

Intelligent Optimization
Algorithms with Applications

智能优化算法
及其应用

王 凌 著

Wang Ling



TUP

清华大学出版社



Springer

施普林格出版社

智能优化算法及其应用

王 凌 著

清华大学出版社

施普林格出版社

(京)新登字 158 号

内 容 简 介

优化技术是一种以数学为基础,用于求解各种工程问题优化解的应用技术。本书系统地叙述模拟退火算法、遗传算法、禁忌搜索、神经网络优化算法、混沌优化、混合优化策略等智能优化算法的基本理论和实现技术以及最新进展和应用,并从结构上对算法进行统一描述,着重强调混合策略的开发与应用。

本书可作为与优化技术相关专业的本科生或研究生的教材,也可供研究人员以及工程技术人员参考。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

书 名: 智能优化算法及其应用

作 者: 王 凌 著

出版者: 清华大学出版社 施普林格出版社

北京清华大学学研大厦,邮编 100084

<http://www.tup.tsinghua.edu.cn>

印刷者: 北京昌平环球印刷厂

发行者: 新华书店总店北京发行所

开 本: 787×1092 1/16 印张: 15 字数: 318 千字

版 次: 2001 年 10 月第 1 版 2001 年 10 月第 1 次印刷

书 号: ISBN 7-302-04499-6/TP·2659

印 数: 0001~4000

定 价: 22.00 元

前言

如今,科学技术正处于多学科相互交叉和渗透的时代。特别是,计算机科学与技术的迅速发展,从根本上改变了人类的生产与生活。同时,随着人类生存空间的扩大以及认识与改造世界范围的拓宽,人们对科学技术提出了新的和更高的要求,其中对高效的优化技术和智能计算的要求日益迫切。

优化技术是一种以数学为基础,用于求解各种工程问题优化的应用技术。作为一个重要的科学分支,它一直受到人们的广泛重视,并在诸多工程领域得到迅速推广和应用,如系统控制、人工智能、模式识别、生产调度、VLSI 技术和计算机工程等。鉴于实际工程问题的复杂性、约束性、非线性、多极小、建模困难等特点,寻求一种适合于大规模并行且具有智能特征的算法已成为有关学科的一个主要研究目标和引人注目的研究方向。

20 世纪 80 年代以来,一些新颖的优化算法,如人工神经网络、混沌、遗传算法、进化规划、模拟退火、禁忌搜索及其混合优化策略等,通过模拟或揭示某些自然现象或过程而得到发展,其思想和内容涉及数学、物理学、生物进化、人工智能、神经科学和统计力学等方面,为解决复杂问题提供了新的思路 and 手段。这些算法独特的优点和机制,引起了国内外学者的广泛重视并掀起了该领域的研究热潮,且在诸多领域得到了成功应用。在优化领域,由于这些算法构造的直观性与自然机理,因而通常被称作智能优化算法(intelligent optimization algorithms),或称现代启发式算法(meta-heuristic algorithms)。

目前,智能优化算法的研究成果相当分散,初涉优化领域的研究工作者需花费较多时间和精力才能掌握这方面的优化技术。为推动和指导在校师生在该领域的研究和应用,本书将模拟退火算法、遗传算法、禁忌搜索、神经网络优化算法及其混合优化策略有机集成,强调各类算法的结构统一性和研究系统性,结合国内外研究成果,从理论、技术和应用几方面进行系统的阐述和深入的分析与讨论,并通过各类实例分析指导算法的应用。

本书内容自成体系,由7章组成。第1章为绪论,主要介绍优化问题和优化算法及其分类、计算复杂性和NP等概念。第2,3和4章分别介绍了模拟退火、遗传算法和禁忌搜索的优化流程、机制与特点、收敛性理论、参数选取与实现技术、算法改进等内容。第5章首先介绍了神经网络的发展、模型和BP算法,进而介绍了基于Hopfield网络的优化算法流程、收敛理论和改进途径,以及基于混沌神经网络和混沌序列的优化方法等内容。第6章将各种邻域搜索算法加以统一描述,通过对算法要素和流程的分析与归纳,提出了邻域搜索算法的统一结构和算法性能评价指标,并综述了有关研究进展。第7章阐述了混合优化策略的开发与设计,着重介绍了一类GASA混合优化策略的设计与分析。第8章则介绍了混合优化策略在函数优化、模型辨识、控制器参数整定、TSP,生产调度、神经网络和光学仪器设计等领域的应用。最后,本书在附录中给出了TSP,Job-shop和Flow-shop的若干Benchmark问题。

本书可作为与优化相关专业的师生、研究人员以及工程技术人员的参考书。由于作者水平有限,对于本书的不足之处,诚望读者批评指正。

感谢清华大学自动化系郑大钟教授、王雄教授、赵千川副教授等给予的热心指导和建议,感谢清华大学出版社和王一玲老师等的大力支持,感谢参与研究工作的有关学生。此外,本书的完成得到了国家自然科学基金(69684001、60074012)、国家攀登计划(970211017)、973国家基础研究项目(G1998020310)以及清华大学骨干人才计划等项目的资助,在此表示衷心感谢。

王凌

清华大学自动化系

E-mail: wangling@proc. au. tsinghua. edu. cn

2000年9月于清华园

目 录

第 1 章 绪论	1
1.1 最优化问题及其分类	1
1.1.1 函数优化问题	1
1.1.2 组合优化问题	10
1.2 优化算法及其分类	12
1.3 邻域函数与局部搜索	13
1.4 计算复杂性与 NP 完全问题	14
1.4.1 计算复杂性的基本概念	14
1.4.2 P, NP, NP-C 和 NP-hard	14
第 2 章 模拟退火算法	17
2.1 模拟退火算法	17
2.1.1 物理退火过程和 Metropolis 准则	17
2.1.2 组合优化与物理退火的相似性	18
2.1.3 模拟退火算法的基本思想和步骤	19
2.2 模拟退火算法的马氏链描述	20
2.3 模拟退火算法的收敛性	21
2.3.1 时齐算法的收敛性	21
2.3.2 非时齐算法的收敛性	26
2.3.3 SA 算法渐进性能的逼近	26
2.4 模拟退火算法关键参数和操作的设计	27

2.5	模拟退火算法的改进	29
2.6	并行模拟退火算法	31
2.7	算法实现与应用	32
2.7.1	组合优化问题的求解	32
2.7.2	函数优化问题的求解	33
第3章	遗传算法	36
3.1	遗传算法的基本流程	36
3.2	模式定理和隐含并行性	38
3.3	遗传算法的马氏链描述及其收敛性	40
3.3.1	预备知识	40
3.3.2	标准遗传算法的马氏链描述	41
3.3.3	标准遗传算法的收敛性	42
3.4	一般可测状态空间上遗传算法的收敛性	44
3.4.1	问题描述	45
3.4.2	算法及其马氏链描述	45
3.4.3	收敛性分析和收敛速度估计	45
3.5	算法关键参数与操作的设计	47
3.6	遗传算法的改进	50
3.7	免疫遗传算法	51
3.7.1	引言	51
3.7.2	免疫遗传算法及其收敛性	52
3.7.3	免疫算子的机理与构造	54
3.7.4	TSP 问题的免疫遗传算法	56
3.8	并行遗传算法	58
3.9	算法实现与应用	59
第4章	禁忌搜索算法	62
4.1	禁忌搜索	62
4.1.1	引言	62
4.1.2	禁忌搜索示例	63
4.1.3	禁忌搜索算法流程	67
4.2	禁忌搜索的收敛性	68
4.3	禁忌搜索的关键参数和操作	70
4.4	并行禁忌搜索算法	75

4.5	禁忌搜索的实现与应用	77
4.5.1	基于禁忌搜索的组合优化	77
4.5.2	基于禁忌搜索的函数优化	78
第5章	神经网络与神经网络优化算法	83
5.1	神经网络简介	83
5.1.1	神经网络发展回顾	83
5.1.2	神经网络的模型	84
5.2	基于 Hopfield 反馈网络的优化策略	89
5.2.1	基于 Hopfield 模型优化的一般流程	89
5.2.2	基于 Hopfield 模型优化的缺陷	90
5.2.3	基于 Hopfield 模型优化的改进研究	90
5.3	动态反馈神经网络的稳定性研究	94
5.3.1	动态反馈网络的稳定性分析	94
5.3.1.1	离散对称动态反馈网络的渐近稳定性分析	95
5.3.1.2	非对称动态反馈网络的全局渐近稳定性分析	99
5.3.1.3	时延动态反馈网络的全局渐近稳定性分析	101
5.3.2	动态反馈神经网络的收敛域估计	103
5.4	基于混沌动态的优化研究概述	105
5.4.1	基于混沌神经网络的组合优化概述	106
5.4.2	基于混沌序列的函数优化研究概述	108
5.4.3	混沌优化的发展性研究	109
5.5	一类基于混沌神经网络的优化策略	110
5.5.1	ACNN 模型的描述	110
5.5.2	ACNN 模型的优化机制	111
5.5.3	计算机仿真研究与分析	112
5.5.4	模型参数对算法性能影响的几点结论	116
第6章	广义邻域搜索算法及其统一结构	118
6.1	广义邻域搜索算法	118
6.2	广义邻域搜索算法的要素	119
6.3	广义邻域搜索算法的统一结构	120
6.4	优化算法的性能评价指标	123
6.5	广义邻域搜索算法研究进展	125
6.5.1	理论研究概述	125

6.5.2	应用研究概述	128
6.5.3	发展性研究	129
第7章	混合优化策略	130
7.1	引言	130
7.2	基于统一结构设计混合优化策略的关键问题	131
7.3	一类 GASA 混合优化策略	132
7.3.1	GASA 混合优化策略的构造出发点	132
7.3.2	GASA 混合优化策略的流程和特点	133
7.3.3	GASA 混合优化策略的马氏链描述	135
7.3.4	GASA 混合优化策略的收敛性	136
7.3.5	GASA 混合优化策略的效率定性分析	141
第8章	混合优化策略的应用	143
8.1	基于模拟退火-单纯形算法的函数优化	143
8.1.1	单纯形算法简介	143
8.1.2	SMSA 混合优化策略	144
8.1.3	算法操作与参数设计	145
8.1.4	数值仿真与分析	146
8.2	基于混合策略的控制器参数整定和模型参数估计研究	149
8.2.1	引言	149
8.2.2	模型参数估计和 PID 参数整定	149
8.2.3	混合策略的操作与参数设计	150
8.2.4	数值仿真与分析	151
8.3	基于混合策略的 TSP 优化研究	154
8.3.1	TSP 的混合优化策略设计	154
8.3.2	基于典型算例的仿真研究	156
8.3.3	对 TSP 的进一步讨论	158
8.4	基于混合策略的加工调度研究	159
8.4.1	基于混合策略的 Job-shop 优化研究	159
8.4.1.1	引言	159
8.4.1.2	JSP 的析取图描述和编码	161
8.4.1.3	JSP 的混合优化策略设计	163
8.4.1.4	基于典型算例的仿真研究	166
8.4.2	基于混合策略的置换 Flow-shop 优化研究	170

8.4.2.1	混合优化策略	170
8.4.2.2	算法操作与参数设计	172
8.4.2.3	数值仿真与分析	172
8.4.3	基于混合策略的一类批量可变流水线调度问题的优化研究	174
8.4.3.1	问题描述及其性质	174
8.4.3.2	混合优化策略的设计	175
8.4.3.3	仿真结果和分析	177
8.5	基于混合策略的神经网络权值学习研究	177
8.5.1	BPSA 混合学习策略	178
8.5.2	GASA 混合学习策略	178
8.5.3	GATS 混合学习策略	179
8.5.4	编码和优化操作设计	180
8.5.5	仿真结果与分析	180
8.6	基于混合策略的神经网络结构学习研究	184
8.6.1	RBF 网络简介	184
8.6.2	RBF 网络结构优化的编码和操作设计	184
8.6.3	RBF 网络结构的混合优化策略	186
8.6.4	计算机仿真与分析	187
8.7	基于混合策略的光学仪器设计研究	189
8.7.1	引言	189
8.7.2	模型设计	190
8.7.3	仿真研究和设计结果	191
附录	Benchmark 问题	193
A:	TSP Benchmark 问题	193
B:	置换 Flow-shop Benchmark 问题	195
C:	Job-shop Benchmark 问题	211
参考文献		217

第 1 章

绪 论

优化技术是一种以数学为基础,用于求解各种工程问题优化的应用技术,作为一个重要的科学分支一直受到人们的广泛重视,并在诸多工程领域得到迅速推广和应用,如系统控制、人工智能、模式识别、生产调度、VLSI 技术、计算机工程等等。实现生产过程的最优化,对提高生产效率与效益、节省资源具有重要的作用。同时,优化方法的理论研究对改进算法性能、拓宽算法应用领域、完善算法体系同样具有重要作用。因此,优化理论与算法的研究是一个同时具有理论意义和应用价值的重要课题。

本章首先对优化问题、优化算法及其分类进行介绍,然后介绍计算复杂性、NP 等概念,为后文介绍优化算法及其应用作准备。需要指出的是,最优化分为最小化和最大化,由于两者可以相互转化,因此若不加特殊说明,本书讨论的最优化仅指最小化。

1.1 最优化问题及其分类

优化方法涉及的工程领域很广,问题种类与性质繁多。归纳而言,最优化问题可分为函数优化问题和组合优化问题两大类,其中函数优化的对象是一定区间内的连续变量,而组合优化的对象则是解空间中的离散状态。

1.1.1 函数优化问题

函数优化问题通常可描述为:令 S 为 R^n 上的有界子集(即变量的定义域), $f: S \rightarrow R$

为 n 维实值函数, 所谓函数 f 在 S 域上全局最小化就是寻求点 $X_{\min} \in S$ 使得 $f(X_{\min})$ 在 S 域上全局最小, 即 $\forall X \in S: f(X_{\min}) \leq f(X)$ 。

算法的性能比较通常是基于一些称为 Benchmark 的典型问题展开的。就函数优化问题而言, 目前文献中常用的 Benchmark 问题如下:

(1) Sphere Model

$$f_1(X) = \sum_{i=1}^{30} x_i^2, \quad |x_i| \leq 100$$

其最优状态和最优值为

$$\min(f_1(X^*)) = f_1(0, 0, \dots, 0) = 0$$

(2) Schwefel's Problem 2.22

$$f_2(X) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|, \quad |x_i| \leq 10$$

其最优状态和最优值为

$$\min(f_2(X^*)) = f_2(0, 0, \dots, 0) = 0$$

(3) Schwefel's Problem 1.2

$$f_3(X) = \sum_{i=1}^{30} \left(\sum_{j=1}^i x_j \right)^2, \quad |x_i| \leq 100$$

其最优状态和最优值为

$$\min(f_3(X^*)) = f_3(0, 0, \dots, 0) = 0$$

(4) Schwefel's Problem 2.21

$$f_4(X) = \max_{i=1}^{30} \{|x_i|\}, \quad |x_i| \leq 100$$

其最优状态和最优值为

$$\min(f_4(X^*)) = f_4(0, 0, \dots, 0) = 0$$

(5) Generalized Rosenbrock's Function

$$f_5(X) = \sum_{i=1}^{29} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \quad |x_i| \leq 30$$

其最优状态和最优值为

$$\min(f_5(X^*)) = f_5(1, 1, \dots, 1) = 0$$

(6) Step Function

$$f_6(X) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2, \quad |x_i| \leq 100$$

其最优状态和最优值为

$$\min(f_6(X^*)) = f_6(0, 0, \dots, 0) = 0$$

(7) Quartic Function i. e. Niose

$$f_7(X) = \sum_{i=1}^{30} ix_i^4 + \text{random}[0, 1), \quad |x_i| \leq 1.28$$

其最优状态和最优值为

$$\min(f_7(X^*)) = f_7(0, 0, \dots, 0) = 0$$

(8) Generalized Schwefel's Problem 2.26

$$f_8(X) = - \sum_{i=1}^{30} (x_i \sin(\sqrt{|x_i|})), \quad |x_i| \leq 500$$

其最优状态和最优值为

$$\min(f_8(X^*)) = f_8(420.9687, 420.9687, \dots, 420.9687) = -12569.5$$

(9) Generalized Rastrigin's Function

$$f_9(X) = \sum_{i=1}^{30} [x_i^2 - 10 \cos(2\pi x_i) + 10], \quad |x_i| \leq 5.12$$

其最优状态和最优值为

$$\min(f_9(X^*)) = f_9(0, 0, \dots, 0) = 0$$

(10) Ackley's Function

$$f_{10}(X) = -20 \exp \left[-0.2 \sqrt{\sum_{i=1}^{30} x_i^2 / 30} \right] - \exp \left(\sum_{i=1}^{30} \cos(2\pi x_i) / 30 \right) + 20 + e, \quad |x_i| \leq 32$$

其最优状态和最优值为

$$\min(f_{10}(X^*)) = f_{10}(0, 0, \dots, 0) = 0$$

(11) Generalized Griewank Function

$$f_{11}(X) = \frac{1}{4000} \sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad |x_i| \leq 600$$

其最优状态和最优值为

$$\min(f_{11}(X^*)) = f_{11}(0, 0, \dots, 0) = 0$$

(12) Generalized Penalized Function

$$f_{12}(X) = \frac{\pi}{30} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{29} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_{30} - 1)^2 \right\} + \sum_{i=1}^{30} u(x_i, 10, 100, 4), \quad |x_i| \leq 50$$

其最优状态和最优值为

$$\min(f_{12}(X^*)) = f_{12}(1, 1, \dots, 1) = 0$$

其中,

$$y_i = 1 + (x_i + 1)/4$$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

(13) Generalized Penalized Function

$$f_{13}(X) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{29} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_{30} - 1)^2 \right\} \\ + \sum_{i=1}^{30} u(x_i, 5, 100, 4), \quad |x_i| \leq 50, y_i, u(\cdot) \text{ 同上}$$

其最优状态和最优值为

$$\min(f_{13}(X^*)) = f_{13}(1, 1, \dots, 1) = 0$$

(14) Shekel's Foxholes Function

$$f_{14}(X) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6} \right]^{-1}, \quad |x_i| \leq 65.56$$

其最优状态和最优值为

$$\min(f_{14}(X^*)) = f_{14}(-32, -32) \approx 1$$

其中,

$$(a_{ij}) = \begin{pmatrix} -32, & -16, & 0, & 16, & 32, & -32, & \dots, & 0, & 16, & 32 \\ -32, & -32, & -32, & -32, & -32, & -16, & \dots, & 32, & 32, & 32 \end{pmatrix}$$

(15) Kowalik's Function

$$f_{15}(X) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2, \quad |x_i| \leq 5$$

其最优状态和最优值为

$$\min(f_{15}(X^*)) \approx f_{15}(0.1928, 0.1908, 0.1231, 0.1358) \approx 0.0003075$$

其中,

$$(a_i) = (0.1957, 0.1947, 0.1735, 0.16, 0.0844, 0.0627, 0.0456, \\ 0.0342, 0.0323, 0.0235, 0.0246)$$

$$(1/b_i) = (0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16)$$

(16) Six-Hump Camel-Back Function

$$f_{16}(X) = 4x_1^2 - 2.1x_1^4 + x_1^6/3 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad |x_i| \leq 5$$

其最优状态和最优值为

$$\begin{aligned} \min(f_{16}(X^*)) &= f_{16}(0.08983, -0.7126) \\ &= f_{16}(-0.08983, 0.7126) \\ &= -1.0316285 \end{aligned}$$

(17) Branin Function

$$f_{17}(X) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10, \\ -5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15$$

其最优状态和最优值为

$$\begin{aligned}\min(f_{17}(X^*)) &= f_{17}(-3.142, 2.275) = f_{17}(3.142, 2.275) \\ &= f_{17}(9.425, 2.425) = 0.398\end{aligned}$$

(18) Goldstein-Price Function

$$\begin{aligned}f_{18}(X) &= [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ &\quad \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)], \\ &\quad |x_i| \leq 2\end{aligned}$$

其最优状态和最优值为

$$\min(f_{18}(X^*)) = f_{18}(0, -1) = 3$$

(19) Hartman's Function

$$f_{19}(X) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2\right], \quad 0 \leq x_j \leq 1$$

其中,

$$\begin{aligned}(a_{ij}) &= \begin{bmatrix} 3, & 10, & 30 \\ 0.1, & 10, & 35 \\ 3, & 10, & 30 \\ 0.1, & 10, & 35 \end{bmatrix}, \quad (c_i) = (1, \quad 1.2, \quad 3, \quad 3.2), \\ (p_{ij}) &= \begin{bmatrix} 0.3689, & 0.1170, & 0.2673 \\ 0.4699, & 0.4387, & 0.7470 \\ 0.1091, & 0.8732, & 0.5547 \\ 0.03815, & 0.5743, & 0.8828 \end{bmatrix}\end{aligned}$$

其最优状态和最优值为

$$\min(f_{19}(X^*)) = f_{19}(0.114, 0.556, 0.852) = -3.86$$

(20) Hartman's Function

$$f_{20}(X) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2\right], \quad 0 \leq x_j \leq 1$$

其中,

$$\begin{aligned}(p_{ij}) &= \begin{bmatrix} 0.1312, & 0.1696, & 0.5569, & 0.0124, & 0.8283, & 0.5886 \\ 0.2329, & 0.4135, & 0.8307, & 0.3736, & 0.1004, & 0.9991 \\ 0.2348, & 0.1415, & 0.3522, & 0.2883, & 0.3047, & 0.6650 \\ 0.4047, & 0.8828, & 0.8732, & 0.5743, & 0.1091, & 0.0381 \end{bmatrix}, \quad (c_i) \text{ 同上}, \\ (a_{ij}) &= \begin{bmatrix} 10, & 3, & 17, & 3.5, & 1.7, & 8 \\ 0.05, & 10, & 17, & 0.1, & 8, & 14 \\ 3, & 3.5, & 1.7, & 10, & 17, & 8 \\ 17, & 8, & 0.05, & 10, & 0.1, & 14 \end{bmatrix}\end{aligned}$$

其最优状态和最优值为

$$\min(f_{20}(X^*)) = f_{20}(0.201, 0.15, 0.477, 0.275, 0.311, 0.657) = -3.32$$

(21) Shekel's Family

$$f(X) = - \sum_{i=1}^m [(x - a_i)(x - a_i)^T + c_i]^{-1}, \quad 0 \leq x \leq 10$$

m 分别取 5, 7 和 10 时, $f(X)$ 分别对应 $f_{21}(X)$, $f_{22}(X)$ 和 $f_{23}(X)$, 各参数见表 1.1.1。其最优状态和最优值为 $\min(f(X_{\text{loc-opt}})) \approx f(a_i) \approx 1/c_i, 1 \leq i \leq m$ 。

表 1.1.1 参 数 表

i	a_{ij}				c_i
	$j=1$	2	3	4	
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

此外, J. D. Schaffer 提出的函数

$$f(X) = \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} - 0.5, \quad |x_i| \leq 100$$

也常作为测试函数, 其最优状态和最优值为

$$\min(f(X^*)) = f(0, 0) = -1$$

此函数在距全局最优点大约 3.14 范围内存在无穷多个局部极小将其包围, 并且函数强烈振荡, 因此一般算法难以得到最优解。

鉴于许多工程问题存在约束条件, 受约束函数的优化问题也一直是优化领域关注的主要对象。常用的受约束测试函数包括:

$$(1) \min g_1(X) = 5 \sum_{i=1}^4 (x_i - x_i^2) - \sum_{i=5}^{13} x_i, \text{ 约束条件为}$$

$$2x_1 + 2x_2 + x_{10} + x_{11} \leq 10$$

$$2x_1 + 2x_3 + x_{10} + x_{11} \leq 10$$

$$2x_2 + 2x_3 + x_{11} + x_{12} \leq 10$$

$$-8x_1 + x_{10} \leq 0$$

$$\begin{aligned}
& -8x_2 + x_{11} \leq 0 \\
& -8x_3 + x_{12} \leq 0 \\
& -2x_4 - x_5 + x_{10} \leq 0 \\
& -2x_6 - x_7 + x_{11} \leq 0 \\
& -2x_8 - x_9 + x_{12} \leq 0 \\
& 0 \leq x_i \leq 1, \quad i = 1, 2, \dots, 9, 13 \\
& 0 \leq x_i \leq 100, \quad i = 10, 11, 12
\end{aligned}$$

其全局最优点和最优值为 $g_1(X^*) = g_1(1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1) = 1$ 。

$$(2) \max g_2(X) = \left| \frac{\sum_{i=1}^n \cos^4(x_i) - 2 \prod_{i=1}^n \cos^2(x_i)}{\sqrt{\sum_{i=1}^n i x_i^2}} \right|, \text{约束条件为}$$

$$\prod_{i=1}^n x_i \geq 0.75, \quad \sum_{i=1}^n x_i \leq 7.5n, \quad 0 \leq x_i \leq 1, \quad i = 1, 2, \dots, n$$

其全局最优解未知。

$$(3) \max g_3(X) = (\sqrt{n})^n \prod_{i=1}^n x_i, \text{约束条件为}$$

$$\sum_{i=1}^n x_i^2 = 1, \quad 0 \leq x_i \leq 1, \quad i = 1, 2, \dots, n$$

其全局最优点和最优值为

$$g_3(X^*) = g_3\left(\frac{1}{\sqrt{n}}, \dots, \frac{1}{\sqrt{n}}\right) = 1$$

(4) $\min g_4(X) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$, 约束条件为

$$\begin{aligned}
& 0 \leq 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5 \leq 92 \\
& 90 \leq 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leq 110 \\
& 20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25 \\
& 78 \leq x_1 \leq 102, \quad 33 \leq x_2 \leq 45, \quad 27 \leq x_i \leq 45, \quad i = 3, 4, 5
\end{aligned}$$

其全局最优点和最优值为

$$g_4(X^*) = g_4(78, 33, 29.995, 45, 36.776) = -30665.5$$

(5) $\min g_5(X) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$, 约束条件为

$$\begin{aligned}
& x_4 - x_3 + 0.55 \geq 0, \quad x_3 - x_4 + 0.55 \geq 0 \\
& 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\
& 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\
& 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0
\end{aligned}$$

$$0 \leq x_i \leq 1200, i = 1, 2; \quad -0.55 \leq x_i \leq 0.55, i = 3, 4$$

其已知最优点

$$X^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$$

最优值为

$$g_5(X^*) = 5126.4981$$

$$(6) \min g_6(X) = (x_1 - 10)^3 + (x_2 - 20)^3, \text{约束条件为}$$

$$(x_1 - 5)^2 + (x_2 - 5)^2 \geq 100, 13 \leq x_1 \leq 100, 0 \leq x_2 \leq 100$$

其已知最优点和最优值为

$$g_6(X^*) = g_6(14.095, 0.84296) = -6961.81381$$

$$(7) \min g_7(X) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\ + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 \\ + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,$$

约束条件为

$$\begin{aligned} 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 &\geq 0 \\ -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 &\geq -120 \\ -10x_1 + 8x_2 + 17x_7 - 2x_8 &\geq 0 \\ -x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 &\geq 0 \\ 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 &\geq 0 \\ -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 &\geq 0 \\ 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} &\geq 0 \\ -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 &\geq -30 \\ -10 \leq x_i \leq 10, i = 1, 2, \dots, 10 \end{aligned}$$

其全局最优解为

$$X^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, \\ 1.321644, 9.828726, 8.280092, 8.375927)$$

最优值为

$$g_7(X^*) = 24.3062091$$

$$(8) \max g_8(X) = \frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}, \text{约束条件为}$$

$$x_1^2 - x_2 + 1 \leq 0$$

$$1 - x_1^2 + (x_2 - 4)^2 \leq 0$$

$$0 \leq x_i \leq 10, i = 1, 2$$

其全局最优解在原点附近,如

$$g_8(0.00015, 0.0225) > 1552$$

$$(9) \min \quad g_9(X) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7,$$

约束条件为

$$\begin{aligned} 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 &\geq 0 \\ 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 &\geq 0 \\ 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 &\geq 0 \\ -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 &\geq 0 \\ -10 \leq x_i \leq 10, \quad i = 1, 2, \dots, 7 \end{aligned}$$

其全局最优解为

$$X^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.624487, \\ 1.038131, 1.594227)$$

最优值为

$$g_9(X^*) = 680.6300573$$

$$(10) \min \quad g_{10}(X) = x_1 + x_2 + x_3, \text{约束条件为}$$

$$\begin{aligned} 1 - 0.0025(x_4 + x_6) &\geq 0 \\ 1 - 0.0025(x_5 + x_7 - x_4) &\geq 0 \\ 1 - 0.01(x_8 - x_5) &\geq 0 \\ x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 &\geq 0 \\ x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 &\geq 0 \\ x_3x_8 - 1250000 - x_3x_5 + 2500x_5 &\geq 0 \\ 100 \leq x_1 \leq 10000, \quad 1000 \leq x_i \leq 10000, \quad i = 2, 3, \\ 10 \leq x_i \leq 1000, \quad i = 4, 5, \dots, 8 \end{aligned}$$

其全局最优解为

$$X^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, \\ 217.9799, 286.4162, 395.5979)$$

最优值为

$$g_{10}(X^*) = 7049.330923$$

$$(11) \min \quad g_{11}(X) = x_1^2 + (x_2 - 1)^2, \text{约束条件为}$$

$$\begin{aligned} x_2 - x_1^2 &= 0 \\ -1 \leq x_i \leq 1, \quad i = 1, 2 \end{aligned}$$

其已知最优解和最优值为

$$g_{11}(X^*) = g_{11}(\pm 0.70711, 0.5) = 0.750000455$$

对于受约束问题,除了局部极小解的存在,影响最优化性能的因素主要包括:

(1) 目标函数所对应曲面的拓扑性质,譬如在相同约束下,线性或凸函数比无规律的函数要容易求解。

(2) 可行区域的疏密程度,通常以可行区域占整个搜索空间的比值来度量;同时,约束在可行区域边界上的变化强度与惩罚项的确定也大有关系。

(3) 目标函数在整个搜索空间中整体最优解与可行区域中最优解之比,特别当整体最优解离可行区域很近时将使其对惩罚项非常敏感。

(4) 在最优解处活跃约束的数目,活跃约束数目越多则最优解离可行区域的边界越近。

许多场合将受约束问题

$$\begin{cases} \min & f(X) \\ g(X) & \geq 0 \\ h(X) & = 0 \\ X & \in S \end{cases}$$

转化为无约束问题来处理,常用的途径有:

(1) 把问题的约束在状态的表达形式中体现出来,并设计专门的算子,使状态所表示的解在搜索过程中始终保持可行性。这种方法最直接,但适用领域有限,算子的设计也较困难。

(2) 在编码过程中不考虑约束,而在搜索过程中通过检验解的可行性来决定解的弃用。这种方法一般只适用于简单的约束问题。

(3) 采用惩罚的方法来处理约束越界问题。这种方法比较通用,适当选择惩罚函数的形式可得到较好的结果。譬如采用罚函数

$$\begin{cases} \min & f(X) + \lambda h^2(X) + \beta [\min\{0, g(X)\}]^2 \\ X & \in S \end{cases}$$

因此对函数优化的讨论通常以无约束问题为主。

1.1.2 组合优化问题

组合优化问题通常可描述为:令 $\Omega = \{s_1, s_2, \dots, s_n\}$ 为所有状态构成的解空间, $C(s_i)$ 为状态 s_i 对应的目标函数值,要求寻找最优解 s^* ,使得 $\forall s_i \in \Omega, C(s^*) = \min C(s_i)$ 。组合优化往往涉及排序、分类、筛选等问题,它是运筹学的一个重要分支。

典型的组合优化问题有旅行商问题(traveling salesman problem, TSP)、加工调度问题(scheduling problem, 如 Flow-shop, Job-shop)、0-1 背包问题(knapsack problem)、装箱问题(bin packing problem)、图着色问题(graph coloring problem)、聚类问题(clustering problem)等。

(1) 旅行商问题

给定 n 个城市和两两城市之间的距离, 要求确定一条经过各城市当且仅当一次的最短路线。其图论描述为: 给定图 $G=(V, A)$, 其中 V 为顶点集, A 为各顶点相互连接组成的边集, 已知各顶点间的连接距离, 要求确定一条长度最短的 Hamilton 回路, 即遍历所有顶点当且仅当一次的最短回路。

(2) 加工调度问题

Job-shop 问题是一类较 TSP 更为复杂的典型加工调度问题, 是许多实际问题的简化模型。一个 Job-shop 可描述为: n 个工件在 m 台机器上加工, O_{ij} 表示第 i 个工件在第 j 台机器上的操作, 相应的操作时间 T_{ij} 为已知, 事先给定各工件在各机器上的加工次序(称为技术约束条件), 要求确定与技术约束条件相容的各机器上所有工件的加工次序, 使加工性能指标达到最优。在 Job-shop 问题中, 除技术约束外, 通常还假定每一时刻每台机器只能加工一个工件, 且每个工件只能被一台机器所加工, 同时加工过程为不间断。若各工件的技术约束条件相同, 一个 Job-shop 问题就转化为较简单的 Flow-shop 问题。进而, 若各机器上各工件的加工次序也相同, 则问题可进一步转化为置换 Flow-shop 问题。

(3) 0-1 背包问题

对于 n 个体积分别为 a_i 、价值分别为 c_i 的物品, 如何将它们装入总体积为 b 的背包中, 使得所选物品的总价值最大。

(4) 装箱问题

即如何以个数最少的尺寸为 1 的箱子装入 n 个尺寸不超过 1 的物品。

(5) 图着色问题

对于 n 个顶点的无环图 G , 要求对其各个顶点进行着色, 使得任意两个相邻的顶点都有不同的颜色, 且所用颜色种类最少。

(6) 聚类问题

m 维空间上的 n 个模式 $\{X_i | i=1, 2, \dots, n\}$, 要求聚成 k 类, 使得各类本身内的点最相近, 譬如要求

$$\chi^2 = \sum_{i=1}^n \|X_i^{(p)} - R_p\| \text{ 最小}$$

其中 R_p 为第 p 类的中心, 即

$$R_p = \sum_{i=1}^{n_p} X_i^{(p)} / n_p$$

其中, $p=1, 2, \dots, k, n_p$ 为第 p 类中的点数。

显然, 上述问题描述均非常简单, 并且有很强的工程代表性, 但最优化求解很困难, 其主要原因是所谓的“组合爆炸”。譬如, 聚类问题的可能划分方式有 $k^n/k!$ 个, Job-shop

的可能排列方式有 $(n!)^m$ 个,基于置换排列描述的 n 城市TSP问题有 $n!$ 种可行排列,即使对无方向性和循环性的平面问题仍有 $(n-1)!/2$ 种不同排列,显然状态数量随问题规模呈超指数增长。若计算机每秒能处理1亿种排列,则穷举20城市问题的 $20!$ 种排列约需几百年,如此巨大的计算量是无法承受的,更不用谈更大规模问题的求解。因此,解决这些问题的关键在于寻求有效的优化算法,也正是问题的代表性和复杂性激起了人们对组合优化理论与算法的研究。

针对组合优化,人们也构造了大量用作测试算法性能的Benchmark问题,本书附录给出了TSP,Job-shop和Flow-shop问题的若干常用Benchmark问题,供读者参考。

1.2 优化算法及其分类

所谓优化算法,其实就是一种搜索过程或规则,它是基于某种思想和机制,通过一定的途径或规则来得到满足用户要求的问题的解。

就优化机制与行为而分,目前工程中常用的优化算法主要可分为:经典算法、构造型算法、改进型算法、基于系统动态演化的算法和混合型算法等。

(1) 经典算法。包括线性规划、动态规划、整数规划和分枝定界等运筹学中的传统算法,其算法计算复杂性一般很大,只适于求解小规模问题,在工程中往往不实用。

(2) 构造型算法。用构造的方法快速建立问题的解,通常算法的优化质量差,难以满足工程需要。譬如,调度问题中的典型构造型方法有:Johnson法、Palmer法、Gupta法、CDS法、Daunenbring的快速接近法、NEH法等。

(3) 改进型算法,或称邻域搜索算法。从任一解出发,对其邻域的不断搜索和当前解的替换来实现优化。根据搜索行为,它又可分为局部搜索法和指导性搜索法。

- 局部搜索法。以局部优化策略在当前解的邻域中贪婪搜索,如只接受优于当前解的状态作为下一当前解的爬山法;接受当前解邻域中的最好解作为下一当前解的最陡下降法等。
- 指导性搜索法。利用一些指导规则来指导整个解空间中优良解的探索,如SA、GA、EP、ES和TS等。

(4) 基于系统动态演化的方法。将优化过程转化为系统动态的演化过程,基于系统动态的演化来实现优化,如神经网络和混沌搜索等。

(5) 混合型算法。指上述各算法从结构或操作上相混合而产生的各类算法。

优化算法当然还可以从别的角度进行分类,如确定性算法和不确定性算法,局部优化算法和全局优化算法等。

1.3 邻域函数与局部搜索

邻域函数是优化中的一个重要概念,其作用就是指导如何由一个(组)解来产生一个(组)新的解。邻域函数的设计往往依赖于问题的特性和解的表达方式(编码),这一点我们将在后文结合具体问题进行分析。由于优化状态表征方式的不同,函数优化与组合优化中的邻域函数的具体方式将明显存在差异。

函数优化中邻域函数的概念比较直观,利用距离的概念通过附加扰动来构造邻域函数是最常用的方式,如 $x' = x + \eta\xi$, 其中 x' 为新解, x 为旧解, η 为尺度参数, ξ 为满足某种概率分布的随机数或白噪声或混沌序列或梯度信息等。显然,采用不同的概率分布(如高斯分布、柯西分布、均匀分布等)或下降策略,将实现不同性质的状态转移。

在组合优化中,传统的距离概念显然不再适用,但其基本思想仍旧是通过一个解产生另一个解。下面对邻域函数给出一般性定义,并以 TSP 为例进行解释。

定义 1.3.1 令 (S, F, f) 为一个组合优化问题,其中 S 为所有解构成的状态空间, F 为 S 上的可行域, f 为目标函数,则一个邻域函数可定义为一种映射,即 $N: S \rightarrow 2^S$ 。其涵义是,对于每个解 $i \in S$,一些“邻近” i 的解构成 i 的邻域 $S_i \subset S$,而任意 $j \in S_i$ 称为 i 的邻域解或邻居。通常约定, $j \in S_i \Leftrightarrow i \in S_j$ 。

通常, TSP 问题的解可用置换排列来表示,如排列 $(1, 2, 3, 4)$ 可表示 4 个城市 TSP 的一个解,即旅行顺序为 1, 2, 3, 4。那么, k 个点的交换就可认为是一种邻域函数。譬如,不考虑由解的方向性和循环性引起的重复性,上述排列的 2 点交换对应的邻域函数将产生新解 $(2, 1, 3, 4)$ 、 $(3, 2, 1, 4)$ 、 $(4, 2, 3, 1)$ 、 $(1, 3, 2, 4)$ 、 $(1, 4, 3, 2)$ 、 $(1, 2, 4, 3)$ 。

基于邻域函数的概念,就可以对局部极小和全局最小进行定义。

定义 1.3.2 若 $\forall j \in S_i \cap F$, 满足 $f(j) \geq f(i)$, 则称 i 为 f 在 F 上的局部极小解;若 $\forall j \in F$, 满足 $f(j) \geq f(i)$, 则称 i 为 f 在 F 上的全局最小解。

局部搜索算法是基于贪婪思想利用邻域函数进行搜索的,它通常可描述为:从一个初始解出发,利用邻域函数持续地在当前解的邻域中搜索比它好的解,若能够找到如此的解,就以之成为新的当前解,然后重复上述过程,否则结束搜索过程,并以当前解作为最终解。可见,局部搜索算法尽管具有通用易实现的特点,但搜索性能完全依赖于邻域函数和初始解,邻域函数设计不当或初值选取不合适,则算法最终的性能将会很差。同时,贪婪思想无疑将使算法丧失全局优化的能力,也即算法在搜索过程中无法避免陷入局部极小。因此,若不在搜索策略上进行改进,那么要实现全局优化,局部搜索算法采用的邻域函数必须是“完全的”,即邻域函数将导致解的完全枚举。而这在大多数情况下是无法实现的,而且穷举的方法对于大规模问题在搜索时间上是不允许的。

鉴于局部搜索算法的上述缺点,智能优化算法,如模拟退火算法、遗传算法、禁忌搜

索、神经优化算法和混沌搜索等,从不同的角度利用不同的搜索机制和策略实现对局部搜索算法的改进,来取得较好的全局优化性能。后文将对这些算法作系统性介绍,并以广义邻域搜索算法的结构进行统一化阐述。

1.4 计算复杂性与 NP 完全问题

1.4.1 计算复杂性的基本概念

由于有些优化算法所需的计算时间和存储空间难以承受,因此算法可解的问题在实践中并不一定可解。如对称 TSP 问题,可能路径为 $(n-1)!/2$,若以路径比较为基本操作,则需 $(n-1)!/2-1$ 次基本操作。对于每秒执行一百万次操作的计算机,当 $n=20$ 时就需 1929 年才能找到最优解。所以,我们必须对计算复杂性理论有所了解,这也是最优化的理论基础。只有了解所研究问题的复杂性,才能有针对性地设计算法,进而提高优化效率。

算法的时间和空间复杂性对计算机的求解能力有重大影响。算法对时间和空间的需要量称为算法的时间复杂性和空间复杂性。问题的时间复杂性是指求解该问题的所有算法中时间复杂性最小的算法的时间复杂性,问题的空间复杂性也可类似定义。按照计算复杂性理论研究问题求解的难易程度,可把问题分为 P 类、NP 类和 NP 完全类。

算法或问题的复杂性一般表示为问题规模 n (如 TSP 问题中的城市数) 的函数,时间复杂性记为 $T(n)$,空间复杂性记为 $S(n)$ 。在算法分析和设计中,沿用实用性的复杂性概念,即把求解问题的关键操作,如加、减、乘、比较等运算指定为基本操作,算法执行基本操作的次数则定义为算法的时间复杂性,算法执行期间占用的存储单元则定义为算法的空间复杂性。在分析复杂性时,可以求出算法的复杂性函数 $p(n)$,也可用复杂性函数主要项的阶 $O(p(n))$ 来表示。若算法 A 的时间复杂性为 $T_A(n)=O(p(n))$,且 $p(n)$ 为 n 的多项式函数,则称算法 A 为多项式算法。时间复杂性不属于多项式时间的算法统称为指数时间算法。

1.4.2 P, NP, NP-C 和 NP-hard

P 类问题指具有多项式时间求解算法的问题类。但迄今为止,许多优化问题仍没有找到求得最优解的多项式时间算法,通常称这种比 P 类问题更广泛的问题为非多项式确定问题,即 NP 问题。NP 的概念通常由判定问题引入,下面介绍相应的若干概念。

定义 1.4.1 实例是问题的特殊表现,所谓实例就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,这些数据占有计算机的空间称为实例的输入长度。

定义 1.4.2 若一个问题的每个实例只有“是”或“否”两种回答,则称该问题为判定问题,并称肯定回答的实例为“是”实例,否定回答的实例为“否”实例或非“是”实例。

定义 1.4.3 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”的判定实例 I 都存在一个字符串 S 是 I 的“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,且验证算法验证 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。

由此可见,判定问题是否属于 NP 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。可见, $P \subset NP$ 。

归约和转换是描述问题特性的常用方法。如果能够将几类问题归结为一个问题,则一旦解决了一个归结后的问题,其他几类问题也就解决了。

定义 1.4.4 给定问题 A_1 和 A_2 ,称 A_1 可多项式转换为 A_2 ,若存在一个多项式函数 $g(x)$ 和一个字符串,满足:(1) 对 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。

定义 1.4.5 给定判定问题 A_1 和 A_2 ,称 A_1 可多项式归约为 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 $f_{H_1}(d(I)) \leq g_2(f_{H_2}(g_1(d(I))))$ 。

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 ,可按如下步骤构造 A_1 的算法:

(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 H_2 求解 I_2 ,此步计算时间为 $f_{H_2}(g_1(d(I_1)))$ 。

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 $g_1(d(I_1)) + f_{H_2}(g_1(d(I_1)))$ 。

进一步,我们给出 NP-C(NP-Complete)和 NP-hard 的概念。

定义 1.4.6 称判定问题 $A \in \text{NP-C}$,若 $A \in \text{NP}$ 且 NP 中的任何一个问题可多项式归约为问题 A 。称问题 A 为 NP-hard,只要上述第二个条件成立。

由此可见, $\text{NP-C} \subset \text{NP-hard}$,而两者的区别仅在于 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,若已知问题可多项式归约为新问题,则新问题为 NP-hard,进而若可验证新问题属于 NP 类,则新问题为

NP-C。

上述四类问题的关系可用图 1.4.1 表示。

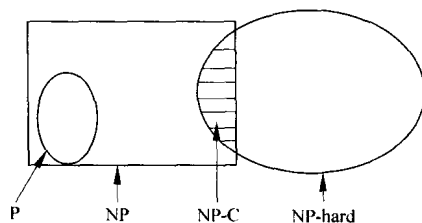


图 1.4.1 四类问题的关系图

NP-C 问题具有重要的实际意义和工程背景，目前已有许多问题被证明为 NP-C，如背包问题、装箱问题、Job-shop 和 Flow-shop 问题、集合覆盖问题、TSP 问题等，有兴趣的读者可阅读 Garey M R 和 Johnson D S 的《Computers and Intractability: A Guide to the Theory of NP-Completeness》一书。这些问题的求解需要兼顾解的质量和求解时间，一方面可以采用简单、直接的策略和规则来构造近似算法；另一方面可设计平均性能良好但在最坏(worst-case)情况下仍然是指数界时间的概率搜索算法。本书后续章节将从理论分析和数值研究等方面对模拟退火、遗传算法、禁忌搜索和神经网络等算法进行介绍，尤其强调算法的实用性技术。

第 2 章

模拟退火算法

模拟退火算法(simulated annealing, 简称 SA)的思想最早是由 Metropolis 等(1953)提出的,1983 年 Kirkpatrick 等将其用于组合优化。SA 算法是基于 Monte Carlo 迭代求解策略的一种随机寻优算法,其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。模拟退火算法在某一初温下,伴随温度参数的不断下降,结合概率突跳特性在解空间中随机寻找目标函数的全局最优解,即在局部优解能概率性地跳出并最终趋于全局最优。模拟退火算法是一种通用的优化算法,目前已在工程中得到了广泛应用,诸如 VLSI、生产调度、控制工程、机器学习、神经网络、图像处理等领域。本章将从优化流程、操作、算法理论与技术等方面对模拟退火算法进行介绍。

2.1 模拟退火算法

模拟退火算法最早是针对组合优化提出的,其目的在于:(1) 为具有 NP 复杂性的问题提供有效的近似求解算法;(2) 克服优化过程陷入局部极小;(3) 克服初值依赖性。模拟退火算法的基本思想出于物理退火过程,因此我们首先简单介绍物理退火过程。

2.1.1 物理退火过程和 Metropolis 准则

简单而言,物理退火过程由以下三部分组成:

