 10.3 蚁群算法的理论分析

蚁群算法是一种概率搜索算法，从数学上对其正确性与可靠性进行证明相对于确定性算法而言比较困难。自 Gutjahr W J 最先从有向图论的角度对一种改进蚁群算法的收敛性进行理论证明至今，人们在其收敛性研究方面已经取得了相当丰富的理论研究成果，但是这些理论成果基本上都是针对一些改进蚁群算法的理论分析，例如 Gutjahr W J<sup>[6, 7]</sup>对一类 GBAS、GBAS/tdev 及 GBAS/tdlb 算法的收敛性进行了证明；Stüenzle T 和 Dorigo M 对  $ACO_{gb, \epsilon_{min}}$  算法<sup>[8]</sup>的收敛性进行了理论证明；Yoo J H 等<sup>[9, 10]</sup>对一类分布式蚂蚁路由算法的收敛性进行了深入的理论分析；孙焘等<sup>[11]</sup>对一类简单蚁群算法的收敛性作了初步研究；丁建立等<sup>[12]</sup>对一种遗传-蚁群算法的收敛性进行了 Markov 理论分析；Hou Y H 等<sup>[13]</sup>基于不动点理论对一类 GACA 的收敛性进行了初步研究；等等。可见，这些证明并没有

把对蚁群算法的理论分析统一到一个共同的框架之下。

此外，在蚁群算法的理论研究方面还存在许多问题需要进一步解决，比如初始化参数选择问题、信息素分配问题、收敛速度问题等，这些均带有经验和直觉性，至今没有经过严格的数学论证；同时，连续域蚁群算法的收敛性证明仍存在许多研究空白。今后蚁群算法的收敛性证明和理论分析仍然是一个非常具有挑战性的研究方向。

## 10.4 蚁群算法的并行实现

蚁群算法本身隐含着一定的并行性，单只蚂蚁在一次循环中独立于其他蚂蚁。因此从本质上说，蚁群算法应以分布式的协同优化计算方式为特征，而在串行计算机上对蚁群算法的进行模拟并不能真正体现蚁群算法的本质特征，因此，进一步的研究工作还应开展蚁群算法的并行机实现。

并行机实现中往往需要在计算与通信之间寻求一种平衡，蚁群算法的并行机实现主要集中在最优的并行化<sup>[14~17]</sup>。所谓最优的并行化是指使用适量的处理机以最小的代价使得并行化蚁群算法性能达到当前情况下的最好，或者说当并行化蚁群算法性能不变时，应竭力减少计算和通信的成本。

这里，用  $P=(P_1, P_2, \dots, P_i, \dots, P_n)$  表示  $n$  台处理机序列，其中  $1 \leq i \leq n$ ， $i \in N$ 。设  $W=(w_1, w_2, \dots, w_i, \dots, w_n)$  表示处理机的处理能力序列，其中  $w_i$  表示处理机  $P_i$  的处理能力，并依赖于  $P_i$  的内存大小、主频等硬件条件，且  $w_i \geq 0$ 。记  $M=(m_1, m_2, \dots, m_i, \dots, m_n)$  表示蚁群规模序列，其中  $m_i$  表示处理机  $P_i$  的上分配到的蚂蚁数目， $m_i \in N$ ，且记  $m=|M|$  为蚁群规模。设  $U=(u_1, u_2, \dots, u_i, \dots)$  表示处理机之间进行信息交换的时间序列。为了减少通信开销，当蚁群算法迭代到第  $u$  代时，对所有的处理机按照下述定义进行相互通信。

设  $R=(R_{u_1}, R_{u_2}, \dots, R_{u_i}, \dots)$  表示处理机之间进行信息交换的关系序列， $R_{u_i}$  是一个  $n \times n$  的对称矩阵，每个元素取 2 个值。例如  $R_{u_i}(i, j)=1$  表示第  $u_i$  代时  $P_i$  和  $P_j$  要进行通信，而  $R_{u_i}(i, j)=0$  表示第  $u_i$  代时  $P_i$  和  $P_j$  无须进行通信。

设  $T=T(W, M, U, R)$  表示并行化蚁群算法在配置参数为  $W, M, U, R$  时的计算成本，一般可以用处理机最长的运行时间来衡量，当处理机之间的处理能力无差异时可省略  $W$ 。在并行化蚁群算法运行过程中，如果每次都要求对所有的处理机进行通信，则在不引起歧义的前提下可省略  $R$ 。对于串行化蚁群算法，用  $T_s=T_s(w, m)$  表示其计算成本，其中  $w$  表示处理机的处理能力。如果考虑并行化蚁群算法中  $\forall w_i=w$ ，那么可简单记作  $T=T(M, U, R)$  和  $T_s=T_s(m)$ 。设  $S=S(W, M, U, R)=\frac{T_s(w, m)}{T(W, M, U, R)}$  表示并行化蚁群算法的加速比。在不考虑处理机间差异的条件下，记  $S=S(W, M, U, R)=\frac{T_s(m)}{T(M, U, R)}$ 。同时，设  $E=$

$E(W, M, U, R) = \frac{S(W, M, U, R)}{n}$  表示并行化蚁群算法的效率。当不考虑处理机间差异时，记  $E = E(M, U, R) = \frac{S(M, U, R)}{n}$ 。若设  $\eta = \eta(W, M, U, R) = S(W, M, U, R) \times E(W, M, U, R)$  为效用函数，该函数表示并行化蚁群算法的计算效率。当不考虑处理机间差异时，记  $\eta = \eta(M, U, R) = S(M, U, R) \times E(M, U, R)$ 。对于并行化蚁群算法的最优化目标，就是求使  $\eta(W, M, U, R)$  达到最大值的  $M, U, R$  组合。

在研究蚁群算法并行机实现问题时，需要解决对蚁群算法并行化过程中并行计算模型的选择以及对蚁群算法的分解、映射方法的改进等问题，还需要解决在对蚁群算法并行化过程中粒度处理标准的问题，从而使并行处理过程具有较好的可扩展性，并具有良好的负载均衡性。目前主要存在如下几个问题：

- (1) 在不考虑处理机间差异、选择同步更新方法（即  $R_{u_i}$  是全 1 方阵）的情况下，如何确定问题规模与加速比之间的关系；在处理机增多的情况下，如何选择处理机的最佳数目以使计算时间减少。
- (2) 在不考虑处理机间差异、选择同步更新方法（即  $R_{u_i}$  是全 1 方阵）的情况下，且当处理机一定时，如何选择最优的蚁群规模  $m$  以使效用函数的值最大。
- (3) 在不考虑处理机间差异、选择异步更新方法（即  $R_{u_i}$  是非全 1 方阵）的情况下，且当问题规模一定时，如何分配处理机数目和每个处理机上的蚁群规模以使效用函数值最大。
- (4) 在其他条件一定的情况下，如何设计一个更好的  $R$  序列也是一个不容忽视的问题。
- (5) 当并行机实现时，局部迭代若干次后需要交换信息。局部迭代次数的增大也会使得蚁群算法早熟的概率增大，而局部迭代次数减小则会增加通信时间，因此还要努力解决如何有效避免蚁群算法的过早停滞和减少通信开销之间的平衡问题。