(1) 加温过程。其目的是增强粒子的热运动,使其偏离平衡位置。当温度足够高时,

固体将熔解为液体,从而消除系统原先可能存在的非均匀态,使随后进行的冷却过程以某一平衡态为起点。熔解过程与系统的熵增过程相联系,系统能量也随温度的升高而增大。

(2) 等温过程。物理学的知识告诉我们,对于与周围环境交换热量而温度不变的封闭系统,系统状态的自发变化总是朝自由能减少的方向进行,当自由能达到最小时,系统达到平衡态。

(3) 冷却过程。其目的是使粒子的热运动减弱并渐趋有序,系统能量逐渐下降,从而得到低能的晶体结构。

固体在恒定温度下达到热平衡的过程可以用 Monte Carlo 方法加以模拟,虽然该方法简单,但必须大量采样才能得到比较精确的结果,因而计算量很大。鉴于物理系统倾向于能量较低的状态,而热运动又妨碍它准确落到最低态的图像,采样时着重取那些有重要贡献的状态则可较快达到较好的结果。因此,Metropolis 等在 1953 年提出了重要性采样法,即以概率接受新状态。具体而言,在温度 t ,由当前状态 i 产生新状态 j ,两者的能量分别为 E_i 和 E_j ,若 $E_j < E_i$ 则接受新状态 j 为当前状态;否则,若概率 $p_r = \exp[-(E_j - E_i)/kt]$ 大于 $[0, 1]$ 区间内的随机数则仍旧接受新状态 j 为当前状态,若不成立则保留状态 i 为当前状态,其中 k 为 Boltzmann 常数。当这种过程多次重复,即经过大量迁移后,系统将趋于能量较低的平衡态,各状态的概率分布将趋于某种正则分布,如 Gibbs 正则分布。同时,我们也可以看到,这种重要性采样过程在高温下可接受与当前状态能量差较大的新状态,而在低温下基本只接受与当前能量差较小的新状态,这与不同温度下热运动的影响完全一致,而且当温度趋于零时,就不能接受比当前状态能量高的新状态。这种接受准则通常称为 Metropolis 准则,它的计算量相对 Monte Carlo 方法要显著减少。

2.1.2 组合优化与物理退火的相似性

前文已指出,所谓组合优化即寻找最优解 s^* ,使得 $\forall s_i \in \Omega, C(s^*) = \min C(s_i)$,其中 $\Omega = \{s_1, s_2, \dots, s_n\}$ 为所有状态构成的解空间, $C(s_i)$ 为状态 s_i 对应的目标函数值。基于 Metropolis 接受准则的优化过程,可避免搜索过程陷入局部极小,并最终趋于问题的全局最优解,如图 2.1.1 所示。而传统的“瞎子爬山”方法显然做不到这一点,从而也对初值具有依赖性。

因此,基于 Metropolis 接受准则的最优化过程与物理退火过程存在一定的相似性,我们用表 2.1.1

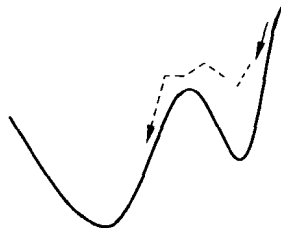


图 2.1.1 基于 Metropolis 接受准则的突跳性搜索

归纳。

表 2.1.1

组合优化	物理退火	组合优化	物理退火
解	粒子状态	Metropolis 抽样过程	等温过程
最优解	能量最低态	控制参数的下降	冷却
设定初温	熔解过程	目标函数	能量

2.1.3 模拟退火算法的基本思想和步骤

1983 年 Kirkpatrick 等意识到组合优化与物理退火的相似性,并受到 Metropolis 准则的启迪,提出了模拟退火算法。归纳而言,SA 算法是基于 Monte Carlo 迭代求解策略的一种随机寻优算法,其出发点是基于物理退火过程与组合优化之间的相似性,SA 由某一较高初温开始,利用具有概率突跳特性的 Metropolis 抽样策略在解空间中进行随机搜索,伴随温度的不断下降重复抽样过程,最终得到问题的全局最优解。

标准模拟退火算法的一般步骤可描述如下:

- (1) 给定初温 $t=t_0$,随机产生初始状态 $s=s_0$,令 $k=0$;
- (2) Repeat:
 - (2.1) Repeat:
 - (2.1.1) 产生新状态 $s_j=Genete(s)$;
 - (2.1.2) if $\min\{1, \exp[-(C(s_j)-C(s))/t_k]\} \geq \text{random}[0,1]$ $s=s_j$;
 - (2.1.3) Until 抽样稳定准则满足;
 - (2.2) 退温 $t_{k+1}=\text{update}(t_k)$ 并令 $k=k+1$;
- (3) Until 算法终止准则满足;
- (4) 输出算法搜索结果。

上述模拟退火算法可用流程框图(如图 2.1.2 所示)直观描述。

从算法结构知,新状态产生函数、新状态接受函数、退温函数、抽样稳定准则和退火结束准则(简称三函数两准则)以及初始温度是直接影响算法优化结果的主要环节。模拟退火算法的实验性能具有质量高、初值鲁棒性强、通用易实现的优点。但是,为寻到最优解,算法通常要求较高的初温、较慢的降温速率、较低的终止温度以及各温度下足够多次的抽样,因而模拟退火算法往往优化过程较长,这也是 SA 算法最大的缺点。因此,在保证一定优化质量的前提下提高算法的搜索效率,是对 SA 进行改进的主要内容。

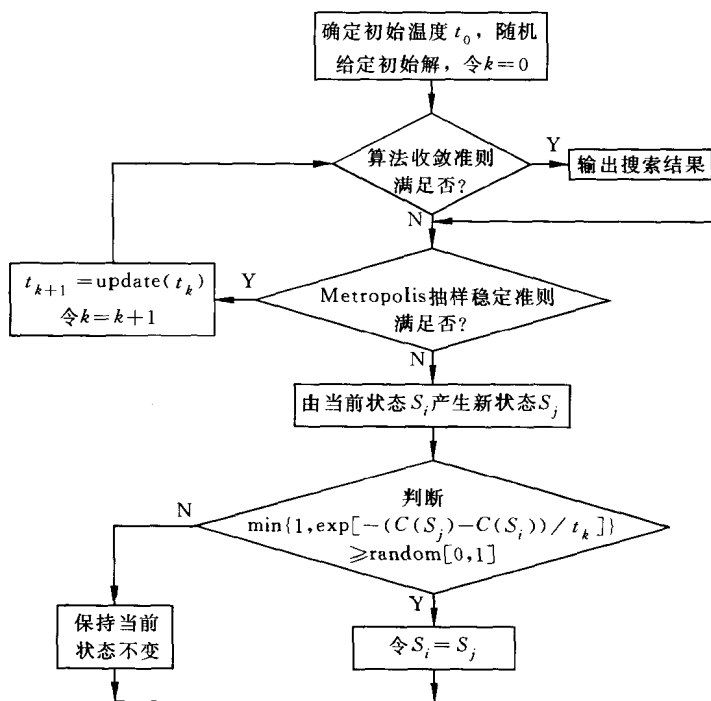


图 2.1.2 标准模拟退火算法的流程图

2.2 模拟退火算法的马氏链描述

令 $\Omega = \{s_1, s_2, \dots\}$ 为所有状态构成的解空间, $X(k)$ 为 k 时刻状态变量的取值。随机序列 $\{X(k)\}$ 称为马氏链, 若 $\forall n \in \mathbb{Z}^+$, 满足

$$\begin{aligned} \Pr\{X(n) = j \mid X(0) = i_0, X(1) = i_1, \dots, X(n-1) = i\} \\ = \Pr\{X(n) = j \mid X(n-1) = i\} \end{aligned}$$

并称

$$p_{i,j}(n-1) = \Pr\{X(n) = j \mid X(n-1) = i\}$$

为一步转移概率, 记 n 步转移概率为

$$p_{i,j}^{(n)} = \Pr\{X(n) = j \mid X(0) = i\}$$

马氏链称为有限状态马氏链, 若解空间有限。马氏链称为时齐的, 若

$$\forall n \in \mathbb{Z}^+, p_{i,j}(n) = p_{i,j}(n-1)$$

我们约定, 以下讨论的 SA 算法对应的马氏链为有限状态马氏链。

考察模拟退火算法的搜索进程, 算法从一个初始状态开始后, 每一步状态转移均是在

当前状态 i 的邻域 N_i 中随机产生新状态 j , 然后以一定概率进行接受的。可见, 接受概率仅依赖于新状态和当前状态, 并由温度加以控制。因此, SA 算法对应了一个马氏链。若固定每一温度 t , 算法均计算马氏链的变化直至平稳分布, 然后下降温度, 则称这种算法为时齐算法。若无需各温度下算法均达到平稳分布, 但温度需按一定的速率下降, 则称这种算法为非时齐法或非平稳马氏链算法。

马氏链可用一个有向图 $G = (V, E)$ 表示, 其中 V 为所有状态构成的顶点集, $E = \{(i, j) | i, j \in V, j \in N_i\}$ 为边集。

记 $g_{i,j}$ 为由状态 i 产生 j 的概率, 则

$$g_{i,j} = \begin{cases} g(i, j)/g(i), & j \in N_i \\ 0, & j \notin N_i \end{cases}$$

其中

$$g(i) = \sum_{j \in N_i} g(i, j)$$

它通常与温度无关。若新状态在当前状态的邻域中以等同概率产生, 则

$$g(i, j)/g(i) = 1/|N_i|$$

其中 $|N_i|$ 为状态 i 的邻域中状态总数。

记 $a_{i,j}$ 为由当前状态 i 接受状态 j 的概率, 接受概率通常定义为

$$a_{i,j} = \min\{1, \exp[-(C(j) - C(i))/t]\}$$

其中 $C(\cdot)$ 为目标函数, t 为温度参数。

记 $p_{i,j}$ 为由状态 i 到状态 j 的转移概率, 则有

$$\forall i, j \quad p_{i,j}(t) = \begin{cases} g_{i,j}a_{i,j}(t), & j \in N_i \text{ and } j \neq i \\ 0, & j \notin N_i \text{ and } j \neq i \\ 1 - \sum_{k \in N_i} p_{i,k}(t), & j = i \end{cases}$$

模拟退火算法要实现全局收敛, 直观上, 它必须满足以下条件: (1) 状态可达性, 即对应马氏链的状态图是强连通的; (2) 初值鲁棒性, 即算法的最终结果不依赖初值; (3) 极限分布的存在性。下面, 我们从理论上对 SA 算法的收敛性进行分析。

2.3 模拟退火算法的收敛性

2.3.1 时齐算法的收敛性

马氏链是描述和分析模拟退火算法的重要数学工具, 在分析时齐算法收敛性之前, 我

们首先给出马氏链理论中的一些相关定义和定理,具体证明过程可参见有关马氏链的书目。

定义 2.3.1 称状态 i 可达状态 j , 若存在 $n \in Z^+$, 使得 $p_{i,j}^{(n)} > 0$, 并记作 $i \rightarrow j$ 。进而, 若 $j \rightarrow i$, 则称状态 i 与 j 相通, 并记作 $i \leftrightarrow j$ 。

显然, 可达性具有递推性质, 即若 $i \rightarrow j$ 且 $j \rightarrow k$, 则 $i \rightarrow k$ 。

进而, 令

$$T_{i,j} = \min\{n \mid X(0) = i, X(n) = j, n \in Z^+\}$$

为状态 i 到达 j 的首达时刻, 其概率定义为

$$f_{i,j}^{(n)} = \Pr\{X(n) = j, X(m) \neq j, m = 1, \dots, n-1 \mid X(0) = i\}$$

则由状态 i 能够到达 j 的概率为

$$f_{i,j} = \sum_{n=1}^{\infty} f_{i,j}^{(n)}$$

显然, $f_{i,j} > 0$ 的充要条件为 $i \rightarrow j$ 。

定义 2.3.2 称状态 i 为常返态, 若 $f_{i,i} = 1$; 称状态 i 为非常返态, 或瞬时态, 若 $f_{i,i} < 1$ 。进而, 当 $f_{i,i} = 1$, 称状态 i 为正常返态, 若 $\mu_i = \sum_{n=1}^{\infty} n f_{i,i}^{(n)} < \infty$; 称状态 i 为零常返态, 若 $\mu_i = \infty$ 。

若从自身出发, 则常返态能够以概率 1 无穷次返回自身, 而非常返态只能有限次返回自身。

定义 2.3.3 若集合 $\{n \mid p_{i,i}^{(n)} > 0, n \in Z^+\}$ 非空, 称此集合的最大公约数 $d(i)$ 为状态 i 的周期。称状态 i 为周期的, 若 $d(i) > 1$; 称状态 i 为非周期的, 若 $d(i) = 1$ 。进而, 称状态 i 为遍历态, 若它为正常返且非周期的。

引理 2.3.1 若 $i \leftrightarrow j$, 则状态 i 与 j 同为周期或非周期的。

定义 2.3.4 称马氏链为不可约的, 若马氏链的所有状态属于同一等价类。不可约链的一个充分条件为, $\forall i, j \in \Omega, \exists n \in Z^+, f_{i,j}^{(n)} > 0$ 。

定义 2.3.5 称 $\{v_j \geq 0, j \in Z^+\}$ 为马氏链的平稳分布, 若一步转移概率满足等式 $v_j = \sum_{i=1}^{\infty} v_i p_{i,j}$ 。

引理 2.3.2 不可约的有限时齐马氏链的状态均为正常返。

引理 2.3.3 非周期不可约的时齐马氏链为正常返的充要条件是存在惟一的平稳分布 $\{v_j \geq 0, j \in Z^+\}$, 此时平稳分布也为极限分布, 并且满足

$$v_j = \sum_{i=1}^{\infty} v_i p_{i,j}^{(n)} = \sum_{i=1}^{\infty} v_i p_{i,j} = \lim_{n \rightarrow \infty} p_{i,j}^{(n)} = 1/\mu_j$$

其中 μ_j 由定义 2.3.2 给出。

定理 2.3.1 若 $\forall i, j \in \Omega, t > 0, a_{i,j}(t) > 0$, 同时 $\exists n \geq 1$, 使得 $s_0, s_1, \dots, s_n \in \Omega, s_0 = i$,

$s_n = j, g_{s_k, s_{k+1}}(t) > 0, k = 0, 1, \dots, n-1$, 其中 $\forall 0 \leq l \neq m \leq n, s_l \neq s_m$, 则时齐 SA 算法对应的有限状态马氏链不可约。

证明 当温度参数 $t > 0$ 给定时, SA 算法对应有限状态的时齐马氏链, 若 $\forall i, j, a_{i,j}(t) > 0$, 则有

$$p_{i,j}^{(n)} \geq p_{s_0, s_1} p_{s_1, s_2} \cdots p_{s_{n-1}, s_n} = g_{s_0, s_1} a_{s_0, s_1} g_{s_1, s_2} a_{s_1, s_2} \cdots g_{s_{n-1}, s_n} a_{s_{n-1}, s_n} > 0$$

因此, $i \rightarrow j$, 进而由状态 i 和 j 的任意性知 $i \leftrightarrow j$ 。从而, 由定义 2.3.4 中不可约链的充分条件可得此定理。

定理 2.3.2 在定理 2.3.1 条件下, 若 $\exists j \neq i \in \Omega$, 使得 $a_{i,j}(t) < 1, g_{i,j}(t) > 0$, 则时齐算法对应的不可约马氏链为非周期的。

证明 考虑状态 i 到自身的转移概率

$$\begin{aligned} p_{i,i}(t) &= 1 - \sum_{l \in \Omega, l \neq i} g_{i,l}(t) a_{i,l}(t) = 1 - \sum_{l \in \Omega, l \neq i, l \neq j} g_{i,l}(t) a_{i,l}(t) - g_{i,j}(t) a_{i,j}(t) \\ &> 1 - \sum_{l \in \Omega, l \neq i, l \neq j} g_{i,l}(t) - g_{i,j}(t) = 1 - \sum_{l \in \Omega, l \neq i} g_{i,l}(t) \geq 0 \end{aligned}$$

因此, $p_{i,i}(t) > 0$, 故状态 i 为非周期。进而, 当定理 2.3.1 条件成立时, 此马氏链不可约, 且 $\forall i, j \in \Omega, i \leftrightarrow j$ 。故由引理 2.3.1 知, 此不可约马氏链为非周期的。

定理 2.3.3 时齐模拟退火算法对应的有限状态马氏链存在平稳分布 $v = (v_1, v_2, \dots, v_{|\Omega|})$, 且

$$\forall i \in \Omega, v_i(t) = \frac{a_{i^*, i}(t)}{\sum_{j \in \Omega} a_{i^*, j}(t)}$$

其中 $i^* \in \Omega_{\text{opt}}$, 若以下条件成立:

- (1) $\forall i, j \in \Omega, g_{i,j}(t)$ 与 t 无关, 且 $g_{i,j} = g_{j,i}$, 同时 $\exists n \geq 1, s_0, s_1, \dots, s_n \in \Omega, s_0 = i, s_n = j$, 使得 $g_{s_k, s_{k+1}}(t) > 0, k = 0, 1, \dots, n-1$;
- (2) $\forall i, j, k \in \Omega$, 若 $C(i) \leq C(j) \leq C(k)$, 则 $a_{i,k}(t) = a_{i,j}(t) a_{j,k}(t)$;
- (3) $\forall i, j \in \Omega, t > 0$, 若 $C(i) \geq C(j)$, 则 $a_{i,j}(t) = 1$; 若 $C(i) < C(j)$, 则 $0 < a_{i,j}(t) < 1$ 。

证明 首先, 若条件(3)成立, 则 $\forall i, j \in \Omega, t > 0, a_{i,j}(t) > 0$ 。进而, 当条件(1)成立时, 定理 2.3.1 的条件满足, 则时齐算法对应的有限状态马氏链不可约。

其次, 若条件(1)和(3)成立, 一定存在两个状态使得定理 2.3.2 条件成立(除非解空间中所有状态均为最优, 此时问题根本没有优化的必要, 因此不属于我们讨论的范畴), 则时齐算法对应的马氏链为非周期的。从而, 根据引理 2.3.3 知, 马氏链存在惟一的平稳分布 $v = (v_1, v_2, \dots, v_{|\Omega|})$ 。

进而, 由于平稳分布是惟一的, 因此要证明此定理下面只需验证定理中 v_i 的定义满足平稳分布的性质, 即

$$v_i(t) = \sum_{j \in \Omega} v_j(t) p_{j,i}(t)$$

推导如下:

$$\begin{aligned}
\sum_{j \in \Omega} v_j p_{j,i}(t) &= \sum_{\substack{j \neq i \\ C(j) \leq C(i)}} \frac{a_{i^*,j}(t)}{\sum_{k \in \Omega} a_{i^*,k}(t)} g_{j,i} a_{j,i}(t) + \sum_{\substack{j \neq i \\ C(j) > C(i)}} \frac{a_{i^*,j}(t)}{\sum_{k \in \Omega} a_{i^*,k}(t)} g_{j,i} a_{j,i}(t) + v_i(t) p_{i,i}(t) \\
&= \sum_{\substack{j \neq i \\ C(j) \leq C(i)}} \frac{a_{i^*,i}(t)}{\sum_{k \in \Omega} a_{i^*,k}(t)} g_{j,i} + \sum_{\substack{j \neq i \\ C(j) > C(i)}} \frac{a_{i^*,j}(t)}{\sum_{k \in \Omega} a_{i^*,k}(t)} g_{j,i} + v_i(t) p_{i,i}(t) \\
&= v_i(t) \sum_{\substack{j \neq i \\ C(j) \leq C(i)}} g_{j,i} + \sum_{\substack{j \neq i \\ C(j) > C(i)}} v_j(t) g_{j,i} \\
&\quad + v_i(t) \left[1 - \sum_{\substack{j \neq i \\ C(j) \leq C(i)}} g_{i,j} a_{i,j}(t) - \sum_{\substack{j \neq i \\ C(j) > C(i)}} g_{i,j} a_{i,j}(t) \right] \\
&= v_i(t) \sum_{\substack{j \neq i \\ C(j) \leq C(i)}} g_{j,i} + \sum_{\substack{j \neq i \\ C(j) > C(i)}} v_j(t) g_{j,i} + v_i(t) - v_i(t) \sum_{\substack{j \neq i \\ C(j) \leq C(i)}} g_{i,j} \\
&\quad - \sum_{\substack{j \neq i \\ C(j) > C(i)}} g_{i,j} v_j(t) = v_i(t)
\end{aligned}$$

注: 通常的接受概率 $a_{i,j} = \min\{1, \exp[-(C(j) - C(i))/t]\}$ 显然满足定理中的条件(2)和(3), 而条件(1)需马氏链对应状态图的强连通性、邻域的对称性(对称性意味着任意两个状态互为邻居或互不为邻居)以及互为邻居的状态的产生概率的等同性来保证。

如果上述定理条件满足, 进而对于状态 i, j , 若当 $C(i) < C(j)$ 时, 式 $\lim_{t \rightarrow 0} a_{i,j}(t) = 0$ 成立, 则考察平稳分布

$$v_i(t) = \frac{a_{i^*,i}(t)}{\sum_{j \in \Omega} a_{i^*,j}(t)}$$

可得

$$\lim_{t \rightarrow 0} v_i(t) = \begin{cases} \frac{1}{|\Omega_{\text{opt}}|}, & i \in \Omega_{\text{opt}} \\ 0, & i \notin \Omega_{\text{opt}} \end{cases}$$

上式说明, 当温度趋于零时, 马氏链以概率 1 收敛到最优状态集, 而收敛到非最优状态的概率为 0。

定理 2.3.4 若定理 2.3.3 中条件(2)和(3)满足, 而条件(1)改为, 马氏链的状态图满足强连通性和邻域的对称性(其中互邻状态的产生概率的等同性无需满足), 且

$$\forall i \in \Omega, g_{i,j}(t) = \begin{cases} \frac{1}{|N_i|}, & j \in N_i \\ 0, & j \notin N_i \end{cases}$$

则马氏链存在平稳分布 $v = (v_1, v_2, \dots, v_{|\Omega|})$, 其中

$$v_i(t) = \frac{|N_i| a_{i^*,i}(t)}{\sum_{j \in \Omega} |N_j| a_{i^*,j}(t)}, \quad i^* \in \Omega_{\text{opt}}$$

证明 首先, 由状态图的强连通性和邻域的对称性、定理 2.3.3 的条件(3)以及引理

2.3.1 知,马氏链为不可约的。

其次,由定理 2.3.3 的条件(3)和修改后的条件(1)知,马氏链为非周期的。

进而,平稳分布存在且是惟一的。下面仅需验证定理中 v_i 的定义满足平稳分布的性质,即

$$v_i(t) = \sum_{j \in \Omega} v_j(t) p_{j,i}(t)$$

推导如下:

$$\begin{aligned} & \sum_{j \in \Omega} v_j p_{j,i}(t) \\ &= \sum_{j \neq i, i \in N_j} \frac{|N_j| a_{i^*,j}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} g_{j,i} a_{j,i}(t) + \sum_{j \in \Omega, i \notin N_j} \frac{|N_j| a_{i^*,j}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} g_{j,i} a_{j,i}(t) \\ & \quad + \frac{|N_i| a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} \left(1 - \sum_{k \in \Omega, k \neq i} g_{i,k} a_{i,k}(t)\right) \\ &= \sum_{j \neq i, i \in N_j} \frac{a_{i^*,j}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} a_{j,i}(t) + \frac{|N_i| a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} \left(1 - \sum_{k \in N_i, k \neq i} g_{i,k} a_{i,k}(t)\right) \\ &= \sum_{j \neq i, i \in N_j} \frac{a_{i^*,j}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} a_{j,i}(t) + \frac{|N_i| a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} - \frac{\sum_{k \in N_i, k \neq i} a_{i,k}(t) a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} \\ &= \sum_{\substack{j \neq i, i \in N_j \\ C(i) \leq C(j)}} \frac{a_{i^*,j}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} + \sum_{\substack{j \neq i, i \in N_j \\ C(i) > C(j)}} \frac{a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} + \frac{|N_i| a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} \\ & \quad - \frac{\sum_{\substack{k \in N_i, k \neq i \\ C(i) \leq C(k)}} a_{i^*,k}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} - \frac{\sum_{\substack{k \in N_i, k \neq i \\ C(i) > C(k)}} a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} \\ &= \frac{|N_i| a_{i^*,i}(t)}{\sum_{k \in \Omega} |N_k| a_{i^*,k}(t)} = v_i(t) \end{aligned}$$

需要指出的是,上述定理仅给出了时齐算法平稳分布存在的充分条件,而非必要条件,因此不能用上述定理的条件去验证平稳分布的不存在性。例如,接受函数

$$a_{i,j}(t) = \left(1 + \exp\left[\frac{C(j) - C(i)}{t}\right]\right)^{-1}$$

显然不满足上述定理的条件,但可以证明,当条件 $\forall i, j \in \Omega, g_{i,j} = g_{j,i}$ 成立时,马氏链的平稳分布仍存在,其分量为

$$v_i(t) = \frac{\exp[-C(i)/t]}{\sum_{j \in \Omega} \exp[-C(j)/t]}$$

2.3.2 非时齐算法的收敛性

区别于时齐算法对各温度下状态序列无限长的条件,非时齐算法通过对退温函数的严格控制以实现收敛目的。由于非时齐 SA 算法是后文讨论的 GASA 混合策略的一种特例,在此仅给出非时齐 SA 算法的一个收敛性定理,具体证明过程以及相关的定义与引理可参见后文混合策略收敛性的相关内容。

定理 2.3.5 令 $S_m = \{i \in V | C(j) \leq C(i), \forall j \in N_i\}$ 为状态空间中目标函数局部极大的点集, $r = \min_{i \in V - S_m} \max_{j \in V} d(i, j)$ 为图 G 的半径,其中 $d(i, j)$ 为图 G 中由顶点 i 到达顶点 j 的最少边数, $L = \max_{i \in V} \max_{j \in N_i} |C(i) - C(j)|$, 若 $t_0 > t_1 > \dots > t_n, \lim_{n \rightarrow \infty} t_n = 0$, 当存在正整数 k_0 使得式 $\sum_{k=k_0}^{\infty} \exp(-rL/t_{k-1}) = \infty$ 成立,则马氏链强遍历,即 SA 算法以概率 1 收敛到问题的全局最优解。

注:为满足上述定理的条件,可设计退温函数 $t_k = \alpha / \ln(k+m), k=0, 1, \dots$, 其中 $2 \leq m < \infty$, 则当 $\alpha \geq rL$ 时, SA 算法以概率 1 收敛到全局最优解。

2.3.3 SA 算法渐进性能的逼近

上述收敛定理表明, SA 算法要以概率 1 实现全局收敛需进行无限次状态转移,在理论上这显然是不实用的。下面,通过介绍 SA 渐进行为的逼近性质来进一步了解 SA 算法。

令 $P(t)$ 为 SA 算法对应时齐马氏链的状态转移矩阵, $v(t)$ 表示 $P(t)$ 的特征值为 1 的左特征向量给定的平稳分布,则研究表明,当 $k \rightarrow \infty$ 时, $\|\alpha(t) - v(t)\| = O(k^s |\lambda_2(t)|^k)$, 其中 $0 < \lambda_2(t) < 1$ 表示 $P(t)$ 的重数为 m_2 的次大特征值, $s = m_2 - 1$ 。可见,算法对平稳分布的逼近速度,或者说算法的收敛速度,取决于 $\lambda_2(t)$ 。然而,对于大规模问题,计算维数很大的 $P(t)$ 的特征值 $\lambda_2(t)$ 很困难,也不切合实际。研究表明,若 $k > K \left[1 + \frac{\ln(\epsilon/2)}{\ln(1-\gamma^*(t))} \right]$, 则 $\|\alpha(t) - v(t)\| < \epsilon$, 其中 $\gamma(t) = \min^+ p_{i,j}(t), K = |\Omega|^2 - 3|\Omega| + 3$ 。这说明,当状态转移次数至少为状态空间规模的平方时,平稳分布才能任意逼近。然而,对于许多大规模问题,状态空间的规模是问题规模的超指数级,因此实现平稳分布的任意逼近需付出指数级时间。

对于非时齐算法,研究表明,对于任意小正数 ϵ ,若退温函数设计为

$$t_k = (1+r)L/\ln(k+m), k=0, 1, \dots, k = O\left[\epsilon^{-\frac{1}{\min(a,b)}}\right]$$

则最优集上的均匀分布能够以任意精度 ϵ 逼近,其中

$$a = \frac{1}{(1+r)[\min_{i,j} g_{i,j}(t)]^{-(1+r)}}, b = \frac{\min_{i \in \Omega_{\text{opt}}} C(i) - C_{\text{opt}}}{(1+r)L}$$

显然,对于大规模问题,这同样会导致指数级的执行时间。譬如,对于 n 城市的 TSP 问题,若采用两点互换邻域结构,则

$$\begin{aligned} r &= n-2 \\ \min_{i,j} g_{i,j} &= 1/[(n-1)(n-2)] \\ a &\approx \frac{1}{n-1} \left[\frac{1}{(n-1)(n-2)} \right]^{n-1} \ll b < \frac{1}{n-1} \end{aligned}$$

若取 $\epsilon=1/n$, 则 $k=O(n^{n^{2n-1}})$, 而 $|\Omega|=O(n!)$, 这意味着 SA 所需花费的时间将大于穷举法。

上述理论研究表明,SA 实现全局收敛的时间性能很差。但是,鉴于许多大规模工程问题的复杂性和 NP 特性,以及工程中要求满意解即可的前提,具有通用特性的模拟退火仍旧是一种有效而实用的优化算法。下面,我们将对 SA 算法的实现技术进行介绍。

2.4 模拟退火算法关键参数和操作的设计

从算法流程上看,模拟退火算法包括三函数两准则,即状态产生函数、状态接受函数、温度更新函数、内循环终止准则和外循环终止准则,这些环节的设计将决定 SA 算法的优化性能。此外,初温的选择对 SA 算法性能也有很大影响。

理论上,SA 算法的参数只有满足算法的收敛条件,才能保证实现的算法依概率 1 收敛到全局最优解。然而,由 SA 算法的收敛性理论知,某些收敛条件无法严格实现,如时齐马氏链的内循环终止准则,即使某些收敛条件可以实现,如非时齐马氏链的更新函数,但也常常会因为实际应用的效果不理想而不被采用。因此,至今 SA 算法的参数选择依然是一个难题,通常只能依据一定的启发式准则或大量的实验加以选取。

1. 状态产生函数

设计状态产生函数(邻域函数)的出发点应该是尽可能保证产生的候选解遍布全部解空间。通常,状态产生函数由两部分组成,即产生候选解的方式和候选解产生的概率分布。前者决定由当前解产生候选解的方式,后者决定在当前解产生的候选解中选择不同状态的概率。候选解的产生方式由问题的性质决定,通常在当前状态的邻域结构内以一定概率方式产生,而邻域函数和概率方式可以多样化设计,其中概率分布可以是均匀分布、正态分布、指数分布、柯西分布等。

2. 状态接受函数

状态接受函数一般以概率的方式给出,不同接受函数的差别主要在于接受概率的形式不同。设计状态接受概率,应该遵循以下原则:

(1) 在固定温度下,接受使目标函数值下降的候选解的概率要大于使目标函数值上升的候选解的概率;

(2) 随温度的下降,接受使目标函数值上升的解的概率要逐渐减小;

(3) 当温度趋于零时,只能接受目标函数值下降的解。

状态接受函数的引入是 SA 算法实现全局搜索的最关键的因素,但实验表明,状态接受函数的具体形式对算法性能的影响不显著。因此,SA 算法中通常采用 $\min[1, \exp(-\Delta C/t)]$ 作为状态接受函数。

3. 初温

初始温度 t_0 、温度更新函数、内循环终止准则和外循环终止准则通常被称为退火历程 (annealing schedule)。

在非时齐 SA 算法收敛性理论中,初温由退温函数 $t_k = \alpha / \log(k + k_0)$ 中的 α 决定,但求解实际问题时很难精确得到 α 值或其下界。

在时齐 SA 算法收敛性理论中,虽没有对初温给出限制,但根据与物理退火过程的类比关系,初温应选择充分大以使几乎所有产生的候选解都能被接受,如此保证最终优良的收敛性。

实验表明,初温越大,获得高质量解的几率越大,但花费的计算时间将增加。因此,初温的确定应折衷考虑优化质量和优化效率,常用方法包括:

(1) 均匀抽样一组状态,以各状态目标值的方差为初温。

(2) 随机产生一组状态,确定两两状态间的最大目标值差 $|\Delta_{\max}|$,然后依据差值,利用一定的函数确定初温。譬如, $t_0 = -\Delta_{\max} / \ln p_r$, 其中 p_r 为初始接受概率。若取 p_r 接近 1, 且初始随机产生的状态能够一定程度上表征整个状态空间时,算法将以几乎等同的概率接受任意状态,完全不受极小解的限制。

(3) 利用经验公式给出。

4. 温度更新函数

温度更新函数,即温度的下降方式,用于在外循环中修改温度值。

在非时齐 SA 算法收敛性理论中,更新函数可采用函数 $t_k = \alpha / \log(k + k_0)$ 。由于温度与退温时间的对数函数成反比,所以温度下降的速度很慢。当 α 取值较大时,温度下降到比较小的值需要很长的计算时间。快速 SA 算法采用更新函数 $t_k = \beta / (1 + k)$, 与前式相比,温度下降速度加快了。但需要强调的是,单纯温度下降速度加快并不能保证算法以较快的速度收敛到全局最优,温度下降的速率必须与状态产生函数相匹配。

在时齐 SA 算法收敛性理论中,要求温度最终趋于零,但对温度的下降速度没有任何限制,但这并不意味着可以使温度下降得很快,因为在收敛条件中要求各温度下产生的候选解数目无穷大,显然这在实际应用时是无法实现的。通常,各温度下产生候选解越多,温度下降的速度可以越快。

目前,最常用的温度更新函数为指数退温,即 $t_{k+1} = \lambda t_k$, 其中 $0 < \lambda < 1$ 且其大小可以

不断变化。

5. 内循环终止准则

内循环终止准则,或称 Metropolis 抽样稳定准则,用于决定在各温度下产生候选解的数目。在非时齐 SA 算法理论中,由于在每个温度下只产生一个或少量候选解,所以不存在选择内循环终止准则的问题。而在时齐 SA 算法理论中,收敛性条件要求在每个温度下产生候选解数目趋于无穷大,以使相应的马氏链达到平稳概率分布,显然在实际应用算法时这是无法实现的。常用的抽样稳定准则包括:

- (1) 检验目标函数的均值是否稳定;
- (2) 连续若干步的目标值变化较小;
- (3) 按一定的步数抽样。

6. 外循环终止准则

外循环终止准则,即算法终止准则,用于决定算法何时结束。设置温度终值 t_c 是一种简单的方法。SA 算法的收敛性理论中要求 t_c 趋于零,这显然是不实际的。通常的做法包括:

- (1) 设置终止温度的阈值;
- (2) 设置外循环迭代次数;
- (3) 算法搜索到的最优值连续若干步保持不变;
- (4) 检验系统熵是否稳定。

由于算法的一些环节无法在实际设计算法时实现,因此 SA 算法往往得不到全局最优解,或算法结果存在波动性。许多学者试图给出选择“最佳”SA 算法参数的理论依据,但所得结论与实际应用还有一定距离,特别是对连续变量函数的优化问题。目前,SA 算法参数的选择仍依赖于一些启发式准则和待求问题的性质。SA 算法的通用性很强,算法易于实现,但要真正取得质量和可靠性高、初值鲁棒性好的效果,克服计算时间较长、效率较低的缺点,并适用于规模较大的问题,尚需进行大量的研究工作。

2.5 模拟退火算法的改进

在确保一定要求的优化质量基础上,提高模拟退火算法的搜索效率(时间性能),是对 SA 算法进行改进的主要内容。可行的方案包括:

- (1) 设计合适的状态产生函数,使其根据搜索进程的需要表现出状态的全空间分散性或局部区域性。
- (2) 设计高效的退火历程。
- (3) 避免状态的迂回搜索。
- (4) 采用并行搜索结构。

(5) 为避免陷入局部极小,改进对温度的控制方式。

(6) 选择合适的初始状态。

(7) 设计合适的算法终止准则。

此外,对模拟退火算法的改进,也可通过增加某些环节而实现。主要的改进方式包括:

(1) 增加升温或重升温过程。在算法进程的适当时机,将温度适当提高,从而可激活各状态的接受概率,以调整搜索进程中的当前状态,避免算法在局部极小解处停滞不前。

(2) 增加记忆功能。为避免搜索过程中由于执行概率接受环节而遗失当前遇到的最优解,可通过增加存储环节,将“Best So Far”的状态记忆下来。

(3) 增加补充搜索过程。即在退火过程结束后,以搜索到的最优解为初始状态,再次执行模拟退火过程或局部趋化性搜索。

(4) 对每一当前状态,采用多次搜索策略,以概率接受区域内的最优状态,而非标准SA的单次比较方式。

(5) 结合其他搜索机制的算法,如遗传算法、混沌搜索等。

(6) 上述各方法的综合应用。

下面介绍一种对退火过程和抽样过程进行修改的两阶段改进策略。

熟知,模拟退火算法在局部极小解处有机会跳出并最终趋于全局最优的根本原因是算法通过概率判断来接受新状态,这在理论上也已得到严格证明,即当初温充分高、降温足够慢、每一温度下抽样足够长、最终温度趋于零时,算法最终以概率1收敛到全局最优解。但由于全局收敛条件难以实现,并且“概率接受”使得当前状态可能比搜索轨迹中的某些中间状态要差,从而实际算法往往最终得到近似最优解,甚至可能比中间经历的最好解差,而且搜索效率较差。

为了不遗失“Best So Far”的状态,并提高搜索效率,改进的做法是:在算法搜索过程中保留中间最优解,并即时更新;设置双阈值使得在尽量保持最优性的前提下减少计算量,即在各温度下当前状态连续 step1 步保持不变则认为 Metropolis 抽样稳定,若连续 step2 次退温过程中所得的最优解均不变则认为算法收敛。

改进算法由改进退火过程和改进抽样过程两部分组成,具体步骤可描述如下:

1. 改进的退火过程

(1) 给定初温 t_0 ,随机产生初始状态 s ,令初始最优解 $s^* = s$,当前状态为 $s(0) = s, i = p = 0$ 。

(2) 令 $t = t_i$,以 t, s^* 和 $s(i)$ 调用下文改进的抽样过程,返回其所得最优解 $s^{*'}$ 和当前状态 $s'(k)$,令当前状态 $s(i) = s'(k)$ 。

(3) 判断 $C(s^*) < C(s^{*'})$? 若是,则令 $p = p + 1$;否则,令 $s^* = s^{*'}$, $p = 0$ 。

(4) 退温 $t_{i+1} = \text{update}(t_i)$,令 $i = i + 1$ 。

(5) 判断 $p > \text{step2}$? 若是,则转第6步;否则,返回第2步。

(6) 以最优解 s^* 作为最终解输出, 停止算法。

2. 改进的抽样过程

(1) 令 $k=0$ 时的初始当前状态为 $s'(0)=s(i)$, 初始最优解 $s^{*'}=s^*$, $q=0$ 。

(2) 由状态 s 通过状态产生函数产生新状态 s' , 计算增量 $\Delta C'=C(s')-C(s)$ 。

(3) 若 $\Delta C'<0$, 则接受 s' 作为当前解, 并判断 $C(s^{*'})>C(s')$? 若是, 则令 $s^{*'}=s'$, $q=0$; 否则, 令 $q=q+1$ 。若 $\Delta C'>0$, 则以概率 $\exp(-\Delta C'/t)$ 接受 s' 作为下一当前状态。若 s' 被接受, 则令 $s'(k+1)=s'$, $q=q+1$; 否则, 令 $s'(k+1)=s'(k)$ 。

(4) 令 $k=k+1$, 判断 $q>\text{step1}$? 若是, 则转第 5 步; 否则, 返回第 2 步。

(5) 将当前最优解 $s^{*'}$ 和当前状态 $s'(k)$ 返回到的改进退火过程。

2.6 并行模拟退火算法

基于并行计算和分布式计算技术的发展, 并行算法的设计已成为算法研究的重要内容。目前, 并行算法的设计主要采用如下两种策略:

- (1) 修改现有串行算法的结构;
- (2) 针对并行计算机的结构特点, 直接设计并程序。

通常, 并行算法的设计需要考虑到存储区分配、同步处理、数据集成与通信等环节。就模拟退火算法而言, 由于算法初始和结束阶段与整个算法进程具有一定的独立性, 抽样过程与退火过程也具有一定的独立性, 因此, 模拟退火算法比较容易实现其并行化方式。直观且可行的方案包括:

1. 操作并行性

所谓操作并行性, 就是将整个算法的各个执行环节分别分配给不同的处理机去完成, 如将状态产生函数、目标值计算、准则判断等指定给不同的处理机执行。由于各环节是串行执行的, 因此各处理机必须按确定的方式执行, 这无疑使整个搜索进程受到很大的限制, 且需花费大量通信时间。

2. 进程并行性

进程并行性包括全过程并行性和子进程并行性两种方式。

所谓全过程并行性, 就是算法首先产生一组初始状态, 然后将各状态发送给不同的处理机, 各处理机独立地进行整个模拟退火搜索过程, 最后经汇总比较得到最终结果。显然, 这仅仅是利用空间资源来弥补单机串行搜索的不足, 而非真正的并行方式。

所谓子进程并行性, 就是由多个处理机同时独立地执行算法的某些进程经综合后继续执行算法的其他环节。譬如, 多个处理机分别对当前状态执行抽样过程, 当所有抽样过程结束后, 综合得到新的当前状态。其中, 各处理机可采用不同的状态产生函数、接受函数, 甚至不同的控制参数, 从而使整个系统的灵活性增强, 充分发挥各处理机的作用, 实现

并行策略的优越性。

3. 空间并行性

所谓空间并行性,就是将整个搜索空间分解成若干个子区域,各子区域分别由不同的处理机执行 SA 的搜索过程,最终综合得到原问题的优化结果。由于各处理机的搜索空间缩小了,对各子问题的搜索效率和可靠性可得以提高,从而可改善对原问题的优化质量与效率。然而,当问题不适合分解或分解不当时,子问题的独立优化将难以反映问题的整体特性。

2.7 算法实现与应用

由于后文将着重介绍由 SA 构造的混合策略的应用,本节对 SA 的应用仅作简单阐述。

2.7.1 组合优化问题的求解

本小节将以 TSP 为例介绍模拟退火算法求解组合优化问题的一个实现方案。

TSP 是典型的 NP 完全问题,求解非常困难。基于 TSP 的问题特性,构造型算法成为最先开发的求解算法,如最近邻点、最近合并、最近插入、最远插入、最近添加、贪婪插入和极小代数法(徐心和, 1990)等。但是,由于构造型算法优化质量较差,迄今为止已开发了许多性能较好的改进型搜索算法,如 n -opt 法,SA,GA,TS,EP 和神经网络等。为了进一步提高算法的全局优化能力,避免搜索过程陷入局部极小,现已提出的改进策略主要有:并行多邻域搜索,平滑优化曲面形状,引进重升温、熵抽样等高级技术等。下面介绍 SA 求解 TSP 的一个方案,具体仿真结果参见第 8 章 GASA 的应用。

1. 编码选择

TSP 的编码策略主要有近邻编码、次序编码、二进制编码、矩阵编码、边编码和路径编码等。近邻编码必须考虑解的合法性;次序编码不利于全局优化;二进制编码不自然,且需附加修正算子来保证解的合法性;矩阵编码存储量很大,且影响遗传算子的优化效率;边编码处理复杂。基此,路径编码是描述 TSP 解的最常用的一种策略。

所谓路径编码,即直接采用城市在路径中的位置来构造用于优化的状态,如路径 5-4-1-7-9-8-6-2-3 对应的路径编码为(5 4 1 7 9 8 6 2 3)。这种编码形式自然直观,易于加入启发式信息,也有利于优化操作的设计。

2. SA 状态产生函数的设计

对于基于路径编码的 SA 状态产生函数操作,可将其设计为:

- (1) 互换操作(SWAP),即随机交换染色体中两不同基因的位置;

(2) 逆序操作(INV),即将染色体中两不同随机位置间的基因串逆序;

(3) 插入操作(INS),即随机选择某个点插入到串中的不同随机位置。

譬如状态为(5 4 1 7 9 8 6 2 3),两随机位置为 2,6,则 SWAP 的结果为(5 8 1 7 9 4 6 2 3),INV 的结果为(5 8 9 7 1 4 6 2 3),INS 的结果为(5 8 4 1 7 9 6 2 3)。

对于对称 TSP,三种算子对串的影响可归纳为以下三种情况:

(1) 若所选择的两位置相邻,则三种算子均改变三条边;

(2) 若所选择的两位置间隔一个点,则 INS 改变三条边,SWAP 和 INV 改变两条边;

(3) 若所选择的两位置间隔两个或两个以上的点,则 SWAP 改变四条边,INV 改变两条边,INS 改变三条边。

由于表示 TSP 解的串很长,若算子中的随机性满足均匀分布,则上述第三种情况发生的概率最大。此时,SWAP 算子更有利于算法的大范围探索,INV 算子则更有利于算法的小范围迁移。

3. SA 状态接受函数的设计

状态接受函数是算法产生概率突跳能力的关键,能够在分布机制指导下避免局部极小。结合基于 SWAP 算子的状态产生函数,为了使搜索过程具有克服陷入局部极小的能力,并满足 SA 算法的对称条件, $\min\{1, \exp(-\Delta/t)\} > \text{random}[0,1]$ 准则是作为接受新状态的条件最常用方案,其中 Δ 为新旧状态的目标值差, t 为“温度”。

4. 初温和初始状态

最常用且可理解的初温确定方案是,首先随机产生一组状态,确定两两状态间的最大目标值差 $|\Delta_{\max}|$,然后由式 $t_0 = -\Delta_{\max}/\ln p_r$,其中 p_r 为初始接受概率(理论上应接近于 1,实际设计时也可取 0.1)。

理论上,初始状态可以随机取,但为了提高优化效率,不妨可采用启发式算法(如 2opt 方法)快速得到一个解,并以此为 SA 的初始状态。

5. 退温函数的设计

理论上,温度应以很慢的速度下降,如对数的倒数方式,但为了避免过于冗长的搜索过程,较好地折衷兼顾优化质量和时间性能,指数退温函数是最常用的退温策略,即 $t_k = \lambda t_{k-1}$, λ 为退温速率。

6. 温度修改准则和算法终止准则的设计

为适应算法性能的动态变化,较好地兼顾算法的优化性能和时间性能,可采用阈值法设计的“温度修改”和“算法终止”两准则。也即,若优化过程中得到的最佳优化值连续 20 代进化保持不变,则进行退温;若最佳优化值连续 20 次退温仍保持不变,则终止搜索过程,以此优化值为算法的优化结果。

2.7.2 函数优化问题的求解

利用模拟退火算法求解函数优化问题与求解组合优化问题的区别主要在于编码和状

态产生函数的不同,其他操作基本类似。

为了避免二进制编码精度受串长限制和存储大的缺点,函数优化中最常用编码方案的是实数编码,即以实数来表示求解状态。

至于状态产生函数,函数优化最常用的方案为 $x(k+1)=x(k)+\eta\xi$,其中 η 为扰动幅度参数, ξ 为随机扰动变量。

随机扰动可服从如下柯西、高斯、均匀分布,甚至是混沌机制的,并将各方案相应地称为 SGC,SGG,SGU 和 SGK。

(1) 柯西分布:概率密度函数为

$$f(\xi) = \frac{1}{\pi} \frac{a}{a^2 + \xi^2}, \quad -\infty < \xi < \infty, \quad a \text{ 为尺度参数}$$

(2) 高斯分布:概率密度函数为

$$f(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp[-\xi^2/(2\sigma)], \quad -\infty < \xi < \infty, \quad \sigma \text{ 为方差}$$

(3) 均匀分布:区间 $[a, b]$ 上均匀分布的概率密度函数为

$$f(\xi) = \begin{cases} 1/(b-a), & \xi \in [a, b] \\ 0, & \text{else} \end{cases}$$

(4) 混沌机制:采用 Logistic 映射 $x_{k+1}=4x_k(1-x_k)$, $x_0 \in (0, 1)$ 。

其中,以原点为中心的柯西和高斯分布函数曲线如图 2.7.1 所示。

由概率密度函数和分布机制知:

(1) 柯西分布在原点处的峰值比高斯分布小,而两端长扁平形状趋于零的速度很慢,因此基于高斯分布的邻域函数产生小扰动的概率较大而产生大扰动的概率几乎为零,基于柯西分布的邻域函数产生小扰动的能力相对高斯分布有所下降而产生大扰动的能力增强;

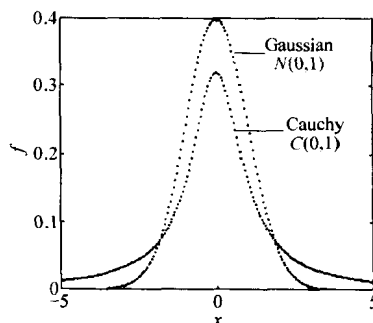


图 2.7.1 柯西和高斯分布概率密度函数

(2) 均匀分布产生一定区域内的任意幅度的

扰动的概率均等,鉴于高斯分布在 ± 5 之外的概率几乎为零,因此可将均匀分布的区间限于 $[-5, 5]$;

(3) 混沌机制在 $(0, 1)$ 范围内按自身规律以一种不重复方式进行轨道遍历,可通过尺度变换将其遍历范围扩充到 $[-5, 5]$ 。

数值仿真研究表明(王凌等, 2000):

(1) 对于单极小且不含平坦区的问题:SAG 的优化效率最差,但 SAG 的最终优化度优于 SAC 和 SAU;在优化开始阶段 SAC 的优化效率和优化度最好。

(2) 对于存在平坦区的问题:SAC 的优化效率最高,其次是 SAU 和 SAG;而 SAK 显

然不适合于求解这类问题,其优化度必须通过加大混沌遍历范围和搜索时间来加以提高。

(3) 对于存在多极小的问题:SAC 的优化度和可靠性最好,然后是 SAK 和 SAU,而 SAG 最差;SAU,SAK 和 SAG 的优化效率相当,而 SAC 的优化效率最低(这是为克服陷入局部极小而付出的时间代价)。

(4) 决定邻域函数性能的关键不是问题的维数,而是函数的形状和局部极小解的数量。

可见,柯西分布的局部搜索性能相对其他邻域函数是可接受的,而其两端较慢趋于零的长扁平形状使其最容易产生大步长扰动,结合 SA 的接受函数使得算法的全局优化度和可靠性最好。此外,就 SAC 而言,最优的分布尺度参数是与问题相关的,它依赖于当前解和全局最优解的距离。由于实际问题的最优解往往未知,因此最优尺度参数也难以确定。为提高优化度,尺度参数应该采用自适应修正的策略。上述研究表明,概率密度函数的形状决定了算法的搜索行为和性能。同时,图 2.7.2 表明:尺度参数变大,则算法产生大步长移动的概率加大,有利于大范围粗搜索,克服局部极小和平坦区的影响;反之,尺度参数变小,则算法产生局部小扰动的概率加大,有利于进行局部趋化性细搜索。极端情况是:尺度参数趋于无穷大将使分布近似为均匀分布,趋于无穷小将使分布近似为冲击函数。因此,可采用“基于退温策略”的尺度修正方案,譬如 $a_k = 2 \times (0.99)^k$ 来改善优化性能(王凌等,2000)。总而言之,鉴于实际问题的复杂性,我们认为 SGC 最具有应用前景。

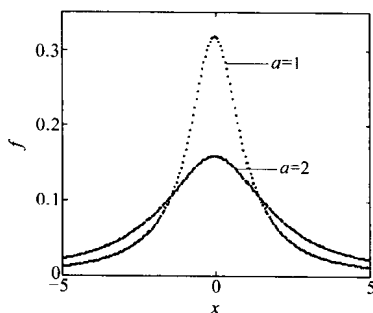


图 2.7.2 不同尺度参数下柯西分布概率密度函数

以上仅对算法的操作作简单设计,有兴趣的读者可利用附录给出的 Benchmark 问题作仿真研究来体会模拟退火算法的性能。此外,后文在介绍混合算法应用时将给出具体的仿真结果,包括与模拟退火算法的比较结果。

第 3 章

遗传算法

近年来,随着人工智能应用领域的不断扩大,传统的基于符号处理机制的人工智能方法在知识表示、信息处理和解决组合爆炸等方面所遇到的困难越来越明显,从而使得寻求一种适合于大规模问题并具有自组织、自适应、自学习能力的算法成为有关学科的一个研究目标。大自然为我们解决各种问题创造了灵感,遗传算法(genetic algorithms,简称 GA)就是 J. Holland 于 1975 年受生物进化论的启发而提出的。GA 是基于“适者生存”的一种高度并行、随机和自适应的优化算法,它将问题的求解表示成“染色体”的适者生存过程,通过“染色体”群的一代代不断进化,包括复制、交叉和变异等操作,最终收敛到“最适应环境”的个体,从而求得问题的最优解或满意解。GA 是一种通用的优化算法,其编码技术和遗传操作比较简单,优化不受限制性条件的约束,而其两个最显著特点则是隐含并行性和全局解空间搜索。目前,随着计算机技术的发展,GA 愈来愈得到人们的重视,并在机器学习、模式识别、图像处理、神经网络、优化控制、组合优化、VLSI 设计、遗传学等领域得到了成功应用。本章将分别对遗传算法的优化流程、操作、算法理论与技术作介绍。

3.1 遗传算法的基本流程

遗传算法是一类随机优化算法,但它不是简单的随机比较搜索,而是通过对染色体的评价和对染色体中基因的作用,有效地利用已有信息来指导搜索有希望改善优化质量的

状态。标准遗传算法的主要步骤可描述如下：

- (1) 随机产生一组初始个体构成初始种群,并评价每一个体的适配值(fitness value)。
- (2) 判断算法收敛准则是否满足。若满足则输出搜索结果;否则执行以下步骤。
- (3) 根据适配值大小以一定方式执行复制操作。
- (4) 按交叉概率 p_c 执行交叉操作。
- (5) 按变异概率 p_m 执行变异操作。
- (6) 返回步骤(2)。

上述算法中,适配值是对染色体(个体)进行评价的一种指标,是 GA 进行优化所用的主要信息,它与个体的目标值存在一种对应关系;复制操作通常采用比例复制,即复制概率正比于个体的适配值,如此意味着适配值高的个体在下一代中复制自身的概率大,从而提高了种群的平均适配值;交叉操作通过交换两父代个体的部分信息构成后代个体,使得后代继承父代的有效模式,从而有助于产生优良个体;变异操作通过随机改变个体中某些基因而产生新个体,有助于增加种群的多样性,避免早熟收敛。

标准遗传算法的流程图描述,如图 3.1.1 所示。

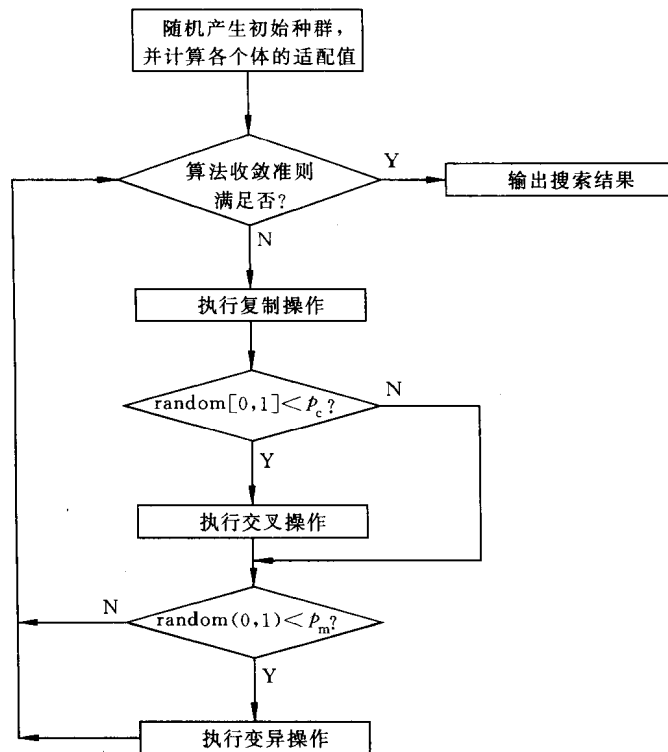


图 3.1.1 标准遗传算法的优化框图

遗传算法利用生物进化和遗传的思想实现优化过程,区别于传统优化算法,它具有以下特点:

(1) GA 对问题参数编码成“染色体”后进行进化操作,而不是针对参数本身,这使得 GA 不受函数约束条件的限制,如连续性、可导性等。

(2) GA 的搜索过程是从问题解的一个集合开始的,而不是从单个个体开始的,具有隐含并行搜索特性,从而大大减小了陷入局部极小的可能。

(3) GA 使用的遗传操作均是随机操作,同时 GA 根据个体的适配值信息进行搜索,无需其他信息,如导数信息等。

(4) GA 具有全局搜索能力,最善于搜索复杂问题和非线性问题。

遗传算法的优越性主要表现在:

(1) 算法进行全空间并行搜索,并将搜索重点集中于性能高的部分,从而能够提高效率且不易陷入局部极小。

(2) 算法具有固有的并行性,通过对种群的遗传处理可处理大量的模式,并且容易并行实现。

3.2 模式定理和隐含并行性

模式定理和隐含并行性是 Holland 为解释基于二进制编码的标准遗传算法(SGA)的能效而建立的,是最早对遗传算法全局收敛性作定性分析的理论基础,它说明了适配值高、长度短、阶数低的图式在后代中至少以指数增长包含该图式的个体数。

基于二进制编码的 SGA 中,个体(或称染色体)是以二进制字符串形式表示的,个体的每一位称为基因,令 X_l 为长度为 l 的二进制串的全体。若用 $*$ 表示通配符,即表示该位置的基因可取 1 或 0,则空间 $V^l = \{0, 1, *\}^l$ 表示所有模式的集合。譬如 $l=5$ 时,模式 $H=01**1$ 表示集合 $\{01001, 01011, 01101, 01111\}$ 。称出现在模式中取确定值的位置的数目为模式的阶,记为 $o(H)$,如 $o(01**1)=3$ 。称模式中第一个取确定值的位置与最后一个取确定值的位置之间的距离为模式的对应长度,记为 $\delta(H)$,如 $\delta(01**1)=4, \delta(*0***)=0$ 。

设种群数目为 N ,令 $X=(B^l)^N$ 为种群状态空间, $H \in V^l$ 为任一模式,记算法中第 t 代种群为 $P(t) = \{x_1(t), x_2(t), \dots, x_N(t)\} \in X$, $f: (B^l)^N \rightarrow R^1$ 为适配值函数,则称

$$f(H, t) = \frac{1}{|H \cap P(t)|} \sum_{x \in H \cap P(t)} f(x)$$

为模式 H 的平均适配值,其中 $|H \cap P(t)|$ 表示 $H \cap P(t)$ 中元素的个数,即 $P(t)$ 中含有 H 中元素的个数,并称 $\bar{f}(t) = \sum_{x \in P(t)} f(x)/N$ 为 $P(t)$ 的平均适配值。

模式定理 设 SGA 采用比例选择策略,交叉概率和变异概率分别为 p_c 和 p_m ,且 p_m 取值较小,模式 H 的定义长度为 $\delta(H)$,阶为 $o(H)$,第 $t+1$ 代种群 $P(t+1)$ 含有 H 中元素个数的期望为 $E[|H \cap P(t+1)|]$,则以下不等式成立:

$$E[|H \cap P(t+1)|] \geq |H \cap P(t)| \frac{f(H,t)}{\bar{f}(t)} \left[1 - p_c \frac{\delta(H)}{l-1} - o(H)p_m \right]$$

证明 由于 SGA 按适配值采用比例选择策略,则 $H \cap P(t)$ 中元素得以选择的期望值为

$$\sum_{x \in H \cap P(t)} \frac{f(x)}{\bar{f}} = |H \cap P(t)| \frac{f(H,t)}{\bar{f}(t)}$$

由于交叉操作是随机选取 1 到 $l-1$ 中的某一位置并交换两父代的对应子串,则属于模式 H 的个体经交叉后仍属于 H 的概率不小于 $1 - p_c \frac{\delta(H)}{l-1}$ 。而每个个体经变异操作后未被破坏的概率为 $(1 - p_m)^{o(H)}$ 。此外, $P(t)$ 不属于 H 的个体也有可能经交叉、变异后属于 H 。因此, $P(t+1)$ 中属于 H 的个体数目的期望值不小于

$$|H \cap P(t)| \frac{f(H,t)}{\bar{f}(t)} \left[1 - p_c \frac{\delta(H)}{l-1} \right] (1 - p_m)^{o(H)}$$

通常 SGA 中 p_m 取值很小,则

$$\begin{aligned} \left[1 - p_c \frac{\delta(H)}{l-1} \right] (1 - p_m)^{o(H)} &\approx \left[1 - p_c \frac{\delta(H)}{l-1} \right] (1 - o(H)p_m) \\ &\geq 1 - p_c \frac{\delta(H)}{l-1} - o(H)p_m \end{aligned}$$

从而,模式定理得证。

推论 在 SGA 中,阶次低、定义长度短且适配值超过平均适配值的模式在种群中的数目的期望值以指数级递增。

尽管模式定理在一定意义上解释了 SGA 的有效性,但还存在以下缺点:

- (1) 模式定理仅适用于基于二进制编码的 SGA,对其他编码方式此定理未必成立;
- (2) 模式定理仅提供一个期望值的下界,仍不能说明算法的收敛性;
- (3) 模式定理对算法参数的选取不能够提供实用的指导,对算法操作也有依赖性。

隐含并行性定理 设 ϵ 为一小正数, $l_s < \epsilon(l-1) + 1$, $N = 2^{l_s/2}$,则 SGA 一次处理的存活概率不小于 $1 - \epsilon$ 且定义长度不大于 l_s 的模式数为 $O(N^3)$ 。

此定理说明 SGA 表面上每代仅对 N 个个体作处理,但实际上并行处理了大约 $O(N^3)$ 个模式,并且无需额外的存储,这正是遗传算法具有高效搜索能力的所在,即隐含并行性。

由于模式定理和隐含并行性定理无助于严格解释算法的收敛性,下面我们将利用马氏链描述遗传算法,并探讨其收敛性。

3.3 遗传算法的马氏链描述及其收敛性

3.3.1 预备知识

定义 3.3.1 称 $n \times n$ 方阵 $A = (a_{ij})$ 为

- (1) 非负的, 记 $A \geq 0$, 若 $a_{ij} \geq 0, i, j = 1, 2, \dots, n$ 。
- (2) 严格正的, 记 $A > 0$, 若 $a_{ij} > 0, i, j = 1, 2, \dots, n$ 。
- (3) 正则的, 若 $A \geq 0$, 且存在自然数 k , 使 $A^k > 0$ 。
- (4) 随机的, 若 $A \geq 0, \sum_{j=1}^n a_{ij} = 1, i = 1, 2, \dots, n$ 。

- (5) 归约的, 若 $A \geq 0$ 且经过相同的行和列初等变换后可转化为 $\begin{bmatrix} C & 0 \\ R & T \end{bmatrix}$ 的形式, 其中

C 和 T 均为方阵。

- (6) 不可约的, 若 $A \geq 0$ 且不归约。
- (7) 稳定的, 若 A 是随机的且所有行相同。
- (8) 列容的, 若 A 是随机的且每一列中至少有一个正数。

引理 3.3.1 若 C, M 和 S 为随机阵, 且 $M > 0, S$ 为列容的, 则 $CMS > 0$ 。

证明 记 $A = CM, B = AS$ 。由于 C 为随机阵, 则 C 的每一行中至少存在一个正数。

从而, $\forall i, j \in \{1, 2, \dots, n\}, a_{ij} = \sum_{k=1}^n c_{ik} m_{kj} > 0$, 即 $A > 0$ 。进而, 由于 S 为列容的, 则

$\forall i, j \in \{1, 2, \dots, n\}, b_{ij} = \sum_{k=1}^n a_{ik} s_{kj} > 0$ 。故 $CMS > 0$ 得证。

引理 3.3.2 若 P 为正则随机阵, 则 P^k 收敛到一个全正稳定随机阵, 即 $P^\infty = [1, 1, \dots, 1]^T [p_1, p_2, \dots, p_n]$, 其中 $[p_1, p_2, \dots, p_n]P = [p_1, p_2, \dots, p_n], \sum_{i=1}^n p_{ij} = 1, p_j = \lim_{k \rightarrow \infty} p_{ij}^{(k)} > 0$, 即 $[p_1, p_2, \dots, p_n]^T$ 是 P^T 的特征值为 1 且各分量均为正数的特征向量。

引理 3.3.3 若

$$P = \begin{bmatrix} C & 0 \\ R & T \end{bmatrix}$$

是随机的, 其中 C 为 m 维严格正随机方阵, $R \neq 0, T \neq 0$, 则

$$P^\infty = \begin{bmatrix} C^\infty & 0 \\ R^\infty & 0 \end{bmatrix}$$

是一个稳定的随机阵, 其中

$$\mathbf{R}^{(k)} = \lim_{k \rightarrow \infty} \sum_{i=0}^{k-1} \mathbf{T}^i \mathbf{R} \mathbf{C}^{k-i}$$

同时,

$$\mathbf{P}^{(k)} = [1, 1, \dots, 1]^T [p_1, p_2, \dots, p_n], \quad \sum_{i=1}^n p_{ij} = 1, \quad p_j = \lim_{k \rightarrow \infty} p_{ij}^{(k)} \geq 0$$

其中 $p_j > 0 (1 \leq j \leq m)$, $p_j = 0 (m+1 \leq j \leq n)$ 。

3.3.2 标准遗传算法的马氏链描述

在 SGA 中,令所有个体形成的有限空间为 Ω ,所有种群对应的整个状态空间为 G ,其中每一种群为一个状态,一旦染色体长度 l 和种群数目 N 给定且有限时,则 Ω 的维数 $|\Omega|$ 有限, G 的维数 $|G| = |\Omega|^N$ 也有限。由于算法中每一代状态的转移依赖于选择(复制)、交叉和变异操作,且与进化代数无关,因此 SGA 可视为一个有限状态的齐次马氏链。

鉴于算法中三个遗传操作是循环往复执行的,我们按“交叉→变异→选择”顺序来考虑算法中状态的转移。从而,表征马氏链的状态转移矩阵 $\mathbf{P}$ 为矩阵 $\mathbf{C}$, $\mathbf{M}$ 和 $\mathbf{S}$ 的乘积,其中 $\mathbf{C}$, $\mathbf{M}$ 和 $\mathbf{S}$ 分别为交叉、变异和选择操作所决定的状态转移矩阵。

1. 交叉操作

交叉操作可视为状态空间上的随机函数,记交叉操作决定的状态转移矩阵为 $\mathbf{C} = (c_{ij})_{|G| \times |G|}$,其中 c_{ij} 为从状态 i 经交叉操作转移到状态 j 的概率。由于一个状态通过交叉操作总要转移到状态空间中的另一个状态,因此 $\sum_{j=1}^{|G|} c_{ij} = 1$ 显然成立,即 $\mathbf{C}$ 为随机矩阵。

2. 变异操作

在 SGA 中,变异操作是独立作用于种群内各个体的每一位基因上,记变异操作决定的状态转移矩阵为 $\mathbf{M} = (m_{ij})_{|G| \times |G|}$,其中 m_{ij} 为从状态 i 经变异操作转移到状态 j 的概率。由于染色体的每个基因具有相同的变异概率 $p_m > 0$,则 $m_{ij} = p_m^{H_{ij}} (1 - p_m)^{N - H_{ij}} > 0$,其中 H_{ij} 为状态 i 与状态 j 之间具有不同基因的位置的总数。例如,状态 $\{(1000), (0111), (1010)\}$ 与状态 $\{(1100), (1110), (0101)\}$ 之间的 H_{ij} 为 7。因此, $\mathbf{M}$ 为严格正的随机矩阵。

3. 选择操作

在 SGA 中,交叉和变异操作后得到的种群经选择操作后又将转移到另一个种群,记选择操作决定的状态转移矩阵为 $\mathbf{S} = (s_{ij})_{|G| \times |G|}$ 。考虑比例选择(轮盘赌)策略,种群中染色体 X_i 被选中的概率为

$$f(X_i) / \sum_{k=1}^N f(X_k) > 0$$

则种群 i 经选择操作后保持不变的概率

$$s_{ii} \geq \prod_{j=1}^N (f(X_j)) / \sum_{k=1}^N f(X_k) > 0$$

因此,转移矩阵 S 是随机且列容的。

通过对上述三个操作的概率描述,我们对 SGA 的马氏链描述有了一个清晰的认识,下面将对算法的收敛性进行阐述。

3.3.3 标准遗传算法的收敛性

对应于有限状态马氏链的一般遗传算法(二进制或十进制编码)均是后文介绍的 GASA 混合策略的一种特例,本节仅讨论基于二进制编码的 SGA 的收敛性。

定理 3.3.1 若 SGA 中交叉概率 $p_c \in [0, 1]$, 变异概率 $p_m \in (0, 1)$, 并且算法采用比例选择策略, 则 SGA 的状态转移矩阵 $P = CMS$ 是严格正的。

证明 利用上节矩阵 C, M 和 S 的定义以及引理 3.3.1 即可。

注:根据上述定理可知,SGA 是一个遍历马氏链,从而存在一个与初值无关的极限分布,由此算法的初始种群可以任意初始化,同时任何时刻从一个状态转移到另一个状态的概率不为 0,即马氏链构成了强连通图。然而,尽管初始种群的随机性在理论上不影响极限分布,但由于实际算法中由于某些环节的近似处理(如终止准则),算法的搜索结果存在一定的波动性,通常算法要执行多次才能得到较可靠的结果。

定理 3.3.2 SGA 不能够以概率 1 收敛到全局最优解。

证明 令 $P(t) = \{X_1(t), \dots, X_N(t)\}$ 为算法第 t 代的种群,其最优适配值为 $Z_t = \max\{f(X_k(t)), k=1, \dots, N\}$, 设 f^* 为全局最优的适配值。由定理 3.3.1 和引理 3.3.2 知,SGA 以任意状态 i 为极限分布的概率 $\lim_{t \rightarrow \infty} p_i^t = p_i^\infty > 0$, 因此, $\lim_{t \rightarrow \infty} P\{Z_t = f^*\} \leq 1 - p_i^\infty < 1$ 。

注:上述定理从概率意义上说明了 SGA 不能够收敛到全局最优,其原因在于算法中最优解的概率遗失。因此,只要在算法中每代保留当前最优解,无论是在选择之前还是在选择之后,算法将最终收敛到全局最优,从而有以下定理。

定理 3.3.3 每代在选择操作后保留最优解的 SGA 以概率 1 收敛到全局最优解。

证明 由于算法采用了保优技术,为方便起见,我们将保留下来的各代的最优个体存放在种群的第一号位置,但它不参与遗传操作。即,在第 t 代进化时的种群为 $(X^*(t-1), X_1(t), \dots, X_N(t))$, 其中 $X^*(t-1)$ 表示算法搜索至第 $t-1$ 代所得到的最佳个体。从而,算法对应马氏链的状态空间维数将变为 $|\Omega|^{N+1}$ 。然后,我们将包含相同的最优解的状态仍按在原状态空间的顺序进行排列,而对包含不同最优解的状态按最优解的适配值从大到小进行排列。由此,新的交叉、变异和选择操作对应的状态转移矩阵可表示为 $C^+ = \text{diag}(C, C, \dots, C)$, $M^+ = \text{diag}(M, M, \dots, M)$ 和 $S^+ = \text{diag}(S, S, \dots, S)$ 。

在选择操作后,算法要将当前种群中的最优解与前一代保留下来最优解进行比较,这一操作也可用矩阵 $U = (u_{ij})$ 表示。显然,状态 $Y(t) = (X^*(t-1), X_1(t), \dots, X_N(t))$ 转移

到 $Y(t+1) = (X^*(t), X_1(t+1), \dots, X_N(t+1))$ 的概率为

$$\Pr(Y(t+1) | Y(t)) = \begin{cases} 1, & \text{if } \max(f(X_1(t)), \dots, f(X_N(t))) = X^*(t) \\ & \text{and } (X_1(t), \dots, X_N(t)) = (X_1(t+1), \dots, X_N(t+1)) \\ 0, & \text{else} \end{cases}$$

从而, 矩阵 U 每一行均有且只有一个元素为 1, 而其他元素为 0。同时, 考虑到状态的排列顺序, 可知矩阵 U 为下三角矩阵。记

$$U = \begin{bmatrix} U_{11} & & & \\ U_{21} & U_{22} & & \\ \vdots & \vdots & \ddots & \\ U_{|\Omega|1} & U_{|\Omega|2} & \dots & U_{|\Omega||\Omega|} \end{bmatrix}$$

其中 U_{ij} 为 $|\Omega|^N \times |\Omega|^N$ 阶矩阵, 且 U_{11} 为单位矩阵。从而, 马氏链的一步转移矩阵为

$$P^+ = C^+ M^+ S^+ U = \text{diag}(CMS, CMS, \dots, CMS)$$

$$= \begin{bmatrix} CMSU_{11} & & & \\ CMSU_{21} & CMSU_{22} & & \\ \vdots & \vdots & \ddots & \\ CMSU_{|\Omega|1} & CMSU_{|\Omega|2} & \dots & CMSU_{|\Omega||\Omega|} \end{bmatrix}$$

显然, P^+ 为可约的随机矩阵, 且 $CMSU_{11}$ 严格正。进而, 考虑到 P^+ 中第一位状态由好到坏的排列顺序可知,

$$R = \begin{bmatrix} CMSU_{21} \\ \vdots \\ CMSU_{|\Omega|1} \end{bmatrix} \neq 0, \quad T = \begin{bmatrix} CMSU_{22} & & \\ \vdots & \ddots & \\ CMSU_{|\Omega|2} & \dots & CMSU_{|\Omega||\Omega|} \end{bmatrix} \neq 0$$

由引理 3.3.3 可知, 不包含最优个体的状态在马氏链的极限分布中的概率为 0, 包含最优个体的状态在马氏链的极限分布中的概率和为 1。从而, 算法将以概率 1 收敛到包含全局最优个体的状态, 也即算法能够以概率 1 搜索到全局最优解。

定理 3.3.4 每代在选择操作前保留最优解的 SGA 以概率 1 收敛到全局最优解。

证明过程类似于定理 3.3.3, 在此不再赘述。

除了讨论算法的收敛性, 对带有保优操作的遗传算法的收敛速度进行估计也是很重要的研究内容。

设带有保优操作的遗传算法对应于有限状态维马氏链 $\{Z_k\}$, 行向量 π 为给定每一状态的概率, P 为一步转移矩阵, 则经 n 次转移后, 各状态的概率为 πP^n 。若种群中全部个体相同且均为最优, 则称此状态为吸收态。显然, 马氏链 $\{Z_k\}$ 中有如下三种性质的状态: (1) 吸收态; (2) 可一步转移到吸收态的非吸收态; (3) 经 P 可一步转移到另一状态, 而此状态转移到吸收态的概率为零。从而, 一步转移矩阵 P 可分解为

$$P = \begin{bmatrix} I_k & 0 \\ R & Q \end{bmatrix}$$

其中, I_k 为描述吸收态的 k 阶单位矩阵, R 为描述能转移到吸收态的状态的 $(|G| - k) \times k$ 阶矩阵, Q 为描述其他状态的 $(|G| - k) \times (|G| - k)$ 阶矩阵。

进而

$$P^n = \begin{bmatrix} I_k & 0 \\ M_n R & Q^n \end{bmatrix}$$

其中 $M_n = I + Q + Q^2 + \cdots + Q^{n-1}$, $\|Q\| < 1$ 。从而,

$$P^\infty = \lim_{n \rightarrow \infty} P^n = \begin{bmatrix} I_k & 0 \\ (I - Q)^{-1} R & 0 \end{bmatrix}$$

由于保优 SGA 算法的全局收敛性, 最优分布 z^* 存在, 且 $z^* = z(0)P^\infty$, $z(0) = [z_1(0), \dots, z_k(0), z_{k+1}(0), \dots, z_{|G|}(0)] = [\bar{z}_1(0), \bar{z}_2(0)]$ 。以下给出 $z(i) = z(0)P^i$ 收敛到 z^* 的速度估计。

定理 3.3.5 设 $\|Q\| = \lambda < 1$, 则概率分布 $z(i)$ 收敛到 z^* 的收敛速度估计为 $\|z(i) - z^*\| \leq O(\lambda^i)$ 。

证明 $\|z(i) - z^*\| = \|z(0)P^i - z(0)P^\infty\|$

$$\begin{aligned} &= \left\| z(0) \left[\begin{bmatrix} I_k & 0 \\ M_i R & Q^i \end{bmatrix} - \begin{bmatrix} I_k & 0 \\ (I - Q)^{-1} R & 0 \end{bmatrix} \right] \right\| \\ &= \left\| z(0) \begin{bmatrix} 0 & 0 \\ (M_i - (I - Q)^{-1}) R & Q^i \end{bmatrix} \right\| \\ &= \|z_2(0) [(M_i - (I - Q)^{-1}) R \quad Q^i]\| \\ &\leq \|\bar{z}_2(0)\| (\|M_i - (I - Q)^{-1}\| \|R\| + \|Q^i\|) \\ &= \|\bar{z}_2(0)\| (\|(I - Q)^{-1} - (I - Q)^{-1} Q^i \\ &\quad - (I - Q)^{-1}\| \|R\| + \|Q^i\|) \\ &= \|\bar{z}_2(0)\| (\|-(I - Q)^{-1}\| \|R\| + 1) \|Q\|^i \\ &\leq \|\bar{z}_2(0)\| \left(\frac{\|R\|}{1 - \|Q\|} + 1 \right) \|Q\|^i = O(\lambda^i) \end{aligned}$$

迄今为止, 遗传算法的理论研究仍主要针对 SGA 模型。高级的遗传算法由于其本身的多样性, 理论方面的研究相当分散, 尚未取得引人注目的结果。同时, 大部分结论都是通过计算机数值仿真来说明的, 数学上严格完整且令人信服的解释仍需努力探索。GA 的收敛性研究, 尤其是如何提高算法的收敛速度和鲁棒性, 仍是具有重要研究价值的课题。

3.4 一般可测状态空间上遗传算法的收敛性

上节基于有限状态马氏链理论和特征值分析讨论了有限状态空间上 SGA 的收敛性

和收敛速度估计,本节讨论一般可测状态空间上遗传算法的收敛性及其收敛速度估计。

3.4.1 问题描述

考虑如下—类优化问题:

$$\max\{f(x); x \in X\} \quad (3.4.1)$$

其中, x 为优化状态, X 为可测的有界紧致空间, $f(x)$ 为有界适配值函数。

假定问题的最优解集 $S_{\text{opt}} = \{x; f(x) = f_{\max}\}$ 非空。令 $\phi(\cdot)$ 为空间 X 的一个测度函数,通常 S_{opt} 仅包含有限解,因此 $\phi(S_{\text{opt}}) = 0$ 。为了分析方便,如果 $\phi(S_{\text{opt}}) = 0$,则用一个具有正测度的扩大集 $S_{\text{opt}-\epsilon} = \{x; |f(x) - f_{\max}| \leq \epsilon\}$ 来代替 S_{opt} ,其中 ϵ 为小正实数。因此,基于上述考虑,在此假设 $\phi(S_{\text{opt}}) > 0$ 。

3.4.2 算法及其马氏链描述

遗传算法的框架同 SGA,为方便起见,按变异→交叉→选择的顺序进行遗传操作。

定义 3.4.1 称 X 为个体空间,称 X 的每个点 x 为一个个体,称乘积空间 $X^N = X \times X \times \cdots \times X$ 为种群空间,称 X^N 中的每一个 $\mathbf{x} = \{x_1, \dots, x_N\}$ 为包含 N 个个体的一个种群,定义种群 $\mathbf{x}$ 的适配值为 $f(\mathbf{x}) = \max\{f(x_i); x_i \in \mathbf{x}\}$,定义全局最优解集为 $S_{\text{opt}}^N = \{\mathbf{x}; \exists x_i \in \mathbf{x}, x_i \in S_{\text{opt}}\}$ 。

假设 $\xi(t)$ 为 t 时刻的种群状态, $\xi_M(t)$, $\xi_C(t)$ 和 $\xi_S(t)$ 分别为变异后、交叉后和选择后的种群状态,则 $\xi(t)$, $\xi_M(t)$, $\xi_C(t)$ 和 $\xi_S(t)$ 均为 X^N 上的随机变量,且 $\xi(t+1) = \xi_S(t)$ 。各变量的演化顺序为 $\xi(t) \rightarrow \xi_M(t) \rightarrow \xi_C(t) \rightarrow \xi_S(t) \rightarrow \xi(t+1)$ 。同时,由于各遗传操作不随时间变化,因此相应的转移概率函数是时不变的。

令变异、交叉和选择操作引起的转移概率函数分别为:

$$P_M(\mathbf{x}, A) = P_M(\xi_M(t) \in A | \xi(t) = \mathbf{x}) \quad (3.4.2)$$

$$P_C(\mathbf{x}, A) = P_C(\xi_C(t) \in A | \xi_M(t) = \mathbf{x}) \quad (3.4.3)$$

$$P_S(\mathbf{x}, A) = P_S(\xi_S(t) \in A | \xi_C(t) = \mathbf{x}) \quad (3.4.4)$$

其中 $A \subset X^N$ 。

从而,遗传算法可用状态空间 X^N 上的平稳马氏链 $\{\xi(t); t \in Z^+\}$ 来描述,其转移概率由 K-C 方程(Kolmogorov-Chapman Equation)给出:

$$P(\mathbf{x}, A) = \int_y \int_z P_M(\mathbf{x}, dy) P_C(y, dz) P_S(z, A) \quad (3.4.5)$$

3.4.3 收敛性分析和收敛速度估计

在此,我们并不讨论遗传操作的具体形式,仅强调它们使算法实现全局收敛的性质。

直观上而言,算法要实现全局收敛,首先要求任意初始种群经有限步转移能够到达全局最优解,其次算法必须有择优操作来防止遗失最优解。因此,我们假设各遗传操作满足如下性质。

性质 3.4.1 对任意时刻 t ,若 $\xi(t) \in S_{\text{opt}}^N$,则 $\xi_M(t+1) \in S_{\text{opt}}^N$;若 $\xi_M(t) \in S_{\text{opt}}^N$,则 $\xi_C(t+1) \in S_{\text{opt}}^N$;若 $\xi_C(t) \in S_{\text{opt}}^N$,则 $\xi_S(t+1) \in S_{\text{opt}}^N$ 。

注:性质 3.4.1 意味着各遗传操作具有择优性。

性质 3.4.2 存在某一时刻 t_0 ,对于任意初始种群 $\xi(1)=x$,任意可测子集 $A \in S_{\text{opt}}^N$ 和某一 $\delta(x) > 0$,转移概率函数满足 $P(1, t_0; x, A) \geq \delta(x)\phi(A)$ 。

注:性质 3.4.2 意味着对任意初始种群,最优集在有限步数内可达。

假设 μ_1, μ_2 为 X^N 上的两个概率测度,定义它们的整体偏零距离(total variation distance)为 $\|\mu_1 - \mu_2\| = \sup_{A \subseteq X^N} |\mu_1(A) - \mu_2(A)|$,其中 A 为 X^N 上的任意可测子集。假设 π 为 X^N 上的一个概率分布,它是以转移函数 $P(x, A)$ 表征的马氏链 $\{\xi(t); t \in Z^+\}$ 的不变量,当且仅当对 X^N 上的所有可测集 A , $\pi(A) = \int_{X^N} P(x, A)\pi(dx)$ 。

定义 3.4.2 对于以转移函数 $P(x, A)$ 表征的马氏链 $\{\xi(t); t \in Z^+\}$,记 μ_1 为初始种群的概率分布 $\xi(1)$, μ_t 为 $\xi(t)$ 的概率分布,若存在 P 的不变概率分布 π 使得 $\lim_{t \rightarrow \infty} \|\mu_t - \pi\| = 0$,则称 μ_t 收敛到 π 。进而,若存在子集 $A \subseteq X^N$,使得 $\pi(A) = 1$,则称 μ_t 全局收敛到 A 。

由此,给出一般可测状态空间上遗传算法的收敛定理和收敛速度估计。

定理 3.4.1 设 X^N 为有界且紧致的可测空间,若以式(3.4.5)表征的马氏链 $\{\xi(t); t \in Z^+\}$ 满足性质 3.4.1 和 3.4.2,则对于任意初始种群 $\xi(1)=x$,马氏链全局收敛到最优集,即存在不变概率测度 π 使得 $\pi(S_{\text{opt}}^N) = 1$ 且 $\lim_{t \rightarrow \infty} \|\mu_t - \pi\| = 0$ 。同时,存在 $t_0 > 0$ 和正实数 $\delta > 0$,使得 $\|\mu_t - \pi\| \leq (1 - \delta)^{\lceil t/t_0 \rceil - 1}$,其中 μ_t 为 $\xi(t)$ 的概率分布, $\lceil t/t_0 \rceil$ 表示取不超过 t/t_0 的最大整数。

在证明此定理之前,先介绍 Minorization 条件、Doebelin 条件和若干引理。

定义 3.4.3 称 X^N 上的转移函数 $P(x, A)$ 在子集 $R \subseteq X^N$ 上满足 Minorization 条件,若存在 X^N 上的一个概率测度 ϕ ,正整数 t_0 和正实数 $\delta > 0$,使得对所有 $x \in R$ 和 R 上的所有可测集 A , $P(1, t_0; x, A) \geq \delta\phi(A)$ 。

定义 3.4.4 定义 3.4.3 中,若 $R = X^N$,则称 Minorization 条件为 Doebelin 条件。

引理 3.4.1 设 X^N 为有界且紧致的可测空间,若以式(3.4.5)表征的马氏链 $\{\xi(t); t \in Z^+\}$ 满足性质 3.4.1 和 3.4.2,则它满足 Doebelin 条件。

证明 定义空间 X^N 上的概率测度 ϕ 如下:

$$\text{对所有可测子集 } A, \phi(A) = \phi(A \cap S_{\text{opt}}^N) / \phi(S_{\text{opt}}^N)$$

对于初始种群 $\xi(1)=x$ 和任意可测子集 $A \subseteq X^N$,考虑如下两种情况:

(1) 若 $\phi(A \cap S_{\text{opt}}^N) = 0$,则 $\phi(A) = 0$,从而 $P(1, t_0; x, A) \geq \phi(A) = 0$ 一直成立。

(2) 若 $\phi(A \cap S_{\text{opt}}^N) > 0$, 则 $\phi(A) > 0$, 由性质 3.4.2 知,

$$P(1, t_0; \mathbf{x}, A) \geq P(1, t_0; \mathbf{x}, A \cap S_{\text{opt}}^N) \geq \delta(\mathbf{x})\phi(A \cap S_{\text{opt}}^N) = \delta(\mathbf{x})\phi(S_{\text{opt}}^N)\phi(A)$$

由于 X^N 为有界且紧致的可测空间, 因此 $\delta = \inf\{\delta(\mathbf{x})\phi(S_{\text{opt}}^N); \mathbf{x} \in X^N\} > 0$ 。

联合考虑上述两种情况得, 对于时刻 t_0 , 正实数 $\delta > 0$ 和初始种群 $\mathbf{x}$, $P(1, t_0; \mathbf{x}, A) \geq \delta\phi(A)$ 成立, 即马氏链满足 Doeblin 条件。

注: t_0 可视为由任意初始种群到 S_{opt}^N 的首达时间, δ 可认为是相应的首达概率。

引理 3.4.2 若以式(3.4.5)表征的马氏链 $\{\xi(t); t \in Z^+\}$ 满足性质 3.4.1 和 3.4.2, 则对 P 的任意不变概率分布 π , $\pi(S_{\text{opt}}^N) = 1$ 。

证明 假设 $\pi(S_{\text{opt}}^N) = 1$ 不成立, 即 $\pi(X^N - S_{\text{opt}}^N) > 0$ 。

由性质 3.4.1, 对于任意种群 $\mathbf{x} \in S_{\text{opt}}^N$ 和集合 $X^N - S_{\\text{opt}}^N$, $P(\mathbf{x}, X^N - S_{\text{opt}}^N) = 0$ 。既然 π 为 P 的任意不变概率分布, 则对任意可测集 A , $\pi(A) = \int_{X^N} P(1, t_0; \mathbf{x}, A)\pi(d\mathbf{x})$ 。简单起见, 记 $\pi(A) = \int_{X^N} P^{t_0}(\mathbf{x}, A)\pi(d\mathbf{x})$ 。取 $A = X^N - S_{\text{opt}}^N$, 结合 $P(\mathbf{x}, X^N - S_{\text{opt}}^N) = 0$ 得,

$$\pi(X^N - S_{\text{opt}}^N) = \int_{X^N} P^{t_0}(\mathbf{x}, X^N - S_{\text{opt}}^N)\pi(d\mathbf{x}) = \int_{X^N - S_{\text{opt}}^N} P^{t_0}(\mathbf{x}, X^N - S_{\text{opt}}^N)\pi(d\mathbf{x})。$$

由性质 3.4.2 知, $P^{t_0}(\mathbf{x}, S_{\text{opt}}^N) > 0$, 即 $P^{t_0}(\mathbf{x}, X^N - S_{\text{opt}}^N) < 1$, 从而得,

$$\begin{aligned} 0 < \pi(X^N - S_{\text{opt}}^N) &= \int_{X^N - S_{\text{opt}}^N} P^{t_0}(\mathbf{x}, X^N - S_{\text{opt}}^N)\pi(d\mathbf{x}) < \int_{X^N - S_{\text{opt}}^N} \pi(d\mathbf{x}) \\ &= \pi(X^N - S_{\text{opt}}^N) \end{aligned}$$

这显然是矛盾的, 因此 $\pi(S_{\text{opt}}^N) = 1$ 是成立的。

引理 3.4.3 对于可测空间 X^N 上以转移函数 $P(\mathbf{x}, A)$ 表征的马氏链 $\{\xi(t); t \in Z^+\}$, 若它满足 Doeblin 条件, 即存在概率测度 ψ , 正整数 t_0 和正实数 $\delta > 0$, 使得 $P(1, t_0; \mathbf{x}, A) \geq \delta\phi(A)$, 则存在惟一不变量 π , 使得 $\|\mu_t - \pi\| \leq (1 - \delta)^{\lceil t/t_0 \rceil - 1}$ 。

由引理 3.4.1 和 3.4.3 知, 以马氏链 $\{\xi(t); t \in Z^+\}$ 表征的遗传算法的收敛速度可估计为 $\|\mu_t - \pi\| \leq (1 - \delta)^{\lceil t/t_0 \rceil - 1}$; 同时由引理 3.4.2 知 $\pi(S_{\text{opt}}^N) = 1$, 即遗传算法全局收敛到 S_{opt}^N 。从而, 定理 3.4.1 成立。

由定理 3.4.1 知, 收敛速度由首达概率 δ 或降低首达时间 t_0 控制, 增加 δ 或降低 t_0 是提高算法收敛速度的途径。

注: 给定任意精度 $\epsilon > 0$, 如果希望 $\|\mu_t - \pi\| \leq \epsilon$, 仅要求 $(1 - \delta)^{\lceil t/t_0 \rceil - 1} \leq \epsilon$, 这意味着当 $t \geq t_0[1 + \ln\epsilon/\ln(1 - \delta)]$ 时, 遗传算法将达到要求精度。

3.5 算法关键参数与操作的设计

通常, 遗传算法的设计是按以下步骤进行的:

- (1) 确定问题的编码方案。
- (2) 确定适配值函数。
- (3) 遗传算子的设计。
- (4) 算法参数的选取。主要包括种群数目、交叉与变异概率、进化代数等。
- (5) 确定算法的终止条件。

下面对关键参数与操作的设计作简单介绍。

1. 编码

编码就是将问题的解用一种码来表示,从而将问题的状态空间与 GA 的码空间相对应,这很大程度上依赖于问题的性质,并将影响遗传操作的设计。由于 GA 的优化过程不是直接作用在问题参数本身,而是在一定编码机制对应的码空间上进行的,因此编码的选择是影响算法性能与效率的重要因素。

函数优化中,不同的码长和码制,对问题求解的精度与效率有很大影响。二进制编码将问题的解用一个二进制串来表示,十进制编码将问题的解用一个十进制串来表示,显然码长将影响算法的精度,而且算法将付出较大的存储量。实数编码将问题的解用一个实数来表示,解决了编码对算法精度和存储量的影响,也便于优化中引入问题的相关信息,它在高维复杂优化问题中得到广泛应用。

组合优化中,由于问题本身的性质,编码方式需要特殊设计,如 TSP 问题中基于置换排列的路径编码、0-1 矩阵编码等。

2. 适配值函数

适配值函数用于对个体进行评价,也是优化过程发展的依据。在简单问题的优化时,通常可以直接利用目标函数变换成适配值函数,譬如将个体 X 的适配值 $f(X)$ 定义为 $M - c(X)$ 或 $e^{-ac(X)}$,其中 M 为一足够大正数, $c(X)$ 为个体的目标值, $a > 0$ 。在复杂问题的优化时,往往需要构造合适的评价函数,使其适应 GA 进行优化。

3. 算法参数

种群数目是影响算法优化性能和效率的因素之一。通常,种群太小则不能提供足够的采样点,以致算法性能很差,甚至得不到问题的可行解;种群太大时尽管可增加优化信息以阻止早熟收敛的发生,但无疑会增加计算量,从而使收敛时间太长。当然,在优化过程中种群数目是允许变化的。

交叉概率用于控制交叉操作的频率。概率太大时,种群中串的更新很快,进而会使高适配值的个体很快被破坏掉;概率太小时,交叉操作很少进行,从而会使搜索停滞不前。

变异概率是加大种群多样性的重要因素。基于二进制编码的 GA 中,通常一个较低的变异率足以防止整个群体中任一位置的基因一直保持不变。但是,概率太小则不会产生新个体,概率太大则使 GA 成为随机搜索。

由此可见,确定最优参数是一个极其复杂的优化问题,要从理论上严格解决这个问题是十分困难的,它依赖于 GA 本身理论研究的进展。Grefenstette(1986)曾将 GA 的参数