## 10.5 蚁群算法的应用领域

从目前公开发表的蚁群算法相关学术论文来看，大约五分之三的论文是蚁群算法在不同优化领域中的应用研究成果。虽然蚁群算法的应用范围已经几乎涉及到各个优化领域，但是还存在很多不足，今后在应用上可着重从如下两方面对其进行研究。

- (1) 纵向而言，蚁群算法的应用深度还不够。从公开发表的研究成果来看，大部分应用还仅仅停留在仿真阶段，而且所做的研究大都是在对实际问题实验条件或约束条件进行简化的前提下进行的。今后尤其在动态优化问题、随机优化问

题、多目标优化问题等方面的研究还需要进一步深化。

(2) 横向而言, 蚁群算法的应用领域还需要进一步拓宽。不同的应用领域都有许多不同的实际问题, 尝试蚁群算法在不同问题中的实际应用也是今后的一项重要研究内容。如何抽象实际问题, 使蚁群算法的求解更接近工程实际是广大蚁群算法研究者们所共同关注的一个关键性问题。

## 10.6 蚁群算法的硬件实现

仿生硬件是并行计算环境下的产物, 蚁群算法的硬件实现是仿生硬件研究领域中一个新的子分支。虽然蚁群算法的硬件实现在理论与实验研究方面均取得了明显进展, 但是还存在以下主要问题<sup>[18]</sup>:

(1) 目前所开发的蚁群算法仿生硬件实现所能处理电路的规模、复杂程度远远比不上常规电路综合方法。对于规模较大、功能较复杂的电路, 目前已开发的蚁群算法仿生硬件的进化时间太长或者在要求的时间内很难找到解, 这是制约其工程应用的主要问题, 今后应该加强对这一问题的进一步研究。

(2) 对于蚁群算法仿生硬件产品的可靠性、可测试性、普适性以及鲁棒性问题也需要进一步研究。虽然蚁群算法硬件不需要对硬件的结构有很深入的了解, 只需要根据其外部特性就可用蚁群算法进行自动寻优。但是, 还需要进一步对蚁群算法仿生硬件进行理论分析与建模, 因为, 有些问题(如模式识别问题)仅凭借对系统外部特性的不完备测试是很难保证蚁群算法仿生硬件系统的可靠性的。

(3) 今后应该研制开发可批量工程应用的蚁群算法仿生硬件芯片, 以进一步推进蚁群算法硬件产品的商业化进程。

(4) 研制蚁群算法的仿生硬件开发平台、建立蚁群算法仿生硬件产品的技术规范等都是今后蚁群算法硬件实现方面一些不可忽视的研究内容。

## 10.7 蚁群算法的智能融合

本书在第9章重点阐述了蚁群算法与遗传算法、人工神经网络、微粒群算法、人工免疫算法的融合策略, 但是从现有的成果来看, 这些智能融合大部分是一些初步尝试, 很多都是针对具体问题来进行的。所解决的问题不同, 其融合策略也就存在着很多差异, 因此应该在现有成果的基础上继续进行深入研究, 努力探索蚁群算法与一种或几种仿生优化算法相融合的统一机制。

另一方面, 蚁群算法与一些新兴的、目前影响不是很大的仿生优化算法(如人工鱼群算法<sup>[19]</sup>、混合蛙跳算法<sup>[20]</sup>、蜂群算法<sup>[21]</sup>、情感计算<sup>[22]</sup>等)之间的融合至今还存在着研究空白, 今后应多尝试将蚁群算法与这些仿生优化算法进行智能融合并应用于解决实际问题。这样不仅容易诞生许多创新性研究成果, 而且也

是一个非常有意义的研究方向。

## 10.8 本章小结

本章讨论了目前蚁群算法研究领域所存在的主要问题，并重点从模型改进、理论分析、并行实现、应用范围、硬件实现、算法融合等方面对蚁群算法在今后进一步的研究方向进行展望。

尽管人们对蚁群算法的研究时间不长，在这一领域还有一些问题需要进一步研究和解决，但是理论研究和实际应用表明它是一种很有前途的仿生优化算法。随着人类认识的进步和社会发展的加速，仿生智能及最优化系统理论将越来越成为科学认识和工程实践的有力工具，因此，关于蚁群算法理论及其应用的研究必将是一个长期的研究课题。相信随着人们对仿生智能系统理论及应用研究的不断深入，蚁群算法这一新兴的仿生优化算法必将展现出更加广阔、更加引人注目的发展前景，“蚁群算法”这朵仿生优化百花园中的“奇葩”必将更加娇艳、更加芬芳！

## 参 考 文 献

- 1 段海滨, 王道波, 朱家强等. 蚁群算法理论及应用研究的进展. 控制与决策, 2004, 19(12): 1321~1326, 1340
- 2 Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behavior. Nature, 2000, 406(6): 39~42
- 3 Jackson D E, Holcombe M, Ratnieks F L W. Trail geometry gives polarity to ant foraging networks. Nature, 2004, 432(7019): 907~909
- 4 Holland J. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press, 1975; MIT Press, 1992
- 5 Dorigo M, Bonabeau E, Theraulaz G. Ant algorithms and stigmergy. Future Generation Computer Systems, 2000, 16(8): 851~871
- 6 Gutjahr W J. A graph-based ant system and its convergence. Future Generation Computer Systems, 2000, 16(8): 873~888
- 7 Gutjahr W J. ACO algorithms with guaranteed convergence to the optimal solution. Information Processing Letters, 2002, 82(3): 145~153
- 8 Stüenzle T, Dorigo M. A short convergence proof for a class of ant colony optimization algorithms. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 358~365
- 9 Yoo J H, La R J, Makowski A M. Convergence results for ant routing. Technical Report CSHCN 2003-46, Institute for Systems Research, University of Maryland, College Park (MD), 2003
- 10 Yoo J H, La R J, Makowski A M. Convergence of ant routing algorithms-results for simple parallel network and perspectives. Technical Report CSHCN 2003-44, Institute for Systems Research, University of Maryland, College Park (MD), 2003
- 11 孙焘, 王秀坤, 刘业欣等. 一种简单蚂蚁算法及其收敛性分析. 小型微型计算机系统, 2003, 21(8):

1524~1526

- 12 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法融合的马尔可夫收敛性分析. 自动化学报, 2004, 30(4): 659~664
- 13 Hou Y H, Wu Y W, Lu L J, et al. Generalized ant colony optimization for economic dispatch of power systems. Proceedings of the 2002 International Conference on Power System Technology, 2002, 1: 225~229
- 14 Dorigo M, Stützle T. Ant colony optimization. Cambridge, MA: MIT Press, 2003
- 15 段海滨. 蚁群算法及其在高性能电动仿真转台参数优化中的应用研究. 南京: 南京航空航天大学博士学位论文, 2005
- 16 秦玲. 蚁群算法的改进与应用. 扬州: 扬州大学硕士学位论文, 2004
- 17 Chu S C, Roddick J F, Pan J S, et al. Parallel ant colony systems. Lecture Notes in Artificial Intelligence, 2003, 2871: 279~284
- 18 段海滨, 王道波, 于秀芬. 蚁群算法硬件实现的研究进展. 控制与决策, 待发表
- 19 李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法. 系统工程理论与实践, 2002, 22(11): 32~38
- 20 Eusuff M M, Lansey K E. Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management, 2003, 129(3): 210~225
- 21 Sato T, Hagiwara M. Bee system: finding solution by a concentrated search. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1997, 4: 3954~3959
- 22 Canamero D. Modeling motivations and emotions as a basis for intelligent behavior. Proceedings of the 1st International Conference on Automation Agents, 1997, 145~155

## 附录 A 基本蚁群算法程序

### A.1 C 语 言 版

```
//*****  
//          Basic Ant Colony Algorithm for TSP          //  
//          C Version                                //  
//*****  
  
# include <iostream.h>  
# include <fstream.h>  
# include <math.h>  
# include <time.h>  
# include <conio.h>  
# include <stdlib.h>  
# include <iomanip.h>  
# define N 50           // city size  
# define M 30           // ant number  
double initao = 1;  
double tao [N][N];  
double detatao [N][N];  
double distance [N][N];  
double yita [N][N];  
int tabu [M][N];  
int route [M][N];  
double solution [M];  
int bestroute [N];  
double BestSolution = 10000000000;  
double alfa, beta, rou, Q;  
int NcMax;  
void initparameter (); // initialize the parameters of basic ACA  
double EvaluateSolution ( int *a ); // evaluate the solution of TSP, and calculate the length  
of path  
void InCityXY ( double x[ ], double y[ ] ); // input the nodes' coordinates of TSP  
  
void main( )  
{
```

```

int NC = 0;
initparameter();
double x[N];
double y[N];
InCityXY(x, y);
for (int i = 0; i < N; i++)
for (int j = i + 1; j < N; j++)
{
    distance[j][i] = sqrt((x[i]-x[j])*(x[i]-x[j])-(y[i]-y[j])*(y[i]-y[j]));
    distance[i][j] = distance[j][i];
}
// calculate the heuristic parameters
for (i = 0; i < N; i++)
    for (i = 0; i < N; i++)
    {
        tao[i][j] = initao;
        if (j != i)
            yita[i][j] = 100/(distance[i][j] + 50-int(distance[i][j]) % 50)
    }
for (int k = 0; k < M; k++)
    for (i = 0; i < N; i++)
        route[k][i] = -1;
        srand(time(NULL));
for (k = 0; k < M; k++)
{
    route[k][0] = (k + N) % N;
    tabu[k][route[k][0]] = 1;
}
// each ant try to find the optimal path
do
{
    int s = 1;
    double partsum;
    double pper;
    double drand;
    // ant choose one whole path
    while (s < N)
    {
        for (int k = 0; k < M; k++)

```

```

{
    int jrand = rand( ) % 3000;
    drand = double (jrand) / 3001;
    partsum = 0; pper = 0;
    for (int j = 0; j < N; j++)
    {
        if(tabu [k][j] == 0)
            partsum += pow(tao[route[k][s-1][j], alfa]*pow(yita[route[k][s-1]][j], beta);
        }
        for (j = 0; j < N; j++)
        {
            if(tabu[k][j] == 0)
                pper += pow(tao[route[k][s-1][j], alfa]*pow(yita[route[k][s-1][j], beta) / partsum;
            if (ppendrand) break;
        }
        tabu [k][j] = 1;
        route [k][s] = j;
    }
    s++;
}
// the pheromone is updated
for (i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
    {
        detatao [i][j] = 0;
    }
for (int k = 0; k < M; k++)
{
    solution [k] = EvaluateSolution (route [k]);
}
for (k = 1; k < M; k++)
{
    if (solution [k] < BestSolution)
    {
        BestSolution = solution [k];
        for (s = 0; s < N; s++)
            BestRoute [s] = route [k][s];
    }
}

```

```

for (k = 0; k<M; k++)
{
    for (s = 0; s< N-1; s++)
        detatao [route[k][s]] [route[k][s]] += Q / solution [k];
    detatao [route[k][N-1]] [route[k][0]] += Q / solution [k];
}

for (i = 0; i<N; i++)
{
    for (j = 0; j<N; j++)
    {
        tao [i][j] = rou * tao [i][j] + detatao [i][j];
        if (tao [i][j]< 0.00001)
            tao [i][j] = 0.00001;
        if (tao [i][j]> 20)
            tao [i][j] = 20;
    }
}

for (k = 0; k<M; k++)
{
    for( int j = 1; j<N; j++)
    {
        tabu[k][route[k][j]] = 0;
        route[k][j] = -1;
    }
    NC++;
}

} while (NC<NcMax);

// output the calculating results
fstream result;
result.open ("Optimal _ Results.dat", ios::app);
if(! result)
{
    cout<<"Can't open the 'Optimal _ Results' file\n";
    abort();
}

result<< "*****" << endl;
result<< "The initialized parameters of ACA are as follows: " << endl;
result<< "alfa = " << alfa << ", beta = " << beta << endl;
result<< "Q = " << Q << ", The maximum iteration number of ACA is: " << NcMax << endl;
result<< " The evaporation parameter of ACA is: " << rou << endl;
result<< "*****" << endl;
for(i = 0; i<N; i++)
{
    result<< bestroute[i] << ". ";
    result<< endl;
}

```

```

result.close( );
}

//***** Function: Evaluate the solution *****
double EvaluateSolution ( int *a)
{
    double dist = 0;
    for( int i = 0; i < N; i + + )
        dist + = distance [*(a + (i + N) % N)][*(a + (i + N + 1) % N)];
    return dist;
}

//***** Function: Set the city coordinates *****
void InCityXY(double x[ ], double y[ ])
{
    fstream inxyfile;
    inxyfile.open('City _ Coordinates.dat', ios :: in);
    if( ! inxyfile)
    {
        cout<< " Can't open the 'City _ Coordinates' file\n";
        abort ( );
    }
    char ch1, ch2;
    while( ! inxyfile.eof ( ))
    {
        inxyfile.get(ch 1);
        if(chl > = '0') break;
        int i = 0, j = 0;
        x[0] = y[0] = 0;
        while (! inxyfile.eof ( ))
        {
            (inxyfile.get (ch1));
            if(chl > = '0' && chl < = '9')
            {
                ch2 = chl;
                while(ch2 > = '0' && ch2 < = '9')
                {