选取作为一个优化问题,提出用 GA 优化 GA 参数的二级数值方法。尽管此方法适用范围较广,但工作量较大,且二级算法本身的参数也有待优化,因此很少得到实际应用。

4. 遗传算子

优胜劣汰是设计 GA 的基本思想,它应在选择、交叉、变异等遗传算子中得以体现,并考虑到对算法效率与性能的影响。

复制操作是为了避免有效基因的损失,使高性能的个体得以更大的概率生存,从而提高全局收敛性和计算效率。最常用的方法是比例复制和基于排名的复制,前者以正比于个体适配值的概率来选择相应的个体,后者则基于个体在种群中的排名来选择相应的个体。至于种群的替换,采纳的方案可以是部分个体的替换,也可以是整个群体的替换。

交叉操作用于组合出新的个体,在解空间中进行有效搜索,同时降低对有效模式的破坏概率。二进制编码中,单点交叉随机确定一个交叉位置,然后对换相应的子串;多点交叉随机确定多个交叉位置,然后对换相应的子串。譬如,父串为 $\{(1011001), (0010110)\}$,若单点交叉位置为 4,则后代为 $\{(1011110), (0010001)\}$;若多点交叉位置为 2,5,则后代为 $\{(1010101), (0011010)\}$ 。十进制编码也类似。实数编码则可采用算术交叉,即 $x'_1 = \alpha x_1 + (1-\alpha)x_2$, $x'_2 = \alpha x_2 + (1-\alpha)x_1$, 其中 $\alpha \in (0,1)$, x_1, x_2 为父代个体, x'_1, x'_2 为后代个体。组合优化中,交叉操作有部分映射交叉、次序交叉、循环交叉等,后文将对它们作简单介绍,读者也可参阅 Goldberg 的著作(1989)。

当交叉操作产生的后代适配值不再进化且没有达到最优时,就意味着算法的早熟收敛。这种现象的根源在于有效基因的缺损,变异操作一定程度上克服了这种情况,有利于增加种群的多样性。二进制或十进制编码中通常采用替换式变异,即用另一种基因替换某位置原先的基因;实数编码中通常采用扰动式变异,即对原先个体附加一定机制的扰动来实现变异;组合优化问题中通常采用互换式、逆序式、插入式变异,这在介绍 SA 时已作说明。

5. 算法的终止条件

GA 的收敛理论说明了 GA 以概率 1 收敛的极限性质,因此我们要追寻的是提高算法的收敛速度,这与算法操作设计和参数选取有关。然而,实际应用 GA 时是不允许让它无停止地发展下去的,而且通常问题的最优解也未必知道,因此需要有一定的条件来终止算法的进程。最常用的终止条件就是事先给定一个最大进化步数,或者是判断最佳优化值是否连续若干步没有明显变化等。

应该认识到,GA 不是一个简单的系统,而是一种复杂的非线性智能计算模型,纯粹用数学方法来预测其运算结果是很难的,而且这方面的工作也远远不够。目前,为兼顾 GA 的优化质量与效率,实际应用 GA 时许多环节一般还只是凭经验解决,这方面还有待更深入的研究与发展。

3.6 遗传算法的改进

遗传算法的主要任务和目的,是设法产生或有助于产生优良的个体成员,且这些成员能够充分表现出解空间中的解,从而使算法效率提高并避免早熟收敛现象。但是,实际应用遗传算法时,往往出现早熟收敛和收敛性能差等缺点。现今的改进方法,大都针对基因操作、种群的宏观操作、基于知识的操作和并行化 GA 进行。

对于复制操作,DeJong(1975)研究了回放式随机采样复制,其缺点是选择误差较大;同时,他又提出了无回放式随机采样复制以降低选择误差。Brindle(1981)提出了一种选择误差更小、操作简单的确定式采样以及无回放式余数随机采样方法。Back(1992)提出了与适配值大小和正负无关的均匀排序策略;同时又提出全局收敛的最优串复制策略,提高了搜索效率,但不适于非线性较强的问题。在交叉操作方面,有 Goldberg(1989)提出的部分匹配交叉算子(partially mapped crossover)、Starkweather 等(1991)提出的增强边缘重组算子(enhanced edge recombination)、Davis(1991)提出的序号交叉算子(order crossover)和均匀排序交叉算子(uniform order-based crossover)、Smith(1985)提出的循环交叉算子(cycle crossover)、Dejong(1975)提出的单点交叉算子(one-point crossover)和多点交叉算子、Syswerda(1989)提出的双点交叉算子(two-point crossover),此外常用的交叉算子还有置换交叉、算术交叉、启发式交叉等。变异操作方面主要发展了自适应变异、多级变异等操作。为了改进算法的性能,一些高级的基因操作也得到了发展和应用,譬如双倍体和显性遗传(diploid and dominance)用以延长曾经适配值高而目前较差的基因块的寿命,并且在变异概率低的情况下也能保持一定的多样性;倒位操作(inversion)用以助长有用基因块的紧密形式;优先策略(elitism)用以把目前解群中最好的解直接放入下一代种群中,如此保证各代种群中总会有目前为止最好的解;静态繁殖(steady-state reproduction)用以在迭代过程中用部分优质子串来更新部分父串作为下一代种群以使优质的父串在下一代中得以保留;没有重串的静态繁殖(steady-state without duplication)用以在形成下一代种群时不含重复的串。此外,还有多倍体结构(multiple chromosome structure)、分离(segregation)、异位(translocation)等操作。在种群宏观操作方面,主要是引入小生存环境和物种形成(niche and speciation)的思想,这在分类问题中得到了应用。在基于知识的操作方面,主要是将问题的特殊信息与 GA 相结合,包括混合算法的构造以及在遗传算子中增加知识的操作等。

遗传算法的结构也是很值得研究的问题。Krishnakumar(1989)为克服群体数目大造成计算时间长的缺点,提出了所谓 μ GA 的小群体方法,仿真结果显示了较高的计算效率和适于动态系统优化的潜力,但尚无严格的理论分析。Schraudolph 等(1992)针对二进制编码优化精度差的缺点,提出了一种类似于对解空间进行尺度变换的参数动态编码

策略,较好地提高了 GA 的精度,但不适于非线性较强的多极小优化问题。Androulakis 等(1991)采用实数编码提出了一种扩展遗传搜索算法,把搜索方向作为独立的变量来处理以解决有约束的优化问题。Poths 等(1994)为克服算法早熟收敛的缺点,提出了类似于并行实现思想的基于变迁和人工选择的遗传算法。Grefenstette(1981)全面研究了 GA 并行化实现的结构问题,并给出了多种结构形式,主要有同步主-仆方法(synchronous master-slave)、亚同步主-仆方法(semi-synchronous master-slave)、分布式异步并发方法(distributed asynchronous concurrent)、网络方法(network)。Goldberg(1989)提出了基于对象设计 GA 并行结构的思想。Muhlenbein 等(1991)用并行遗传算法求解高维多极小函数的全局最小解,从而提供了 GA 求解高度复杂优化问题的有力实例。

在函数优化方面,Goldberg(1989)引入分享(sharing)思想将解空间分成若干子空间,然后在子空间中产生子群体成员分别进行优化,以求得整个问题的解,避免算法只收敛到某个局部解;DeJong(1975)提出聚集(crowding)的思想,根据群成员中的相似性来部分替换群体中的个体成员,从而将一些个体成员分别聚集于各个群集中,然后在各个群集中分别求解问题的局部解以实现与分享思想相同的目标。然而,这些方法的应用还有一定的限度,对于解是随机分布的情况就不易奏效。由此,孟庆春等(1995)提出门限变换思想,在选择个体成员进入下一代时,引入门限变换函数将某些优良成员周围的成员传到下一代以达到除劣增优的目的,从而避免搜索的盲目性。至于有约束优化问题的求解,目前的处理方法主要有:(1)把问题的约束在“染色体”的表示形式中体现出来,并设计专门的遗传算子,使“染色体”所表示的解在 GA 运行过程中始终保持可行性。这种方法最直接,但适用领域有限,算子的设计也较困难。(2)在编码过程中不考虑约束,而在 GA 运行过程中通过检验解的可行性来决定解的弃用。这种方法一般只适用于简单的约束问题。(3)采用惩罚的方法来处理约束越界问题。但到目前为止,采用 GA 求解高维、多约束、多目标的优化问题仍是一个没有很好解决的课题,它的进展将会推动 GA 在许多工程领域的应用。

3.7 免疫遗传算法

本节介绍一种基于免疫的改进遗传算法,它是生命科学中免疫原理与传统遗传算法的结合。算法的核心在于免疫算子的构造,而免疫算子又是通过接种疫苗和免疫选择两个步骤来完成的。同时,在理论上免疫算法是概率 1 收敛的。

3.7.1 引言

遗传算法是一类基于“产生+测试”方式的迭代搜索算法,尽管算法在一定条件具有

全局收敛特性,但算法的交叉、变异、选择等操作一般都是在概率意义下随机进行的,虽保证了种群的群体进化性,但一定程度上不可避免退化现象的出现。此外,尽管遗传算法具有通用性的一面,但却忽视了问题特征信息的辅助作用,同时相对固定的遗传操作使得对不同问题的求解缺少灵活性。大量研究表明,仅仅依赖于以遗传算法为代表的进化算法在模拟人类智能化处理事物能力方面还远远不足,还需要更深入地挖掘和利用人类的智能资源,而免疫遗传算法就是将生命科学中免疫的原理与遗传算法相结合来提高算法的整体性能,并有选择、有目的地利用待求解问题中的一些特征信息来抑制优化过程中退化现象的出现。

3.7.2 免疫遗传算法及其收敛性

1. 免疫遗传算法

类似于生物自然科学的免疫理论,免疫算子分为全免疫(full immunity)和目标免疫(target immunity),两者分别对应于生命科学中非特异性免疫和特异性免疫。其中,全免疫指种群中每个个体在遗传算子作用后,对其每一环节进行一次免疫操作;目标免疫则指在进行了遗传操作后,经过一定的判断,个体仅在作用点处发生免疫反应。前者主要应用于个体进化的初始阶段,而在进化过程中基本上不发生,否则将很有可能产生“同化”现象;后者一般将伴随进化的全过程,它是免疫的一个基本算子。

实际的操作过程中,首先对所求解的问题(即抗原, antigen)进行具体分析,从中提取最基本的特征信息(即疫苗, vaccine);其次,对特征信息进行处理,将其转化为求解问题的一种方案(由此方案得到的所有解的集合称为基于上述疫苗所产生的抗体, antibody);最后,将此方案以适当的形式转化为免疫算子,以实施具体操作。需要说明的是,待求解问题的特征信息往往不止一个,也就是说针对某一特定抗原所能提取的疫苗也可能不止一个,那么在接种疫苗过程中可以随机地选取一种疫苗进行注射,也可将多个疫苗按照一定的逻辑关系进行组合后再注射。总而言之,免疫的思想主要是在合理提取疫苗的基础上,通过接种疫苗和免疫选择两个操作来完成的。前者以提高适应度,后者以防止种群的退化。

(1) 接种疫苗

对于个体 x ,对它接种疫苗是指按照先验知识来修改其某些基因位上的基因,使所得个体以较大的概率具有更高的适应度。

实现考虑以下两种特殊情况:

其一,若个体 y 的每一基因位上的信息都是错误的,即每一位码都与最佳个体不同,则对任一个体 x , x 转移到 y 的概率为 0;

其二,若个体 x 的每一基因位都是正确的,即 x 已是最佳个体,则 x 以概率 1 转移为 x 。

假设种群 $c = (x_1, \dots, x_{n_0})$, 对种群 c 接种疫苗是指在 c 中按比例 $\alpha (0 < \alpha \leq 1)$ 随机抽取 $n_a = \alpha n$ 个个体而进行的操作。疫苗是从对问题的先验知识中提炼出来的, 它所包含的信息量及其正确性对算法的性能起着重要的作用。

(2) 免疫选择

该操作分两步完成。第一步是免疫检测, 即对接种了疫苗的个体进行检测, 若其适应度仍不如父代, 说明在交叉、变异的过程中出现了严重的退化现象。此时该个体将被父代中所对应的个体所取代; 如果子代适应度优于父代则进行第二步处理。第二步是退火选择, 即在当前子代种群 $E_k = (x_1, \dots, x_{n_0})$ 中以概率 $P(x_i) = e^{f(x_i)/T_k} / \sum_{i=1}^{n_0} e^{f(x_i)/T_k}$ 选择个体 x_i 进入新的父代种群, 其中 $f(x_i)$ 为 x_i 的适应度, $\{T_k\}$ 为趋于 0 的温度序列。

至此, 给出免疫遗传算法如下, 其流程如图 3.7.1 所示。

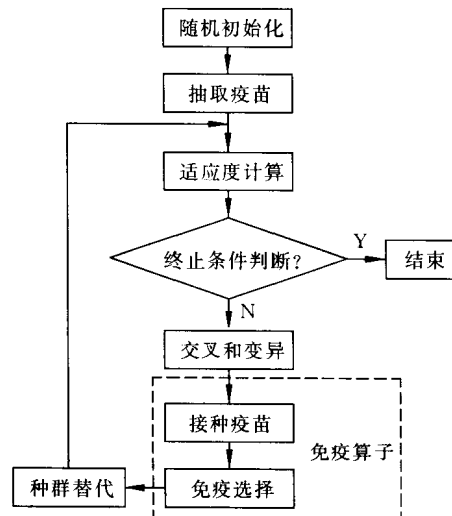


图 3.7.1 免疫遗传算法流程图

- (1) 随机产生初始父代种群 A_1 。
- (2) 根据先验知识抽取疫苗。
- (3) 若当前种群中包含了最佳个体, 则结束算法; 否则进行以下步骤。
- (4) 对当前第 k 代父代种群 A_k 进行交叉操作, 得到种群 B_k 。
- (5) 对种群 B_k 进行变异操作, 达到种群 C_k 。
- (6) 对种群 C_k 进行接种疫苗操作, 得到种群 D_k 。
- (7) 对种群 D_k 进行免疫选择操作, 得到新一代父代种群 A_{k+1} , 返回步骤 3。

2. 免疫遗传算法的收敛性

设种群的规模为 n_0 且在算法中保持不变, 种群中所有个体均为 l 位的 q 进制编码,

算法中的交叉操作选择一点或多点均可,变异操作是对每个基因位以概率 P_M 相互独立地进行变异。算法的状态转移情况可用随机过程 $A_k \xrightarrow{\text{交叉}} B_k \xrightarrow{\text{变异}} C_k \xrightarrow{\text{接种疫苗}} D_k \xrightarrow{\text{疫苗选择}} A_{k+1}$ 来表示,其中 A_k 到 D_k 的状态转移构成一个马氏链。设 X 为个体搜索空间,种群可视为状态空间 $S=X^n$ 中的一个点, $|S|$ 为 S 的规模, $s_i \in S$ 为一个种群, V_k^i 表示随机变量 V 在第 k 代处于状态 s_i , 记 $S^* = \{x \in X; f(x) = \max_{x_i \in X} f(x_i)\}$ 为最优状态集。

定义 3.7.1 若对任意初始分布,均有 $\lim_{k \rightarrow \infty} \sum_{s_i \cap S^* \neq \emptyset} P(V_k^i) = 1$, 则称算法收敛。

该定义表明:所谓算法收敛,即指当算法迭代到足够多的次数以后,群体中包含全局最佳个体的概率接近于 1,也即算法以概率 1 收敛。

定理 3.7.1 免疫遗传算法是概率 1 收敛的,若算法中略去免疫算子,则该算法将不再保证收敛到全局最优值,或者说它是强不收敛的。

证明 参见文献(Jiao et al, 2000)。

3.7.3 免疫算子的机理与构造

1. 免疫算子的机理

免疫算子是由接种疫苗和免疫选择两部分操作构成的。其中,疫苗指的是依据人们对待求问题所具备的先验知识而从中提取出的一种基本的特征信息,抗体是指根据这种特征信息而得出的一类解。前者可看作是对待求的最佳个体所能匹配模式(schema)的一种估计,后者则是对这种模式进行匹配而形成的样本。从对算法的描述中不难发现,疫苗的正确选择对算法的运行效率具有十分重要的意义。它如同遗传算法的编码一样,是免疫操作得以有效地发挥作用的基础与保障。但需说明的是,选取疫苗的优劣,生成抗体的好坏,只会严重影响到免疫算子中接种疫苗作用的发挥,不至于涉及到算法的收敛性。因为免疫遗传算法的收敛性,归根结底是由免疫算子中的免疫选择来保证的。

下面考察免疫选择在算法运行过程所起到的作用。

定理 3.7.2 在免疫选择作用下,若疫苗使抗体适应度得到提高,且高于当前群体的平均适应度,则疫苗所对应的模式将在群体中呈指数级扩散;否则,它将被遏制或呈指数级衰减。

证明 参考遗传算法模式定理的证明,参见文献(Jiao et al, 2000)。

可见,免疫选择在加强接种疫苗方面具有积极作用,在消除其负面影响方面具有鲁棒性。考虑到免疫遗传算法的应用对象主要是 NP 类问题,而这类问题在规模较小时一般易于求解,或者说易于发现其局部条件下的求解规律。因此,在选取疫苗时,既可以根据问题的特征信息来制作免疫疫苗,也可在具体分析的基础上考虑降低原问题的规模,增设一些局部条件来简化问题,用简化后的问题求解规律来作为选取疫苗的一种途径。但是,

在实际的选取过程中应考虑到:一方面,原问题局域化处理越彻底,则局部条件下的求解规律就越明显,这时虽然易于获取疫苗,但寻找所有这种疫苗的计算量会显著增加;另一方面,每个疫苗都是利用某一局部信息来探求全局最优解,即估计该解在某一分量上的模式,所以没有必要对每个疫苗做到精确无误。因此一般可以根据对原问题局域化处理的具体情况,选用目前通用的一些迭代优化算法来提取疫苗。

2. 免疫算子的执行算法

为表述方便,令 $a_{H,k}^i$ 为对第 k 代第 i 个个体 a_k^i 接种疫苗后所得到的抗体, P_I 为个体接种疫苗的概率, P_V 为更新疫苗的概率, $V(a_k^i, h_j)$ 表示按模式 h_j 修改个体 a_k^i 上基因的接种疫苗操作, n 和 m 分别为群体和疫苗的规模。那么,在针对某一待求问题而构造和应用免疫算子时所进行的过程如下所示:

(1) 抽取疫苗:

(1.1) 分析待求问题,搜集特征信息;

(1.2) 依据特征信息估计特定基因位上的模式 $H = \{h_j; j=1, 2, \dots, m\}$

(2) 令 $k=0, j=0$;

(3) While (Conditions = True)

(3.1) 若 $\{P_V\} = \text{True}$, 则 $j=j+1$;

(3.2) $i=0$;

(3.3) for($i \leq n$)

(3.3.1) 接种疫苗: $a_{H,k}^i = V_{\{P_I\}}\{a_k^i, h_j\}$;

(3.3.2) 免疫检验: 若 $a_{H,k}^i < a_{k-1}^i$, 则 $a_k^i = a_{k-1}^i$; 否则 $a_k^i = a_{H,k}^i$;

(3.3.3) $i=i+1$;

(3.3.4) 退火选择: $A_{k+1} = S(A_k), k=k+1$;

其中,停机条件可以采用最大迭代次数或统计个体最佳适应度连续不变的最大次数。

3. 免疫疫苗的选取示例

前文简要说明了免疫疫苗的一般选取策略,在此结合 TSP 问题具体讨论其具体过程与步骤。

(1) 分析待求问题,搜集特征信息

假设在某一时刻,从一城市出发欲前往下一个目标城市,一般首先考虑距离当地路程最近的城市,如果它已被访问过,则可选择剔除该城市之外的距离最小的城市,并以此类推。尽管这种贪婪策略不能保证全局最优,但在仅包含几个城市的一个很小范围内,往往不失为一个较好的策略。当然,能否作为最终的解决方案,还需要进一步的判断。

(2) 根据特征信息制作免疫疫苗

基于上述认识,就 TSP 问题的特点而言,在最终的解决方案中,即在最佳路径里必然包括而且在很大程度上包括了相邻城市间距离最短的路径。显然,这种特点可作为求解问题时以供参考的一种特征信息或知识,从而视为从问题中抽取疫苗的一种途径。因此,

在具体实施过程中,只需使用一般的循环迭代方法找出所有城市的邻近城市即可(当然,某一城市可能是两个或多个城市的邻近城市,也可能都不是)。需要强调的是,疫苗不是一个个体,故不能做为问题的解,它仅仅具备个体在某些基因位上的特征。

(3) 接种疫苗

不失一般性,设 TSP 问题中所有与城市 A_i 距离最近的城市为 A_j ,并且二者非直接连接而是处于某一路径的两段: $A_{i-1}-A_i-A_{i+1}$ 和 $A_{j-1}-A_j-A_{j+1}$,如图 3.7.2 中实线所示。

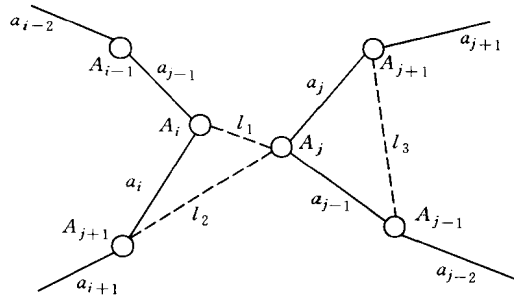


图 3.7.2 TSP 问题的疫苗作用机理示意图

则,当前的遍历路径为 $\pi = \{A_0, \dots, A_{i-1}, A_i, A_{i+1}, \dots, A_{j-1}, A_j, A_{j+1}, \dots, A_N\}$,其对应的路径

长度为 $D_\pi = \sum_{k=1}^{i-1} a_k + a_i + \sum_{k=i+1}^{j-2} a_k + a_{j-1} + a_j + \sum_{k=j+1}^N a_k$ 。在免疫概率 P_i 发生条件下,对

城市 A_i 而言,免疫算子将把其邻近城市 A_j 排列为其下一个城市,而使原先的遍历路径调整为 $\pi_c = \{A_0, \dots, A_{i-1}, A_i, A_j, A_{i+1}, \dots, A_{j-1}, A_{j+1}, \dots, A_N\}$,相应的路径长度变化为

$D_{\pi_c} = \sum_{k=1}^{i-1} a_k + l_1 + l_2 + \sum_{k=i+1}^{j-2} a_k + l_3 + \sum_{k=j+1}^N a_k$ 。比较两路径长度,因为 A_j 是所有城市中

(即全局中)与城市 A_i 距离最近的点,在由 $A_i-A_j-A_{i+1}$ 所构成的三角形中 l_1 一定为最短边或次短边(此时 l_2 一定为最短边,因为若 $a_i < l_1$,则与 A_i 最近的城市为 A_{i+1} 而非 A_j),而在 A_{j-1}, A_j 和 A_{j+1} 之间却不一定具有这个性质。所以在多数情况下, l_3 较 $a_{j-1} + a_j$ 的减少量要大于 $l_1 + l_2$ 较 a_i 的增加量,而且更加重要的是在这一个局部环境内,算子对路径做了一次最佳调整。当然,这次调整究其能否对整个路径有所贡献,还有待于选择机制的进一步判断。但是,从分析过程中不难得出 $P(D_{\pi_c} < D_\pi) \gg P(D_{\pi_c} > D_\pi)$,式中 $P(A)$ 表示事件 A 发生的概率。上述所谓的“调整”过程,即为 TSP 问题求解时基于某一特定疫苗的免疫注射过程。

3.7.4 TSP 问题的免疫遗传算法

本节以平面 TSP 问题为例,介绍免疫遗传算法的一种实现。

1. 编码与适应度函数

为方便与直观起见,可采用 N 个城市的访问次序为问题的编码。

适应度函数可采用式 $f(\pi)=76.5 \times L \sqrt{N} / D_{\pi}$ 计算,其中 L 为包含所有城市的最小正方形的边长, D_{π} 为在排列 π 的路径长度。

2. 交叉与变异算子

采用两点交叉,其中交叉点的位置随机确定。

对于变异操作,算法中加入了对遗传个体基因型特征的继承性和对进一步优化所需个体特征的多样性进行评测的环节,基此设计一种部分路径变异法。该方法每次选取全长路径的一段,路径子段的起点与终点由评测的结果估算确定。具体操作为采用连续 n 次的调换方式,其中 n 的大小由遗传代数 K 决定,如 $n=\lceil N/M+e^{-\alpha K} \rceil$,其中 M 为路径子段的数目, α 为表示快慢程度的常数。

3. 免疫算子

免疫算子包括全免疫和目标免疫两种,对具体问题应视所能提取疫苗的性质而决定采用何种免疫操作。对于 TSP 问题,要找到适用于整个抗原(即全局问题求解)的疫苗极为困难,所以不妨采用目标免疫。具体而言,在求解问题之前先从每个城市点的周围各点中选取一个路径最近的点,以此作为算法执行过程中对该城市点进行目标免疫操作时所注入的疫苗。每次遗传操作后,随机抽取一些个体注射疫苗,然后进行免疫检测,即对接种了疫苗的个体进行检测:若适应度提高,则继续;反之,若其适应度仍不如父代,说明在交叉、变异的过程中出现了严重的退化现象,这时该个体将被父代中所对应的个体所取代。在选择阶段,先计算其被选中的概率,后进行相应的条件判断。

4. 仿真研究

以著名的 75 城市 TSP 为例,取群体规模为 100,交叉概率在 0.5~0.85,变异概率在 0.2~0.01,个体接种疫苗概率 0.2~0.3,更新疫苗概率 0.5~0.8 之间随进化过程自行调整; M 和 α 分别取 10 和 0.04,退温函数为 $T_k=\ln(T_0/k+1)$, $T_0=100$,其中 k 为进化代数。在基本参数保持不变的前提下,对通用遗传算法和免疫算法进行比较,若群体的最佳适应值在连续 100 次迭代中保持不变则认为结束搜索。同时,计算过程中每隔 10 代记

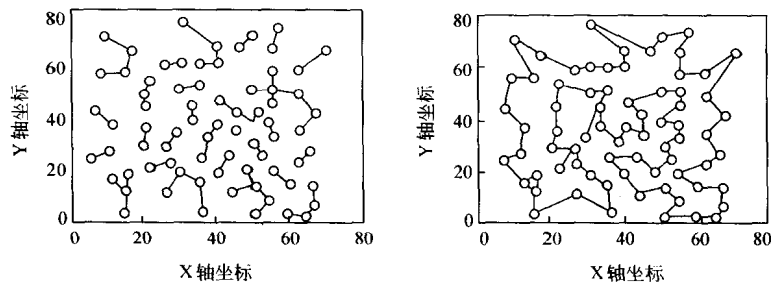


图 3.7.3 优化 75 城市 TSP 问题的免疫疫苗和优化结果

录一次进化结果。图 3.7.3 显示了免疫遗传算法优化 75 城市 TSP 问题的免疫疫苗和优化结果。仿真结果表明,免疫遗传算法经 940 代首次出现后来被认定的最佳个体,而通用遗传算法经 3550 代才出现该最佳个体,同时发现免疫遗传算法对提高搜索效率,消除标准遗传算法在后期的振荡现象具有明显的效果。

3.8 并行遗传算法

GA 的内在并行性在 Holland 提出 GA 时就得到了认识,因此,在并行计算机上实现 GA 是提高算法性能和效率的有效途径。Grefenstette(1981)对 GA 并行化实现的结构问题进行了全面研究,并给出了多种结构形式,在此仅对其中最基本的三种并行方案进行简单阐述。

1. 同步主仆式

遗传算法的同步主仆式并行执行方案,如图 3.8.1 所示。

在这种并行方式中,一个主过程协调若干个仆过程。其中,主过程控制选择、交叉和变异的执行,仆过程仅执行适配值的计算。这种并行化方式很直观,且易于实现,但是也存在两个主要的缺点:若各仆过程计算适配值的时间存在明显差异时,将会造成整个系统长时间的等待;整个系统可靠性较差,对主过程状况的依赖性较大。

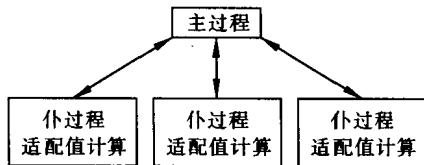


图 3.8.1 同步主仆式并行遗传算法

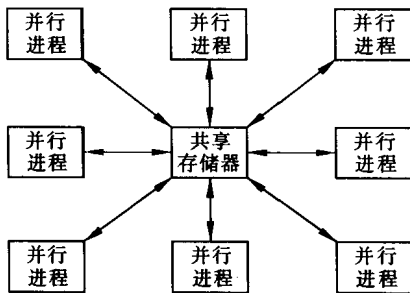


图 3.8.2 异步并发式并行遗传算法

2. 异步并发式

遗传算法的异步并发式并行执行方案,如图 3.8.2 所示。

在这种并行方式中,通过存取一个共享存储器,若干个同样的处理机彼此无关地执行各个遗传算子和适配值的计算。只要存在一个并行进程,同时共享存储器可继续运行,则整个系统就可进行有效的处理。显然,这种方式不易实现,但可大大提高系统的可靠性。

3. 网络式

遗传算法的网络式并行执行方案,如图 3.8.3 所示。

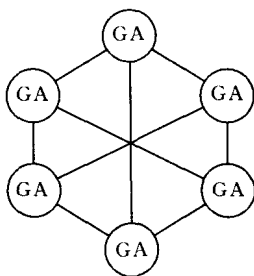


图 3.8.3 网络式并行遗传算法

在这种并行方式中,若干个无关的遗传算法分别在独立的存储器上进行独立的遗传操作和适配值计算,同时各个子群体在每一代中发现的最佳个体通过相应的通信网络传送给其他子群体。与前两种方式相比,由于存在通信时的间断,此方式的连接带宽缩小了。但是,由于各独立过程的自治性,系统的可靠性提高了。

3.9 算法实现与应用

由于后文将着重介绍由 GA 构造的混合优化策略的若干应用,在此仅对 GA 应用于组合优化和函数优化时的实现问题作简单介绍。

1. 组合优化问题的求解

本小节将以置换 Flow-shop 问题为例介绍遗传算法的实现方案。

置换 Flow-shop 问题是一类典型的加工调度问题,而且是一个 NP-hard 问题,它通常可描述为: n 个工件在 m 台机器上流水加工,在每一时刻,每台机器只能加工一个工件且每一工件只能在某一台机器上加工,同时在每台机器上所有工件必须以同一顺序加工,已知工件 i 在机器 j 上的加工时间为 $t_{i,j}$, $i=1,2,\dots,n$, $j=1,2,\dots,m$, 希望确定各工件的加工顺序,使得最大加工完成时间 $c_{\max}$ (makespan) 或 E/T 等性能指标最小。令工件 i 在机器 j 上的加工完成时间为 $T_{i,j}$, 加工顺序为 $\pi=\{\pi_1, \pi_2, \dots, \pi_n\}$, 则

$$\begin{cases} T_{\pi_1,1} = t_{\pi_1,1}, \dots, T_{\pi_1,k} = \sum_{l=1}^{k-1} t_{\pi_1,l} \\ T_{\pi_i,1} = T_{\pi_{i-1},1} + t_{\pi_i,1} \\ T_{\pi_i,k} = \max\{T_{\pi_{i-1},k}, T_{\pi_i,k-1}\} + t_{\pi_i,k} \\ i = 2, 3, \dots, n; k = 2, 3, \dots, m \\ c_{\max} = T_{\pi_n,m} \end{cases}$$

(1) 编码选择

鉴于 Flow-shop 问题与 TSP 问题的相似性,路径编码仍是最常用的方案,即用路径来表示工件的加工顺序,如路径 5-4-1-7-9-8-6-2-3 对应的路径编码为(5 4 1 7 9 8 6 2 3)。

(2) 初始种群

理论上,初始种群可以随机选取,这也是最常用的办法,但提高效率可借助启发式方法(如 NEH 方法)快速产生解,并由此产生 GA 的初始种群。

(3) 适配值函数和选择操作的设计

适配值函数用于对各状态的目标值作适当变换,以体现各状态性能的差异。可以采用 $f_x = \exp(-c_x)$ 为适配值函数,其中 c_x 为状态 x 的加工性能指标。

为使赋予适配值高的个体有较高的生存概率,采用比例选择策略,即产生随机数 $\xi \in [0, 1]$,若

$$\sum_{j=1}^{i-1} f_j / \sum_{j=1}^{\text{Pop_size}} f_j < \xi \leq \sum_{j=1}^i f_j / \sum_{j=1}^{\text{Pop_size}} f_j$$

则选择状态 i 进行复制。

(4) 交叉操作

基于路径编码的交叉操作可用 Non-ABEL 群置换操作、次序交叉、循环交叉和部分映射交叉(PMX)等(Goldberg, 1989),下面对它们作简单介绍,具体步骤可参见相关书目。

部分映射交叉首先随机地在父代个体中选取两个交叉点,并交换交叉点之间的片段,再根据该段内的位置确定部分映射,在各父代个体上先填入无冲突的位置,而对有冲突的位置分别执行部分映射直到没有冲突,从而获得后代个体。譬如两个父代个体为 $p_1 = [2\ 6\ 4\ 7\ 3\ 5\ 8\ 9\ 1]$, $p_2 = [4\ 5\ 2\ 1\ 8\ 7\ 6\ 9\ 3]$,若交叉位置为 3,7,则后代个体为 $q_1 = [2\ 3\ 4\ 1\ 8\ 7\ 6\ 9\ 5]$, $q_2 = [4\ 1\ 2\ 7\ 3\ 5\ 8\ 9\ 6]$ 。

Non-ABEL 方法采用如下公式得到后代个体: $q_1[i] = p_1[p_2[i]]$, $q_2[i] = p_2[p_1[i]]$,若父代个体同上,则后代个体为 $q_1 = [7\ 3\ 6\ 2\ 9\ 8\ 5\ 1\ 4]$, $q_2 = [5\ 7\ 1\ 6\ 2\ 8\ 9\ 3\ 4]$ 。

次序交叉与 PMX 非常类似,它首先随机确定两个交叉位置,并交换交叉点之间的片段,其他位置根据父代个体中位置的相对顺序来确定。譬如,若父代个体和交叉点同上,则后代个体为 $q_1 = [4\ 3\ 5\ 1\ 8\ 7\ 6\ 9\ 2]$, $q_2 = [2\ 1\ 6\ 7\ 3\ 5\ 8\ 9\ 4]$ 。

循环交叉将另一个父代个体作为参照以对当前父代个体中的位置进行重组,先与另一父代个体实现一个循环链,并将对应的位置填入相应的位置,循环组成后再将另一个父代个体的位置填入相同的位置。若父代个体同上,则后代个体为 $q_1 = [2\ 5\ 4\ 1\ 8\ 7\ 6\ 9\ 3]$, $q_2 = [4\ 6\ 2\ 7\ 3\ 5\ 8\ 9\ 1]$ 。

交叉操作的目的是组合出继承父代有效模式的新个体,进行解空间中的有效搜索。但是,Non-ABEL 群置换操作产生后代方式简单,过分打乱了父串,不利于保留有效模式;次序交叉和循环交叉对父串的修改幅度也较大。PMX 算子在一定程度上满足 Holland 图式定理的基本性质,子串能够继承父串的有效模式。

(5) 变异操作

对于基于路径编码的变异操作可采用互换操作(SWAP)、逆序操作(INV)、插入操作(INS)方案,前文对此已作了介绍,在此不再赘述。

(6) 算法终止准则

由于最大进化代数很难合适设置,为适应算法性能的动态变化,较好地兼顾算法的优化性能和时间性能,可采用阈值法设计算法终止准则,即若最佳优化值连续若干代进化仍保持不变,则终止搜索过程。

2. 函数优化问题的求解

遗传算法求解函数优化问题与组合优化问题的区别主要在编码和交叉、变异操作上,其他环节基本类似。

对于二进制编码策略,编码采用长度为 l 的二进制串 S 来表示 $[a, b]$ 区间内的变量 x , 则 $x = a + \frac{[S]_2}{2^l - 1}(b - a)$ 。显然,基于这种编码方式的优化精度受串的长度影响,而且算法存储量较大。交叉操作可采用单点或多点交叉,变异操作则采用单个或多个基因变换法。

对于实数编码策略,算法直接用实数来表示变量,交叉操作可采用前文介绍的算术交叉,而变异则可采用附加扰动方式,具体可参考模拟退火算法求解函数优化时的状态产生函数的设计。

以上仅是算法的一个实现方案,后文有关混合策略的应用将给出具体仿真结果,有兴趣的读者也可利用附录中的 Benchmark 问题进行仿真研究,并对算法操作作多样化设计,从而体会 GA 的优化性能和各操作对算法的影响。

通过本章对遗传算法的介绍,我们认识到,GA 不只是一种单纯的优化算法,而是一种以生物进化思想为基础的一般方法论,是解决复杂问题的有力工具。GA 不是传统的确定性计算工具,动态的复杂问题的求解也不可能是确定的,应建立新的评价标准,这在后文也将给予简单介绍。GA 的理论正在深入,应用日趋广泛,但它仅是生物进化系统的简单近似模拟,其本身的发展也是不断进化的过程,理论研究需要引入新的数学工具并吸收生物学的最新成果,应用研究的成败依赖于对 GA 及其所解决问题的深刻理解。随着 GA 理论愈来愈完善,应用领域愈来愈拓宽,在人工生命、其他优化计算技术等领域与 GA 相结合后,GA 将会发挥其更大的潜力。

第 4 章

禁忌搜索算法

禁忌搜索(Tabu Search 或 Taboo Search, 简称 TS)的思想最早由 Glover(1986)提出,它是对局部邻域搜索的一种扩展,是一种全局逐步寻优算法,是对人类智力过程的一种模拟。TS 算法通过引入一个灵活的存储结构和相应的禁忌准则来避免迂回搜索,并通过藐视准则来赦免一些被禁忌的优良状态,进而保证多样化的有效探索以最终实现全局优化。相对于模拟退火和遗传算法,TS 是又一种搜索特点不同的 meta-heuristic 算法。迄今为止,TS 算法在组合优化、生产调度、机器学习、电路设计和神经网络等领域取得了很大的成功,近年来又在函数全局优化方面得到较多的研究,并大有发展的趋势。本章将主要介绍禁忌搜索的优化流程、原理、算法收敛理论与实现技术等内容。

4.1 禁忌搜索

4.1.1 引言

局部邻域搜索是基于贪婪思想持续地在当前解的邻域中进行搜索,虽然算法通用易实现,且容易理解,但其搜索性能完全依赖于邻域结构和初始解,尤其容易陷入局部极小而无法保证全局优化性。针对局部邻域搜索,为了实现全局优化,可尝试的途径有:以可控性概率接受劣解来逃逸局部极小,如模拟退火算法;扩大邻域搜索结构,如 TSP 的 2opt 扩展到 k-opt;多点并行搜索,如进化计算;变结构邻域搜索(Mladenovic et al, 1997);另外,就是采用 TS 的禁忌策略尽量避免迂回搜索,它是一种确定性的局部极小突跳

策略。

禁忌搜索是人工智能的一种体现,是局部邻域搜索的一种扩展。禁忌搜索最重要的思想是标记对应已搜索到的局部最优解的一些对象,并在进一步的迭代搜索中尽量避开这些对象(而不是绝对禁止循环),从而保证对不同的有效搜索途径的探索。禁忌搜索涉及到邻域(neighborhood)、禁忌表(tabu list)、禁忌长度(tabu length)、候选解(candidate)、藐视准则(aspiration criterion)等概念,我们首先用一个示例来理解禁忌搜索及其各重要概念,而后给出算法的一般流程。

4.1.2 禁忌搜索示例

组合优化是 TS 算法应用最多的领域。置换问题,如 TSP、调度问题等,是一大批组合优化问题的典型代表,在此用它来解释简单的禁忌搜索算法的思想和操作。对于 n 元素的置换问题,其所有排列状态数为 $n!$,当 n 较大时搜索空间的大小将是天文数字,而禁忌搜索则希望仅通过探索少数解来得到满意的优化解。

首先,我们对置换问题定义一种邻域搜索结构,如互换操作(SWAP),即随机交换两个点的位置,则每个状态的邻域解有 $C_n^2 = n(n-1)/2$ 个。称从一个状态转移到其邻域中的另一个状态为一次移动(move),显然每次移动将导致适配值(反比于目标函数值)的变化。其次,我们采用一个存储结构来区分移动的属性,即是否为禁忌“对象”。在以下示例中:考虑 7 元素的置换问题,并用每一状态的相应 21 个邻域解中最优的 5 次移动(对应最佳的 5 个适配值)作为候选解;为一定程度上防止迂回搜索,每个被采纳的移动在禁忌表中将滞留 3 步(即禁忌长度),即次移动在以下连续 3 步搜索中将被视为禁忌对象;需要指出的是,由于当前的禁忌对象对应状态的适配值可能很好,因此在算法中设置判断,若禁忌对象对应的适配值优于“best so far”状态,则无视其禁忌属性而仍采纳其为当前选择,也就是通常所说的藐视准则(或称特赦准则)。

第 1 步:当前解 适配值:0

2	5	7	3	4	6	1
---	---	---	---	---	---	---

候选解

禁忌表

	2	3	4	5	6	7
1						
	2					
		3				
			4			
				5		
					6	

SWAP 与当前解的适配值差

5,4	6*
7,4	4
3,6	2
2,3	0
4,1	-1

第 1 步可理解为:随机初始状态为(2 5 7 3 4 6 1),其适配值为 10,禁忌表被初始化为空,由于在当前解由 SWAP 操作得到的最佳 5 个候选解中(5 4)互换后得到解的适配值为 16,因此当前解更新为(2 4 7 3 5 6 1),并把(5 4)加入到禁忌表中。

第 2 步:当前解 适配值:16

2	4	7	3	5	6	1
---	---	---	---	---	---	---

候选解

禁忌表

	2	3	4	5	6	7
1						
	2					
		3				
			4	3		
				5		
					6	

SWAP 与当前解的适配值差

3,1	2*
2,3	1
3,6	-1
7,1	-2
6,1	-4

对于第 2 步,当前状态为(2 4 7 3 5 6 1),其适配值为 16,此时禁忌表中(4 5)处为 3,表示它将被禁忌 3 步。此时,当前解的候选解中(3 1)互换将使适配值增加 2,从而当前解更新为(2 3 7 1 5 6 3),并把(3 1)加入到禁忌表中。

第 3 步:当前解 适配值:18

2	4	7	1	5	6	3
---	---	---	---	---	---	---

候选解

禁忌表

	2	3	4	5	6	7
1		3				
	2					
		3				
			4	2		
				5		
					6	

SWAP 与当前解的适配值差

1,3	-2T
2,4	-4*
7,6	-6
4,5	-7T
5,3	-9

对于第 3 步,当前解为(2 3 7 1 5 6 3),其适配值为 18,由于(4 5)已被禁忌了一步,因此它在禁忌表中的值减至 2(当值减为 0 时解禁),而(1 3)对应的值为 3。此时,由于当前解的 5 个最佳候选解都不能使适配值得到提高,而禁忌算法却无视这一点(这也是算法实现局部解突跳的一个关键点),同时由于(1 3)和(4 5)是禁忌对象,因此算法在候选解集中选择非禁忌的最佳候选解为下一个当前状态,即互换(2 4)导致的解,其适配值降为 14,并把(2 4)加入禁忌表。

第 4 步:当前解 适配值:14

4	2	7	1	5	6	3
---	---	---	---	---	---	---

候选解

禁忌表

	2	3	4	5	6	7
1		2				
	2		3			
		3				
			4	1		
				5		
					6	

SWAP 与当前解的适配值差

4,5	6T*
5,3	2
7,1	0
1,3	-3T
2,6	-6

对于第 4 步,当前解为(4 2 7 1 5 6 3),其适配值为 14,(4 5)、(1 3)在禁忌表中的值相应减少,而(2 4)对应的值为 3。此时,当前解的 5 个最佳候选解中,虽然互换(4 5)是禁忌对象,但由于它导致的适配值为 20,优于“best so far”状态,因此算法仍选择它为下一个当前状态,即(5 2 7 1 4 6 3),并重新置(4 5)在禁忌表中的值为 3,这就是藐视准则为防止遗失最优解的作用。进而,搜索过程转入第 5 步,并按相同的机理持续到算法终止条件成立。

第 5 步:当前解 适配值:20

5	2	7	1	4	6	3
---	---	---	---	---	---	---

候选解

禁忌表

	2	3	4	5	6	7
1		1				
	2		2			
		3				
			4	3		
				5		
					6	

SWAP 与当前解的适配值差

7,1	0*
4,3	-3
6,3	-5
5,4	-6T
2,6	-8

可见,简单的禁忌搜索是在邻域搜索的基础上,通过设置禁忌表来禁忌一些已经历的操作,并利用藐视准则来奖励一些优良状态,其中邻域结构、候选解、禁忌长度、禁忌对象、藐视准则、终止准则等是影响禁忌搜索算法性能的关键。需要指出的是:

(1) 首先,由于 TS 是局部邻域搜索的一种扩充,因此邻域结构的设计很关键,它决定了当前解的邻域解的产生形式和数目,以及各个解之间的联系。

(2) 其次,出于改善算法的优化时间性能的考虑,若邻域结构决定了大量的邻域解(尤其对大规模问题,如 TSP 的 SWAP 操作将产生 C_n^2 个邻域解),则可以仅尝试部分互

换的结果,而候选解也仅取其中的少量最佳状态。

(3) 禁忌长度是一个很重要的关键参数,它决定禁忌对象的任期,其大小直接进而影响整个算法的搜索进程和行为。同时,以上示例中,禁忌表中禁忌对象的替换是采用FIFO方式(不考虑藐视准则的作用),当然也可以采用其他方式,甚至是动态自适应的方式。

(4) 藐视准则的设置是算法避免遗失优良状态,激励对优良状态的局部搜索,进而实现全局优化的关键步骤。

(5) 对于非禁忌候选状态,算法无视它与当前状态的适配值的优劣关系,仅考虑它们中间的最佳状态为下一步决策,如此可实现对局部极小的突跳(是一种确定性策略)。

(6) 为了使算法具有优良的优化性能或时间性能,必须设置一个合理的终止准则来结束整个搜索过程。

此外,在许多场合禁忌对象的被禁次数(frequency)也被用于指导搜索,以取得更大的搜索空间。禁忌次数越高,通常可认为出现循环搜索的概率越大。譬如:

第 k 步:当前解 适配值:12							候选解		
1	3	6	2	7	5	4			
禁忌表							SWAP 与当前解 惩罚值 的适配值差		
	1	2	3	4	5	6	7		
1				3				1,4	3
2								2,4	-1
3	3					2		3,7	-3
4	1	5					1	1,6	-5
5		4		4				6,5	-4
6			1		2				
7				3					
								3T	-6
								-3*	
								-5	
								-6	