```

```

        switch(i)
    {
        case 0: break;
        case 1: x[j] = x[j] * 10 + (double(ch2)-48); break;
        case 2: y[j] = y[j] * 10 + (double(ch2)-48); break;
    }
    inxyfile.get (ch2);
}
i = ( ++ i) % 3;
if (i == 0 && j<N-1)
{
    j++;
x[j] = 0;
y[j] = 0;
}
}
}
}
}

```

## A. 2 Matlab 语言版

```

% *****
%
%          Basic Ant Colony Algorithm for TSP
%
%          Matlab Version
%
% *****

% % ***** Initialization phase ***** %

global A0;
global n; % city number
global g;
m = str2double(get(handles.edit_antsum, 'String'));% set ant number by using Matlab GUI
initao = str2double(get(handles.edit_tao, 'String'));
alpha = str2double(get(handles.edit_alpha, 'String'));
beta = str2double(get(handles.edit_beta, 'String'));
Q = str2double(get(handles.edit_q, 'String'));
rou = str2double(get(handles.edit_rou, 'String'));
NCmax = str2double(get(handles.edit_ncmax, 'String'));
A = reshape(A0, 3, n); % get the city coordinates in TSP

```

```

x = A(2, :);      % get X-coordinate
y = A(3, :);      % get Y-coordinate
for i = 1:n
    for j = 1:n
        distance(i, j) = sqrt((x(i)-x(j))*(x(i)-x(j)) + (y(i)-y(j))*(y(i)-y(j)));
    end;
end;
for i = 1: n
    for j = 1: n
        if j~ = i
            tao(i, j) = initao;
            yita(i, j) = 1/distance(i, j);
        end;
    end;
end;
initao = initao.*ones(n, n);
detatao = zeros(n, n);
bestsolution = 1000000000000000;

% *****This is the phase in which ants build their tours***** %

for NC = 1:NCmax
    tabu = zeros(m, n);
    for k = 1:m
        tabu(k, g(NC, k)) = 1;
        maxp(1) = 0;
        for j = 1:n
            if tabu(k, j) == 0
                psum_medium0(1, j) = (tao(g(NC, k), j)^alpha).*(yita(g(NC, k), j)^beta);
            else
                psum_medium0(1, j) = 0;
            end;
        end;
        psum_medium = psum_medium0.';
        psum(k, 1) = sum(psum_medium(:, 1));
        for j = 1:n
            if tabu(k, j) == 0
                p(1, j) = (tao(g(NC, k), j)^alpha).*(yita(g(NC, k), j)^beta)/psum(k, 1);
            else

```

```

p(:, j) = 0;
end;
if maxp(1)<p(1, j)
    maxp(1) = p(1, j);
end;
end;
Linear_index = find(maxp(1) == p(1, :));
size1 = [1, n];
[r_index, c_index] = ind2sub(size1, Linear_index(1));
solution_medium(k, 1) = distance(g(NC, k), c_index(1));
route(k, 1) = c_index(1);
tabu(k, c_index(1)) = 1;
tao(g(NC, k), c_index(1)) = (1-rou)*tao(g(NC, k), c_index(1)) + rou*initao(g(NC, k), c_index(1)); % Local updating for the first iteration
tao(c_index(1), g(NC, k)) = (1-rou)*tao(c_index(1), g(NC, k)) + rou*initao(c_index(1), g(NC, k));
solution(k) = solution_medium(k, 1);
for s = 2:(n-1)
    c_index(s) = 0;
    r_index(s) = 0;
    Linear_index(s) = 0;
    maxp(s) = 0;
    for j = 1:n
        if tabu(k, j) == 0
            psum_medium0(s, j) = (tao(route(k, s-1), j)^alpha).*(yita(route(k, s-1), j)^beta);
        else
            psum_medium0(s, j) = 0;
        end;
    end;
    psum_medium = psum_medium0.';
    psum(k, s) = sum(psum_medium(:, s));
    for j = 1:n
        if tabu(k, j) == 0
            p(s, j) = (tao(route(k, s-1), j)^alpha).*(yita(route(k, s-1), j)^beta)/psum(k, s);
        else
            p(s:(n-1), j) = 0;
        end;
    end;

```

```

        if maxp(s)<p(s, j)
            maxp(s) = p(s, j);
        end;
    end;
    Linear_index = find(maxp(s) == p);
    size2 = [n-1, n];
    [r_index(s), c_index(s)] = ind2sub(size2, Linear_index(1));
    solution_medium(k, s) = distance(c_index(s-1), c_index(s));
    route(k, s) = c_index(s);
    tabu(k, c_index(s)) = 1;
    tao(c_index(s-1), c_index(s)) = (1-rou)*tao(c_index(s-1), c_index(s))
+ rou*initao(c_index(s-1), c_index(s));
    tao(c_index(s), c_index(s-1)) = (1-rou)*tao(c_index(s), c_index(s-1))
+ rou*initao(c_index(s), c_index(s-1));
    solution(k) = solution(k) + solution_medium(k, s);
end;
tao(c_index(n-1), g(NC, k)) = (1-rou).*tao(c_index(n-1), g(NC, k)) + rou.*initao(c_index(n-1), g(NC, k)); % Local updating for other iterations
tao(g(NC, k), c_index(n-1)) = (1-rou).*tao(g(NC, k), c_index(n-1)) + rou.*initao(g(NC, k), c_index(n-1));
solution_medium(k, n) = distance(c_index(n-1), g(NC, k));
solution(k) = solution(k) + solution_medium(k, n);
solution_NC(NC, k) = solution(k);
bestsolution_NC(NC) = solution_NC(NC, 1);
end;
for k = 1:m
    if bestsolution_NC(NC)>solution_NC(NC, k);
        bestsolution_NC(NC) = solution_NC(NC, k);
    end;
end;

```

\* \* \*\*\*\*\*In this phase global updating occurs\*\*\*\*\* \*

```

Linear_index1 = find(bestsolution_NC(NC) == solution_NC(NC, :));
size3 = [NC, m];
[r_index1(NC), c_index1(NC)] = ind2sub(size3, Linear_index1(1));
bestroute_NC(NC, :) = route(c_index1(NC), :);
[aa, bb] = size(Linear_index1);
for i = 1:bb