上述禁忌表左下角矩阵中各元素表示禁忌对象到当前搜索步数的被禁次数,而右上角矩阵中各元素表示当前禁忌对象的任期,即(4 1)、(6 3)、(7 4)为当前最近的3次禁忌对象。考虑到被禁次数,可以用惩罚值替代与当前解的适配值差来作决策,从而驱动搜索分散到其他区域。譬如,对每一性能无改进的非禁忌的SWAP操作,设置其惩罚值为它对应的搜索状态与当前状态的适配值的差减去其被禁次数。那么,对于第k步搜索,首先,(1 4)是禁忌对象且不满足藐视准则,因此不被采纳为下一步决策;其次,虽然(2 4)对应的状态与当前状态的差在非禁忌状态中是最好的,但它已被禁忌5次,因此将其惩罚值设置为-6;(3 7)是非禁忌对象,其惩罚值为-3,这在所有候选解中是最好的,因此取其为下一步决策,也即新的当前解将是(1 7 6 2 3 5 4)。显然,这种操作的目的是避免搜索过分集中在某些操作上,即避免搜索很大程度上集中在某区域中。

4.1.3 禁忌搜索算法流程

通过上述示例的介绍,基本上了解了禁忌搜索的机制和步骤。简单 TS 算法的基本思想是:给定一个当前解(初始解)和一种邻域,然后在当前解的邻域中确定若干候选解;若最佳候选解对应的目标值优于“best so far”状态,则忽视其禁忌特性,用其替代当前解和“best so far”状态,并将相应的对象加入禁忌表,同时修改禁忌表中各对象的任期;若不存在上述候选解,则选择在候选解中选择非禁忌的最佳状态为新的当前解,而无视它与当前解的优劣,同时将相应的对象加入禁忌表,并修改禁忌表中各对象的任期;如此重复上述迭代搜索过程,直至满足停止准则。

条理化些,则简单禁忌搜索的算法步骤可描述如下:

- (1) 给定算法参数,随机产生初始解 x ,置禁忌表为空。
- (2) 判断算法终止条件是否满足?若是,则结束算法并输出优化结果;否则,继续以下步骤。
- (3) 利用当前解 x 的邻域函数产生其所有(或若干)邻域解,并从中确定若干候选解。
- (4) 对候选解判断藐视准则是否满足?若成立,则用满足藐视准则的最佳状态 y 替代 x 成为新的当前解,即 $x = y$,并用与 y 对应的禁忌对象替换最早进入禁忌表的禁忌对象,同时用 y 替换“best so far”状态,然后转步骤 6;否则,继续以下步骤。
- (5) 判断候选解对应的各对象的禁忌属性,选择候选解集中非禁忌对象对应的最佳状态为新的当前解,同时用与之对应的禁忌对象替换最早进入禁忌表的禁忌对象元素。
- (6) 转步骤(2)。

同时,上述算法可用如下流程框图更直观地描述,如图 4.1.1。

我们可以明显地看到,邻域函数、禁忌对象、禁忌表和藐视准则,构成了禁忌搜索算法的关键。其中,邻域函数沿用局部邻域搜索的思想,用于实现邻域搜索;禁忌表和禁忌对象的设置,体现了算法避免迂回搜索的特点;藐视准则,则是对优良状态的奖励,它是对禁忌策略的一种放松。需要指出的是,上述算法仅是一种简单的禁忌搜索框架,对各关键环节复杂和多样化的设计则可构造出各种禁忌搜索算法。同时,算法流程中的禁忌对象,可以是搜索状态,也可以是特定搜索操作,甚至是搜索目标值等。

同时,与传统的优化算法相比,TS 算法的主要特点是:

- (1) 在搜索过程中可以接受劣解,因此具有较强的“爬山”能力;
- (2) 新解不是在当前解的邻域中随机产生,而或是优于“best so far”的解,或是非禁忌的最佳解,因此选取优良解的概率远远大于其他解。

由于 TS 算法具有灵活的记忆功能和藐视准则,并且在搜索过程中可以接受劣解,所以具有较强的“爬山”能力,搜索时能够跳出局部最优解,转向解空间的其他区域,从而增强获得更好的全局最优解的概率,所以 TS 算法是一种局部搜索能力很强的全局迭代寻

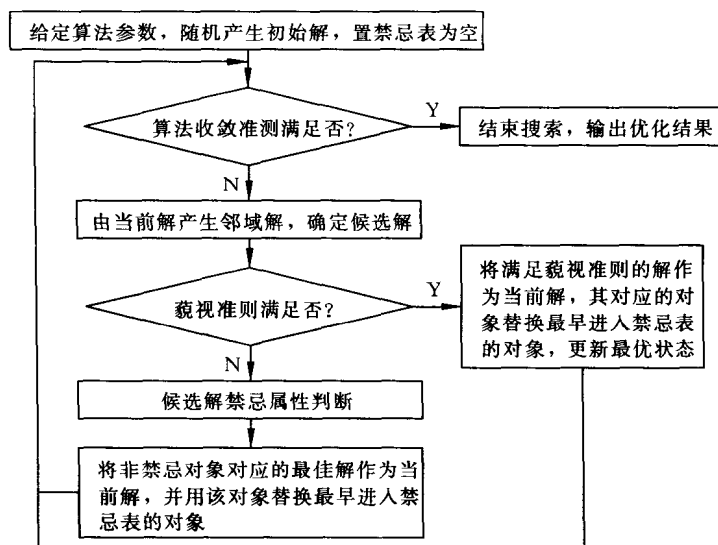


图 4.1.1 简单禁忌搜索算法的流程图

优算法。但是, TS 也有明显的不足, 即: (1) 对初始解有较强的依赖性, 好的初始解可使 TS 在解空间中搜索到好的解, 而较差的初始解则会降低 TS 的收敛速度; (2) 迭代搜索过程是串行的, 仅是单一状态的移动, 而非并行搜索。为了进一步改善禁忌搜索的性能, 一方面可以对禁忌搜索算法本身的操作和参数选取进行改进, 另一方面则可以与模拟退火、遗传算法、神经网络以及基于问题信息的局部搜索相结合。

4.2 禁忌搜索的收敛性

迄今为止, 禁忌搜索算法在许多领域得到了成功应用。尽管许多文献通过仿真研究来探讨参数和操作对算法性能的影响, 但其理论研究还远不完善, 尤其是对收敛性、收敛度、收敛速度等的研究。

本节针对如下一类问题来介绍禁忌搜索算法的收敛性:

$$(P) \text{ Minimize } c(x), x \in X$$

其中, $c(\cdot)$ 为目标函数, X 为有限状态集 (即搜索状态空间), x 为搜索状态。

出于图论的观点, 有限状态优化问题的禁忌搜索可用图 $G_N = (V, A)$ 表示, 其中 N 为邻域结构, V 为所有状态构成的顶点集, $A = \{e; \forall x, e = (x, x'), \text{ if } x' \in N(x)\}$ 为相应的边集。令 k 为算法迭代步数, S_k 为算法至第 k 步迭代所搜索到的所有解的集合, 下面来讨论如下一类 TS 算法的收敛性。

(1) 随机产生初始解 $x \in X$, 令 $k=1$, 当前解 $x_k = x$ 。

(2) 若 x_k 的所有邻域解已被测试过, 即 $N(x_k) \subseteq S_k$, 则选取最早被测试的解 x_{i^*} , 即 $x_{k+1} = x_{i^*}$, $i^* = \min\{i; x_i \in N(x_k)\}$, 更新 x_{i^*} 的测试时刻, 令 $S_{k+1} = S_k - \{x_{i^*}\} + \{x_{k+1}\}$ 。

(3) 若第(2)步不成立, 则按一定准则选取未测试解 $x_{k+1} \in (N(x_k) - S_k)$, 令 $S_{k+1} = S_k + \{x_{k+1}\}$ 。

(4) 若算法终止准则满足, 则结束搜索并输出结果; 否则, 令 $k=k+1$, 并转步骤 2。

上述算法中的第(2)、(3)步可认为是禁忌搜索的抽象描述。另外, 从图论的观点来看, 当循环不可避免时选择过去最早测试的解, 也即选择构成最大循环长度的顶点, 其中长度定义为一个顶点通过邻域到达另一个顶点的最少转移步数。同时, 在此规定算法终止条件为所有状态被测试, 即 $S_k = X$ 。

熟知, 对邻域搜索算法, 邻域结构的设计是影响算法质量和效率的关键因素。为了证明算法的收敛性, 我们首先对邻域结构作如下假设:

(1) 邻域结构是对称的, 即 $\forall x, x' \in X, x \in N(x') \Leftrightarrow x' \in N(x)$;

(2) 图 G_N 是强连通的, 即对任意 $x, x' \in X$, 一定存在一条由 x 到 x' 的路径。

需要指出的是, 上述两条条件并不过强, 组合优化问题的许多邻域结构均满足这些条件, 我们在讨论模拟退火算法收敛性时也曾涉及到这些条件。

引理 4.2.1 序列 $\{S_k\} (k \geq 1)$ 非降, 且收敛到极限 S^* 。

证明 由 $\{S_k\} (k \geq 1)$ 的构造途径知, $S_k \subseteq S_{k+1}$, 因此 $\{S_k\} (k \geq 1)$ 是非降的。同时, 由于搜索空间为有限集, 因此 S_k 必然收敛到某一极限 S^* 。

引理 4.2.2 若邻域结构满足上述两个假设, 即对称性和连通性, 同时存在迭代步数 k^* 使得对所有 $x \in S_{k^*}$, $N(x) \subseteq S_{k^*}$ 成立, 则 $S_{k^*} = X$, 即算法收敛。

证明 采用反证法。设 $S_{k^*} \neq X$, 则由图的连通性知, 至少存在一条路径将 $X - S_{k^*}$ 中的一个解(记为 y)与 S_{k^*} 中的一个解(记为 x)连通。令 y 到 x 的路径为 $P(y, x) = (y_1 = y, y_2, \dots, y_p = x)$, 其中 $y_{i+1} \in N(y_i), i = 1, \dots, p-1$ 。考虑路径 $P(y, x)$ 中的第一个解 y_{i^*} 使得 $y_{i^*} \in X - S_{k^*}, y_{i^*+1} \in S_{k^*}$ 且 $y_{i^*+1} \in N(y_{i^*})$ 。由邻域结构的对称性知, $y_{i^*} \in N(y_{i^*+1})$ 。但是, 既然 $y_{i^*+1} \in S_{k^*}$, 则 $N(y_{i^*+1}) \subseteq S_{k^*}$, 进而 $y_{i^*} \in S_{k^*}$ 。显然有矛盾产生, 因此引理成立。

下面, 我们来证明极限集 S^* 满足对所有 $x \in S^*, N(x) \subseteq S^*$ 。首先将 S_k 分为两个不重叠的子集 A_k 和 $B_k, A_k = \{x_i \in S_k; i < h_k\}, B_k = S_k - A_k = \{x_i \in S_k; h_k \leq i \leq k\}, h_k = \min\{h; N(x_i) \subseteq S_k, i \leq h \leq k\}$ 。若不存在 h 使得对所有 $p \leq h, N(x_p) \subseteq S_k$, 则令 $h_k = 0$ 。对于一个大 k , 算法将足够次执行步骤 2 来产生解, 因此 h_k 意义明确。

引理 4.2.3 $\{A_k\} (k \geq 1)$ 和 $\{B_k\} (k \geq 1)$ 是单调的, 并且分别收敛到 A^* 和 B^* , 其中 A^* 和 B^* 构成集合 S^* 的一个分解。

证明 首先, 由集合的构造知, 对所有 $k \geq 1, A_{k+1} \subseteq A_k, B_k \subseteq B_{k+1}$, 因此 $\{A_k\} (k \geq 1)$ 是非增序列, 而 $\{B_k\} (k \geq 1)$ 是非降序列。同时, 由于 $\{A_k\} (k \geq 1)$ 和 $\{B_k\} (k \geq 1)$ 的界分别为

空集和 X , 且 $A_k \cap B_k = \emptyset$, $A_k \cup B_k = S_k$, 因此两集合分别将收敛到两个极限集, 且构成 S^* 的一种分区。

引理 4.2.4 令 A^* 和 B^* 分别为 $\{A_k\} (k \geq 1)$ 和 $\{B_k\} (k \geq 1)$ 的极限, 则

- (1) 对所有 $x \in B^*$, $N(x) \cap A^* = \emptyset$;
- (2) 对所有 $x \in A^*$, $N(x) \cap B^* = \emptyset$;
- (3) 对所有 $x \in B^*$, $N(x) \subseteq B^*$;
- (4) $A^* = \emptyset$ 。

证明

(1) 假设存在解 $x_k \in B^*$, 使得 $N(x_k) \cap A^* \neq \emptyset$ 。既然 $x_k \in B^*$, 则有 $h_k \leq k$, $N(x_k) \subseteq S_k$ 。则按算法步骤 2, 在 $k+1$ 步迭代时有 $x_{k+1} = x_{i^*}$, $i^* = \min\{i; x_i \in N(x_k)\}$ 。既然 $N(x_k) \cap A^* \neq \emptyset$ 且 $A^* \subseteq S_k$, 则 i^* 可表示为 $i^* = \min\{i; x_i \in N(x_k) \cap A^*\}$ 。这意味着 $x_{k+1} \in A^* \cap B^*$ 。这显然与引理 4.2.3 相矛盾。因此, 对所有 $x \in B^*$, $N(x) \cap A^* = \emptyset$ 。

(2) 假设存在 $x_k \in A^*$, $N(x_k) \cap B^* \neq \emptyset$ 。令 $x_i \in N(x_k) \cap B^*$, 由于 $x_i \in B^*$, 则 $N(x_i) \cap A^* = \emptyset$ 。由于 $x_k \in A^*$, 且 $x_k \in N(x_i)$, 则 $x_k \in N(x_i) \cap A^*$ 。从而矛盾产生, 因此对所有 $x \in A^*$, $N(x) \cap B^* = \emptyset$ 。

(3) 令 $x_k \in B^*$, 由 B_k 的定义知, $k > h_k$, $N(x_k) \subseteq S_k$ 。另一方面, 由引理 4.2.3 知, $S_k \subseteq S^* = A^* \cup B^*$, 这意味着 $N(x_k) \subseteq A^* \cup B^*$ 。由于 $N(x_k) \cap A^* = \emptyset$, 则 $N(x_k) \subseteq B^*$ 。

(4) 采用反证法。考虑 $x_{i^*} \in A^*$, $i^* = \max\{i; x_i \in A^*\}$, 这意味着 $i^* < h^* = h_k$ 。从而, 存在路径 $P(x_{i^*}, x_{h^*})$ 连通 x_{i^*} 和 x_{h^*} , 此路径上 x_{i^*} 的下一个继承解属于 B^* , 且是 x_{i^*} 的邻域解, 从而导致 $N(x_{i^*}) \cap B^* \neq \emptyset$, 这显然与结论(2)矛盾。因此, $A^* = \emptyset$ 。

定理 4.2.1 假设 X 为有限空间, 且对邻域结构的两个假设成立, 则禁忌搜索算法必然收敛且找到最优解。

证明 由引理 4.2.3 以及引理 4.2.4 的(3)和(4)知, 对所有 $x \in S^*$, $N(x) \subseteq S^*$ 。由引理 4.3.1 知, S_k 具有极限 S^* , 这意味着存在某一迭代步数 k^* , 使得对所有 $k > k^*$, $S^* = S_k$ 。进而, 由引理 4.3.2 知定理 4.3.1 成立。

上述收敛定理的意义可归纳为: 若有限状态空间对由当前解的邻域解集中的非禁忌或满足藐视准则的候选解集是连通的(即任意两个状态可通过有限步邻域搜索互达), 并且禁忌表的大小充分大, 则禁忌搜索一定能够达到全局最优解。然而, 要使禁忌表的大小充分大, 即遍历所有状态, 显然这在时间上是不可承受的。同时, 上述定理的证明是在禁忌准则和藐视准则的抽象意义上进行的, 并没有指出算法操作对性能的具体作用, 尤其没有涉及到算法效率。因此在理论上, 操作和参数对算法性能的影响以及算法搜索效率有待进一步研究, 这也有助于开发高效的禁忌搜索算法。

4.3 禁忌搜索的关键参数和操作

一般而言, 要设计一个禁忌搜索算法, 需要确定算法的以下环节:

- (1) 初始解和适配值函数;
- (2) 邻域结构和禁忌对象;
- (3) 候选解选择;
- (4) 禁忌表及其长度;
- (5) 藐视准则;
- (6) 集中搜索和分散搜索策略;
- (7) 终止准则。

本节主要从实现技术上介绍禁忌搜索算法最基本的操作和参数的常用设计原则和方法,包括适配值函数、禁忌对象、禁忌长度、候选解、藐视准则、禁忌频率和终止准则等,而其他环节和具体的实现方案将在后文阐述。

1. 适配值函数

类似于遗传算法,禁忌搜索的适配值函数也是用于对搜索状态的评价,进而结合禁忌准则和藐视准则来选取新的当前状态。显然,目标函数直接作为适配值函数是比较容易理解做法。当然,目标函数的任何变形都可作为适配值函数,譬如对极小化问题可将状态 x 的适配值 $f(x)$ 定义为 $M - c(x)$ 或 $e^{-ac(x)}$,其中 M 为一足够大正数, $c(x)$ 为目标值, $a > 0$ 。

若目标函数的计算比较困难或耗时较多,如一些复杂工业过程的目标函数值需要一次仿真才能获得,此时可采用反映问题目标的某些特征值来作为适配值,进而改善算法的时间性能。当然,选取何种特征值要视具体问题而定,但必须保证特征值的最佳性与目标函数的最优性一致。

2. 禁忌对象

所谓禁忌对象就是被置入禁忌表中的那些变化元素,而禁忌的目的则是为了尽量避免迂回搜索而多探索一些有效的搜索途径。归纳而言,禁忌对象通常可选取状态本身或状态分量或适配值的变化等。

(1) 以状态本身或其变化作为禁忌对象是最为简单、最容易理解的途径。具体而言,当状态由 x 变化到状态 y 时,将状态 y (或 $x \rightarrow y$ 的变化) 视为禁忌对象,从而在一定条件下禁止了 y (或 $x \rightarrow y$ 的变化) 的再度出现。

例:五城市 TSP 问题,设禁忌长度为 3,以 SWAP 为邻域操作(解的起点固定为 1),规定 $x \notin N(x)$,候选解选当前解邻域解集中的最佳 4 个解,问题的距离矩阵如下:

$$(d_{ij}) = \begin{bmatrix} 0 & 10 & 15 & 6 & 2 \\ 10 & 0 & 8 & 13 & 9 \\ 15 & 8 & 0 & 20 & 15 \\ 6 & 13 & 20 & 0 & 5 \\ 2 & 9 & 15 & 5 & 0 \end{bmatrix}$$

第 1 步:当前状态为(1 2 3 4 5;45),其中 45 为目标值。此时禁忌表 $H = \emptyset$,候选解

集为 $\{(1\ 3\ 2\ 4\ 5;43)、(1\ 4\ 3\ 2\ 5;45)、(1\ 2\ 5\ 4\ 3;59)、(1\ 2\ 3\ 5\ 4;44)\}$,则选取 $(1\ 3\ 2\ 4\ 5;43)$ 为新的当前解,并令 $H=\{(1\ 3\ 2\ 4\ 5;43)\}$ 。

第2步:当前解为 $(1\ 3\ 2\ 4\ 5;43)$, $H=\{(1\ 3\ 2\ 4\ 5;43)\}$,候选解集为 $\{(1\ 3\ 2\ 5\ 4;43)、(1\ 4\ 2\ 3\ 5;44)、(1\ 2\ 3\ 4\ 5;45)、(1\ 3\ 5\ 4\ 2;58)\}$,则选取 $(1\ 3\ 2\ 5\ 4;43)$ 为新的当前解,并令 $H=\{(1\ 3\ 2\ 4\ 5;43)、(1\ 3\ 2\ 5\ 4;43)\}$ 。

第3步:当前解为 $(1\ 3\ 2\ 5\ 4;43)$, $H=\{(1\ 3\ 2\ 4\ 5;43)、(1\ 3\ 2\ 5\ 4;43)\}$,候选解集为 $\{(1\ 3\ 2\ 4\ 5;43)、(1\ 2\ 3\ 5\ 4;44)、(1\ 5\ 2\ 3\ 4;45)、(1\ 4\ 2\ 5\ 3;58)\}$,则选取 $(1\ 2\ 3\ 5\ 4;44)$ 为新的当前解,并令 $H=\{(1\ 3\ 2\ 4\ 5;43)、(1\ 3\ 2\ 5\ 4;43)、(1\ 2\ 3\ 5\ 4;44)\}$ 。

第4步:当前解为 $(1\ 2\ 3\ 5\ 4;44)$, $H=\{(1\ 3\ 2\ 4\ 5;43)、(1\ 3\ 2\ 5\ 4;43)、(1\ 2\ 3\ 5\ 4;44)\}$,候选解集为 $\{(1\ 3\ 2\ 5\ 4;43)、(1\ 5\ 3\ 2\ 4;44)、(1\ 2\ 3\ 4\ 5;45)、(1\ 2\ 4\ 5\ 3;58)\}$,则选取 $(1\ 5\ 3\ 2\ 4;44)$ 为新的当前解,并令 $H=\{(1\ 3\ 2\ 5\ 4;43)、(1\ 2\ 3\ 5\ 4;44)、(1\ 5\ 3\ 2\ 4;44)\}$ 。

第5步:当前解为 $(1\ 5\ 3\ 2\ 4;44)$, $H=\{(1\ 3\ 2\ 5\ 4;43)、(1\ 2\ 3\ 5\ 4;44)、(1\ 5\ 3\ 2\ 4;44)\}$,候选解集为 $\{(1\ 5\ 4\ 2\ 3;43)、(1\ 2\ 3\ 5\ 4;44)、(1\ 5\ 3\ 4\ 2;44)、(1\ 5\ 2\ 3\ 4;45)\}$,则选取 $(1\ 5\ 4\ 2\ 3;43)$ 为新的当前解。

(2) 状态的变化包含了多个状态分量的变化,因此以状态分量的变化为禁忌对象将扩大禁忌的范围,并可减少相应的计算量。譬如,对置换问题,SWAP操作引起的两点互换意味着状态分量的变化,这就可作为禁忌对象;对高维函数优化问题,则可将某一维分量本身或其变化作为禁忌对象。对于上述例子:

第1步:当前状态为 $(1\ 2\ 3\ 4\ 5;45)$, $H=\emptyset$,候选解集为 $\{(1\ 3\ 2\ 4\ 5;43)、(1\ 4\ 3\ 2\ 5;45)、(1\ 2\ 5\ 4\ 3;59)、(1\ 2\ 3\ 5\ 4;44)\}$,则选取 $(1\ 3\ 2\ 4\ 5;43)$ 为新的当前解,并令 $H=\{(2\ 3)\}$ 。

第2步:当前解为 $(1\ 3\ 2\ 4\ 5;43)$, $H=\{(2\ 3)\}$,候选解集为 $\{(1\ 3\ 2\ 5\ 4;43)、(1\ 4\ 2\ 3\ 5;44)、(1\ 2\ 3\ 4\ 5;45)、(1\ 3\ 5\ 4\ 2;58)\}$,则选取 $(1\ 3\ 2\ 5\ 4;43)$ 为新的当前解,并令 $H=\{(2\ 3)、(4\ 5)\}$ 。

第3步:当前解为 $(1\ 3\ 2\ 5\ 4;43)$, $H=\{(2\ 3)、(4\ 5)\}$,候选解集为 $\{(1\ 3\ 2\ 4\ 5;43)、(1\ 2\ 3\ 5\ 4;44)、(1\ 5\ 2\ 3\ 4;45)、(1\ 4\ 2\ 5\ 3;58)\}$,这时候候选解集中有多个状态被禁忌,因此选取 $(1\ 5\ 2\ 3\ 4;45)$ 为新的当前解,并令 $H=\{(2\ 3)、(4\ 5)、(3\ 5)\}$ 。

当然,状态分量变化的方向也是可以同时考虑的,即视 $2\rightarrow3$ 和 $3\rightarrow2$ 为不同。

(3) 类似等高线的原理,以适配值或其变化为禁忌对象则将处于同一适配值的状态视为相同状态,这在函数优化中经常采用。由于一个值的变化隐含着多个状态的变化,因此这种情况下的禁忌范围相对于状态的变化将有所扩大。对于上述例子:

第1步:当前状态为 $(1\ 2\ 3\ 4\ 5;45)$, $H=\emptyset$,候选解集为 $\{(1\ 3\ 2\ 4\ 5;43)、(1\ 4\ 3\ 2\ 5;45)、(1\ 2\ 5\ 4\ 3;59)、(1\ 2\ 3\ 5\ 4;44)\}$,则选取 $(1\ 3\ 2\ 4\ 5;43)$ 为新的当前解,并令 $H=\{43\}$ 。

第2步:当前解为 $(1\ 3\ 2\ 4\ 5;43)$, $H=\{43\}$,候选解集为 $\{(1\ 3\ 2\ 5\ 4;43)、(1\ 4\ 2\ 3\ 5;44)、(1\ 2\ 3\ 4\ 5;45)、(1\ 3\ 5\ 4\ 2;58)\}$,则选取 $(1\ 4\ 2\ 3\ 5;44)$ 为新的当前解,并令 $H=$

{43,44}。

第3步: 当前解为(1 4 2 3 5;44), $H=\{43,44\}$, 候选解集为{(1 3 2 4 5;43)、(1 4 5 3 2;44)、(1 5 2 3 4;45)、(1 4 3 2 5;45)}, 这时候候选解集中有多个状态被禁忌。

可见, 以状态本身为禁忌对象比以状态分量或适配值为禁忌对象的禁忌范围要小, 从而给予的搜索范围要大, 容易造成计算时间的增加。然而, 在禁忌长度和候选解集大小相同且较小的情况下, 后两者也会因禁忌范围过大而使搜索陷入局部极小。

3. 禁忌长度和候选解

禁忌长度和候选解集的大小是影响 TS 算法性能的两个关键参数。所谓禁忌长度, 即禁忌对象在不考虑藐视准则情况下不允许被选取的最大次数(通俗些, 可视为对象在禁忌表中的任期), 对象只有当其任期为 0 时才被解禁。候选解集则通常是当前状态的邻域解集的一个子集。在算法的构造和计算过程中, 一方面要求计算量和存储量尽量少, 这就要求禁忌长度和候选解集的尽量小; 但是, 禁忌长度过短将造成搜索的循环, 候选解集过小将容易造成早熟收敛, 即陷入局部极小。

禁忌长度的选取与问题特性、研究者的经验有关, 它决定了算法的计算复杂性。

一方面, 禁忌长度 t 可以是定常不变的, 如将禁忌长度固定为某个数(譬如 $t=3$ 等), 或者固定为与问题规模相关的一个量(譬如 $t=\sqrt{n}$, n 为问题维数或规模), 如此实现很方便、简单。

另一方面, 禁忌长度也可以是动态变化的, 如根据搜索性能和问题特性设定禁忌长度的变化区间 $[t_{\min}, t_{\max}]$ (譬如 $[3, 10]$ 、 $[0.9\sqrt{n}, 1, 1\sqrt{n}]$ 等), 而禁忌长度则可按某种原则或公式在其区间内变化。当然, 禁忌长度的区间大小也可随搜索性能的变化而动态变化。

一般而言, 当算法的性能动态下降较大时, 说明算法当前的搜索能力比较强, 也可能当前解附近极小解形成的“波谷”较深, 从而可设置较大的禁忌长度来延续当前的搜索行为, 并避免陷入局部极小。大量研究表明, 禁忌长度的动态设置方式比静态方式具有更好的性能和鲁棒性, 而更为合理高效的设置方式还有待进一步研究。

候选解通常在当前状态的邻域中择优选取, 但选取过多将造成较大的计算量, 而选取过少则容易造成早熟收敛。然而, 要做到整个邻域的择优往往需要大量的计算, 如 TSP 的 SWAP 操作将产生 C_n^2 个邻域解, 因此可以确定性或随机性地在部分邻域解中选取候选解, 具体数据大小则可视问题特性和对算法的要求而定。

4. 藐视准则

在禁忌搜索算法中, 可能会出现候选解全部被禁忌, 或者存在一个优于“best so far”状态的禁忌候选解, 此时藐视准则将使某些状态解禁, 以实现更高效的优化性能。在此给出藐视准则的几种常用方式。

(1) 基于适配值的准则。全局形式(最常用的方式): 若某个禁忌候选解的适配值优于“best so far”状态, 则解禁此候选解为当前状态和新的“best so far”状态; 区域形式: 将搜索空间分成若干个子区域, 若某个禁忌候选解的适配值优于它所在区域的“best so far”

状态,则解禁此候选解为当前状态和相应区域的新“best so far”状态。该准则可直观理解为算法搜索到了一个更好的解。

(2) 基于搜索方向的准则。若禁忌对象上次被禁时使得适配值有所改善,并且目前该禁忌对象对应的候选解的适配值优于当前解,则对该禁忌对象解禁。该准则可直观理解为算法正按有效的搜索途径进行。

(3) 基于最小错误的准则。若候选解均被禁忌,且不存在优于“best so far”状态的候选解,则对候选解中最佳的候选解进行解禁,以继续搜索。该准则可直观理解为对算法死锁的简单处理。

(4) 基于影响力的准则。在搜索过程中不同对象的变化对适配值的影响有所不同,有的很大,有的较小,而这种影响力可作为一种属性与禁忌长度和适配值来共同构造藐视准则。直观的理解是,解禁一个影响力大的禁忌对象,有助于在以后的搜索中得到更好的解。需要指出的是,影响力仅是一个标量指标,可以表征适配值的下降,也可以表征适配值的上升。譬如,若候选解均差于“best so far”状态,而某个禁忌对象的影响力指标很高,且很快将被解禁,则立刻解禁该对象以期待更好的状态。显然,这种准则需要引入一个标定影响力大小的度量和一个与禁忌任期相关的阈值,无疑增加了算法操作的复杂性。同时,这些指标最好是动态变化的,以适应搜索进程和性能的变化。

5. 禁忌频率

记忆禁忌频率(或次数)是对禁忌属性的一种补充,可放宽选择决策对象的范围。譬如,如果某个适配值频繁出现,则可以推测算法陷入某种循环或某个极小点,或者说现有算法参数难以有助于发掘更好的状态,进而应当对算法结构或参数作修改。在实际求解时,可以根据问题和算法的需要,记忆某个状态的出现的频率,也可以是某些对换对象或适配值等出现的信息,而这些信息又可以是静态的,或者是动态的。

静态的频率信息主要包括状态、适配值或对换等对象在优化过程中出现的频率,其计算相对比较简单,如对象在计算中出现的次数,出现次数与总迭代步数的比,某两个状态间循环的次数等。显然,这些信息有助于了解某些对象的特性,以及相应循环出现的次数等。

动态的频率信息主要记录从某些状态、适配值或对换等对象转移到另一些状态、适配值或对换等对象的变化趋势,如记录某个状态序列的变化。显然,对动态频率信息的记录比较复杂,而它所提供的信息量也较多。常用的方法如下:

(1) 记录某个序列的长度,即序列中的元素个数,而在记录某些关键点的序列中,可以按这些关键点的序列长度的变化来进行计算。

(2) 记录由序列中的某个元素出发后再回到该元素的迭代次数。

(3) 记录某个序列的平均适配值,或者是相应各元素的适配值的变化。

(4) 记录某个序列出现的频率等。

上述频率信息有助于加强禁忌搜索的能力和效率,并且有助于对禁忌搜索算法参数

的控制,或者可基此对相应的对象实施惩罚(如前文示例)。譬如,若某个对象频繁,则可以增加禁忌长度来避免循环;若某个序列的适配值变化较小,则可以增加对该序列所有对象的禁忌长度,反之则缩小禁忌长度;若最佳适配值长时间维持下去,则可以终止搜索进程而认为该适配值已是最优值。

此外,许多改进的禁忌搜索算法还根据频率等信息在算法中增加集中搜索(intensification)和分散搜索(diversification)机制,以增强算法的搜索质量和效率。其中,集中搜索机制强调算法对优良区域的重点搜索,如基于最优或次优状态进行重新初始化或进行多步搜索,加强对应取得最优状态的算法参数的被选取概率等;分散收缩机制则强调拓宽搜索范围,尤其是那些未探索的区域,这与增强遗传算法种群多样性有些类似。显然,集中搜索和分散搜索在某些层面上是矛盾的,而两者对算法性能都有很大影响,因此作为一个较好的禁忌搜索算法,应当具有合理平衡集中搜索与分散搜索的能力,这也是许多论文实现对禁忌搜索算法性能改进的突破口,在此不再作展开介绍。

6. 终止准则

与模拟退火、遗传算法一样,禁忌搜索也需要一个终止准则来结束算法的搜索进程,而严格实现理论上的收敛条件,即在禁忌长度充分大的条件下实现状态空间的遍历,这显然是不切合实际的,因此实际设计算法时通常采用近似的收敛准则。常用方法如下:

(1) 给定最大迭代步数。此方法简单易操作,但难以保证优化质量。

(2) 设定某个对象的最大禁忌频率。即,若某个状态、适配值或对换等对象的禁忌频率超过某一阈值,则终止算法,其中也包括最佳适配值连续若干步保持不变的情况。

(3) 设定适配值的偏离幅度。即,首先有估界算法估计问题的下界,一旦算法中最佳适配值与下界的偏离值小于某规定幅度时,则终止搜索。

4.4 并行禁忌搜索算法

近年来,禁忌搜索算法在许多领域得到了广泛而成功的应用,譬如生产调度、电路设计、电讯工程、神经网络、交通工程等。随着并行计算技术和并行计算机的发展,为满足求解大规模问题的需要,禁忌搜索算法的并行实施也得到了研究和发展。鉴于并行禁忌搜索算法的发展还处于初级阶段,本节对此仅作简单介绍。

相对前文介绍的串行禁忌搜索算法,对算法初始化、参数设置、通讯策略等环节实施不同的并行化方案,则可构造出不同类型的并行禁忌搜索算法。目前,比较认可的一种分类可用图 4.4.1 描述。

1. 基于空间分解的并行策略

(1) 搜索空间的分解策略。即,通过搜索空间分解将原问题分解为若干子问题,各子问题用不同的禁忌搜索算法进行求解,从而实现并行化。

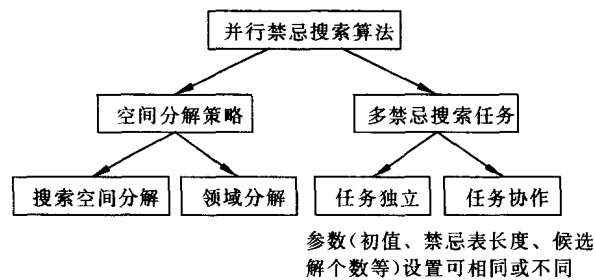


图 4.4.1 并行禁忌搜索算法的分类

(2) 邻域的分解策略。即,每一代中用多种方法对邻域分解所得的各子集进行评价,从而实现对最佳邻域解的搜索的并行化。

显然,这类基于空间分解的并行策略在实施时对同步的要求很高。

2. 基于多禁忌搜索任务的并行策略

顾名思义,这类并行策略中包含多个禁忌搜索算法在运行,而各算法可以使用相同或不同的算法参数(如初值、禁忌表长度、候选解个数等)。同时,各任务可以以不存在通讯的独立方式运行,也可以以协作的方式运行(譬如最优解的共享)。

上述分类中,空间分解或邻域分解策略具有问题依赖性,仅对部分问题适用,而多任务策略比较通用,当然分解策略和多任务策略也可以结合使用。此外,在多处理机情况下,鉴于各任务的数量和定位相对并行机的负荷状态,如静态或动态,又可将并行禁忌搜索分为以下三类。

(1) 非自适应方式。指任务的数量和定位在编译时就生成的情况,且各任务相应的处理机的定位在算法运行过程中是不变的,即静态调度方案。譬如,根据处理机的个数将邻域分解成相应数目的子集。但是,当各处理机的负荷严重不平衡时,必然造成部分处理机的长时间空闲而影响算法的整体搜索效率。需要指出的是,目前所提出的并行禁忌搜索算法大部分仍属于这类运行方式。

(2) 半自适应方式。指任务的数量在编译时就固定,而定位却在运行时确定或改变的情况。其目的是提高非自适应并行方式的性能,其手段则是在处理机间动态地重新分配负荷以实现负荷动态平衡。

(3) 自适应方式。指任务的生成完全是动态变化的情况,如当处理机空闲时则自动生成任务,而当处理机繁忙时则取消任务。Talbi 等(1998)提出了一种自适应的并行禁忌搜索算法,算法由并行而独立的子禁忌搜索算法构成,且各子算法基于不同的参数进行初始化,而各串行任务间不需要通信,并通过对典型 QAP 问题的高效求解验证了算法的有效性。

虽然并行禁忌搜索目前还处于发展阶段,但可以预见,随着应用领域的扩大、理论研究的深入以及并行计算技术的进一步发展,并行禁忌搜索算法将会得到长足的发展和

完善。

4.5 禁忌搜索的实现与应用

随着禁忌搜索算法十多年的发展,它已在许多领域得到广泛且成功的应用。本节将简单介绍禁忌搜索算法在组合优化和函数优化中的实现及应用。

4.5.1 基于禁忌搜索的组合优化

置换 Flow-shop 问题是一类典型的组合优化问题,而且是典型 NP-complete 问题,前文已介绍了基于遗传算法的求解方案。基于禁忌搜索算法的一般设计原则,其算法可以按如下方案实现。

1. 初始解

与模拟退火、遗传算法类似,禁忌搜索的初始解通常可以随机产生,但也可以基于问题信息借助一些启发式方法产生以保证一定的初始性能。就置换 Flow-shop 问题而言,NEH 方法(Nawaz 等, 1983)是一种性能较好的快速构造性方法,借助于 NEH 方法进行初始化已为许多算法所采用(Ben-Daya 等, 1998)。

2. 邻域结构

邻域结构的设计通常与问题有关。就置换 Flow-shop 这类以置换为搜索状态的组合优化问题,常用的方法是互换(SWAP)、插入(INSERT)、逆序(INVERSE)等操作,这与模拟退火的状态产生函数和遗传算法的变异操作相同,而相应的操作自然可成为算法中的禁忌对象。当然,不同的操作将导致邻域解个数及其变化情况的不同,对搜索质量和效率有一定的影响,但目前尚无一般定论。

3. 候选解的选择

首先需要确定的是候选解的数量,然后确定最佳候选解的选取。原则上应当作到对当前解的邻域解的遍历,但当问题规模较大时,当前解的邻域解数量将很大,而考虑到邻域搜索的效率,通常仅取当前解的邻域解集的一个子集作为候选解集。至于最佳候选解的选取,一般的做法是选择候选解集中的满足藐视准则或非禁忌的最优状态。对于置换 Flow-shop 问题而言,状态的最优性可以简单地依据 makespan 等目标值来确定,对于一些目标复杂的问题,则可以借助设计者规定的适配值函数来确定。

4. 禁忌表及其长度

禁忌表是针对禁忌对象所设计的一种结构,而禁忌表长度是算法中的关键参数,前文已介绍了它的若干选取方案。固定长度法将长度固定为某个值,但鉴于动态长度较固定长度表现出的更优越的性能,建议尝试自适应长度法,譬如根据目标值更新的情况或禁忌

频率信息来适当增加或缩短禁忌表长度,具体值或变化标准可以因问题而异。

5. 藐视准则

为了防止优良解的遗失,采用最简单而常用的藐视准则,即若某个状态的性能优于“best so far”状态,则无视其禁忌属性,直接选取它为当前状态。

6. 集中搜索和分散搜索策略

集中搜索策略用于加强对优良解的邻域的进一步探索,简单的处理手段可以是在一定步数的迭代后基于最佳状态进行重新初始化,并对其邻域进行多步趋化性搜索。分散搜索策略则用于拓宽搜索区域尤其是未知区域,简单的处理手段可以是对算法的重新随机初始化,或者是根据频率信息对一些已知对象进行惩罚。读者可以按兴趣尝试多样化的设计方案。

7. 终止条件

兼顾搜索质量和效率,常用的终止条件是给定最优状态连续保持不变的最大持续迭代步数,其具体大小视问题的规模和难度而定。

大量研究表明禁忌搜索算法具有模拟退火、遗传算法等智能优化算法相当的性能,甚至更优越,在此不予介绍仿真结果,有兴趣的读者可以利用本书附录中的 benchmark 问题来测试基于上述设计方案的禁忌搜索算法的性能,也可以同时探讨一些算法参数和操作对性能的影响,并对算法作改进研究。

禁忌搜索算法在组合优化中的应用领域非常广阔,置换 Flow-shop 问题仅是其中的一个典型代表,但调度问题至今仍是禁忌搜索算法应用最广泛而成功的一个领域。譬如, Ponnambalam 等(2000)提出了求解 job shop 的一个禁忌搜索算法; Nowicki 等(1996)和 (Ben-Daya 等, 1998)分别提出了求解 flow shop 的禁忌搜索算法; Verhoeven(1998)研究了资源受限的调度问题的禁忌搜索算法; Lutz 等(1998)利用禁忌搜索算法确定生产线中缓冲区的位置和大小; James 等(1998)利用禁忌搜索算法改善 E/T 调度问题的优化性能; Costamagna 等(1998)提出了设计电讯网络的一个禁忌搜索算法; Martins 等(1998)将禁忌搜索算法应用于公交车网络的设计问题; Talbi 等(1998)设计了 QAP 问题的一个自适应并行禁忌搜索算法; Rego(1998)研究了 TSP 问题的禁忌搜索算法。此外,禁忌搜索还被应用到图分区、频带分配、0-1 背包问题、时间表设计、满意问题、电力系统设计与调度、聚类问题、间歇化工设计、卫星通讯、计算机结构设计、光学工程等领域,在此不再一一列举。需要指出的是,禁忌搜索算法虽然尚处发展之中,但其应用领域远远不止于此,而且大有拓宽的趋势。

4.5.2 基于禁忌搜索的函数优化

禁忌搜索算法是一种有效的组合优化算法, Glover 最早也是针对组合优化提出的。目前,相比于禁忌搜索算法在组合优化中的应用,它在函数优化问题中的应用还很少。函

数优化与组合优化的最大区别在于状态和邻域的特征。函数优化时,状态通常以实数的形式表征,而邻域结构通常定义为以 x 为中心, r 为半径的球,记作 $B(x, r)$,从而所有满足 $\|x' - x\| \leq r$ 的点 x' 均为 x 的邻域解,其中 $\|\cdot\|$ 为 Euclidean 范数。同时,在产生 η 个邻域解时,为了对邻域实现分布较均匀的探索,可以定义一系列半径分别为 $h_0, h_1, \dots, h_\eta$ 的同心球,然后分别在 η 个区域 C_i 中各产生一个点, $C_i(x, h_{i-1}, h_i) = \{x'; h_{i-1} \leq \|x' - x\| \leq h_i\}, i=1, 2, \dots, \eta$ 。更容易实现的一种做法是用超方体替代球,用类似的方法产生规定数目的邻域解,如此通过比较点的座标即可简单地区分点所属的范围。

下面介绍 Chelouah 等(2000)提出的一种增强连续禁忌搜索算法(ECTS),算法尤其强调集中化搜索和分散化搜索的重要性。ECTS 算法采用超方体邻域结构,搜索进程主要由 5 部分组成,即参数设置、分散搜索、最佳有希望区域的搜索、集中搜索、算法终止和结果输出。ECTS 算法的主要搜索行为可描述如下:在搜索过程中,禁忌表记录中心点位置,通过检验每个新产生的邻域解的中心是否已被记录在禁忌表中来判断其禁忌属性,而算法只接受非禁忌的邻域解;同时,算法对最佳的“有希望区域”(promising areas)实施分散化搜索,而凡是目标函数出现不可接受的恶化时就进行新的有希望区域的检测(譬如当所有相关目标函数值的恶化超过设定阈值时,则认为当前解位于有希望区域的球的中心),而相应的中心位置将被存储在希望表中(promising list);为了避免搜索过程回溯到已搜索过的区域,算法对有希望区域的中心也进行禁忌属性判断,即对新检测到的有希望区域的中心判断是否处于希望表中;若在规定步数中没有检测到新的有希望区域(即分散化搜索的结束条件),算法将确定存储于希望表中的最佳有希望区域,然后在该区域进行集中化搜索,即该区域的中心成为新的当前解,并在其小区域中进行禁忌搜索,直到满足算法的终止准则。下面,我们给出 ECTS 的算法框架,并对算法的主要搜索环节进行介绍。

1. ECTS 算法框架

由算法框架可明显地区分算法的 5 个主要进程,其中分散化搜索和集中化搜索环节利用了相同的禁忌搜索流程,即产生邻域解、选择最佳邻域解、表的更新和参数的调整。同时,尽管算法没有明确地给出藐视准则的作用,但却隐含在有希望区域的定义和搜索中,当然也可像前文介绍的组合优化一样,在最佳邻域解的判断时设置藐视准则。

2. 分散化搜索

分散化搜索从一个初始解出发,在其邻域中产生规定数目的邻域解(产生方法如前文介绍),并规定只有不属于禁忌表的邻域解才可接受,然后接受非禁忌的最佳邻域解为新的当前解,而无视它与先前解的优劣性,同时如前文介绍的那样检测和产生一个新的有希望解,如果该解不属于希望球则用之定义一个新的有希望区域,如果算法接受该新的有希望区域,则把希望表中的最差区域替换掉,当新的有希望区域在规定连续步数内未检测到,则终止上述分散化搜索,并根据希望表确定最佳的有希望区域。可以说,算法中禁忌表和希望表的同时使用激励了搜索过程远离起点和对同一有希望区域的搜索,从而起到

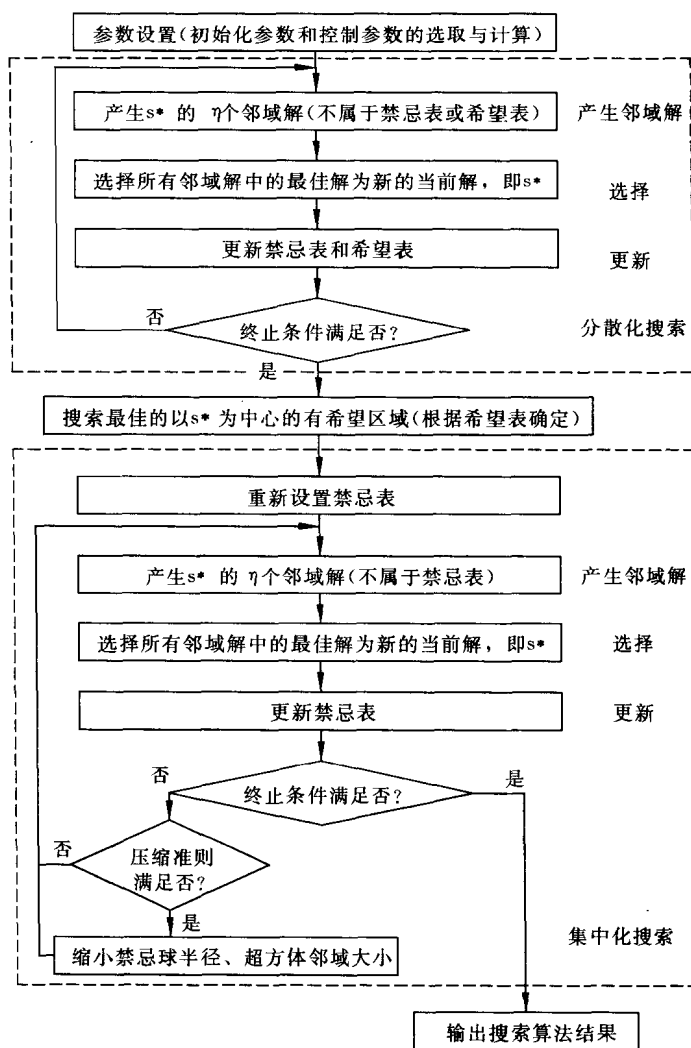


图 4.5.1 ECTS 算法框架

分散化搜索的目的。

3. 最佳有希望区域搜索

最佳有希望区域通过三个步骤来确定。首先,计算希望表中所有解的目标函数值的平均值;其次,将目标值高于平均值的解消去;最后,将禁忌球半径和超方体邻域大小缩半,并对余留的有希望解执行“产生邻域解、选取最佳解”的搜索过程,然后用最佳邻域解替换有希望解(仅在邻域解优于有希望解时执行),而当整个希望表被扫描过后,算法将移去最不重要的有希望区域。上述过程通过禁忌球半径和超方体邻域大小的再次缩半而重