```

```

[r_index1_tao(NC, i), c_index1_t(NC, i)] = ind2sub(size3, Linear_index1(i));
detatao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i), 1)) = Q/solution(c
_index1_t(NC, i));
detatao(route(c_index1_t(NC, i), 1), g(NC, c_index1_t(NC, i))) = Q/solution(c
_index1_t(NC, i)); tao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i),
1)) = (1-rou).*tao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i),
1)) + rou.*detatao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i),
1)); tao(route(c_index1_t(NC, i), 1), g(NC, c_index1_t(NC, i))) = (1-
rou).*tao(route(c_index1_t(NC, i), 1), g(NC, c_index1_t(NC, i))) + rou.
*detatao(route(c_index1_t(NC, i), 1), g(NC, c_index1_t(NC, i)));
detatao(route(c_index1_t(NC, i), n-1), g(NC, c_index1_t(NC, i))) = Q/solution(c
_index1_t(NC, i));
detatao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i), n-1)) =
Q/solution(c_index1_t(NC, i));
tao(route(c_index1_t(NC, i), n-1), g(NC, c_index1_t(NC, i))) = (1-rou).**
tao(route(c_index1_t(NC, i), n-1), g(NC, c_index1_t(NC, i))) + rou.**
detatao(route(c_index1_t(NC, i), n-1), g(NC, c_index1_t(NC, i)));
tao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i), n-1)) = (1-rou).**
tao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i), n-1)) + rou.**
detatao(g(NC, c_index1_t(NC, i)), route(c_index1_t(NC, i), n-1));

for s = 2:n-1
    detatao(route(c_index1_t(NC, i), s), route(c_index1_t(NC, i), s-1)) = Q/
solution(c_index1_t(NC, i));
    detatao(route(c_index1_t(NC, i), s-1), route(c_index1_t(NC, i), s)) = Q/
solution(c_index1_t(NC, i)); tao(route(c_index1_t(NC, i), s), route(c_
index1_t(NC, i), s-1)) = (1-rou).*tao(route(c_index1_t(NC, i), s), route(c
_index1_t(NC, i), s-1)) + rou.*detatao(route(c_index1_t(NC, i), s), route
(c_index1_t(NC, i), s-1)); tao(route(c_index1_t(NC, i), s-1), route(c_
index1_t(NC, i), s)) = (1-rou).*tao(route(c_index1_t(NC, i), s-1), route(c
_index1_t(NC, i), s)) + rou.*detatao(route(c_index1_t(NC, i), s-1), route
(c_index1_t(NC, i), s));
end;
end;
if bestsolution>bestsolution_NC(NC)
    bestsolution = bestsolution_NC(NC);
end;
Linear_index2 = find(bestsolution == bestsolution_NC);
size4 = [1, NC];

```

```

[r_index2, c_index2] = ind2sub(size4, Linear_index2(1));
bestroute(1, :) = bestroute_NC(c_index2, :);
initao = tao;
end;

% % ***** Output the best result of TSP***** %

for NC = 1:NCmax
    bestroute_NC(NC, n) = g(NC, c_index1(NC));
    t(NC) = NC;
end;
bestroute(1, n) = bestroute_NC(c_index2, n);
plot(x(bestroute(n)), y(bestroute(n)), '*');
hold on;
for i = 1:n
    plot(x(i), y(i), 'o');
    hold on;
end;
hold on;
line([x(bestroute(n)) x(bestroute(1))], [y(bestroute(n)) y(bestroute(1))]);
hold on;
for j = 1:(n-1)
    line([x(bestroute(j)) x(bestroute(j+1))], [y(bestroute(j)) y(bestroute(j+1))]);
    hold on;
end;
hold off;
xlabel('X coordinate');
ylabel('Y coordinate');
title('Best tour in NCmax iterations');

% ****%
% Function: Open and import file of city coordinates in TSP %
% ****%

[fname, pname, filterindex] = uigetfile('*.*', 'All Files (*.*)');
set(handles.text_filename, 'String', filename);
fid = fopen(filename, 'r');
if fid == -1;
    warndlg('Can't open the file', 'WARN');

```

```

fclose(fid);
end;
[A0, COUNT] = fscanf(fid, ' %g');
n = COUNT/3;
set(handles.edit_citysum, 'String', n);
fclose(fid);
m = str2double(get(handles.edit_antsum, 'String'));
NCmax = str2double(get(handles.edit_ncmax, 'String'));
for NC = 1: NCmax
    for k = 1: m
        g(NC, k) = fix((n-1)*rand(1)) + 1;
    end;
end.

```

### A. 3 Visual Basic 语言版

```

'*****
' Basic Ant Colony Algorithm for TSP
' Visual Basic Version
'*****

Public MaxAnts 'Maximum ant number
Public MaxCities 'Maximum city number
Public Alpha As Double
Public Beta As Double
Public Rou As Double
Public Tao0 As Double
Public MaxIter As Integer
Public W As Double
Public Sigma As Double
Public CalcTimes As Double
Public Q0 As Double
Public Type Tour _ Of _ Ant
    fromCity As Integer
    toCity As Integer
    Prob As Double
End Type
Public Type Ant _ ACA
    Tour( ) As Tour _ Of _ Ant

```

```

StartingCity As Integer
CurrentCity As Integer
Visited( ) As Boolean
LengthOfPath As Double
End Type

Public Type City_Type
    x As Double
    y As Double
End Type

Public Ant( ) As Ant_ACA
Public City( ) As City_Type
Public Dis( ) As Double
Public Tao( ) As Double
Public NTao( ) As Boolean
Public SignInitRan As Boolean
Public CityXMax As Double, CityXMin As Double, CityYMax As Double, CityYMin As Double
'*****
' Function: Initialized the basic parameters of ACA
'*****

Public Function Init_ACA( )
    Dim TSPFile As String
    Alpha = Val(frmACA.txtAlpha.Text)
    Beta = Val(frmACA.txtBeta.Text)
    Rou = Val(frmACA.txtRou.Text)
    Tao0 = Val(frmACA.txtTao0.Text)
    Sigma = Val(frmACA.txtSigma)
    W = Val(frmACA.txtW.Text)
    CalcTimes = Val(frmACA.txtCalcTime.Text)
    Q0 = Val(frmACA.txtQ0.Text)
    MaxAnts = Val(frmACA.txtMaxAnts.Text)
    TSPFile = frmACA.lstCityData.Text + ".txt"
    Open TSPFile For Input As #1
    Input #1, MaxCities
    ReDim City(1 To MaxCities)
    ReDim Ant(1 To MaxAnts)
    ReDim Dis(1 To MaxCities, 1 To MaxCities)
    ReDim Tao(1 To MaxCities, 1 To MaxCities)
    ReDim NTao(1 To MaxCities, 1 To MaxCities)
    For i = 1 To MaxAnts

```

```

ReDim Ant(i).Tour(1 To MaxCities)
ReDim Ant(i).Visited(1 To MaxCities)

Next i
For i = 1 To MaxCities
    Input #1, a
    Input #1, City(i).x
    Input #1, City(i).y

Next i
Close #1      'Prepare for init PictureBoxes
CityXMin = City(1).x; CityXMax = City(1).x
CityYMin = City(1).y; CityYMax = City(1).y
For i = 2 To MaxCities
    If City(i).x > CityXMax Then
        CityXMax = City(i).x
    Else
        If City(i).x < CityXMin Then
            CityXMin = City(i).x
        End If
    End If
    If City(i).y > CityYMax Then
        CityYMax = City(i).y
    Else
        If City(i).y < CityYMin Then
            CityYMin = City(i).y
        End If
    End If
End If

Next i
For i = 1 To MaxCities
    For j = 1 To MaxCities
        Tao(i, j) = Tao0
        NTao(i, j) = False
    Next j
Next i
For i = 1 To MaxAnts
    If SignInitRan = True Then
        Ant(i).StartingCity = Int(Rnd * MaxCities) + 1
    Else
        Ant(i).StartingCity = 1
    End If