复进行,直到只有一个有希望区域存在。

4. 集中化搜索

集中化搜索过程首先重新设置禁忌表,并将上一进程得到的有希望区域定义为搜索域,该区域的中心作为当前解,然后再次执行“产生邻域解、选取最佳解、禁忌表更新”的搜索过程,直到目标函数值在规定连续迭代步数内保持不变。然后,算法将超方体邻域大小和禁忌球半径缩半,重新设置禁忌表,并以最佳解为起点重复上述搜索过程,直到达到最大迭代步数或目标值连续若干步保持不变。

5. 参数设置

通过对算法主要进程的上述描述,我们对算法有了较明确的理解。算法参数分为初始化参数和控制参数,其中部分为用户给定参数,部分为算法计算参数,具体内容如表 4.5.1 所示,下面讨论它们的选取和计算问题。

表 4.5.1 ECTS 算法参数列表

用户选取的初始化参数	各状态变量的搜索区域
	初始搜索状态
	禁忌表内容
	希望表内容
需计算的初始化参数	初始超方体搜索的最小边界长度 δ
	接受有希望区域的初始阈值
	最佳初始点
	每步迭代中当前解的邻域解数目 η
	未检测到有希望区域的最大连续迭代步数
	目标值未得到改善的最大连续迭代步数
	性能无改进时超方体邻域大小和禁忌球半径的连续收缩最大值 最大迭代步数
用户选取的控制参数	禁忌表长度 N_t
	希望表长度 N_p
	允许计算初始禁忌球半径的参数 ρ_t
	允许计算初始希望球半径的参数 ρ_p
需计算的控制参数	允许计算初始超方体邻域大小的参数 ρ_n
	禁忌球初始半径 ϵ_t
	希望球初始半径 ϵ_p
	超方体邻域初始大小

(1) 初始化参数的确定

- 变量的搜索区域一般根据问题给出;

- 初始解一般随机产生;
- 禁忌表初始化为空;
- 对于希望表的构造,算法对随机产生的点,判断它是否不属于已生成的希望球,若否则以其作为初始希望球的中心,如此在整个解空间均匀地产生 N_p 个采样点,并且将其中的最佳点定义为最佳初始解;
- 接受有希望区域的初始阈值设置为 N_p 个采样点的平均目标函数值;
- 设置邻域解数量 η 为决策变量数目的 2 倍,并且至少为 10;
- 设置未检测到新的有希望区域的最大连续迭代步数为决策变量数目的 2 倍;
- 设置目标值未改善的最大连续迭代步数为决策变量数目的 5 倍;
- 设置性能无改进时邻域大小和禁忌球半径的连续收缩最大值为决策变量数的 2 倍;
- 设置算法的最大迭代步数为决策变量数的 50 倍。

(2) 控制参数的确定

- 禁忌表长度设置为 7;
- 希望表长度设置为 10;
- ρ_t, ρ_p 和 ρ_n 分别设置为 100, 50 和 5;
- $\epsilon_t = \delta / \rho_t, \epsilon_p = \delta / \rho_p$;
- 超方体邻域初始大小通过将搜索空间分成 ρ_n 来得到。

对于函数优化问题,设计禁忌搜索算法的目的主要是克服局部极小的影响以高效实现全局优化。ECTS 算法的性能在一些 benchmark 问题的测试上得到了充分的验证,在此不予列出计算结果,有兴趣的读者可以利用本书前文给出的 benchmark 问题或一些实际工程问题来测试禁忌搜索算法的性能,并探讨算法的改进以及与其他方法的比较。当然,相比于 ECTS 较复杂的搜索框架,禁忌搜索算法完全可采纳它在组合优化中的简单框架。譬如, Sexton 等(1998)在优化神经网络结构时采纳 TS 的一般搜索框架,将各变量的邻域定义为其 $\pm 0.1\%$ 的范围,目标值和各变量 $\pm 0.01\%$ 的范围定义为禁忌区域,并用藐视准则来激活具有最佳目标值的状态,同时通过修改搜索步长来实现集中化搜索和分散化搜索。需要指出的是,禁忌搜索在函数优化中的应用远不限于 benchmark 问题和神经网络的设计,在此不予展开介绍,况且该领域还处发展之中。

第 5 章

神经网络与神经网络优化算法

人工神经网络是近年来得到迅速发展的一个前沿课题。神经网络由于其大规模并行处理、容错性、自组织和自适应能力和联想功能强特点,已成为解决很多问题的有力工具,对突破现有科学技术的瓶颈,更深入探索非线性等复杂现象起到了重大作用,已广泛应用在许多工程领域。人工神经元是生物神经元特性及功能的数学抽象,神经网络通常指由大量简单神经元互连而构成的一种计算结构,它在某种程度上可以模拟生物神经网络的工作过程,从而具备解决实际问题的能力。神经网络优化算法就是利用神经网络中神经元的协同并行计算能力来构造的优化算法,它将实际问题的优化解与神经网络的稳定状态相对应,把对实际问题的优化过程映射为神经网络系统的演化过程。本章首先回顾神经网络的发展,然后介绍神经网络的模型和 BP 学习算法,进而着重介绍 Hopfield 反馈神经网络、收敛理论和相应的优化算法,最后介绍混沌序列优化和混沌神经网络算法。

5.1 神经网络简介

5.1.1 神经网络发展回顾

神经网络的研究至今已有近 60 年的历史,其发展道路曲折,目前已得到较深入而广泛的研究与应用。

1943 年,心理学家 McCulloch 和数学家 Pitts(1943)合作提出了形式神经元的数学

模型,即 MP 模型,他们利用逻辑的数学工具研究客观世界的事件在形式神经网络中的表达。1949 年,心理学家 Hebb(1949)通过对大脑神经细胞、学习和条件反射的观察与研究,提出了改变神经元连接强度的 Hebb 规则。50 年代末,Rosenblatt(1959)设计发展了 MP 模型,提出了多层感知机,即 Perceptron(1962),试图模拟动物和人脑的感知和学习能力。60 年代初,Widrow(1960)提出了自适应线性单元模型,即 Adaline,以及一种有效的网络学习方法,即 Widrow-Hebb 规则,或称为 δ 规则。鉴于上述研究,神经网络引起了许多科学家的兴趣。但是,1962 年,当时具有较高学术地位的人工智能创始人 Minsky 和 Papert 出版了《Perceptron》一书,对以感知机为代表的神经网络的功能和局限性从数学上进行了深入分析,并指出 Perceptron 只能进行线性分类求解一阶谓词问题,同时寻找多层感知机的有效学习算法并不乐观。鉴于上述观点,许多学者放弃了对神经网络的研究,从而使神经网络的研究陷入低潮。然而,在此期间仍有一些学者坚持着对神经网络的研究,譬如 Grossberg(1969)提出了自适应共振理论,即 ART 模型,并发展成 ART1,ART2 和 ART3 三个版本;Kohonen(1984)提出了自组织映射理论,即 SOM,并对联想存储器进行了研究;Fukushima(1980)提出了神经认知网络理论;Werbos 提出了误差反传理论,即 BP;Amari(1978)则致力于神经网络数学理论的研究。所有这些具有开创性的研究成果虽然在当时并未引起很大的影响,但为神经网络此后的发展奠定了很好的理论基础。

1982 年,Hopfield 通过引入能量函数的概念,研究网络的动力学性质,并用电子线路设计出相应的网络,从而开拓了神经网络用于联想记忆和优化计算的新途径,进而掀起了神经网络新的研究热潮,为神经优化的研究奠定了基础。1986 年,Rumelhart 和 MacClelland 等提出了 PDP 理论,尤其是发展了多层前向网络的 BP 算法,为解决多层前向网络的学习问题开辟了有效途径,并成为迄今应用最普遍的学习算法。随后,大量开拓性的研究工作又大大发展了神经网络的模型与学习算法,并加深了人们对神经网络的认识。同时,神经网络的应用研究也渗透到控制工程、机器学习、优化计算、信号处理、模式识别和经济等领域。

目前,各类与神经网络相关的学术期刊和会议相应问世,有关神经网络及其应用的论文大量涌现,神经网络已成为国际上的一个研究热点。

5.1.2 神经网络的模型

1. 单层感知机

单层感知机模型如图 5.1.1 所示,它具有简单的模式识别能力,但只能解决线性分类,而不能解决非线性问题。

通常,单层感知机的学习算法如下:

- (1) 给出初始权值 w_i 和阈值 θ ;

(2) 给定连续输入样本 $x_i, i=0, \dots, n-1$ 和目标输出 $c(t)$;

(3) 计算实际输出 $y(t) = f(\sum_{i=0}^{n-1} w_i(t)x_i(t) - \theta)$, 其中 $f(\cdot)$ 为神经元激励函数;

(4) 调整权值

$$w_i(t+1) = w_i(t) + \eta[c(t) - y(t)]x_i(t)$$

其中,

$$c(t) = \begin{cases} 1 & \text{对 A 类输入样本} \\ -1 & \text{对 B 类输入样本} \end{cases}, 0 \leq \eta < 1$$

(5) 返回第(2)步。

对于上述算法,若函数是线性可分的,则经过有限步迭代后算法收敛到正确的权值。

2. 多层感知机

(1) 多层感知机的描述

一个 M 层的多层感知机(前向网络)可描述如下:

① 网络包含一个输入层(定义为第 0 层)和 $M-1$ 个隐层,最后一个隐层称为输出层。

② 第 l 层包含 N_l 个神经元和一个阈值单元(定义为每层的第 0 单元),输出层不含阈值单元。

③ 第 $l-1$ 层第 i 个单元到第 l 层第 j 个单元的权值表为 $w_{ij}^{l-1,l}$ 。

④ 第 l 层($l>0$)第 j 个($j>0$)神经元的输入定义为 $x_j^l = \sum_{i=0}^{N_{l-1}} w_{ij}^{l-1,l} y_i^{l-1}$, 输出定义为 $y_j^l = f(x_j^l)$, 其中 $f(\cdot)$ 为隐单元激励函数,常采用 Sigmoid 函数,即 $f(x) = [1 + \exp(-x)]^{-1}$ 。输入单元一般采用线性激励函数 $f(x) = x$, 阈值单元的输出始终为 1。

⑤ 目标函数通常采用

$$E = \sum_{p=1}^P E_p = \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^{N_{M-1}} (y_{j,p}^{M-1} - t_{j,p})^2$$

其中 P 为样本数, $t_{j,p}$ 为第 p 个样本的第 j 输出分量。

对于典型的三层前向网络,其结构如图 5.1.2 所示。

(2) BP 算法

网络学习归结为确定网络的结构和权值,使目标函数值最小。网络学习是一个复杂的非线性优化问题,已经证明属于 NP 难问题,即使在结构给定的情况下也是如此。

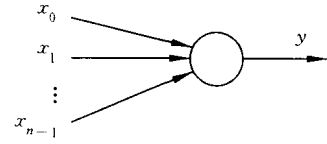


图 5.1.1 单层感知机模型

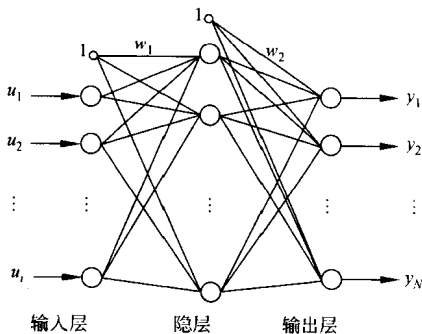


图 5.1.2 三层前向神经网络

BP 算法(Rumelhart et al, 1986)是前向神经网络经典的有监督学习算法,它的提出对前向神经网络的发展起过历史性的推动作用。对于上述 M 层的 FNN, BP 算法可由下列迭代式描述,具体推导可参见神经网络的相关书目。

$$\begin{aligned} w_{ij}^{l-1,l}(k+1) &= w_{ij}^{l-1,l}(k) - \alpha \partial E / \partial w_{ij}^{l-1,l}(k) \\ &= w_{ij}^{l-1,l}(k) - \alpha \sum_{p=1}^P \delta_{j,p}^l(k) y_{i,p}^{l-1}(k) \end{aligned} \quad (5.1.1)$$

$$\delta_{j,p}^l(k) = \begin{cases} [y_{j,p}^l(k) - t_{j,p}^l] f'[x_{j,p}^l(k)], & l = M-1 \\ f'[x_{j,p}^l(k)] \sum_{m=1}^{N_{l+1}} \delta_{m,p}^{l+1}(k) w_{jm}^{l+1}(k), & l = M-2, \dots, 1 \end{cases} \quad (5.1.2)$$

其中, α 为学习率。

实质上, BP 算法是一种梯度下降法, 算法性能依赖于初始条件, 学习过程易于陷入局部极小。数值仿真研究表明, BP 算法的学习速度、精度、初值鲁棒性和网络推广性能都较差, 不能满足应用的需要。下面对此作简单阐述和分析。

(3) BP 算法收敛缓慢的主要原因

① BP 算法利用梯度信息来调整权值, 在误差曲面平坦处, 导数值较小使得权值调整幅度较小, 从而误差下降很慢; 在曲面曲率大处, 导数值较大使得权值调整幅度较大, 会出现跃冲极小点现象, 从而引起振荡。

② 神经元的总输入偏离阈值太远时, 总输入就进入激励函数非线性特性的饱和区。此时若实际输出与期望输出不一致, 激励函数较小的导数值将导致算法难以摆脱“平台”区。

③ 由于网络结构的复杂性, 不同权值和阈值对同一样本的收敛速度不同, 从而使得整体学习速度缓慢。

(4) 克服 BP 算法训练缓慢的常用方法

① 改变学习步长(Xu et al, 1992)。等效于对权值的改变, 从而改变误差曲面的形状, 缩短到达极小点的路径而加速收敛。

② 加动量项和改变动量因子(Rumelhart et al, 1986)。使权值变化更平滑而有利于加速收敛, 但动量因子需适当选择或自适应改变。

③ 适当选择神经元激励函数(Hush et al, 1992)和初始权值、阈值(Lee et al, 1993), 并对输入样本的归一化处理。目的是避免“平台”现象的出现。

④ 采用合适的训练模式(如逐一式、批处理、跳跃式等)。避免权值收敛速度的不平衡现象(Xu et al, 1992)。

⑤ 采用高阶导数信息、最优滤波法和启发式算法。如二阶导数法(Parker, 1987)、共轭梯度法(Kramer et al, 1988)、准牛顿法(Watrous, 1987)、扩展 Kalman 算法和 delta-bar-delta 算法等。

(5) BP 算法易陷入局部极小的原因和改进措施

由于不能保证目标函数在权空间中的正定性,而误差曲面往往复杂且无规则,存在多个分布无规则的局部极小点,因而基于梯度下降的 BP 算法易于陷入局部极小。

改进措施主要有:

- ① 引入全局优化技术;
- ② 平坦化优化曲面以消除局部极小;
- ③ 设计合适网络使其满足不产生局部极小条件。

(6) BP 算法的推广性能及其改进措施

网络的推广性能差主要表现为,网络能够很好地实现训练样本的输入输出映射,但不能保证对未训练的样本输入得到理想的输出。

改进方法主要有:

- ① 引入与问题相关的先验知识对权值加以限制;
- ② 产生虚拟“瓶颈层”,以便对权矩阵的秩施加限制;
- ③ 对目标函数附加惩罚项以强制无用权值趋于零;
- ④ 动态修改网络结构,对推广函数与目标函数进行多目标优化等。

3. Hopfield 网络

1982 年, Hopfield 开创性地在物理学、神经生物学和计算机科学等领域架起了桥梁,提出了 Hopfield 反馈神经网络模型(HNN),证明在高强度连接下的神经网络依靠集体协同作用能自发产生计算行为。Hopfield 网络是典型的全连接网络,通过在网络中引入能量函数以构造动力学系统,并使网络的平衡态与能量函数的极小解相对应,从而将求解能量函数极小解的过程转化为网络向平衡态的演化过程。尤其是通过对 TSP 问题的成功求解,开辟了神经网络模型在计算机科学应用中的新天地,动态反馈网络从而受到广泛的研究和关注,被广泛应用于优化问题中,且已设计出了专用的硬件电路。

(1) 离散型 Hopfield 网络

离散型 Hopfield 网络的输出为二值型,网络采用全连接结构。表 $v_1, v_2, \dots, v_n$ 为各神经元的输出, $w_{1i}, w_{2i}, \dots, w_{ni}$ 为各神经元与第 i 神经元的连接权值, θ_i 为第 i 神经元的阈值,则有

$$v_i = f\left(\sum_{j=1, j \neq i}^n w_{ji} v_j - \theta_i\right) = f(u_i) = \begin{cases} 1, & u_i \geq 0 \\ -1, & u_i < 0 \end{cases} \quad (5.1.3)$$

能量函数定义为 $E = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1, j \neq i}^n w_{ij} v_i v_j + \sum_{i=1}^n \theta_i v_i$, 则其变化量为

$$\Delta E = \sum_{i=1}^n \frac{\partial E}{\partial v_i} \Delta v_i = \sum_{i=1}^n \Delta v_i \left(-\sum_{j=1, j \neq i}^n w_{ji} v_j + \theta_i\right) \leq 0 \quad (5.1.4)$$

也就是说,能量函数总是随神经元状态的变化而下降的。

(2) 连续型 Hopfield 网络

连续型 Hopfield 网络如图 5.1.3 所示。

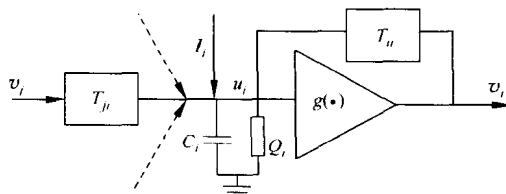


图 5.1.3 连续型 Hopfield 网络

网络的动态方程提出可简化描述如下：

$$\begin{cases} C_i \frac{du_i}{dt} = \sum_{j=1}^n T_{ji} v_j - \frac{u_i}{R_i} + I_i \\ v_i = g(u_i) \end{cases} \quad (5.1.5)$$

其中, u_i, v_i 分别为第 i 神经元的输入和输出, $g(\cdot)$ 为具有连续且单调增性质的神经元激励函数, T_{ij} 为第 i 神经元到第 j 神经元的连接权, I_i 为施加在第 i 神经元的偏置, $C_i > 0$ 和 Q_i 为相应的电容和电阻, $1/R_i = 1/Q_i + \sum_{j=1}^n T_{ji}$ 。

定义能量函数

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n T_{ij} v_i v_j - \sum_{i=1}^n I_i v_i + \sum_{i=1}^n \int_0^{v_i} g^{-1}(v) dv / R_i$$

则其变化量

$$\frac{dE}{dt} = \sum_{i=1}^n \frac{\partial E}{\partial v_i} \frac{dv_i}{dt}$$

其中,

$$\begin{aligned} \frac{\partial E}{\partial v_i} &= -\frac{1}{2} \sum_{j=1}^n T_{ij} v_j - \frac{1}{2} \sum_{j=1}^n T_{ji} v_j + \frac{u_i}{R_i} - I_i \\ &= -\frac{1}{2} \sum_{j=1}^n (T_{ij} - T_{ji}) v_j - \left(\sum_{j=1}^n T_{ji} v_j - \frac{u_i}{R_i} + I_i \right) \\ &= -\frac{1}{2} \sum_{j=1}^n (T_{ij} - T_{ji}) v_j - C_i \frac{du_i}{dt} = -\frac{1}{2} \sum_{j=1}^n (T_{ij} - T_{ji}) v_j - C_i g^{-1'}(v_i) \frac{dv_i}{dt} \end{aligned}$$

于是, 当 $T_{ij} = T_{ji}$ 时,

$$\frac{dE}{dt} = - \sum_{i=1}^n C_i g^{-1'}(v_i) \left(\frac{dv_i}{dt} \right)^2 \leq 0$$

且当 $\frac{dv_i}{dt} = 0$ 时 $\frac{dE}{dt} = 0$ 。因此, 随时间的增长, 神经网络在状态空间中的解轨迹总是向能量函数减小的方向变化, 且网络的稳定点就是能量函数的极小点。

连续型 Hopfield 网络广泛用于联想记忆和优化计算问题。当用于联想记忆时, 能量

函数是给定的,网络的运行过程是通过确定合适的权值以满足最小能量函数的要求;当用于优化计算时,网络的连接权值是确定的,首先将目标函数与能量函数相对应,然后通过网络的运行使能量函数不断下降并最终达到最小,从而得到问题对应的极小解。

(3) Boltzman 机

Boltzman 机是离散 Hopfield 网络的一种变型,通过对离散 Hopfield 网络加以扰动,使其以概率的形式表达,而网络的模型方程不变,只是输出值类似于 Boltzman 分布以概率分布取值。

离散 Hopfield 网络的输出为 $v_i = \text{sgn}(\sum w_{ji}v_j - \theta_i)$ 。对于 Boltzman 机,设内部状态 $I_i = \sum_{\substack{j=1 \\ j \neq i}}^n w_{ji}v_j - \theta_i$, $p_i(0)$ 和 $p_i(1)$ 分别为神经元 i 输出值为 0 和 1 时的概率,即 $p_i(0) = \frac{1}{1 + e^{-I_i/T}}$, $p_i(1) = 1 - p_i(0)$, 其中 T 为类似温度的扰动。

可见,当 T 较大时, $p_i(0)$ 和 $p_i(1)$ 几乎相等;当 T 较小时,两概率值将由系统内部状态决定。因此在系统运行时,先将 T 置于较大值以使系统能够跃过能量较大的状态来避免陷入局部极小,然后将 T 值逐渐减小,从而使系统最终收敛到全局能量最小的状态。这与模拟退火算法的机制几乎一样。

(4) 随机 Hopfield 网络模型

随机网络模型是连续 Hopfield 网络的一种变型,即在每个输入上加扰动。设扰动参数为 T , 扰动幅度用 T 与神经元自身电位的函数 $q(u_i, u_0)$ 的乘积表示,模型方程为

$$C_i \frac{du_i}{dt} = -\frac{u_i}{R_i} + \sum_j T_{ji}v_j + I_i + \sqrt{2Tq(u_i, u_0)}u_i$$

当 T 加大时,扰动加大。当有关神经元正要完全兴奋或完全抑制时,表现在 $q(u_i, u_0)$ 和 v_i 上扰动也加大。这样系统就有足够的时间选择兴奋或抑制的状态。当系统冷却时,网络将会收敛到全局能量最小的状态。

5.2 基于 Hopfield 反馈网络的优化策略

5.2.1 基于 Hopfield 模型优化的一般流程

Hopfield 网络是一种非线性动力学模型,通过引入类似 Lyapunov 函数的能量函数概念,把神经网络的拓扑结构(用连接权矩阵表示)与所求问题(用目标函数描述)对应起来,转换成神经网络动力学系统的演化问题。因此,在用 Hopfield 网络求解优化问题之前,必须将问题映射为相应的神经网络。譬如对 TSP 问题的求解,首先将问题的合法解

映射为一个置换矩阵,并给出相应的能量函数,然后将满足置换矩阵要求的能量函数的最小值与问题的最优解相对应。

对于一般性问题,通常需要以下几方面工作:

- (1) 选择合适的问题表示方法,使神经网的输出与问题的解相对应;
- (2) 构造合适的能量函数,使其最小值对应问题的最优解;
- (3) 由能量函数和稳定条件设计网络参数,如连接权值和偏置参数等;
- (4) 构造相应的神经网络和动态方程;
- (5) 用硬件实现或软件模拟。

5.2.2 基于 Hopfield 模型优化的缺陷

基于 Hopfield 模型设计优化算法的出发点在于以下两方面:

- (1) 神经网络是稳定的,网络势必收敛到渐近平衡点;
- (2) 神经网络的渐近平衡点恰好是能量函数的极小点。

但是,从稳定性理论分析,Hopfield 模型优化方法并不严格,虽然借助 Lyapunov 方法的思想构造能量函数,但并不知道网络平衡点的位置,因此不能保证能量函数的极值点就是问题真正的全局极小点,从而往往出现 Hopfield 网络不能收敛到全局极小的情况。然而,在 Lyapunov 稳定理论中,平衡点与函数极点预先已对应,必然把运动轨迹趋于平衡点与 Lyapunov 函数趋于极小值相对应,从而保证稳定理论的正确性。稳定性分析仍然是动态反馈网络应用的理论依据。利用 HNN 优化求解,若网络参数选择不好,往往得不到最优解,或是次优度很差,甚至出现“非法”或“冻结”现象。“非法”指结果不符合置换矩阵要求,“冻结”意味着算法不收敛。因此,研究参数对算法性能的影响很有意义,尽管分析权矩阵特征值和特征向量可给出一些参数选择的约束,但目前尚无一般而实用的选择方法。

鉴于理论上的不严格性,同时由于传统 Hopfield 网络仍采用梯度下降策略,因此基于 Hopfield 网络的优化计算通常会导致以下问题(王凌等,1999):

- (1) 网络最终收敛到局部极小解,而非问题的全局最优解;
- (2) 网络可能会收敛到问题的不可行解;
- (3) 网络优化的最终结果很大程度上依赖于网络的参数,即参数鲁棒性较差。

5.2.3 基于 Hopfield 模型优化的改进研究

标准 Hopfield 模型需要 n^2 个神经元,计算复杂性至少为 $O(n^4)$ 。基于 Hopfield 模型的优化计算优化性能较差,且对初值、参数、网络分布和激励函数的鲁棒性较差,同时算法收敛性标志不严格。为改进 Hopfield 模型的性能,大量学者以 TSP 为例在算法和网

络结构方面作了研究。针对 TSP 问题, Hopfield 最先设计的能量函数如下:

$$E = \frac{A}{2} \sum_x \sum_i \sum_{j \neq i} v_{xi} v_{xj} + \frac{B}{2} \sum_i \sum_x \sum_{y \neq x} v_{xi} v_{yi} + \frac{C}{2} \left(\sum_x \sum_i v_{xi} - n \right)^2 + \frac{D}{2} \sum_x \sum_i \sum_{y \neq x} d_{xy} v_{xi} (v_{y,i+1} + v_{y,i-1}) \quad (5.2.1)$$

相应的神经网络动力学方程和网络的连接权矩阵 T 、外加偏置 I 可描述为

$$\begin{cases} \frac{du_{xi}}{dt} = -\frac{u_{xi}}{\tau} - A \sum_{j \neq i} v_{xj} - B \sum_{y \neq x} v_{yi} - C \left(\sum_x \sum_i v_{xi} - n \right) - D \sum_{y \neq x} d_{xy} (v_{y,i+1} + v_{y,i-1}) \\ v_{xi} = g(u_{xi}) = \frac{1}{2} [1 + \tanh(u_{xi}/u_0)] \end{cases} \quad (5.2.2)$$

$$\begin{cases} T_{xi,yj} = -A \delta_{xy} (1 - \delta_{ij}) - B \delta_{ij} (1 - \delta_{xy}) - C - D d_{xy} (\delta_{j,i+1} + \delta_{j,i-1}) \\ I_{xi} = Cn \end{cases} \quad (5.2.3)$$

采用如下参数: $\tau = 1, A = B = D = 500, C = 200, u_0 = 0.02, u_{xi} = u_{00} + \delta u_{xi}$, 其中 $-0.1u_0 \leq \delta u_{xi} \leq 0.1u_0, u_{00}$ 在 $t=0$ 时满足 $\sum_x \sum_i v_{xi} = n$ 。尽管 Hopfield 在 $n=10$ 的 20 组不同初值的优化实验中 16 次得到次优解 (Hopfield et al, 1985), 但后来许多研究表明 HNN 存在严重的不稳健性, 网络参数和初始条件严重影响计算结果, 甚至得不到最优或合法解。

以下是庄镇泉等 (1992) 给出算法仿真的具体步骤。其中, δu_{xi} 为初始随机扰动用以打破各神经元间的平衡, λ 为迭代步长控制 u_{xi} 的变化速率。假设 $n=10$, 各点横、纵坐标均取 $(0, 200)$ 内随机数, 选取 $A=B=D=500, C=A/\sqrt{n}, u_0=0.02, \lambda=10^{-5}$, 在 25 次随机实验中, 36% 的解较优, 52% 的解较差, 12% 出现“非法”或“冻结”。

- (1) 确定网络参数 A, B, C, D, λ 和初值 $u_0, u_{xi}, \delta u_{xi}$ 。
- (2) 由 Sigmoid 激励函数计算每个神经元的输出 v_{xi} 。
- (3) 计算能量函数 E 和 du_{xi}/dt 的值。
- (4) 以 $u_{xi}^{k+1} = u_{xi}^k + \lambda du_{xi}/dt$ 确定新的 u_{xi} 和 v_{xi} 。
- (5) 判断能量函数是否满足稳定条件。若到一定条件仍不稳定则认为“冻结”而退出算法, 否则转 (2); 若满足稳定条件则转 (6)。
- (6) 输出最优路径、路径长度以及执行时间。

利用上述算法, 并以上述参数为基准通过改变参数研究参数对算法的影响, 每次只改变一个参数, 仿真结果如表 5.2.1~表 5.2.4 所示。

通过上述仿真结果, 参数对算法性能的影响归纳如下:

- (1) u_0 的影响: 由表 5.2.1 知, u_0 下降导致寻优时间缩短, 但 u_0 太小会导致路径不优或“非法”路径的出现; u_0 太大又可能出现“冻结”现象。原因在于 u_0 相当于放大器增益, u_0 太小使 Sigmoid 激励函数趋于离散阶跃函数, 从而不易获得优解; 反之使激励函数过于平坦, 神经元状态不易收敛到“0”或“1”。

表 5.2.1 u_0 对算法性能的影响

参数 u_0	优化结果
0.03	“冻结”
0.02	$d=521.00, t=11.81s$
0.01	$d=521.00, t=9.01s$
0.005	$d=563.63, t=7.42s$
0.002	“非法”

表 5.2.2 λ 对算法性能的影响

参数 λ	优化结果
1.5×10^{-5}	“非法”
1.0×10^{-5}	$d=521.00, t=11.81s$
0.5×10^{-5}	$d=521.00, t=16.20s$
0.1×10^{-5}	“冻结”

表 5.2.3 C 对算法性能的影响

参数 C	优化结果
0.6A	“非法”
0.4A	$D=521.00, t=5.05s$
$A/\sqrt{10}$	$D=521.00, t=11.8s$
0.2A	“冻结”

表 5.2.4 D 对算法性能的影响

参数 D	优化结果
1.5A	“非法”
1.2A	$d=521.00, t=16.2s$
0.8A	$D=521.00, t=3.46s$
0.1A	$D=584.73, t=2.31s$

(2) λ 的影响:由表 5.2.2 知, λ 减小导致寻优时间增加,太小的 λ 会引起“冻结”现象或路径不优; λ 太大则会引起“非法”路径。从 u_{xi} 的迭代式知, λ 的大小决定 u_{xi} 变化大小的快慢, λ 大使 u_{xi} 变化剧烈而易出现“非法”现象, λ 小使 u_{xi} 更新过于慢而易出现“冻结”现象。

(3) 参数 A, B 的影响:由于算法设置 $A=B$,且 C, D 与 A 呈正比关系,因此改变 A, B 意味着同时改变能量函数的各系数,即间接地改变 λ 值,出现的现象必然与改变 λ 相同。仿真结果也如此,在此不再罗列。

(4) 参数 C 的影响:由表 5.2.3 知, C 下降导致寻优时间增加,过小的 C 值会引起“冻结”现象; C 太大而导致“非法”路径的出现。原因在于 C 的大小影响能量函数中置换矩阵约束项的权重, C 太大使能量函数变化剧烈,尽管计算过程缩短,但路径可能出现“非法”;相反则导致收敛太慢而出现“冻结”。

(5) 参数 D 的影响:由表 5.2.4 知, D 下降使寻优时间缩短, D 太小则引起路径不优; D 过大则会引起“非法”路径。由能量函数知, D 的改变即能量函数中目标项权重改变, D 太大使目标项权重过大而导致与约束项不一致,从而出现“非法”; D 太小使目标项权重降低,势必出现路径不优的现象。

归纳而言,基于 HNN 模型优化的不稳健性主要来源于:

- (1) 初值的不稳健性,各神经元初值大小和分布情况影响寻优结果;
- (2) 模型参数的不稳健性,参数直接影响能量函数中约束项和目标项在优化过程中的地位和重要程度;
- (3) 问题结构的不稳健性,相同模型参数对不同结构的问题会导致不同的收敛性;
- (4) 激励函数的不稳健性,主要是函数形态的影响;

(5) 算法收敛性标志的不严格性。

常用的改进方案包括：

(1) 在迭代过程中设置神经元输出变“0”和“1”的阈值，一旦达到阈值相应的输出就设置为“0”或“1”，从而提高算法的收敛速度；

(2) 当每行或每列出现一个“1”时，相应行或列的其余元置“0”；

(3) 取消神经元动态方程中的自反馈项，以减小能量的消耗；

(4) 用离散化或其他形态的激励函数代替连续型 Sigmoid 函数，改善收敛速度和性能；

(5) 改变能量函数形式或置换矩阵的定义等。

基于上述思路的研究成果有：Wilson 等(1988)针对随机扰动的缺点，归一化城市坐标到(0,1)区间，并引入一个偏置(如式 5.2.4)到初值来改善优化性能；Brandt 等(1988)针对问题的表征方式提出三下标法，用 v_{ijk} 表示从第 i 个城市到第 j 个城市恰是第 k 次访问，这样总路径长度可表示为 $\sum_i \sum_j \sum_k d_{ij} v_{ijk}$ ，数据项 d_{ij} (城市 i 和 j 间的距离)仅是能量函数中一次项的系数，与权矩阵无关，但需要 n^3 个神经元数，计算量较大；Szu(1991)在能量函数方面提出了一个强化约束公式(如式 5.2.5)，表示置换矩阵每行每列只有一个“1”，及时唤醒全零元行(列)中的神经元，从而抑制过多的“1”出现，减少非法路径；Xu 等(1991)改变置换矩阵的定义，用 v_{xy} 表示 x 与 y 之间的连接， $v_{xy}=1$ 表示 x 与 y 相连，否则不连，这样可用 $n(n-1)/2$ 个神经元建立模型，能量函数如式(5.2.6)，从而神经元间的连接复杂性较 Hopfield 网络有所降低，并消除了一些冗余项，提高了计算效率；Xu 等(1991)还提出了广义神经网络，通过自适应地消除回路得到全局最优解；Wang 等(1991)采用时变能量函数(如式(5.2.7))，自适应地改变能量函数中某个参数，参数的缓慢变化可避免陷入局部极小，且随时间加大而最终趋于 1，从而得到有效优化解，文中指出这种时变模型能较大幅度提高得到优良解的比例。

$$\text{bias}(i, j) = \cos \left[\arctan \left(\frac{y_i - 0.5}{x_i - 0.5} \right) + \frac{2\pi(j-1)}{n} \sqrt{(x_i - 0.5)^2 + (y_i - 0.5)^2} \right] \quad (5.2.4)$$

$$E = \sum_x \left(\sum_i v_{xi} - 1 \right)^2 + \sum_i \left(\sum_x v_{xi} - 1 \right)^2 \quad (5.2.5)$$

$$E = \frac{A}{2} \sum_x \left(\sum_{y \neq x} v_{xy} - 2 \right)^2 + \frac{B}{2} \sum_y \left(\sum_{x \neq y} v_{xy} - 2 \right)^2 + C \sum_{x=1}^n \sum_{y=x+1}^n v_{xy} d_{xy} \quad (5.2.6)$$

$$E = \frac{A}{2} \sum_x \left(\sum_i v_{xi} - S \right)^2 + \frac{B}{2} \sum_i \left(\sum_x v_{xi} - S \right)^2 + \frac{C}{2} \sum_x \sum_y \sum_i d_{xy} v_{xi} (v_{y,i+1} + v_{y,i-1}) + \frac{A+B}{2} \sum_x \sum_i v_{xi} (1 - v_{xi}) \quad (5.2.7)$$

由于基于 Hopfield 模型的优化过程中网络动态的变化保证能量函数单调非增，因此往往收敛到局部解，而 SA 为克服这一困难提供了思路。所以，如果把两种算法合理结

合,就能提高优化性能和时间性能。例如,用 Hopfield 算法构成主算法较快得到可行解,用 SA 概率性逃离局部极小点而转移到目标函数的其他极小点,从而提高最终的优化性能。

同时,利用 GA 并行搜索的优点,结合 Hopfield 网络也是一种可行方法。将混沌动态引入 Hopfield 模型,结合混沌动态的遍历性和 Hopfield 模型的梯度下降过程,同样可构造有效的优化方法。

此外,将竞争网络和 Hopfield 网络结合,竞争网络实现约束函数, Hopfield 网络实现具有简化约束的能量函数,用信号时延小的竞争网络控制调节 Hopfield 网络的输出,一旦 Hopfield 网络的输出大于竞争网络神经元的阈值就采用“全胜策略”通过竞争网络开始搜索最高输出,这也是提高算法性能的方法。

不可否认,基于 Hopfield 模型的优化计算具有简单、规范、快速等优点,但为了克服其优化性和鲁棒性差等缺点,还需要做大量的研究工作。修改能量函数形式、确立网络参数选择原则,以及把 Hopfield 模型与其他方法合理结合等,仍是有关 Hopfield 网络的重要研究课题。可以预见,鉴于人类智能活动快速并行等优点,反馈神经网络将在决策、控制和判断等方面发挥强大的作用。

5.3 动态反馈神经网络的稳定性研究

动态反馈神经网络,如 Hopfield 网络(HNN),是一类具有联想功能且能产生优化计算行为的非线性动力学模型。它的提出推动了神经网络硬件和神经计算机的研究与发展。但是,基于 HNN 的优化过程,难以实现全局优化,对初始条件和网络参数的鲁棒性差,不利于反馈网络的应用。下面从理论上研究离散、非对称、时延等网络的稳定条件,为网络结构和参数设计提供准则和指导,并对一般连续反馈网络研究网络稳定域范围的估计算法。

5.3.1 动态反馈网络的稳定性分析

稳定性分析,不论是针对联想记忆而进行的吸引子局部稳定性的研究,还是针对优化计算进行的全局稳定性的研究,都可用来指导动态反馈网络的设计和应用(焦李成,1992)。

一个连续动态反馈神经网络可描述为

$$\begin{cases} \dot{x}_i(t) = -a_i x_i(t) + s_i + \sum_{j=1}^n w_{ij} y_j(t) \\ y_i(t) = \sigma_i[x_i(t)/\eta_i], \quad i = 1, 2, \dots, n \end{cases} \quad (5.3.1)$$

其中, $x_i(t)$ 和 $y_i(t)$ 分别为神经元的输入和输出状态, a_i 为自反馈系数, s_i 为阈值, w_{ij} 为连接权值, η_i 为激励增益 ($\eta_i > 0$), $\delta_i(\cdot)$ 为激励函数。

上述描述的向量方程形式为

$$\dot{\mathbf{X}} = -\mathbf{A}\mathbf{X} + \mathbf{S} + \mathbf{W}\mathbf{Y} \quad (5.3.2)$$

其中, $\mathbf{X} = [x_1, \dots, x_n]^T$, $\mathbf{Y} = [y_1, \dots, y_n]^T = \mathbf{G}(\mathbf{X}) = [\sigma_1(x_1/\eta_1), \dots, \sigma_n(x_n/\eta_n)]^T$, $\mathbf{A} = \text{diag}\{a_1, \dots, a_n\}$, $\mathbf{S} = [s_1, \dots, s_n]^T$, $\mathbf{W} = (w_{ij})_{n \times n}$ 。

在此约定, 激励函数为连续可微、有界且严格单调递增。因此, 其逆函数也为连续可微且严格单调递增。对于连续反馈神经网络, 当 $\mathbf{W}^T = \mathbf{W}$ 且 $a_i > 0, i = 1, 2, \dots, n$ 时, 网络状态必渐近收敛到系统的平衡点。

由于数字实现必须进行离散化, 且实际网络难以实现对称条件, 网络传输存在时延, 因此下面只讨论离散、非对称和时延网络的稳定性。并且, 令激励函数满足 $0 < \sigma'_i(\cdot) \leq \beta_i$, 或 $(\sigma_i^{-1})'(\cdot) \geq 1/\beta_i$, 其中符号“'”表示求导运算, 再令 $\mathbf{B} = \text{diag}\{\eta_1/\beta_1, \dots, \eta_n/\beta_n\}$ 。

5.3.1.1 离散对称动态反馈网络的渐近稳定性分析

令 τ 为仿真步长, 基于 Euler 法, 可导出式(5.3.1)的离散模型为

$$\begin{cases} x_i(k+1) = (1 - a_i\tau)x_i(k) + \tau s_i + \sum_{j=1}^n \tau w_{ij} \cdot y_j(k) \\ y_i(k) = \sigma_i(x_i(k)/\eta_i) \text{ or } x_i(k) = \eta_i \sigma_i^{-1}(y_i), \quad i = 1, 2, \dots, n \end{cases} \quad (5.3.3)$$

对应的向量方程形式为

$$\mathbf{X}(k+1) = (\mathbf{I} - \tau\mathbf{A})\mathbf{X}(k) + \tau(\mathbf{S} + \mathbf{W}\mathbf{Y}) \quad (5.3.4)$$

其中, $\mathbf{I}$ 为单位阵。

定义 5.3.1 考虑离散系统 $\mathbf{X}(k+1) = f(\mathbf{X}(k))$, 称 $\mathbf{X}$ 为系统的 m 周期点, 如果 $f^m(\mathbf{X}) = \mathbf{X}$ 且 $f^i(\mathbf{X}) \neq \mathbf{X}, 1 \leq i < m$; 称 $\mathbf{X}$ 为系统的不动点, 如果 $f(\mathbf{X}) = \mathbf{X}$ 。

定义 5.3.2 考虑离散神经网络(5.3.3), 称系统以同步迭代方式运行, 若所有神经元的状态是并行同步更新的; 称系统以异步迭代方式运行, 若每一时刻只有一个神经元状态发生变化。

定理 5.3.1 考虑离散动态反馈神经网络(5.3.3), 以同步迭代方式运行的网络必渐近收敛到系统的不动点, 如果满足 $\mathbf{W}^T = \mathbf{W}$ 和以下任一条件:

(1) 当 $0 \leq a_i \cdot \tau \leq 1, i = 1, \dots, n$ 时, $\mathbf{W} + \mathbf{A}\mathbf{B}$ 或 $\tau\mathbf{W} + 2(\mathbf{I} - \tau\mathbf{A})\mathbf{B}$ 为正定。

(2) 当 $a_i < 0, i = 1, \dots, n$ 时, $\tau\mathbf{W} + 2\mathbf{B}$ 为正定。

进一步, 若 $\eta_i = \eta, \beta_i = \beta$, 则条件(2)可简化为 $2\eta/\beta\tau > -\lambda_{\min}(\mathbf{W})$; 若 $a_i = a$, 则条件(1)可简化为 $a\eta/\beta > -\lambda_{\min}(\mathbf{W})$ 或 $2(1 - \tau a)\eta/\beta\tau > -\lambda_{\min}(\mathbf{W})$ 。

证明 构造有界能量函数 $V(\mathbf{Y})$ 为

$$V(\mathbf{Y}) = -\frac{1}{2} \sum_{i,j} \tau w_{i,j} y_i(k) y_j(k) - \sum_i \tau s_i y_i(k) + \sum_i \tau a_i \eta_i \int_0^{y_i(k)} \sigma_i^{-1}(x) dx \quad (5.3.5)$$

令 $\Delta y_i = y_i(k+1) - y_i(k)$, 若系统从 k 到 $k+1$ 时刻以同步方式运行, 则有

$$\begin{aligned}\Delta V(\mathbf{Y}) &= V(\mathbf{Y}(k+1)) - V(\mathbf{Y}(k)) \\ &= -\frac{1}{2} \sum_{i,j} \tau w_{i,j} (\Delta y_i \Delta y_j + 2\Delta y_i y_j(k)) - \sum_i \tau s_i \Delta y_i + \sum_i \tau a_i \eta_i \int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx\end{aligned}$$

利用 $x_i(k) = \eta_i \sigma_i^{-1}(y_i(k))$, 可知上式中:

$$\begin{aligned}-\sum_i \tau s_i \Delta y_i &= -\sum_i \Delta y_i [x_i(k+1) - (1 - \tau a_i) x_i(k) - \sum_j \tau w_{i,j} y_j(k)] \\ &= -\sum_i \Delta y_i [\eta_i \sigma_i^{-1}(y_i(k+1)) - (1 - \tau a_i) \eta_i \sigma_i^{-1}(y_i(k))] + \sum_{i,j} \tau w_{i,j} \Delta y_i y_j(k)\end{aligned}$$

从而

$$\begin{aligned}\Delta V(\mathbf{Y}) &= -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j + \sum_i \tau a_i \eta_i \int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx \\ &\quad - \sum_i \eta_i \Delta y_i \sigma_i^{-1}(y_i(k+1)) + \sum_i (1 - \tau a_i) \eta_i \Delta y_i \sigma_i^{-1}(y_i(k))\end{aligned}\quad (5.3.6)$$

下面, 分为三种情况加以证明:

(1) 当 $0 \leq a_i \tau \leq 1, i=1, \dots, n$ 时, 式(5.3.6)可改写为:

$$\begin{aligned}\Delta V(\mathbf{Y}) &= -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j + \sum_i \tau a_i \eta_i \int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx - \sum_i \tau a_i \eta_i \Delta y_i \sigma_i^{-1}(y_i(k+1)) \\ &\quad - \sum_i (1 - \tau a_i) \eta_i \Delta y_i [\sigma_i^{-1}(y_i(k+1)) - \sigma_i^{-1}(y_i(k))] \\ &\leq -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j - \sum_i \tau a_i \eta_i \left[\Delta y_i \sigma_i^{-1}(y_i(k+1)) - \int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx \right]\end{aligned}$$

将 $\int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx$ 在 $y_i(k+1)$ 处展开, 有

$$\begin{aligned}&\int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx - \Delta y_i \sigma_i^{-1}(y_i(k+1)) + \frac{[\Delta y_i]^2}{2} \sigma_i^{-1'}(\xi_i) \\ &\geq \int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx - \Delta y_i \sigma_i^{-1}(y_i(k+1)) + [\Delta y_i]^2 / 2\beta_i\end{aligned}$$