```

```

Ant(i).CurrentCity = 0
Ant(i).LengthOfPath = 0
For j = 1 To MaxCities
    Ant(i).Tour(j).fromCity = 0
    Ant(i).Tour(j).toCity = 0
Next j
Ant(i).Visited(i) = False
Ant(i).Tour(1).fromCity = Ant(i).StartingCity
Next i
For i = 1 To MaxCities
    For j = 1 To MaxCities
        Dis(i, j) = Sqr((City(i).x-City(j).x)^2 + (City(i).y-City(j).y)^2)
    Next j
Next i
End Function
'*****
' Function: Initialized iteration
'*****'

Public Function Iteration_Init( ) As Integer
    For i = 1 To MaxAnts
        If SignInitRan = True Then
            Ant(i).StartingCity = Int(Rnd * MaxCities) + 1
        Else
            Ant(i).StartingCity = 1
        End If
        Ant(i).CurrentCity = 0
        Ant(i).LengthOfPath = 0
        For j = 1 To MaxCities
            Ant(i).Tour(j).fromCity = 0
            Ant(i).Tour(j).toCity = 0
            Ant(i).Visited(j) = False
        Next j
        Ant(i).Tour(1).fromCity = Ant(i).StartingCity
    Next i
End Function
'*****
' Function: Select the city according to state transition rule
'*****'

Public Function SelectCity(ByVal n As Integer, ByVal NoTour As Integer) As Integer

```

```

Dim STao As Double, P As Double, Sp As Double
Dim STaoMax As Double, ArgSTaoMax As Integer
Randomize Time
P = Rnd
If P <= Q0 Then
    STaoMax = 0
    j = Ant(n).CurrentCity
    For i = 1 To MaxCities
        If Ant(n).Visited(i) = False Then
            If STaoMax < Tao(j, i) Then
                STaoMax = Tao(j, i)
                ArgSTaoMax = i
            End If
        End If
    Next i
    SelectCity = ArgSTaoMax
    Exit Function
End If
STao = 0
j = Ant(n).CurrentCity
For i = 1 To MaxCities
    If Ant(n).Visited(i) = False Then
        STao = STao + (Tao(j, i) ^ Alpha) * ((1 / Dis(j, i)) ^ Beta)
    End If
Next i
If STao = 0 Then
    MsgBox "Error! Travel has been completed, but the ants are still running. STao = 0"
    SelectCity = -1
    Exit Function
End If
Randomize Time
P = Rnd * STao
Sp = 0
For i = 1 To MaxCities
    If Ant(n).Visited(i) = False Then
        Sp = Sp + (Tao(j, i) ^ Alpha) * ((1 / Dis(j, i)) ^ Beta)
        If Sp >= P Then
            SelectCity = i
            Ant(n).Tour(NoTour).Prob = ((Tao(j, i) ^ Alpha) * ((1 / Dis(j, i)) ^ Beta)) / STao
        End If
    End If
Next i
SelectCity = ArgSTaoMax
Exit Function
End If

```

```

        Exit Function
    End If
End If
Next i
MsgBox "Error! STao>Sp"
SelectCity = -1
End Function

'*****
' Function: Pheromone local updating
'*****

Public Function Local_Update(ByVal i As Integer, j As Integer)
    Tao(i, j) = (1-Rou) * Tao(i, j) + Rou * Tao0
    Tao(j, i) = Tao(i, j)
End Function

'*****
' Function: Pheromone global updating
'*****


Public Function PhUpdate(ByVal n As Integer) As Integer
    Dim aa As Double, bb As Double
    For i = 1 To MaxCities
        For j = 1 To MaxCities
            Tao(i, j) = (1-Rou) * Tao(i, j)
            NTao(i, j) = False
            NTao(j, i) = False
            Tao(j, i) = Tao(i, j)
        Next j
    Next i
    For i = 1 To MaxCities
        aa = Ant(n).Tour(i).fromCity
        bb = Ant(n).Tour(i).toCity
        Tao(aa, bb) = Tao(aa, bb) + W / Ant(n).LengthOfPath
        NTao(aa, bb) = True
        NTao(bb, aa) = True
        Tao(bb, aa) = Tao(aa, bb)
    Next i
    PhUpdate = 1
End Function

'*****
' Function: Tour length calculation
'*****
```

```

Public Function CalcLen(ByVal n As Integer) As Double
    Dim aa As Integer, bb As Integer, cc As Double
    For i = 1 To MaxCities
        aa = Ant(n).Tour(i).fromCity
        bb = Ant(n).Tour(i).toCity
        cc = cc + Dis(aa, bb)
    Next i
    CalcLen = cc
End Function

'***** Output the best result of TSP *****'

Public Sub Draw_XOY()
    Dim StepX As Double, StepY As Double
    frmACA.AxisBestLenX.ScaleX      (Val(frmACA.txtBestLenXMax.Text))
    Val(frmACA.txtBestLenXMin.Text))
    frmACA.AxisBestLenY.ScaleY      (Val(frmACA.txtBestLenYMax.Text))
    Val(frmACA.txtBestLenYMin.Text))
    If Val(frmACA.txtBestLenNX.Text) > 0 And Val(frmACA.txtBestLenNY.Text) > 0 Then
        StepX = frmACA.AxisBestLenX.Width/Val(frmACA.txtBestLenNX.Text)
        StepY = frmACA.AxisBestLenY.Height/Val(frmACA.txtBestLenNY.Text)
        For i = 1 To Val(frmACA.txtBestLenNX.Text)-1
            frmACA.AxisBestLenX.Line (StepX * i, 0)-(StepX * i, frmACA.AxisBestLenX.Height)
        Next i
        For i = 1 To Val(frmACA.txtBestLenNY.Text)-1
            frmACA.AxisBestLenY.Line (0, StepY * i)-(frmACA.AxisBestLenY.Width, StepY * i)
        Next i
    End If
End Sub

Public Sub Draw_Best_Graph(ByVal i As Integer, ByVal k As Double)
    If i = 1 Then
        frmACA.picBestLen.PSet (i, k)
    Else
        frmACA.picBestLen.Line-(i, k)
    End If
End Sub

```

## 附录 B 相关网站

<http://www.aco-metaheuristic.org/> : **Ant Colony Optimization**  
[http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html/](http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html) : **Ant Colony Optimization**  
<http://www.metaheuristics.org/> : **Metaheuristics Network Web**  
<http://iridia.ulb.ac.be/~mdorigo/HomePageDorigo/> : **Marco Dorigo**  
<http://www.elet.polimi.it/pagel.do?stu32.values=en&dau1.oid=80&UserCtxParam=0&GroupCtxParam=0&ctx1=it&crc=934028162/> : **Alberto Colomi**  
<http://www.idsia.ch/~luca/> : **Luca Maria Gambardella**  
<http://www3.csr.unibo.it/~maniezzo/> : **Vittorio Maniezzo**  
<http://www.intellektik.informatik.tu-darmstadt.de/~tom/> : **Thomas Stützle**  
<http://homepage.univie.ac.at/walter.gutjahr/> : **Walter Gutjahr**  
<http://www.idsia.ch/~gianni/> : **Gianni Di Caro**  
<http://iridia.ulb.ac.be/~mbiro/> : **Mauro Birattari**  
<http://www.sci.unich.it/%7Earoli/> : **Andrea Roli**  
<http://iridia.ulb.ac.be/~aroli/> : **Andrea Roli**  
<http://www.lsi.upc.es/~cblum/> : **Christian Blum**  
<http://www2.create.human.nagoya-u.ac.jp/~ari/> : **Takaya Arita**  
<http://www.scs.carleton.ca/~arpwhite/> : **Tony White**  
<http://www.cs.up.ac.za/cs/engel/> : **Andries P Engelbrecht**  
<http://www.research.att.com/~dsj/> : **David S. Johnson**  
<http://www.eyckelhof.nl/> : **Casper Joost Eyckelhof**  
[http://www.csail.mit.edu/index.php/](http://www.csail.mit.edu/index.php) : **MIT Computer Science and Artificial Intelligence Laboratory**  
[http://www.fi.uib.no/~antonych/glob.html/](http://www.fi.uib.no/~antonych/glob.html) : **Global Optimization**  
<http://www.swarm-bots.org/> : **Swarm-bots Project**  
<http://iridia.ulb.ac.be/~ants/> : **International Workshops on Ant Algorithms**  
[http://alphard.ethz.ch/hafner/PPS/PPS2001/AntFarm/links.html/](http://alphard.ethz.ch/hafner/PPS/PPS2001/AntFarm/links.html) : **Ant Farm**  
<http://www.agentland.com/> : **Agent Land**  
<http://www.math.wisc.edu/~propp/ant.c/> : **Ant C Program**  
<http://www.cs.cmu.edu/~ACO/> : **ACO Program Home Page**  
[http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html/](http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html) : **ACO Software**  
[http://chern.ie.nthu.edu.tw/Ant\\_Algorithms.htm/](http://chern.ie.nthu.edu.tw/Ant_Algorithms.htm) : **Ant Algorithms Resources**  
<http://dsp.jpl.nasa.gov/members/payman/swarm/> : **Swarm Intelligence**  
<http://www.swarm.org/> : **Swarm Wiki**  
[http://www.sciencenews.org/articles/20001111/bob10.asp/](http://www.sciencenews.org/articles/20001111/bob10.asp) : **Calculating Swarms**

[http://www.densis.fee.unicamp.br/~moscato/TSPLIB\\_home.html](http://www.densis.fee.unicamp.br/~moscato/TSPLIB_home.html) : **TSPLIB Home Page**  
<http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/> : **TSPLIB**  
<http://www.tsp.gatech.edu/> : **TSP Web**  
[Ftp://ftp.zib.de/pub/Packages/mp-testdata/tsp/tsplib/index.html](ftp://ftp.zib.de/pub/Packages/mp-testdata/tsp/tsplib/index.html) : **Mirror of TSPLIB**  
<http://www.tsp.gatech.edu/world/countries.html#CH> : **National TSP**  
<http://www.research.att.com/~dsj/chfsp/atsp.html> : **Asymmetric TSP**  
<http://www.sce.carleton.ca/netmanage/tony/swarm-presentation/sld025.htm> : **Ant Colony Algorithm Slider**  
<http://www.ai.mit.edu/projects/ants/> : **Ant Microrobots**  
<http://www.omnetpp.org/filemgmt/singlefile.php?lid=48/> : **Ant Net Algorithm**  
[http://www.sciencenews.org/pages/sn\\_arc99/1\\_2\\_99/bob2.htm](http://www.sciencenews.org/pages/sn_arc99/1_2_99/bob2.htm) : **Agents of Cooperation**

## 附录 C 基本术语(中英文对照)及缩略语

### B

不动点

fixed-point

不感兴趣弧段

uninteresting edge, UE

并行遗传算法

parallel genetic algorithm, PGA

### C

车间作业调度问题

job-shop scheduling problem, JSP

车辆路径问题

vehicle routing problem, VRP

粗糙数据约简

rough data reduction, RDR

超熵

hyper entropy

差别感觉阈限

contrast sensor threshold, CST

测试弧段

testable edge, TE

操作

operation

查准率

precision

查全率

recall

### D

调度问题

scheduling problem

单机排序

machine scheduling

地雷探测问题

mine detection problem, MDP

点覆盖问题

point covering problem, PCP

多背包问题

multiple knapsack problem, MKP

多态蚁群算法

polymorphic ant colony algorithm, PACA

多目标优化问题

multi-objective optimization problem, MOP

多供货点 VRP

vehicle routing problem with multi-depot

多路划分

multi-way partitioning

带聚类处理的蚁群算法

clustering processing ant colony algorithm,

CPACA

赌轮选择

roulette wheel selection

对称 TSP

symmetric traveling salesman problem

单机总误时排序问题

single machine total tardiness problem,

SMTTP

短期负荷预测

short-term load forecasting, STLF

递归神经网络

recurrent neural network, RNN

### E

二维格模型蛋白质折叠问题

2D HP protein folding problem

### F

非确定性单带图灵机

non-deterministic one-tape Turing machine, NDTM	国防部 Department of Defense, DoD
非对称 TSP asymmetric traveling salesman problem	
非支配排序遗传算法 non-dominated sorting genetic algorithm, NSGA	
非线性 PID nonlinear PID, NLPID	
非线性回归 nonlinear regression, NLR	
飞行仿真转台 flight simulator	
服务质量 quality of service, QoS	
仿生硬件 bio-inspired hardware, BHW	
反向传播 back propagation, BP	
<b>G</b>	
故障识别 fault identification	
规则集 rule set	
归类结构 subsumption architecture	
轨迹平滑机制 trail smoothing mechanism	
广义指派问题 generalized assignment problem, GAP	
广义蚁群算法 generalized ant colony algorithm, GACA	
光纤网络路由 optical networks routing	
光谱解析 spectra analyzing problem, SAP	
工作 job	
<b>H</b>	
环境 environment	
混流装配线 sequencing mixed models on an assembly line, SMMAL	
混沌遗传算法 chaos genetic algorithm, CGA	
灰度 gray value	
呼叫接纳控制 call admission control, CAC	
活跃标志 active flag, AF	
<b>J</b>	
居里夫人杰出成就奖 Marie Curie Excellence Award	
机器人路径规划 robot path planning	
机器 machine	
机组最优投入问题 unit commitment, UC	
几何约束 geometric constraint problem, GCP	
几乎处处 almost surely, A. S.	
基于混合行为的蚁群算法 hybrid behavior based ant colony algorithm, HBACA	
基于距离的聚类算法 distance-based clustering algorithm, DCA	
极性 polar, P	
集合覆盖问题 collection coverage problem	