其中, ξ_i 为 $y_i(k)$ 和 $y_i(k+1)$ 之间的值。于是, 得到

$$\Delta V(\mathbf{Y}) \leq -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j - \sum_i \tau a_i \eta_i [\Delta y_i]^2 / 2\beta_i = -\frac{\tau}{2} \Delta \mathbf{Y}^T [\mathbf{W} + \mathbf{A}\mathbf{B}] \Delta \mathbf{Y}$$

这表明, 当 $0 \leq a_i \tau \leq 1, i=1, \dots, n$ 时, 则 $\Delta V(\mathbf{Y}) \leq 0$ 仅当 $\mathbf{W} + \mathbf{A}\mathbf{B}$ 正定, $\Delta V(\mathbf{Y}) = 0$ 当且仅当 $\Delta \mathbf{Y} = 0$ 。从而, 系统可渐近收敛到不动点。进一步, 若 $a_i = a, \eta_i = \eta, \beta_i = \beta$, 则正定条件可简化为 $a\eta/\beta > -\lambda_{\min}(\mathbf{W})$ 。

(2) 当 $0 \leq a_i \tau \leq 1, i=1, \dots, n$ 时, 式(5.3.6)又可改写为

$$\begin{aligned}\Delta V(\mathbf{Y}) &= -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j + \sum_i \tau a_i \eta_i \int_{y_i(k)}^{y_i(k+1)} \sigma_i^{-1}(x) dx \\ &\quad - \sum_i \tau a_i \eta_i \Delta y_i \sigma_i^{-1}(y_i(k+1)) - \sum_i \eta_i \Delta y_i [\sigma_i^{-1}(y_i(k+1)) - \sigma_i^{-1}(y_i(k))]\end{aligned}$$

设 ζ_i, θ_i 取值于 $y_i(k+1)$ 和 $y_i(k)$ 之间, 且由于 $\sigma_i^{-1}(\cdot)$ 为严格单调增, 因此成立:

$$\Delta y_i [\sigma_i^{-1}(y_i(k+1)) - \sigma_i^{-1}(y_i(k))] \geq \Delta y_i [\sigma_i^{-1}(\theta_i) - \sigma_i^{-1}(y_i(k))] \geq 0$$

由中值定理, 可得

$$\begin{aligned} \Delta V(\mathbf{Y}) &= -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j + \sum_i \tau a_i \eta_i \Delta y_i [\sigma_i^{-1}(\theta_i) - \sigma_i^{-1}(y_i(k))] \\ &\quad - \sum_i \eta_i \Delta y_i [\sigma_i^{-1}(y_i(k+1)) - \sigma_i^{-1}(y_i(k))] \\ \Delta V(\mathbf{Y}) &\leq -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j - \sum_i (1 - \tau a_i) \eta_i \Delta y_i [\sigma_i^{-1}(y_i(k+1)) - \sigma_i^{-1}(y_i(k))] \\ &= -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j - \sum_i (1 - \tau a_i) \eta_i [\Delta y_i]^2 \sigma_i^{-1}(\zeta_i) \\ &\leq -\frac{1}{2} \Delta \mathbf{Y}^T [\tau \mathbf{W} + 2(\mathbf{I} - \tau \mathbf{A}) \mathbf{B}] \Delta \mathbf{Y} \end{aligned} \quad (5.3.7)$$

因此, 当 $0 \leq a_i \tau \leq 1, i=1, \dots, n$ 时, 则 $\Delta V(\mathbf{Y}) \leq 0$ 仅当 $\tau \mathbf{W} + 2(\mathbf{I} - \tau \mathbf{A}) \mathbf{B}$ 正定, $\Delta V(\mathbf{Y}) = 0$ 当且仅当 $\Delta \mathbf{Y} = 0$ 。由此, 系统可渐近收敛到不动点。进一步, 若 $a_i = a, \eta_i = \eta, \beta_i = \beta$, 则正定条件可简化为 $2(1 - \tau a) \eta / \beta \tau > -\lambda_{\min}(\mathbf{W})$ 。

(3) 当 $a_i < 0, i=1, \dots, n$ 时, 由式(5.3.7)得

$$\begin{aligned} \Delta V(\mathbf{Y}) &\leq -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j - \sum_i \eta_i \Delta y_i [\sigma_i^{-1}(y_i(k+1)) - \sigma_i^{-1}(y_i(k))] \\ &= -\frac{1}{2} \sum_{i,j} \tau w_{i,j} \Delta y_i \Delta y_j - \sum_i \eta_i [\Delta y_i]^2 \sigma_i^{-1}(\zeta_i) \\ &\leq -\frac{1}{2} \Delta \mathbf{Y}^T [\tau \mathbf{W} + 2\mathbf{B}] \Delta \mathbf{Y} \end{aligned}$$

因此, 当 $a_i < 0, i=1, \dots, n$ 时, 则 $\Delta V(\mathbf{Y}) \leq 0$ 仅当 $\tau \mathbf{W} + 2\mathbf{B}$ 正定, $\Delta V(\mathbf{Y}) = 0$ 当且仅当 $\Delta \mathbf{Y} = 0$ 。从而, 系统可渐近收敛到不动点。进一步, 若 $\eta_i = \eta, \beta_i = \beta$, 则正定条件可简化为 $2\eta / \beta \tau > -\lambda_{\min}(\mathbf{W})$ 。

推论 5.3.1 若离散动态反馈神经网络(5.3.3)满足 $\mathbf{W}^T = \mathbf{W}$ 和下述的条件之一, 则以同步迭代方式运行的网络必渐近收敛到系统的不动点。

(1) 当 $0 \leq a_i \tau \leq 1, i=1, \dots, n$ 时,

$$w_{i,i} + a_i \eta_i / \beta_i > \left| \sum_{j \neq i}^n w_{i,j} \right|$$

或

$$w_{i,i} + 2(1 - \tau a_i) \eta_i / \beta_i \tau > \left| \sum_{j \neq i}^n w_{i,j} \right|, \quad i = 1, 2, \dots, n$$

(2) 当 $a_i < 0, i=1, \dots, n$ 时,

$$w_{i,i} + 2\eta_i / \beta_i \tau > \left| \sum_{j \neq i}^n w_{i,j} \right|$$

证明 利用定理 5.3.1 和 Gerschgorin 定理,可直接导出此推论。

定理 5.3.2 考虑离散动态反馈神经网络(5.3.3),则以异步迭代方式运行的网络必渐近收敛到系统的不动点,如果满足 $W^T = W$ 和以下的任一条件:

(1) 当 $0 \leq a_i \tau \leq 1, i=1, \dots, n$ 时,

$$a_i \eta_i / \beta_i > -\min_{i=1}^n w_{i,i}$$

或 $2(1 - \tau a_i) \eta_i / \beta_i \tau > -\min_{i=1}^n w_{i,i} \quad i=1, 2, \dots, n;$

(2) 当 $a_i < 0, i=1, \dots, n$ 时,

$$2\eta_i / \beta_i \tau > -\min_{i=1}^n w_{i,i}$$

证明 不失一般性,设由 k 到 $k+1$ 时刻,仅第 l 个神经元状态发生变化,能量函数的选择同于式(5.3.5),则有

$$\begin{aligned} \Delta V(Y) = & -\frac{\tau}{2} w_{l,l} (\Delta y_l)^2 - (1 - \tau a_l) \eta_l \Delta y_l [\sigma_l^{-1}(y_l(k+1)) - \sigma_l^{-1}(y_l(k))] \\ & - \tau a_l \eta_l \Delta y_l \sigma_l^{-1}(y_l(k+1)) + \tau a_l \eta_l \int_{y_l(k)}^{y_l(k+1)} \sigma_l^{-1}(x) dx \end{aligned}$$

此后,类似于定理 5.3.1 的推证,即可证得此结论。

定理 5.3.3 考虑离散动态网络(5.3.3),若 $a_i \tau = 1, i=1, \dots, n$ 和 $W^T = W$,则以任何迭代方式运行的网络所得吸引子只能是不动点或 2 周期点。

证明 当 $a_i \tau = 1, i=1, \dots, n$ 时,有 $x_i(k+1) = \tau(\sum_j w_{i,j} y_j + s_i)$ 。进而,通过取有界能量函数 $V(Y)$,可导出其增量 $\Delta V(Y)$,有

$$\begin{aligned} V(Y) = & -\sum_{i,j} \tau w_{i,j} y_i(k) y_j(k-1) - \sum_i \tau s_i [y_i(k) - y_i(k-1)] \\ & + \sum_i \eta_i \left[\int_0^{y_i(k)} \sigma_i^{-1}(x) dx + \int_0^{y_i(k-1)} \sigma_i^{-1}(x) dx \right] \\ \Delta V(Y) = & -\sum_i \left[\Delta_2 y_i(k) \left(\sum_j \tau w_{i,j} y_j(k) + \tau s_i \right) \right] + \sum_i \eta_i \int_{y_i(k-1)}^{y_i(k+1)} \sigma_i^{-1}(x) dx \\ = & -\sum_i \eta_i \Delta_2 y_i(k) \sigma_i^{-1}(y_i(k+1)) + \sum_i \eta_i \int_{y_i(k-1)}^{y_i(k+1)} \sigma_i^{-1}(x) dx \\ = & \sum_i \eta_i \Delta_2 y_i(k) [\sigma_i^{-1}(\zeta_i) - \sigma_i^{-1}(y_i(k+1))] \end{aligned}$$

上式推导中已经利用了中值定理。其中, $\Delta_2 y_i(k) = y_i(k+1) - y_i(k-1)$, ζ_i 取值于 $y_i(k+1)$ 和 $y_i(k-1)$ 之间。

进而,由于 $\sigma_i^{-1}(\cdot)$ 为严格单调增,故当 $\Delta_2 y_i(k) \neq 0$ 时, $\Delta_2 y_i(k)$ 和 $\sigma_i^{-1}(\zeta_i) - \sigma_i^{-1}(y_i(k+1))$ 必互为异号,从而 $\Delta V(Y) < 0$; 当 $\Delta_2 y_i(k) = 0$ 时,则 $\Delta V(Y) = 0$ 。由此,离散动态反馈网络(5.3.3)的吸引子,只能或是不动点,或是 2 周期点。

注: 若将激励函数 $\sigma_i(\cdot)$ 取为 Sigmoid 函数 $y_i(t) = 1/[1 + \exp(-x_i(t)/\eta_i)]$, 则有 $\beta_i = 1/4$, 上述各个定理中的条件还可进一步简化。

5.3.1.2 非对称动态反馈网络的全局渐近稳定性分析

上节中,我们已经建立了对应于对称权矩阵的系统的不动点为渐近稳定的条件。但是,由于 VLSI 技术难以实现权矩阵的对称性,使得结果的工程应用受到一定的限制。此外,当网络用于优化计算时,总是希望网络能够渐近收敛到全局稳定平衡态,来得到对应问题的全局最优解。因此,下面的讨论中,着重于研究非对称权矩阵反馈网络的全局渐近稳定性。

设 $\mathbf{X}^* = [x_1^*, \dots, x_n^*]^T$ 为系统(5.2.2)的平衡态,则 $\mathbf{A}\mathbf{X}^* = \mathbf{S} + \mathbf{W}\mathbf{G}(\mathbf{X}^*)$ 。进而,令

$$\mathbf{Z} = \mathbf{X} - \mathbf{X}^* = [z_1, \dots, z_n]^T$$

$$\mathbf{H}(\mathbf{Z}) = [\sigma_1[(z_1 + x_1^*)/\eta_1] - \sigma_1(x_1^*/\eta_1), \dots, \sigma_n[(z_n + x_n^*)/\eta_n] - \sigma_n(x_n^*/\eta_n)]^T$$

则系统方程(5.3.2)可改写为

$$\dot{\mathbf{Z}} = -\mathbf{A}\mathbf{Z} + \mathbf{W}\mathbf{H}(\mathbf{Z})$$

其中 $\mathbf{Z}=0$ 为系统的平衡态。

同理,系统方程(5.3.4)可改写为 $\mathbf{Z}(k+1) = (\mathbf{I} - \tau\mathbf{A})\mathbf{Z}(k) + \tau\mathbf{W}\mathbf{H}(\mathbf{Z}(k))$, 系统的平衡态为 $\mathbf{Z}=0$ 。

定理 5.3.4 设 $0 < \sigma'_i(\cdot) \leq \beta_i, a_i > 0, i=1, \dots, n$, 令 $d = \max_i \{\beta_i/\eta_i\}, a_{\min} = \min_i \{a_i\}$, 则当 $\|\mathbf{W}\| < a_{\min}/d$ 时,连续动态反馈神经网络(5.2.2)的平衡点存在且惟一,并必为全局渐近稳定。

证明 由中值定理知

$$|\sigma_i(x_1/\eta_i) - \sigma_i(x_2/\eta_i)| = |\sigma'_i(\xi_i)/\eta_i(x_1 - x_2)| \leq d |x_1 - x_2|$$

由此可得

$$\|\mathbf{G}(\mathbf{X}_1) - \mathbf{G}(\mathbf{X}_2)\| \leq d \|\mathbf{X}_1 - \mathbf{X}_2\|$$

再考虑映射 $\mathbf{A}^{-1}[\mathbf{W}\mathbf{G}(\mathbf{X}) + \mathbf{S}]: \mathbb{R}^n \rightarrow \mathbb{R}^n$, 则有

$$\begin{aligned} \|\mathbf{A}^{-1}[\mathbf{W}\mathbf{G}(\mathbf{X}_1) + \mathbf{S}] - \mathbf{A}^{-1}[\mathbf{W}\mathbf{G}(\mathbf{X}_2) + \mathbf{S}]\| &\leq \|\mathbf{A}^{-1}\mathbf{W}\| \|\mathbf{G}(\mathbf{X}_1) - \mathbf{G}(\mathbf{X}_2)\| \\ &\leq \|\mathbf{A}^{-1}\| \|\mathbf{W}\| d \|\mathbf{X}_1 - \mathbf{X}_2\| \\ &\leq (d \|\mathbf{W}\| / a_{\min}) \|\mathbf{X}_1 - \mathbf{X}_2\| \end{aligned}$$

因此, $\|\mathbf{W}\| < a_{\min}/d$ 时,此映射为收缩映射。进而, $\mathbf{A}^{-1}[\mathbf{W}\mathbf{G}(\mathbf{X}) + \mathbf{S}] = \mathbf{X}$, 即 $\mathbf{A}\mathbf{X} = \mathbf{W}\mathbf{G}(\mathbf{X}) + \mathbf{S}$ 存在惟一的平衡点。而由中值定理知, $\|\mathbf{H}(\mathbf{Z})\| \leq d \|\mathbf{Z}\|$ 。

现取 Lyapunov 函数为 $V(\mathbf{Z}) = \mathbf{Z}^T \mathbf{Z}$, 则有

$$\begin{aligned} \dot{V}(\mathbf{Z}) &= \dot{\mathbf{Z}}^T \mathbf{Z} + \mathbf{Z}^T \dot{\mathbf{Z}} = (-\mathbf{A}\mathbf{Z} + \mathbf{W}\mathbf{H}(\mathbf{Z}))^T \mathbf{Z} + \mathbf{Z}^T (-\mathbf{A}\mathbf{Z} + \mathbf{W}\mathbf{H}(\mathbf{Z})) \\ &= -2\mathbf{Z}^T \mathbf{A}\mathbf{Z} + 2\mathbf{Z}^T \mathbf{W}\mathbf{H}(\mathbf{Z}) \\ &\leq -2\|\mathbf{A}\| \|\mathbf{Z}\|^2 + 2\|\mathbf{W}\| d \|\mathbf{Z}\|^2 \\ &= -2(a_{\min} - d \|\mathbf{W}\|) \|\mathbf{Z}\|^2 \end{aligned}$$

故当 $\|\mathbf{W}\| < a_{\min}/d$ 时, $\dot{V}=0$ 当且仅当 $\|\mathbf{Z}\|=0$, 有 $\dot{V} < 0$, 从而系统为全局渐近稳定。

定理 5.3.5 设 $0 < \sigma'_i(\cdot) \leq \beta_i, a_i > 0, i = 1, \dots, n, d = \max_i \{\beta_i / \eta_i\}, a_{\min} = \min_i \{a_i\}$, 则当 $\|W\| < a_{\min}/d$ 时, 离散动态反馈神经网络(5.3.4)存在惟一的平衡点。

证明 类似于定理 5.3.4 的证明, 可得

$$\begin{aligned} & \|Z_1(k+1) - Z_2(k+1)\| \\ &= \|(I - \tau A)Z_1(k) + \tau WH(Z_1(k)) - (I - \tau A)Z_2(k) - \tau WH(Z_2(k))\| \\ &\leq \|I - \tau A\| \|Z_1(k) - Z_2(k)\| + \tau \|W\| d \|Z_1(k) - Z_2(k)\| \\ &\leq (1 - \tau a_{\min} + \tau d \|W\|) \|Z_1(k) - Z_2(k)\| \end{aligned}$$

故当 $\|W\| < a_{\min}/d$ 时, $\|Z_1(k+1) - Z_2(k+1)\| < \|Z_1(k) - Z_2(k)\|$ 。从而, 离散动态网络存在惟一平衡点 $Z=0$ 且为渐近稳定。

定理 5.3.6 若定理 5.3.5 的条件成立, 离散动态反馈神经网络(5.3.4)为全局渐近稳定的一个充分条件是: 存在正定阵 P 和标量 $\omega > 0$, 使得

$$(1 + \omega)(I - \tau A)P(I - \tau A) - P + (1 + \omega^{-1})\tau^2 d^2 W^T P W < 0$$

证明 取 $V = Z^T P Z$, 则

$$\begin{aligned} \Delta V[Z(k)] &= Z^T(k+1)PZ(k+1) - Z^T(k)PZ(k) \\ &= [(I - \tau A)Z(k) + \tau WH(Z(k))]^T P [(I - \tau A)Z(k) \\ &\quad + \tau WH(Z(k))] - Z^T(k)PZ(k) \\ &= Z^T(k)[(I - \tau A)P(I - \tau A) - P]Z(k) - 2\tau Z^T(k)(I - \tau A)PWH(Z(k)) \\ &\quad + \tau^2 H^T(Z(k))W^T PWH(Z(k)) \end{aligned}$$

由于 P 正定, 故对任意 $\omega > 0$, 有

$$\begin{aligned} & [\omega^{1/2}(I - \tau A)Z + \tau\omega^{-1/2}WH(Z)]^T P [\omega^{1/2}(I - \tau A)Z + \tau\omega^{-1/2}WH(Z)] \\ &\geq 0 \Rightarrow \omega Z^T(I - \tau A)P(I - \tau A)Z + \tau^2 \omega^{-1}H^T(Z)W^T PWH(Z) \\ &\geq -2\tau Z^T(I - \tau A)PWH(Z) \end{aligned}$$

再由中值定理, 可得 $\|H(Z)\| \leq d\|Z\|$, 从而有

$$\begin{aligned} \Delta V[Z(k)] &\leq Z^T(k)[(1 + \omega)(I - \tau A)P(I - \tau A) - P]Z(k) \\ &\quad + (1 + \omega^{-1})\tau^2 H^T(Z(k))W^T PWH(Z(k)) \\ &\leq Z^T(k)[(1 + \omega)(I - \tau A)P(I - \tau A) - P + (1 + \omega^{-1})\tau^2 d^2 W^T P W]Z(k) \end{aligned}$$

由此, 就证明了, 当 $(1 + \omega)(I - \tau A)P(I - \tau A) - P + (1 + \omega^{-1})\tau^2 d^2 W^T P W$ 为负定时, 离散动态反馈神经网络是全局渐近稳定的。

推论 5.3.2 离散动态网络(5.3.4)为全局渐近稳定, 如果满足以下条件之一:

$$(1) \lambda_{\max}[(1 + \omega)(I - \tau A)^2 - I + (1 + \omega^{-1})\tau^2 d^2 W^T W] < 0$$

$$\text{或 } \lambda_{\max}[(1 + \omega^{-1})\tau^2 d^2 W^T W] < 1 - (1 + \omega)(1 - \tau a_{\min})^2$$

$$(2) \begin{cases} (1 + \omega^{-1})\tau^2 d^2 \sum_{k=1}^n w_{k,i}^2 + (1 + \omega)(1 - \tau a_i)^2 - 1 < 0 \\ (1 + \omega^{-1})\tau^2 d^2 \sum_{j=1}^n \sum_{k=1}^n |w_{k,i} w_{k,j}| < 1 - (1 + \omega)(1 - \tau a_i)^2, \quad i = 1, \dots, n \end{cases}$$

证明 直接利用定理 5.3.6 (令 $P=I$) 和 Gershgorin 定理, 即可证得。

5.3.1.3 时延动态反馈网络的全局渐近稳定性分析

在神经网络的 VLSI 实现中, 信号的传输必定存在一定的时延, 因此对时延网络的研究是具有工程意义的。下面, 限于讨论如下—类的时延动态反馈网络:

$$\dot{X}(t) = -AX(t) + W_1 G(X(t)) + W_2 G(X(t-\tau)) + S \quad (5.3.8)$$

其中,

$$A = \text{diag}\{a_1, \dots, a_n\}, a_i > 0, S = [s_1, \dots, s_n]^T, W_1 = (w_{ij}^{[1]})_{n \times n}, W_2 = (w_{ij}^{[2]})_{n \times n},$$

$$W = W_1 + W_2, G(X) = [g_1(x_1), \dots, g_n(x_n)]^T, g_i(x) = \sigma_i(x/\eta_i)$$

令 $X^* = [x_1^*, \dots, x_n^*]^T$ 为平衡点, 且取变换 $Z = X - X^*$, 则 $Z=0$ 为变换后的平衡点。记

$$\begin{aligned} H(Z(\cdot)) &= [h_1(z_1(\cdot)), \dots, h_n(z_n(\cdot))]^T \\ &= [g_1(z_1(\cdot) + x_1^*) - g_1(x_1^*), \dots, g_n(z_n(\cdot) + x_n^*) - g_n(x_n^*)]^T \end{aligned}$$

则网络方程可改写为

$$\dot{Z}(t) = -AZ(t) + W_1 H(Z(t)) + W_2 H(Z(t-\tau)) \quad (5.3.9)$$

定理 5.3.7 设 $0 < \sigma'_i(\cdot) \leq \beta_i, a_i > 0, i=1, \dots, n, d = \max_i \{\beta_i/\eta_i\}, a_{\min} = \min_i \{a_i\}$, 则当不等式 $d(\|W_1\| + \|W_2\|) < a_{\min}$ 成立时, 时延动态网络(5.3.8)对任意时延均为全局渐近稳定。

证明

(1) 先证明平衡点的惟一性。由中值定理知:

$$|g_i(x_1) - g_i(x_2)| = |\sigma'_i(\zeta_i)/\eta_i(x_1 - x_2)| \leq d |x_1 - x_2|$$

由此可得,

$$\|G(X_1) - G(X_2)\| \leq d \|X_1 - X_2\|$$

考虑映射

$$A^{-1}[WG(X) + S]: R^n \rightarrow R^n$$

则有

$$\begin{aligned} \|A^{-1}[WG(X_1) + S] - A^{-1}[WG(X_2) + S]\| &\leq \|A^{-1}\| \|W\| \|G(X_1) - G(X_2)\| \\ &\leq \|A^{-1}\| (\|W_1\| + \|W_2\|) d \|X_1 - X_2\| \\ &\leq \frac{d(\|W_1\| + \|W_2\|)}{a_{\min}} \|X_1 - X_2\| \end{aligned}$$

因此, 当 $d(\|W_1\| + \|W_2\|) < a_{\min}$ 时, 此映射为收缩映射。而由收缩映射定理知, $A^{-1}[WG(X) + S] = X$, 即 $AX - WG(X) - S = 0$ 有惟一的平衡点。从而, $Z=0$ 为系统(5.3.9)的惟一平衡点。

(2) 再证明系统的全局渐近稳定性。令

$$V(Z(t)) = Z^T(t)Z(t)$$

由中值定理知

$$\|H(Z(\cdot))\| \leq d \|Z(\cdot)\|$$

于是,有

$$\begin{aligned} \dot{V}(Z(t)) &= -2Z^T(t)AZ(t) + 2Z^T(t)W_1H(Z(t)) + 2Z^T(t)W_2H(Z(t-\tau)) \\ &\leq -2a_{\min} \|Z(t)\|^2 + 2d \|W_1\| \|Z(t)\|^2 + 2d \|Z(t)\| \|W_2\| \|Z(t-\tau)\| \end{aligned}$$

按照 Razumikhin-type 定理(Hale,1977)的步骤:

假定 $\forall q > 1, V(Z(t-\tau)) \leq q^2 V(Z(t))$, 即 $\|Z(t-\tau)\| \leq q \|Z(t)\|$;

由于 $d(\|W_1\| + \|W_2\|) < a_{\min}$, 令 $d\|W_2\| = \lambda(a_{\min} - d\|W_1\|)$, $\lambda \in (0, 1)$, 则

$$\dot{V}(Z(t)) \leq -2(1-\lambda q) \|Z(t)\|^2 (a_{\min} - d\|W_1\|)$$

由此,取 $1 < q < 1/\lambda$, $\dot{V}(Z(t)) = 0$ 当且仅当 $Z(t) = 0$, 则知 $\dot{V}(Z(t)) < 0$; 且 $\|Z(t)\| \rightarrow \infty$ 时, $V(Z(t)) \rightarrow \infty$. 因此,根据 Razumikhin-type 定理知,时延动态反馈网络为全局渐近稳定,且与时延值无关。

注:当 $W_2 = 0$ 时,时延网络可转化为非时延网络。相应地,稳定条件可简化为 $d\|W_1\| < a_{\min}$,与上一节中的分析结果一致。

定理 5.3.8 设所有神经元激励函数相同,且 $0 < \sigma'_i(\cdot) \leq \beta_i$, 则时延动态反馈神经网络对任意时延值和外部激励均为全局渐近稳定的条件是

$$\beta \sum_{j=1}^n (|w_{ij}^{[1]}| + |w_{ij}^{[2]}| + |w_{ji}^{[1]}| + |w_{ji}^{[2]}|) < 2\eta a_i, \quad i = 1, \dots, n$$

证明 首先,证明 $Z=0$ 为系统(5.3.9)的惟一平衡点。由于 $\sigma(\cdot)$ 为严格单调增,则 $h(z_i)$ 与 z_i 同号。而由中值定理得, $h^2(z_i) \leq \beta/\eta h(z_i)z_i$ 。

反设系统(5.3.9)存在平衡点 Z^* , 满足 $H(Z^*) \neq 0$ 。那么,由于 Z^* 满足

$$AZ^* - W_1H(Z^*) - W_2H(Z^*) = 0$$

对上式左乘 $\beta/\eta H^T(Z^*)$ 可得

$$\beta/\eta [H^T(Z^*)AZ^* - H^T(Z^*)(W_1 + W_2)H(Z^*)] = 0$$

并且,因为

$$\begin{aligned} \beta/\eta H^T(Z^*)AZ^* &= \sum_{i=1}^n a_i \beta/\eta h(z_i)z_i \\ &\geq \sum_{i=1}^n a_i h^2(z_i) = H^T(Z^*)AH(Z^*) \end{aligned}$$

所以

$$H^T(Z^*)[A - \beta/\eta(W_1 + W_2)]H(Z^*) \leq 0 \quad (5.3.10)$$

也即

$$\begin{aligned} 0 &\geq \sum_i a_i h^2(z_i^*) - \beta/\eta \sum_i \sum_j (w_{i,j}^{[1]} + w_{i,j}^{[2]}) h(z_i^*) h(z_j^*) \\ &\geq \sum_i a_i h^2(z_i^*) - \beta/\eta \sum_i \sum_j (|w_{i,j}^{[1]}| + |w_{i,j}^{[2]}|) \frac{h^2(z_i^*) + h^2(z_j^*)}{2} \end{aligned}$$

$$= \frac{1}{2} \sum_i \left[2a_i - \beta/\eta \sum_j (|w_{ij}^{[1]}| + |w_{ji}^{[1]}| + |w_{ij}^{[2]}| + |w_{ji}^{[2]}|) \right] h^2(z_i^*)$$

显然,当

$$\beta \sum_{j=1}^n (|w_{ij}^{[1]}| + |w_{ij}^{[2]}| + |w_{ji}^{[1]}| + |w_{ji}^{[2]}|) < 2\eta a_i, \quad i = 1, \dots, n$$

时,只有 $H(Z^*)=0$ 才能使上式成立。因此,平衡点必满足 $H(Z^*)=0$ 。而由于 $h(z_i)$ 与 z_i 同号,这意味着 $Z=0$ 为系统(5.3.9)的惟一平衡点。

下面,证明系统的稳定性。取正定能量函数为

$$V(Z(t)) = \frac{1}{2} \sum_{i=1}^n \left\{ z_i^2(t) + \sum_{j=1}^n \beta/\eta |w_{ij}^{[2]}| \int_{t-\tau_j}^t z_j^2(\zeta) d\zeta \right\}$$

其导数为

$$\begin{aligned} \dot{V}(Z(t)) &= \sum_i \left\{ z_i(t) \left[-a_i z_i(t) + \sum_j w_{ij}^{[1]} h(z_j(t)) + \sum_j w_{ij}^{[2]} h(z_j(t-\tau_j)) \right] \right. \\ &\quad \left. + \frac{1}{2} \beta/\eta \sum_j w_{ij}^{[2]} [z_j^2(t) - z_j^2(t-\tau_j)] \right\} \\ &\leq \sum_i \left\{ -a_i z_i^2(t) + \beta/\eta \left(\sum_j |w_{ij}^{[1]}| |z_i(t)| |z_j(t)| \right) \right. \\ &\quad \left. + \sum_j |w_{ij}^{[2]}| (|z_i(t)| |z_j(t-\tau_j)|) + \frac{1}{2} \sum_j |w_{ij}^{[2]}| (z_j^2(t) - z_j^2(t-\tau_j)) \right\} \\ &\leq \sum_i \left\{ -a_i z_i^2(t) + \frac{1}{2} \beta/\eta \sum_j |w_{ij}^{[1]}| (z_i^2(t) + z_j^2(t)) + \frac{1}{2} \beta/\eta \sum_j |w_{ij}^{[2]}| (z_i^2(t) \right. \\ &\quad \left. + z_j^2(t-\tau_j)) \right\} + \sum_i \frac{1}{2} \beta/\eta \sum_j |w_{ij}^{[2]}| (z_j^2(t) - z_j^2(t-\tau_j)) \\ &= -\frac{1}{2} \sum_i \left[2a_i - \beta/\eta \sum_j (|w_{ij}^{[1]}| + |w_{ji}^{[1]}| + |w_{ij}^{[2]}| + |w_{ji}^{[2]}|) \right] z_i^2(t) \end{aligned}$$

由此可知,当

$$\beta \sum_{j=1}^n (|w_{ij}^{[1]}| + |w_{ij}^{[2]}| + |w_{ji}^{[1]}| + |w_{ji}^{[2]}|) < 2\eta a_i, \quad i = 1, \dots, n$$

时, $\dot{V}(Z(t))=0$ 当且仅当 $Z(t)=0$, 即 $\dot{V}(Z(t))<0$; 且 $\|Z(t)\| \rightarrow \infty$ 时, $V(Z(t)) \rightarrow \infty$ 。因此,时延动态反馈网络对任意时延均是全局渐近稳定的。

注:若网络连接权值同时满足对称条件,则稳定条件还可简化为

$$\beta \sum_{j=1}^n (|w_{ij}^{[1]}| + |w_{ij}^{[2]}|) < \eta a_i, \quad i = 1, \dots, n$$

5.3.2 动态反馈神经网络的收敛域估计

在将动态网络用于联想存储时,局部稳定平衡态可作为存储信息的吸引子所构成的

并行神经记忆结构。对此,神经网络的设计不仅涉及到对整体稳定性的研究,而且还要对其各平衡态的稳定吸引域大小进行定量的估计,以得到较好的容错性和动态范围。

讨论连续动态反馈神经网络: $\dot{Z} = -AZ + WH(Z)$, 且 $Z=0$ 为网络的平衡点, $H=[h_1, \dots, h_n]^T$ 。假定激励函数满足 $|\sigma_i(\cdot)| \leq M, a_i > 0, i=1, \dots, n$, 则 $|h_i(\cdot)| \leq 2M$ 。

设 $Z \in S$, S 为以原点为中心且半径为 r 的闭域, 并令

$$a = \min_i \{a_i\}, \quad \|Z(t)\| = \max_i \{|z_i(t)|\}, \quad T_i = \max_j \{|w_{ij}|\}, \quad T = \max_i \{T_i\},$$

$$\lambda_i = \sum_{j=1}^n w_{ij} h_j(z_j), \quad \lambda = [\lambda_1, \dots, \lambda_n]^T, \quad p_i^r = T_i \max_{y \in S} \sigma'_i(y_j) / \eta_i, \quad p^r = \max_i \{p_i^r\}$$

那么,由中值定理可知,

$$|h_i(z_i)| \leq (\max_{y \in S} \sigma_i(y_i)) |z_i| / \eta_i$$

从而

$$|\lambda_i| \leq \sum_{j=1}^n |w_{ij}| (\max_{y \in S} \sigma_j(z_j) / \eta_j) |z_j| \leq \sum_{j=1}^n p_i^r |z_j| \leq np_i^r \|Z\|$$

所以

$$\|\lambda\| \leq np^r \|Z\|$$

再令 $\|Z(t_0)\| = \delta$, 则因为

$$\begin{aligned} \|\lambda\| &\leq \max_i \{|\lambda_i|\} \leq \max_i \left\{ \sum_{j=1}^n |w_{ij}| \cdot 2M \right\} \\ &\leq \max_i \left\{ \sum_{j=1}^n T_j 2M \right\} \leq \max_i \{2nMT_i\} \leq 2nMT \end{aligned}$$

同时,由系统 t 时刻的状态可表为

$$Z(t) = e^{-A(t-t_0)} Z(t_0) + \int_{t_0}^t e^{-A(t-\tau)} WH d\tau$$

所以

$$\begin{aligned} \|Z(t)\| &\leq e^{-a(t-t_0)} \delta + \int_{t_0}^t e^{-a(t-\tau)} \|\lambda\| d\tau \leq e^{-a(t-t_0)} \delta + \int_{t_0}^t e^{-a(t-\tau)} 2nMT d\tau \\ &= \delta e^{-a(t-t_0)} + [1 - e^{-a(t-t_0)}] \cdot 2nMT/a \end{aligned}$$

因此,可得 $\lim_{t \rightarrow \infty} \|Z(t)\| \leq 2nMT/a$ 。这说明,只要初始状态位于 S 域,网络的最终状态必将进入以原点为中心且半径为 $R=2nMT/a$ 的区域。

Gronwall 引理 (Balabanian et al, 1969) 若 $\alpha(t) \leq \beta + \int_0^t (\varphi(\tau)\alpha(\tau) + \theta(\tau)) d\tau$, 其中 β 为正常数, $\alpha(t)$, $\varphi(t)$ 和 $\theta(t)$ 为非负连续函数, 则 $\alpha(t) \leq \beta \exp \left\{ \int_0^t [\varphi(\tau) + \theta(\tau)/\beta] d\tau \right\}$ 。

下面,在上述引理的基础上,我们来讨论最大收敛域的构造。假定 $\|Z(t_0)\| = \delta \leq r$, r 为网络在平衡点 $Z=0$ 的收敛半径, 即 $\|Z(t)\| \leq r$ 。

因为

$$\|Z(t)\| \leq \delta e^{-a(t-t_0)} + \int_{t_0}^t e^{-a(t-\tau)} np^r \|Z\| d\tau$$

所以

$$\|Z(t)\| e^{at} \leq \delta e^{at_0} + \int_0^t e^{a\tau} n p^r \|Z(t)\| d\tau$$

从而由引理知

$$\|Z(t)\| e^{at} \leq \delta e^{at_0} \exp\left\{\int_0^t n p^r d\tau\right\} = \delta e^{at_0} e^{n p^r (t-t_0)}$$

所以

$$\|Z(t)\| \leq \delta e^{(n p^r - a)(t-t_0)} \leq r e^{(n p^r - a)(t-t_0)}$$

若 $n p^r < a$, $Z(t)$ 将最终收敛到平凡解 ($Z=0$)。

由于 $r_1 > r_2 \Rightarrow p^r_1 \geq p^r_2$, 因此最大收敛域的构造归结为寻求满足 $n p^r < a$ 的最大 r 值。

基此, 给出估计收敛域的算法。

- (1) 给定初始半径 $r = 2nMT/a$, 变化量 Δ 和预定最大收敛半径 R 。
- (2) 若 $n p^r < a$, 则 $\text{flagadd} = 1, \text{flagsub} = 0$; 否则 $\text{flagsub} = 1, \text{flagadd} = 0$ 。
- (3) 若 $\text{flagadd} = 1$, 进行以下步骤:
 - (3.1) $r = r + \Delta$ 。
 - (3.2) 若 $n p^r < a$ 且 $r < R$, 则返回(3.1); 否则, 转(3.3)。
 - (3.3) 输出最大收敛半径 r 并转(5)。
- (4) 若 $\text{flagsub} = 1$, 进行以下步骤:
 - (4.1) $r = r - \Delta$ 。
 - (4.2) 若 $n p^r > a$ 且 $r > 0$, 则返回(4.1); 否则, 转(4.3)。
 - (4.3) 输出最大收敛半径 r 并转(5)。
- (5) 结束算法。

如果平衡点为非原点, 则通过平移处理, 可将其移到原点, 从而可应用上述算法对其收敛域进行估计。由于构造算法中利用了大量的不等式, 因此所得的收敛域有一定的保守性, 更为精确的估计有待于进一步的研究。

5.4 基于混沌动态的优化研究概述

混沌是一种普遍的非线性现象, 其行为复杂且类似随机, 但存在精致的内在规律性。混沌的发现, 对科学的发展具有空前而深远的影响。近年来, 混沌控制(Ott et al, 1990)、混沌同步(Pecora et al, 1990)和混沌神经网络(Aihara et al, 1990)受到了广泛关注, 并展现出诱人的应用与发展前景。

混沌具有其独特性质: (1) 随机性, 即混沌具有类似随机变量的杂乱表现; (2) 遍历性, 即混沌能够不重复地历经一定范围内的所有状态; (3) 规律性, 即混沌是由确定性的迭代式产生的。介于确定性和随机性之间, 混沌具有丰富的时空动态, 系统动态的演变可

导致吸引子的转移。最重要的是,混沌的遍历性特点可作为搜索过程中避免陷入局部极小的一种优化机制,这与模拟退火的概率性劣向转移和禁忌搜索的禁忌表检验存在明显的区别。因此,混沌已成为一种新颖的优化工具,并受到广泛重视和大量研究。

5.4.1 基于混沌神经网络的组合优化概述

鉴于在高强度连接下的神经网络依靠集体协同作用能自发产生计算行为,Hopfield等(1982)提出了反馈型的 Hopfield 神经网络,并用于求解组合优化问题。但是,基于 HNN 的优化计算通常会导致以下问题:单纯梯度下降策略易使网络最终收敛到局部极小解,而非问题的全局最优;网络可能会收敛到问题的不可行解;与能量函数相关的参数难以确定,而且参数鲁棒性和初值鲁棒性较差。从而,HNN 的应用受到了很大的限制,尤其对复杂或大规模的问题。

受生物神经元混沌特性的启发,通过在 HNN 中引入混沌动态,Aihara 等(1990)提出了混沌神经网络模型(chaotic neural network,CNN)。基于 Euler 离散化的 HNN,通过增加一个大的自反馈项,Nozawa(1992)得到了类似的 CNN 模型。借鉴混沌动态的遍历性特点,搜索过程不受能量障碍的限制,从而可有效避免优化过程陷入局部极小解。对 TSP 的优化研究,验证了 CNN 相对随机算法的有效性。从而,混沌神经网络成为了改进 HNN 的优化效率和质量,乃至解决组合优化问题的有效工具,并在近几年受到优化领域的广泛关注与研究。

假设 HNN 模型的动态方程及其离散方程如下:

$$dx_i/dt = -\partial E/\partial y_i = -x_i/\tau + \sum_j w_{ij}y_j + I_i \quad (5.4.1)$$

$$x_i(k+1) = (1 - \Delta t/\tau)x_i(k) + \Delta t(\sum_j w_{ij}y_j + I_i) \quad (5.4.2)$$

其中, x, y 分别为神经元的输入和输出, Δt 为离散化步长。

借鉴模拟退火的退温策略,通过控制 CNN 的参数以控制网络的动态,是利用 CNN 求解组合优化的一类有效途径。Chen 等(1995)提出了混沌模拟退火算法,即先利用瞬时混沌动态进行全局遍历性搜索,而后再转入梯度下降搜索,并成功解决了 TSP, N-queen, MSP(maintenance scheduling problem)等问题。其模型的动态方程可描述如下:

$$\begin{cases} x_i(k+1) = \gamma x_i(k) + \alpha(\sum_{j \neq i} w_{ij}y_j(k) + I_i) - z(k)(y_i(k) - I_0) \\ z(k+1) = (1 - \beta)z(k), \beta \in [0, 1] \end{cases} \quad (5.4.3)$$

其中, z 为控制参数, I_0 为一定的阈值。

通过改变 Euler 离散化 HNN 的时间步长,Wang 等(1998)提出了混沌模拟退火算法,其中时间步长作为分岔参数用以控制网络的动态,通过对时间步长的衰减控制,网络动态同样能够经历一个由混沌到逆分岔再到稳定搜索的过程,而前一阶段的搜索恰好用

以避免优化过程陷入局部极小。其模型的动态方程可描述如下：

$$\begin{cases} x_i(k+1) = (1 - \Delta t(k)/\tau)x_i(k) + \Delta t(k) \left(\sum_j w_{ij}y_j(k) + I_i \right) \\ \Delta t(k+1) = \beta \Delta t(k), \beta \in (0, 1] \end{cases} \quad (5.4.4)$$

通过在 HNN 中引入外部机制产生的混沌噪声来构成混沌神经网络,是其用于组合优化问题的另一种途径。这种方法基于混沌时间序列永不可精确重复的规律,当外部引入的混沌的幅度足够小时,能量函数表面上的寻优过程在具有丰富时空特性的混沌动态的激励下能够产生避免陷入局部极小的能力,并保持对问题的原始描述不变。通常,这种策略被称为“外方法”(external approaches),上文介绍的 CNN 策略相应地被称为“内方法”(internal approaches)。这类模型的动态方程可描述如下:

$$x_i(k+1) = (1 - \Delta t/\tau)x_i(k) + \Delta t \left(\sum_j w_{ij}y_j + I_i - \eta_i(k) \right) \quad (5.4.5)$$

其中, η 为归一化的外部混沌序列,譬如 Logistic 映射。

王凌等(2000)将“内模型”和“外模型”集成为一个模型,同时区别于上述算法所采用的指数退温策略,即 $z(k) = \lambda z(k-1)$,他们对控制参数 $z(k)$ 提出了另一种控制策略,即 $z(k) = z(k-1) / \ln[e + \lambda(1 - z(k-1))]$,使得 $z(k+1)/z(k)$ 随 k 增大而加快,进而来控制 $z(k)$ 的下降速率。如此设计目的是,使混沌动态有足够长的进程以提高粗搜索性能,并利用 z 较小时退温速率的加快以加快梯度搜索来改善时间性能,此外他们分析和归纳了模型参数对优化性能的影响,并给出了若干参数选取的规律性结论。

在“内模型”方面,Zhou 等(1997)提出的混沌退火算法也采用了类似的优化机制,但其混沌动态是通过引入非线性项而产生的。在理论研究方面,Chen 等(1999)对混沌神经网络和离散回归网络的吸引集及其特性进行了分析,并得到了网络具有全局搜索能力的充分条件;Tokuda 等(1997, 1998)建议以转换期引起的间歇动态现象作为混沌在局部极小间切换的潜在机制。

在“外模型”方面,Zhou 等(1997)利用 Henon 映射产生的混沌时间序列作为外部噪声,算法解决 100 城市 TSP 的性能优于 Boltzmann 机。Hayakawa 等(1995)利用不同分岔参数下的 Logistic 映射产生外部噪声,研究了噪声对优化的影响,并得到了一些能够产生良好优化能力的分岔参数值。通过在 HNN 中引入不同类型的混沌噪声,Asai 等(1995)进一步研究了混沌噪声出现时网络对局部极小的搜索动态,并观察和比较了混沌噪声与随机噪声下算法对最优解的跟踪能力。Hasegawa 等(1997)认为,不同方式产生的混沌噪声只要具有相同的自相关性就能产生几乎相同的优化能力。

可见,由于基于混沌的粗搜索的进行,CNN 算法最终梯度下降的范围缩小到最优解或满意解的局部邻域,因此在优化性能方面 CNN 相对 HNN 具有较大的优越性,这也通过大量数值研究得到了验证。但是,基于 CNN 的优化也存在一定的缺陷:首先,基于 CNN 的优化过程和应用仍受到 HNN 的限制,CNN 仍需将问题的解映射成适合网络描述的结构,如置换码映射到矩阵码;其次,CNN 包容了 HNN 的所有参数,并增加了混沌

项及其控制过程,算法复杂性相对增加。

5.4.2 基于混沌序列的函数优化研究概述

轨道遍历性,即混沌序列能够不重复地历经一定范围内的所有状态,是混沌用于函数优化的根本出发点。通常,基于混沌动态的搜索过程分为如下两个阶段:

首先,基于确定性迭代式产生的遍历性轨道对整个解空间进行考察。当满足一定终止条件时,认为搜索过程中发现的最佳状态(best so far)已接近问题的最优解(只要遍历性搜索轨道足够长,这种情况总能实现),并以此作为第二阶段的搜索起始点。

其次,以第一阶段得到的结果为中心,通过附加小幅度的扰动进一步进行局部区域内的细搜索,直至算法终止准则满足。其中,所附加的扰动可以是混沌变量,或者是基于高斯分布或柯西分布或均匀分布等的随机变量,也可以是按梯度下降机制计算产生的偏置值。

基于上述思想,李兵等(1997)利用类似载波的方法将 Logistic 映射产生的混沌变量引入到优化变量中,同时将混沌运动的遍历范围转换到优化变量的定义域,然后利用混沌变量进行搜索。其具体步骤如下:

(1) 令 $k=1, k'=1$, 对 Logistic 映射, 即 $x_{n+1}=4x_n(1-x_n)$ 中的 x_n 分别赋予 i 个具有微小差异的初值, 从而得到 i 个轨迹不同的混沌变量 $x_{i,n+1}$ 。

(2) 通过式 $x'_{i,n+1}=c_i+d_ix_{i,n+1}$, 用载波的方法将选定的混沌变量 $x_{i,n+1}$ 转换为用于优化的混沌变量 $x'_{i,n+1}$, 其中 c_i, d_i 为常数, 对变量进行尺度变换。

(3) 用混沌变量进行迭代搜索。令 $x_i(k)=x'_{i,n+1}$, 计算相应的性能指标 $f(k)$, 令 $x_i^*=x_i(0), f^*=f(0)$ 。若 $f(k)\leq f^*$, 则令 $f^*=f(k), x_i^*=x_i(k)$; 否则放弃 $x_i(k)$ 。令 $k=k+1$ 。