set covering problem, SCP	离散鞅
加权最小碰集问题	discrete martingale
weighted minimum hitting set problem, WMHSP	邻域
具有感觉和知觉特征的蚁群算法	neighbor
sensational and consciousness algorithm, SCA	<b>M</b>
绝对感觉阈限	密集非递阶 dense heterarchy
absolute sensor threshold, AST	美国航空航天局
解传送单元	National Aeronautics and Space Administra-tion, NASA
solution forwarding unit, SFU	蒙特卡罗
均匀杂交	Monte Carlo, MC
uniform crossover	<b>N</b>
经济负荷分配	NP
economic dispatch, ED	non-deterministic poly-nominal
进化蒙特卡罗	NP-C
evolution Monte Carlo, EMC	NP-complete
静态随机存储器	内在并行性
static random access memory, SRAM	inherent parallelism
<b>K</b>	内含并行性
可编程逻辑器件	implicit parallelism
programmable logic device, PLD	<b>O</b>
可重配置硬件	欧洲人工生命会议
reconfigurable hardware	European Conference on Artificial Life, ECAL
抗体	<b>P</b>
antibody	PID
<b>L</b>	proportional-integral-differential
旅行商问题	频率分配问题
traveling salesman problem, TSP	frequency assignment problem, FAP
连续函数优化	片上系统
continuous function optimization	system-on-a-chip, SoC
连续交互式蚁群算法	配置逻辑模块
continuous interacting ant colony algorithm,	configurable logic block, CLB
CIACA	匹配标志
链路状态路由	match flag, MF
link state-routing, LS	
利用	
exploitation	

## Q

确定性单带图灵机

deterministic one-tape turing machine,

DTM

全球定位系统

global positioning system, GPS

期望值

expected value

强度阈限

intensity threshold, IT

起始点

start point

启发式蚁群算法

heuristic ant colony algorithm, HACA

亲-疏水格点

hydrophobic and polarmonomers, HP

亲和力

affinity

群体-蚁群优化

population-based ant colony optimization,

P-ACO

## R

冗余分配问题

redundancy allocation problem, RAP

人机结合

human-computer cooperation

人机合作

man-machine synergy

人机结合蚁群/遗传算法

human-computer cooperative ant colony/genetic algorithm, HCAGA

人工神经网络

artificial neural network, ANN

人工免疫算法

artificial immune algorithm, AIA

人工鱼

artificial fish

人工鱼群算法

artificial fish-swarm algorithm, AFA

## S

数据挖掘

data mining

疏水性

hydrophobic, H

输入/输出模块

input/output block, IOB

时间相关挥发系数的 GBAS/tdev

GBAS with time-dependent evaporation factor

时间相关信息素下界的 GBAS/tedb

GBAS with time-dependent lower pheromone bound

时间间隔

makespan

时间乘以误差绝对值积分

integrated time absolute error, ITAE

熵

entropy

随机扰动蚁群算法

ant system with random perturbation behavior, RPAS

双路划分

bi-partitioning

## T

图

graph

图形着色问题

graph coloring problem, GCP

图像处理

image processing

图搜索蚂蚁系统

graph-based ant system, GBAS

凸壳问题

convex hull problem

探索	蚁群遗传算法	
exploration	ant colony algorithm genetic algorithm, ACAGA	
通道	有向连接网络路由	
channel	connection-oriented network routing	
梯度	有时间窗车辆路径问题	
gratitude	vehicle routing problem with time window, VRPTW	
<b>W</b>		
无连接网络路由	有向无环图	
connection-less network routing	directed acyclic graph, DAG	
武器目标分配问题	约束满足问题	
weapon-target assignment problem, WTAP	constraint satisfaction problem, CSP	
网格分割问题	岩土工程	
mesh partitioning problem, MPP	geotechnical engineering	
韦伯定律	圆排列问题	
Weber's law	circle permutation problem, CPP	
微粒群算法	遗传算法	
particle swarm optimization, PSO	genetic algorithm, GA	
<b>X</b>		
信息素	遗传算法与蚁群算法动态融合	
pheromone	dynamic combination of genetic algorithm and ant algorithm, DCG3A	
序列排序问题	云模型	
sequential ordering problem, SOP	cloud model	
系统辨识	演化硬件	
system identification	evolvable hardware, EHW	
行为	异步传输模式	
action	asynchronous transfer mode, ATM	
行为控制法	用法参数控制	
behavior-control paradigm, BCP	usage parameter control, UPC	
现场可编程门阵列	<b>Z</b>	
field programmable gate array, FPGA	最短公共超串问题	
	shortest common supersequence	
<b>Y</b>		
蚁群算法	最好弧段	
ant colony algorithm, ACA	best edge, BE	
蚁群系统	最大独立集问题	
ant colony system, ACS	maximum independent set problem, MISP	
	最大最小蚂蚁系统	
	MAX-MIN ant system, MMAS	

最早准备时间	智能体
earliest ready time, ERT	agent
最早开始时间	转台控制
earliest start time, EST	turntable control
最早完成时间	状态空间
earliest completion time, ECT	state space
指派问题	杂交蚁群系统
quadratic assignment problem, QAP	hybrid ant colony system, HACS

## 附录 D (词一首)

### 鹧鸪天·蚁群算法

段海滨

万物精灵奇纤巧，  
鞠躬尽瘁堪杰豪。  
觅食筑穴勤劳作，  
仿生学界蓬碧草。

苦探索，  
历辛劳，  
寻优之径觅隐遥。  
蚁群算法奇葩绽，  
五洲育苗更艳娇。

2005年8月于北航

注释：“鹧鸪天”是词牌名，“蚁群算法”是词的题目。

这首词分上阙和下阙，上阙主要是对蚂蚁的歌颂；下阙主要是对蚁群算法的描述。

词中，“奇纤巧”是对蚂蚁本身技能的真实写照，同时也映射蚁群算法构思之奇特与巧妙；“鞠躬尽瘁”形容蚂蚁辛勤劳动的风格；“蓬碧草”意指目前仿生优化领域对蚁群觅食、筑巢、分工等多种行为研究的蓬勃开展；“觅隐遥”有双重含义：一是指蚁群利用信息素从无序到有序寻找最优路径需要经历艰苦的尝试，另一方面指研究蚁群算法离不开艰辛的探索；“五洲育苗更艳娇”意指蚁群算法创立十余年来，已经吸引了各国学者对其进行了多方面的深入研究和广泛应用，今后这一新兴的仿生优化算法必将更加欣欣向荣！

(该词发表于2005年11月30日《南航报》(总第887期)“长空”副刊)