(4) 若经过步骤(3)的若干步搜索 f^* 都保持不变, 则按式 $x''_{i,n+1}=x_i^*+a_ix_{i,n+1}$ 进行第二次载波, 反之则返回步骤(3)。其中 a_i 为调节常数, 使得 $a_ix_{i,n+1}$ 为小幅度混沌变量。

(5) 用二次载波后的混沌变量继续进行迭代搜索。令 $x_i(k')=x''_{i,n+1}$, 计算相应的性能指标 $f(k')$ 。若 $f(k')\leq f^*$, 则令 $f^*=f(k'), x_i^*=x_i(k')$; 否则放弃 $x_i(k')$, 令 $k'=k'+1$ 。

(6) 若算法终止准则满足, 则输出最优解 x^* ; 否则返回步骤(5)。

张彤等(1999)为提高优化性能, 提出了一种变尺度的混沌优化方法, 其特点在于: 算法根据进程, 不断缩小优化变量的搜索区域, 不断改变第二阶段搜索的调节参数。王子才等(1999)将混沌的遍历性机制引入模拟退火算法, 使得搜索过程同时具有两者的优点, 其特点在于: 首先, 算法产生一混沌序列, 在进行混沌遍历性搜索的同时, 结合模拟退火算法的接受函数来确定初始温度, 使得算法在此初温下具有较好的随机性(即在初温下算法几乎能接受任意状态); 其次, 利用混沌细搜索策略作为模拟退火的状态产生函数, 同时对优化变量的转移方式进行了改进以略去传统方法的越界处理, 并对调节参数实施自适应控

制策略。Choi 等(1998)将混沌动态引入最陡下降法进行函数优化,并采用了并行搜索结构,其中混沌动态用于跳出局部极小,而最陡下降法用于在局部极小进行细搜索,同时还给出了调整混沌突跳幅度的自适应机制。

基于混沌动态的函数优化并非局限于上述搜索流程,Zhou 等(1997)将一种非线性特性引入传统的梯度下降策略,并使其在一定的控制参数下产生混沌行为,从而通过遍历性搜索避免陷入局部极小;进而,通过对控制参数的“退温”处理,使混沌行为逐渐消失并最终进入确定性的梯度下降以快速得到最优解。相比二阶段混沌优化方法,这种方法更为简单,利用对参数的控制过程替代了搜索过程的二阶段分割,同时梯度下降使得搜索对优化曲面有一定的认知性,但显然不适合于不可导的函数问题。

可见,上述函数问题的混沌优化策略思路直观,容易程序化实现,比较适合于连续变量的函数优化问题。但是它也存在明显的缺点,即当搜索起始点选择不合适或遍历区间很大或控制参数及其控制策略选取不合适时,搜索结果很难达到或接近最优解,或者说算法可能需要花费很长的时间才能取得较好的优化性能。因此,为了使混沌优化具有更为卓越的性能,如何选择搜索起点、如何缩小搜索空间、如何设计好局部搜索方式、如何设计好两个阶段的终止准则、如何选取合适的初始控制参数及其控制策略,仍是提高上述基于混沌动态的优化算法性能的关键。

5.4.3 混沌优化的发展性研究

无论是函数优化,还是组合优化,算法利用混沌的本质都是借助其遍历性来避免搜索过程陷入局部极小的。因此,如何利用和控制好混沌动态,是这类算法产生良好性能的关键。纵观上述研究进展,可见混沌优化研究目前还存在大量不足之处,许多方面有待改进和完善。下面,我们从几个方面进行讨论和归纳,并指出值得进一步研究的内容。

(1) 目前,许多结论都是根据对特定问题的数值仿真研究得到的,不仅缺乏严格的理论论证,而且其普遍性也值得怀疑。因此,混沌优化的理论研究是很重要且迫切的研究课题,尤其是从理论上研究混沌的引入与控制对算法全局搜索能力的影响,其研究成果将有助于指导混沌优化算法的设计。同时,也应该注重对混沌更深入的了解和对混沌理论、分岔理论等本身的研究,这无疑对混沌优化的发展会产生推动作用。

(2) 混沌神经网络类型众多,有必要建立统一的模型和研究理论体系,这有助于开发新型的网络模型,并推动系统化的理论和方法的研究。最近,Kwok 等(1999)对内、外方法的 CNN 结构及其对优化过程的影响建立了一种统一的研究方法,并提出了 CNN 的统一框架。他们将能量函数 E 统一分解描述为 $E = E_{\text{Hop}} + H$,且满足条件 $dy/dt = -\partial E/\partial x$,其中 E_{Hop} 为传统 HNN 的能量函数, H 是依赖不同 CNN 而构造的附加能量项, x 和 y 为神经元的输入和输出。

(3) 注重算法本身环节的实用性改进工作和规律性结论的归纳。首先,我们可以围

绕混沌机制、模型和控制参数、控制策略、局部搜索策略(邻域结构)、收敛准则等环节,进行多样化设计与比较研究(两阶段方法中还应对粗搜索的满意度进行合理衡量,以及设计好两个阶段的转换时机与合理过度),进而归纳各环节和参数对搜索行为和算法性能的影响规律,再返回来指导相应的改进工作。

(4) 注重搜索结构的改进,以提高优化效率和性能。一方面,可以采用并行搜索结构,用多轨道遍历性搜索来提高优化性能;另一方面,可以采用混合优化结构,如将混沌优化策略与遗传算法、模拟退火、禁忌搜索等方法结合使用,通过吸收多种方法的优点来提高优化性能;此外,优化特定问题时,也应注重有效利用与问题相关的规则性搜索或信息来辅助搜索的进行,譬如将 2opt 方法与 CNN 相结合来优化 TSP 问题等。

(5) 注重混沌优化方法与其他类型的各种优化方法的比较研究,进而吸收其他方法的优点来改进混沌优化算法本身。

(6) 目前,混沌优化方法的应用还局限于较小规模的问题,或是一些典型的 Benchmark 问题。算法的设计与研究最终是面向实际应用的,因此不仅要注重实用性算法的研究,而且应注重拓宽算法的实际工程应用领域,尤其是探讨算法在复杂和大规模问题中的应用潜力。

5.5 一类基于混沌神经网络的优化策略

本节介绍一类基于退火策略的混沌网络模型,在此简称 ACNN 模型(王凌等, 2000)。其机制可归纳为:首先,在 Hopfield 网络中引入混沌机制,通过形成混沌动态,“随机地”进行粗搜索;再结合退火策略,通过控制混沌动态使其产生逆分岔并最终退出,一旦混沌动态消失便转入常规 Hopfield 模型梯度优化。

5.5.1 ACNN 模型的描述

通常, Hopfield 网络求解优化问题的连续模型(Hopfield et al, 1985)为

$$du_{xi}(t)/dt = -u_{xi}(t)/\tau + \beta' \left[\sum_y \sum_j w_{xi,yj} v_{yj}(t) + I_{xi} \right] \quad (5.5.1)$$

其基于 Euler 法的离散模型为

$$\begin{aligned} u_{xi}(k+1) &= (1 - \Delta t/\tau) u_{xi}(k) + \Delta t \beta' \left[\sum_y \sum_j w_{xi,yj} v_{yj}(k) + I_{xi} \right] \\ &= \alpha u_{xi}(k) + \beta \left[\sum_y \sum_j w_{xi,yj} v_{yj}(k) + I_{xi} \right] \end{aligned} \quad (5.5.2)$$

进而,将混沌机制引入 Hopfield 网络,同时结合退火策略控制混沌动态,构造如下的具有自组织特性和克服局部极小能力的基于退火策略的混沌网络模型:

$$\begin{cases} u_{xi}(k) = \alpha u_{xi}(k-1) + \beta \left[\sum_y \sum_j w_{xi,yj} v_{yj}(k-1) + I_{xi} - s_1 \right] \\ \quad - z(k-1)[v_{xi}(k-1) - s_2] & (5.5.3) \\ v_{xi}(k) = 1/[1 + \exp(-\mu u_{xi}(k))], \quad x, i, y, j = 1, 2, \dots, n & (5.5.4) \\ z(k) = z(k-1)/\ln[e + \lambda(1 - z(k-1))], \quad k = 1, 2, \dots, m & (5.5.5) \end{cases}$$

其中,式(5.5.3)为网络动态方程,式(5.5.4)为神经元激励函数,式(5.5.5)为退温函数, $u_{xi}(k)$ 和 $v_{xi}(k)$ 为 (x, i) 神经元在 k 时刻的输入和输出, $w_{xi,yj}$ 为连接权, τ 为时间常数, $z < 1$ 为控制参数, $\alpha (0 \leq \alpha \leq 1)$ 为阻尼因子, μ 为激励函数零输入时的斜率, I_{xi} 为外部偏置, s_1, s_2 为设定阈值或外加混沌量, $\beta (\beta > 0)$ 和 λ 分别为尺度和退温参数。可见,这类模型可以很简单地转换为上节介绍的“内模型”或“外模型”,或者是两者的结合。

5.5.2 ACNN 模型的优化机制

ACNN 模型综合了随机性和确定性算法的优点,优化过程分为基于混沌的“粗搜索”和基于 Hopfield 网络的“细搜索”两个阶段。尤其是,粗搜索中的混沌搜索同时具有随机性和轨道遍历性,具有克服局部极小的能力。随机性保证大范围搜索能力,轨道遍历性使算法能按系统自身的行为不重复地遍历所有可能状态,有利于克服一般随机算法中以分布遍历性为机制搜索带来的局限性。由此,算法在机制上保证了同时具有克服陷入局部极小的能力和良好的搜索效率。

以下,对两阶段优化机制给出具体的描述。

1. 粗搜索阶段

选定控制参数使系统一开始就进入大范围意义下的较强混沌动态,利用混沌搜索的随机性和轨道遍历性在大范围内按其自身规律进行遍历性搜索。搜索过程按混沌轨道进行遍历,不受目标函数限制,从而有很强避免陷入局部极小的能力。

伴随混沌搜索的进行,对控制参数“退温”,退温到一定程度并由系统的自组织能力,混沌动态将退出而进入逆分岔阶段。此后,搜索过程将限制在一些周期解,且能随控制参数的进一步减小而缩小搜索区域。

当逆分岔现象随控制参数的进一步退温而达到周期 1 形态时,算法得到一个易于得到全局意义下最优解的状态,粗搜索过程结束,转入 Hopfield 模型的梯度搜索。

2. 细搜索阶段

随着粗搜索结束,系统方程中控制参数决定项的作用减弱,优化过程基本按能量函数的梯度下降方向进行。寻优过程将由粗搜索得到的状态在小范围内按 Hopfield 模型的梯度下降机制进一步搜索,从而较快收敛到全局意义下较满意的解。

应当指出,合理选取退温函数对保证算法的优化效果具有重要影响。从退温函数可看出,用于控制参数 $z(k)$ 相对下降速率的 $z(k+1)/z(k)$ 的相对下降速率随 k 增大而逐渐

加快,也即 α 较大时“退温”较慢,而 α 较小时“退温”加快。如此设计的目的,首先是为了使混沌动态有足够长的进程,以提高混沌粗搜索性能;其次,利用 α 较小时退温速率的加快,以加快梯度搜索过程,从而改善时间性能。下节的仿真研究表明,这种退温策略能够产生较好的优化效果。

5.5.3 计算机仿真研究与分析

下面,以典型的 10 城市和 30 城市 TSP(Fogel,1993)为例,利用 16MB RAM 的 Pentium 586/133 计算机,以 Borland C++ 为仿真环境,研究 ACNN 模型的性能。

参照 Hopfield 求解 TSP 的思路,利用置换矩阵描述问题的解,置换矩阵的每一个元代表相应神经元的输出,只有当置换矩阵的每行每列均只含 1 个“1”且其余均为“0”时才代表一个合法解。

能量函数 E 设计如下:

$$E = \frac{A}{2} \left\{ \sum_{x=1}^n \left(\sum_{i=1}^n v_{xi} - 1 \right)^2 + \sum_{i=1}^n \left(\sum_{x=1}^n v_{xi} - 1 \right)^2 \right\} + \frac{B}{2} \sum_{x=1}^n \sum_{\substack{y=1 \\ y \neq x}}^n \sum_{i=1}^n (v_{y,i+1} + v_{y,i-1}) v_{xi} d_{xy} \quad (5.5.6)$$

1. 基于 10 城市 TSP 的 ACNN 与 Hopfield 模型的优化性能比较

(1) HNN 模型的研究结果

① Hopfield 等原始 HNN 模型(1985),20 次随机实验得到 16 次合法解,10 次满意解;

② Wilson 等重复 Hopfield 的实验(1988),100 次随机实验只有 15 次在 1000 步迭代内收敛到合法解,且仍为次优解;

③ 庄镇泉等的改进 HNN 模型(1992),100 次随机实验只得到 47 次合法解,12 次满意解。

(2) ACNN 模型的研究结果

模型参数取为: $A=B=1$; $s_1=1$, $s_2=0.65$; $\mu=250$;

神经元初始输入范围为 $(-1.0, 1.0)$;

能量函数变化值连续 10 次小于 0.01 作为收敛条件;

其他参数见表 5.5.1。

各参数下 ACNN 模型 50 次随机仿真的统计结果见表 5.5.1。

仿真结果表明:

(1) ACNN 模型全局优化性能有明显的提高,如第 1,5,9 组参数下每次实验均得到最优解;

表 5.5.1 10 城市 TSP 的 ACNN 模型优化统计性能

序号	β	α	λ	$z(0)$	最优解 数目	次优解 数目	非法解 数目	波动率%	平均迭代 步数
1	0.02	0.88	0.01	0.05	50	0	0	0	216
2	0.018	0.88	0.01	0.05	49	1	0	0.22	239
3	0.015	0.88	0.01	0.05	42	5	3	1.26	293
4	0.022	0.88	0.01	0.05	43	7	0	1.71	194
5	0.02	0.90	0.01	0.05	50	0	0	0	200
6	0.02	0.86	0.01	0.05	47	3	0	0.93	225
7	0.02	0.88	0.008	0.05	48	2	0	0.63	259
8	0.02	0.88	0.012	0.05	47	3	0	0.74	187
9	0.02	0.88	0.01	0.06	50	0	0	0	267
10	0.02	0.88	0.01	0.04	39	11	0	1.93	147

(2) 不适当参数下 ACNN 模型出现的次优解和非法解的几率远好于 Hopfield 模型, 如仅第 3 组参数下出现 3 次非法解(约占 6%);

(3) ACNN 模型的波动性能和时间性能较 Hopfield 模型有较大提高。

仿真结果中,也有少量的非法解出现,通过对仿真过程的深入研究,发现其主要原因:

- (1) 混沌动态过分或不充分;
- (2) 网络还未完全稳定,即收敛条件设置不合理。

2. 基于 30 城市 TSP 的 ACNN 模型与 SA 的性能比较

ACNN 模型参数选取为: $A=1, B=0.5, s_1, s_2, \mu$ 和初值范围同前, $\alpha=0.9, \beta=0.015, \lambda=0.007, z(0)=0.08$,能量函数变化值连续 10 次小于 10^{-5} 为收敛条件。

SA 算法参数选取为:初温 5,终温 0.01,退温速率 0.8,抽样步数 1000,以 SWAP 操作进行状态转移。

仿真前,首先将各坐标归一化到 $[0,1]$ 区间,结果输出时再折合回原始数据计算。各算法 50 次随机仿真的统计结果见下表。

表 5.5.2 基于 30 城市 TSP 的 ACNN 模型与 SA 的性能比较

算法	最优解次数	相对误差小于 3%的次优解次数	非法解次数	波动率%	平均迭代步数
ACNN	15	15	0	6.27	653
SA	3	10	0	10.47	28000

仿真结果表明,ACNN 模型的优化性能、时间性能和对初值的鲁棒性均优于 SA。由此可见,ACNN 模型的性能同时优于确定性 Hopfield 网络和有指导性随机搜索方法。这

正好反映了所给出的算法中“确定性和随机性搜索机制相混合”的优点。

为了显示 ACNN 模型的寻优过程和性能,在此给出了若干仿真曲线和图形,包括:

(1) ACNN 模型得到的 10 城市和 30 城市 TSP 的最优解,如图 5.5.1。

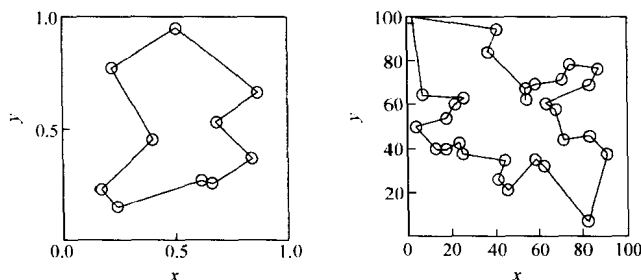


图 5.5.1 ACNN 模型得到的 10 城市和 30 城市 TSP 的最优解

(2) 优化过程中神经元输出动态和能量函数的变化过程,如图 5.5.2 和图 5.5.3。其中,神经元输出状态由混沌到逆分岔,再演变到确定性的梯度下降,这正是算法的搜索机制的体现;优化初始阶段能量函数值不是单调降的,其变化完全按混沌动态的演化而变化,不受能量障碍的影响,这正是算法避免局部极小的能力的原因。

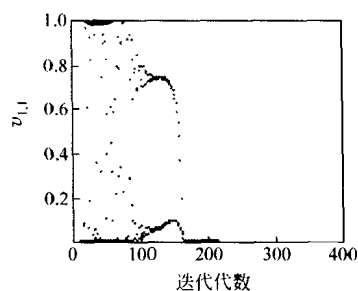


图 5.5.2 第 1 组参数下神经元输出 $v_{1,1}$ 随 z 下降的演变过程

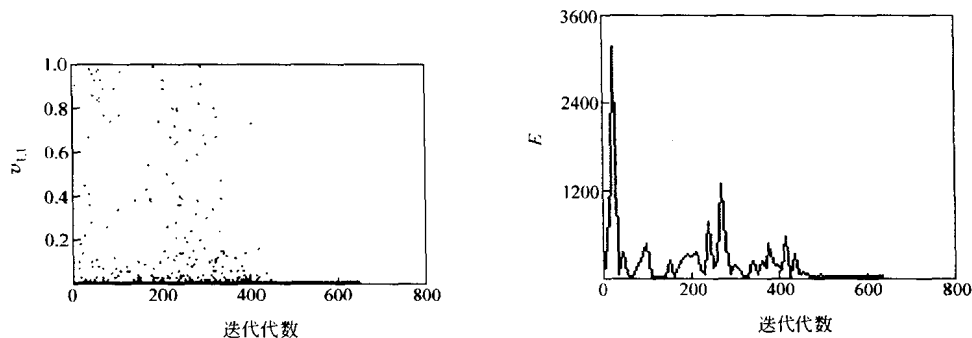


图 5.5.3 30 城市 TSP 求解时神经元输出 $v_{1,1}$ 和能量函数 E 随 z 下降的变化曲线

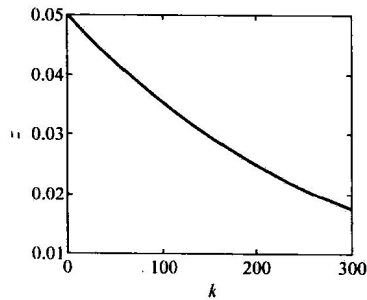


图 5.5.4 优化 10 城市 TSP 的退温方式, $z(0)=0.05$,
 $z(k)=z(k)/\ln[e+0.01\times(1-z(k-1))]$

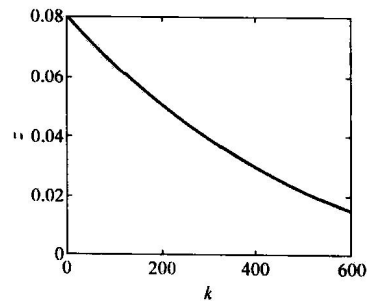


图 5.5.5 优化 30 城市 TSP 的退温方式, $z(0)=0.08$,
 $z(k)=z(k)/\ln[e+0.007\times(1-z(k-1))]$

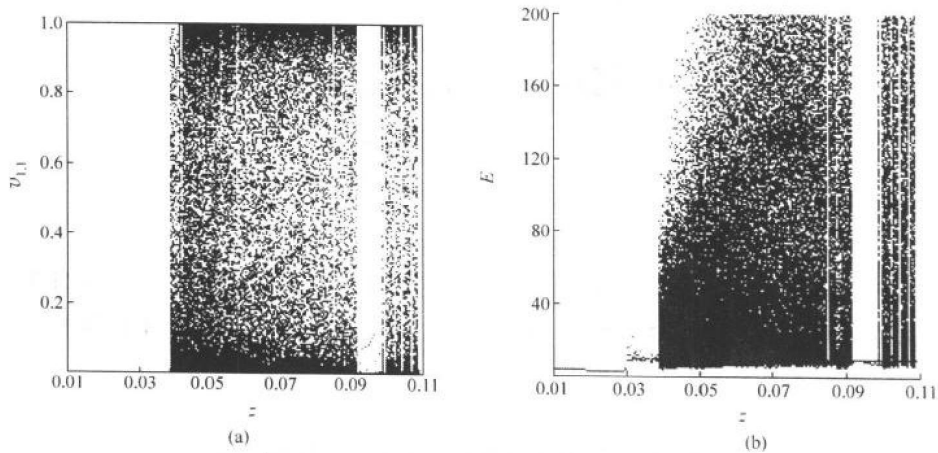


图 5.5.6 神经元输出 $v_{1,1}$ 和能量函数在 $z=0.11\sim 0.01$ 的分岔图(各 z 下初始输入均取 $(-1.0, 1.0)$ 间的同一组随机数, 绘制第 5000 步至 10000 步的结果, 城市数为 30, $A=B=1$, $s_1=1, s_2=0.65, \mu=250, \alpha=0.88, \beta=0.02, \lambda=0.01$)

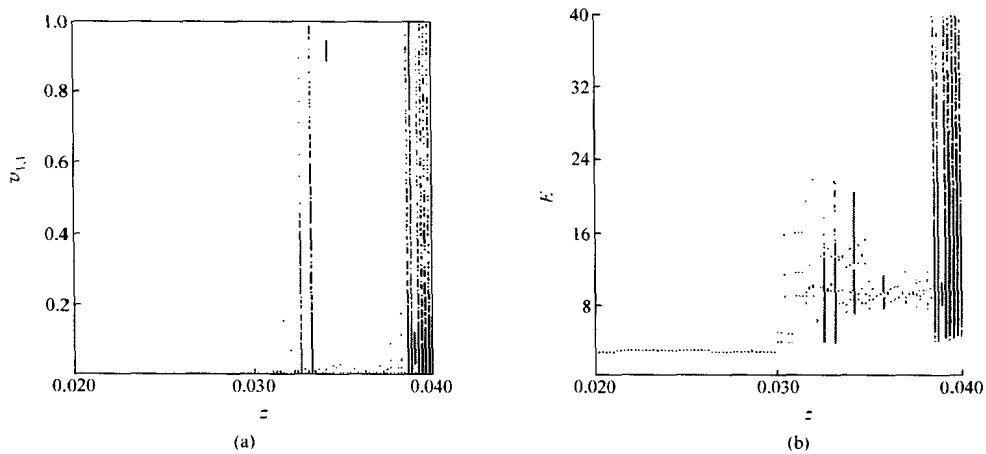


图 5.5.7 神经元输出 $v_{1,1}$ 和能量函数在 $z=0.04 \sim 0.02$ 的分岔图(参数同上)

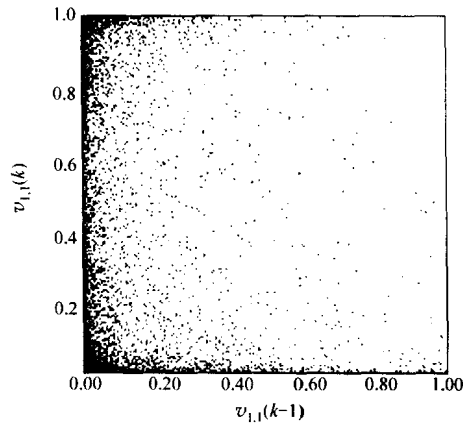


图 5.5.8 神经元输出 $v_{1,1}$ 在 $z=0.05$ 的相图(参数同前)

(3) ACNN 模型中温度的下降方式,如图 5.5.4 和图 5.5.5 所示。

(4) 神经元状态和能量函数随控制参数变化的分岔图(如图 5.5.6 和图 5.5.7 所示),以及特定参数下神经元状态的相图(图 5.5.8)。从这些图形,也可看到控制参数对系统动态特性的影响。

5.5.4 模型参数对算法性能影响的几点结论

通过对仿真结果和 ACNN 模型的分析 and 归纳,可导出如下的模型参数对算法性能影响的有关结论。

1. 尺度参数 β 的影响

β 增大有利于加速收敛,但优化质量会有所下降。从网络动态方程知, β 增大使能量函数对动态方程的影响增大。若影响过大,将不能产生充分的混沌动态,从而影响搜索性能,如次优解增多;反之,太小的 β 使能量函数的变化不能充分影响动态的演变,从而搜索过程难以收敛到对应最小能量值的最优解,同时过分的混沌搜索将导致优化过程变长。

2. 阻尼因子 α 的影响

α 增大有利于加速收敛,并减少算法收敛到次优解的概率。通过对式(5.5.2)的分析知, α 减小相当于增大仿真步长 Δt ,若算法中 β 未变,则 Δt 增大相当于 β' 减小,从而使能量函数的作用变弱,进而导致迭代步数增大和优化性能变差。同理, α 过大虽能加速收敛,但会出现类似 β 过大的现象。

3. 退温参数 λ 的影响

λ 增大有利于加速收敛,但优化质量会下降。 λ 的大小反映控制参数 z 下降速率的快慢,过大的 λ 会使混沌动态消失过快,从而 ACNN 模型过早转化为 Hopfield 模型,进而收敛到局部极小或非法状态;反之,混沌动态持续太久,将严重影响收敛速度。

4. 初始控制参数 $z(0)$ 的影响

$z(0)$ 减小有利于加快收敛,但算法收敛到次优解的概率将增加。由动态方程和退温函数知, $z(0)$ 减小使 $z(k+1)/z(k)$ 减小,即加快了 z 的下降过程,但一旦混沌行为不充分,则必然影响优化性能;反之, $z(0)$ 过大将使其下降过程过慢,从而混沌对系统动态演化过程的影响过大,收敛必然缓慢。

此外,若问题的规模增大,神经元数量和系统动态的复杂性均将增加。仿真表明,此时参数 B 应选择得比参数 A 小些,以均衡能量函数中各项的作用;同时, $z(0)$ 需适当增加, β 和 λ 需适当减小。显然,上述结论为参数的选择提供了指导,而选取范围的定量探讨仍是有待于进一步研究的课题。

第 6 章

广义邻域搜索算法及其统一结构

本章首先提出广义邻域搜索算法的概念,进而提出广义邻域搜索算法的统一结构,为开发新型单一和混合型的优化算法提供一条指导途径。同时,本章介绍了评价算法性能的多项指标和综合性能指标,并对广义邻域搜索算法的研究进展进行了概述。

6.1 广义邻域搜索算法

优化算法的研究与开发是优化技术研究的关键内容。传统的邻域搜索算法,即利用邻域结构进行逐步优化的局部搜索算法,其优化流程可描述如下:

算法从一个初始解 x 出发,然后利用状态发生器持续地在解 x 的邻域中搜索比它好的解。若能够找到如此的解,就以之替代解 x 成为新的当前解,然后重复上述过程;否则结束搜索过程,以当前解作为最终解。

纵观上述算法流程,邻域搜索算法具有以下特点:

- (1) 算法通用易实现。只要设计好状态发生器(邻域函数),就能求解组合或函数优化问题。
- (2) 算法性能对邻域函数和初值具有依赖性。邻域函数设计不同或初值选取不同,则算法最终的性能将会有差异。
- (3) 算法的局部优化特性。由于采用贪婪策略,若当前解为其邻域中的最佳状态,则算法必然因其而终止,或者说算法没有避免搜索过程陷入局部极小的能力。

邻域搜索算法的改进途径可以是：

- (1) 采用并行搜索结构；
- (2) 设计复杂的邻域函数；
- (3) 替换贪婪策略等。

可以说,相对于梯度下降法、趋化性搜索算法而言,模拟退火算法、遗传算法、禁忌搜索、进化规划、进化策略、混沌搜索、模糊优化、基于规则的启发式算法等,都是对传统邻域搜索算法的改进和变型。同时,近年来出于提高优化性能的需要,在诸多研究领域又开发了许多混合优化算法。为了区别于传统的邻域搜索算法,我们将上述通过不同途径而构成的改进算法统称为广义邻域搜索算法。尽管各个算法在优化过程的细节上存在差异,但在优化流程上呈现出很大的共性。在此,我们将广义邻域搜索算法的优化流程描述为:算法从若干初始解出发,在算法参数控制下由当前状态的邻域中产生出若干个候选解,并以某种策略在当前解和候选解中确定新的当前状态。伴随控制参数的调节,重复执行上述搜索过程,直至满足算法终止准则,结束搜索过程并输出优化结果。显然,模拟退火、遗传算法、禁忌搜索、混沌搜索等方法都是上述流程的具体实现。

6.2 广义邻域搜索算法的要素

可以说,广义邻域搜索算法优化流程的统一描述为构造优化算法提供了一种框架。在具体设计算法时,需要考虑以下几方面关键问题,并遵循一定的指导性结论。

1. 搜索机制的选择

搜索机制是构造算法框架和实现优化的关键,是决定算法搜索行为的根本点。基于局部优化的贪婪机制可用于构造局部优化算法,如梯度下降法、爬山法等;基于概率分布的优化机制可用于设计概率意义下的全局搜索算法,如 GA,SA 等;基于系统动态演化的优化机制可用于设计具有遍历性和自学习能力的优化算法,如混沌搜索、神经网络算法等。

2. 搜索方式的选择

搜索方式决定着优化的结构,即每代有多少解参与优化。并行搜索方式以多点同时或交叉优化,来取得较好的优化性能,但计算和存储量较大,如 GA,EP 和神经网络等;串行搜索方式可视为并行方式的一个特例,优化进程中始终只有一个当前状态,处理较为简单,但优化效率一般较差,如 SA,TS 等。

3. 邻域函数的设计

邻域函数决定了邻域结构和邻域解的产生方式。算法对问题解的不同描述方式,使解空间的优化曲面形状和解的分布有所差异,会直接影响邻域函数的设计,进而影响算法的搜索行为。同时,即使在编码机制确定的情况下,邻域结构也可采用不同的形式,以考

虑新状态产生的可行性、合法性和对搜索效率的影响,如基于路径编码的 TSP 优化中可利用互换、逆序和插入等多种邻域结构。在确定邻域结构后,当前状态邻域中候选解的产生方式既可以是确定性的,也可以是随机性的,甚至是混沌性的。

4. 状态更新方式的设计

更新方式是指以何种策略在新旧状态中确定新的当前状态,是决定算法整体优化特性的关键步骤之一。基于确定性的状态更新方式的搜索,一般难以穿越大的能量障碍,容易陷入局部极小;而随机性的状态更新方式,尤其是概率性劣向转移,往往能够取得较好的全局优化性能。

5. 控制参数的修改准则和方式的设计

控制参数是决定算法的搜索进程和行为的又一关键因素。合适的控制参数应有助于增强算法在邻域结构中的优化能力和效率,同时也必须以一定的准则和方式进行修改以适应算法性能的动态变化。一般而言,在当前控制参数难以使算法性能取得较大提高时,就应考虑修改参数;同时,参数的修改幅度必须使算法性能的动态变化具有一定的平滑性,以实行算法行为在不同参数下的良好过渡。算法收敛理论为参数设计提供了指导,而实际设计时也可根据优化过程的动态性能按规则自适应调节参数。

6. 算法终止准则的设计

终止准则是判断算法是否收敛的标准,决定了算法的最终优化性能。算法收敛理论为终止准则提供了明确的设计方案,但是基于理论分析所得的收敛准则往往是很苛刻的,甚至难以应用。实际设计时,应兼顾算法的优化质量和搜索效率等多方面性能,或根据问题需要着重强调算法的某方面性能,采用与算法性能指标相关的近似收敛准则,如给定最大迭代步数、最优解的最大凝滞步数和最小偏差阈值等。

上述分析表明,通过对上述各关键步骤的多样化设计,可以构造多种单一性广义邻域搜索算法。

6.3 广义邻域搜索算法的统一结构

广义邻域搜索算法是对一类优化算法的统称,符合其优化流程的算法很多。由于各种算法的搜索机制、特点和适用域存在一定的差异,实际应用时为选取适合问题的具有全面优良性能的算法,往往依赖于足够的经验和大量的实验结论。造成这种现象的根本原因是,优化算法的研究缺乏系统化。特别是,目前不同算法各自孤立的研究现状,不利于开发新型混合机制的优化算法,也不利于算法应用领域的拓宽。因此,建立统一的算法结构和研究体系,就成为一件很有必要的事。

同时,我们应该认识到,如何改进算法结构和搜索机制以提高算法的优化性能,是目前广义邻域搜索算法研究的主流。在算法设计时总是希望优化算法具有各种优良的性

能,对复杂问题尤其要求算法具有避免陷入局部极小的全局优化能力。但是也应该看到,依赖于单一邻域结构的搜索算法一般难以实现高效的优化,这也正是基于混合邻域结构的搜索算法日益受到重视的原因。鉴于算法混合对提高优化精度与效率的潜力,近年来学术界提出了类型和名称繁多的混合算法,混合优化算法也成为一些学术会议专门的研讨方向。“No Free Lunch”定理说明了没有一种方法对任何问题都是最有效的,即各方法均有其相应的适用域,而算法的混合正是拓宽其适用域和提高性能的有效手段。L. Davis 还指出“hybridize where possible”。由于算法结构缺乏统一性,现今对混合算法的构造还缺乏指导和系统化,这阻碍了算法的发展和应用,也是我们探讨广义邻域搜索算法统一结构的一个根本性出发点。

基于并行和分布式计算机技术的发展,为了使优化算法适合于求解大规模复杂优化问题,可以对优化过程作两方面分解处理。

(1) 基于优化空间的分层:把原优化问题逐层分解成若干子问题,利用有效算法首先对各子问题进行并行化求解,最后逆向逐层综合成原问题的解。

(2) 基于优化进程的分层:把优化过程在进程层次上分成若干个阶段,各阶段分别采用不同的搜索算法或邻域函数进行优化。

针对上述思想,目前混合算法的结构类型主要可归纳为串行、镶嵌、并行及混合结构。

串行结构是一种最简单的结构,如图 6.3.1 所示。串行结构的混合算法就是吸收不同算法的优点,用一种算法的搜索结果作为另一种算法的起点依次来对问题进行优化,其目的主要是在保证在一定优化质量的前提下提高优化效率。设计串行结构的混合算法需要解决的问题主要是确定算法的转换时机。

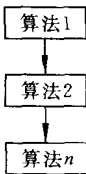


图 6.3.1 混合算法的串行结构

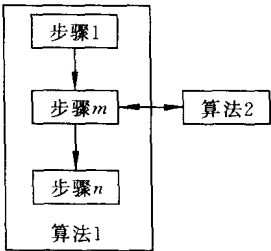


图 6.3.2 混合算法的镶嵌结构

混合算法的镶嵌结构如图 6.3.2 所示,它表现为一种算法作为另一种算法的一个优化操作或用作搜索性能的评价器。前者混合的思想主要是鉴于各种算法优化机制的差异,尤其是互补性,进而克服单一算法早熟和陷入局部极小。设计镶嵌结构的混合算法需要解决的问题主要是子算法与嵌入点的选择。

混合算法的并行结构如图 6.3.3 所示,它包括同步式并行、异步式并行和网络结构。前两方式中有一个算法作为主过程(算法 A),其他算法作为子过程,子过程间一般不发生

通信。同步方式中主过程与子算法是一种主仆关系,各子算法的搜索过程相对独立,而且可以采纳不同的搜索机制,但与主过程的通信必须保持同步。异步方式中各子算法通过共享存储器彼此无关地进行优化,与主过程的通信不受其他子算法的限制,其可靠性有所提高。网络方式中各算法分别在独立的存储器上执行独立的搜索,算法间的通信是通过网络相互传递的,由于网络式结构纯粹是一种并行实现方式,我们不将其纳入混合算法框架。问题分解与综合以及

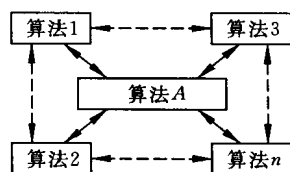


图 6.3.3 混合算法的并行结构

进程间的通信问题是设计并行结构的混合算法需解决的主要问题。

基于以上阐述与分析,我们提出如图 6.3.4 所示的广义邻域搜索算法及其组合型优化算法的统一优化结构。这种结构的统一性主要体现在以下几方面:

1. 单一邻域搜索流程

在统一结构中,将各种单一广义邻域搜索算法进行统一模块化描述,包含构成广义邻域搜索流程的所有关键步骤。

2. 进程层次串行组合邻域搜索,即 SNSA 流程

SNSA 流程,体现了优化过程在进程层次上的分解,是在进程层次上对各种混合算法的统一描述。通过适当的接口处理,利用多种子算法,可构造出多种混合算法。

3. 问题分解和预处理以及子问题的综合过程

它们体现了优化过程在空间层次上的分解,是基于“divide and conquer”思想的算法的统一描述。通过问题分解,可降低求解复杂性,有利于提高优化效率。

4. 整体解的进一步 NS 优化

这是对“原问题经分解求解”到“综合的处理”的手段所造成的全局优化质量一定程度降低的一个补充,同时也用于在统一结构中融进基于问题信息的构造性启发式搜索算法。

譬如大规模 TSP 的求解,鉴于问题整体求解的复杂性,在设计算法时可以先考虑空间的分解,利用聚类的方法将问题分解为若干子问题,然后先用启发式方法快速得到子问题的近似解,而后以其为初始状态利用 GA, SA, TS 等方法和规则性搜索在一定的混合方式下进行指导性优化,待各子问题求解完毕用邻近原则确定问题的整体解,再采用局部改进算法对其作进一步加工以得到问题的最终解。

可以看出,通过综合广义邻域搜索算法及其混合算法的共性,来导出和建立统一结构,在结构上为算法的系统化研究和分析提供了一种有效的途径,为开发新型高效的混合算法提供了指导。下一章将介绍如何依据上述统一结构来构造混合优化策略,并着重介绍由遗传算法和模拟退火构成的 GASA 混合优化策略。

$$E_{m, \text{off-line}} = \frac{c_b - c^*}{c^*} \times 100\% \quad (6.4.1)$$

其中, c_b 为算法多次运行所得的最佳优化值, c^* 为问题的最优值。当最优值为未知时, 可用已知最佳优化值来代替。该指标用以衡量算法对问题的最佳优化度, 其值越小意味着算法的优化性能越好。

定义算法的在线最优性能指标如下:

$$E_{m, \text{on-line}} = \frac{c_b(k) - c^*}{c^*} \times 100\% \quad (6.4.2)$$

其中, $c_b(k)$ 为算法运行第 k 代时的最佳优化值。该指标用以衡量算法的动态最佳优化度, 其值越小意味着算法的优化性能越好。

2. 时间性能指标

定义算法的时间性能指标如下:

$$E_s = \frac{I_a T_0}{I_{\max}} \times 100\% \quad (6.4.3)$$

其中, I_a 为算法多次运行所得的满足终止条件时的迭代步数平均值, $I_{\max}$ 为给定的最大迭代步数阈值, T_0 为算法一步迭代的平均计算时间。搜索率用以衡量算法对问题解的搜索快慢程度即效率, 在 $I_{\max}$ 固定情况下 E_s 值越小说明算法收敛速度越快。

3. 鲁棒性指标

通常, “波动率 E_f ” 被用作鲁棒性指标。我们定义离线初值鲁棒性指标如下:

$$E_{f1, \text{off-line}} = \frac{c_a - c^*}{c^*} \times 100\% \quad \text{或} \quad E_{f2, \text{off-line}} = \text{STDEV}(c_i^*) \quad (6.4.4)$$

其中, c_a 为算法多次运行所得的平均值, c_i^* 为算法第 i 次运行得到的最优值, $\text{STDEV}()$ 为均方差。波动率 E_{f1} 用以衡量算法在随机初值下对最优解的逼近程度, E_{f2} 用以衡量算法性能对随机初值和操作的依赖程度, 两者值越小说明算法的鲁棒性(或可靠性)越高。

定义在线波动性指标如下:

$$E_{f1, \text{on-line}} = \frac{c_a(k) - c^*}{c^*} \times 100\% \quad \text{或} \quad E_{f2, \text{on-line}} = \text{STDEV}(c_i^*(k)) \quad (6.4.5)$$

其中, $c_a(k)$ 为算法运行第 k 代所得的平均值, $c_i^*(k)$ 为算法第 i 次运行在第 k 代得到的最优值。在线指标用以衡量算法对随机初值和操作的动态依赖程度。

基于上述三个性能指标, 优化算法的综合性能指标 E 即为它们的加权组合:

$$E = \alpha_m E_m + \alpha_s E_s + \alpha_f E_f \quad (6.4.6)$$

其中, α_m , α_s 和 α_f 分别为优化性能指标、时间性能指标和鲁棒性指标的加权系数, 且满足 $\alpha_m + \alpha_s + \alpha_f = 1$ 。

综合性能指标值越小表明算法的综合性能越好, 以此作为实际应用时选择算法的一个标准。这也不难理解, 因为工程中对算法性能的要求往往因问题而异, 例如离线优化追求较高的优化性能指标, 在线优化追求较高的时间性能指标和鲁棒性指标。因此, 在不同

场合,除评价算法的各项单一指标外,可通过适当调整各加权系数来反映问题对算法的要求,并计算算法的综合性能,为算法的选取和性能比较提供合理的依据。

6.5 广义邻域搜索算法研究进展

SA,GA 和 TS 作为具有全局优化性能的典型 Meta-heuristic 算法代表,与人工神经网络统称为四大现代启发式算法,前文我们将其统一称为广义邻域搜索算法,具有区别于常规算法的搜索机制和特点。SA 基于物理学中固体物质的退火过程与组合优化的相似性,是基于 Monte Carlo 迭代求解的一种全局概率型搜索算法,首先由 Kirkpatrick 等(1983)用于组合优化。GA 是 Holland(1975)研究自然遗传现象与人工系统的自适应行为时,借鉴“优胜劣汰”的生物进化与遗传思想而提出的一种全局性并行搜索算法,GA 注重父代与子代遗传细节上的联系(强调染色体操作),类似的 EP 和 ES 则侧重父代与子代表现行为上的联系(强调物种层的行为变化)。TS 是 Glover(1989)模拟智力过程而提出的一种具有记忆功能的全局逐步优化算法。其中,SA 是一种串行优化算法,每步仅随机尝试当前状态邻域中的一个状态,同时通过控制“温度”来控制状态更新概率,从而在搜索过程中具有避免陷入局部极小的能力并最终趋于全局最优。GA 是一种并行优化算法,尤其具有很大的隐性并行性,在码空间以一定概率通过对种群的不断选择、交叉和变异操作达到群体并行进化目的。TS 是一种串行优化算法,通过记忆近期操作的存储结构来避免状态的重复出现,并结合藐视准则实现全局快速搜索。初始温度、退温函数、状态产生方式、抽样稳定准则是影响 SA 性能的关键因素;种群数目、复制、交叉和变异操作和操作概率是影响 GA 性能的关键因素;禁忌表大小、邻域函数结构与数量是影响 TS 性能的关键因素。至于算法的收敛准则,通常采用结合收敛性条件且注重搜索效率的启发式判据,如给定最大迭代步数、最优解连续不变的最大次数、搜索序列对应的平均值与最优值的最小偏差等。下面就此类算法的理论与应用研究作简要综述,并指出若干发展性研究内容。

6.5.1 理论研究概述

广义邻域搜索算法的理论研究主要包括算法的收敛性、收敛速度估计,以及为改善算法性能而进行的控制参数与搜索算子的选择、算法结构研究、算法比较性研究等。

1. 收敛性与鲁棒性研究

算法的全局收敛性研究是算法理论研究的主要方面。尽管模式定理(Holland, 1975)抓住了 GA 机制的本质,定性说明了适配值高、长度短且阶数低的模式数量随时间至少以指数增长,但它不能严格说明算法的全局收敛性。基于 SA,GA 和概率型 TS 的马氏链描述,马氏链已成为分析收敛性的有力工具。业已证明在退温函数满足一定条件时

SA 算法以概率 1 收敛到全局最优。Goldberg 等(1987)首先用马氏链分析 GA 的收敛性;Eiben 等(1991)用马氏链证明了保优 GA 全局收敛;Rudolph(1994)用齐次马氏链证明了标准 GA 不具全局收敛性而保优 GA 全局收敛;Qi 等(1994)用马氏链分析了种群规模无穷时浮点数编码的 GA 的收敛性。Faigle 等(1992)得到了概率型 TS 的一些收敛结果。王凌等(1998)证明了将 SA 溶于 GA 的混合算法在退温函数的控制下全局收敛;张讲社等(1997)证明了父代参加竞争是 SA 和 GA 组合算法收敛的充要条件,数值仿真表明了混合策略较强的避免陷入局部极小能力和快速收敛性。

算法收敛速度估计的研究能从理论上对算法提供评判标准和改进方向,Mitra 等(1986)研究了 SA 的收敛速度,Buck 等(1992)研究了 GA 达到全局最优解的时间复杂性,彭宏等(1997)得到了保优 GA 收敛速度的一个粗略估计。算法鲁棒性研究主要是探讨算法性能受环境或算法参数的影响,目前大都以仿真或应用研究进行探讨。Keane(1995)对多峰问题研究了 GA 的收敛性和鲁棒性,王凌等(1997)以 Flow-shop 为例研究了 SA 的初值鲁棒性。然而,目前对算法严格的理论研究均基于简单的理想模型,研究成果与实际算法实施仍有距离,对复杂或实际算法的理论研究,尤其是收敛速度估计和鲁棒性的研究仍需大力探讨。

2. 算法参数的选择研究

算法参数是影响算法性能和效率的关键,如何确定最优参数使算法性能最佳本身就是一个极其复杂的优化问题。由于参数空间的庞大和各参数间的相关性,尚无确定最优参数的一般方法,目前在求解实际问题时主要凭经验选取算法参数。

SA 的收敛性要求初温应足够高以使得解空间各状态能以几乎相同的概率出现,每一温度下的抽样次数应足够多或与状态接受函数相关的退温速度应足够慢以使得马氏链能达到平稳分布,同时终止温度应足够小以保证算法最终收敛到全局极小。王凌等(1997)以 Flow-shop 为例研究了 SA 初温、退温速率、邻域结构和初始状态对算法性能的影响;Park 等(1998)提出了利用非线性规划确定 SA 最优参数的系统化流程,并以图分区、调度等问题,通过与凭实验经验选取参数的 SA 算法的比较研究验证了有效性。

对于遗传算法,种群数影响算法的有效性,太小了则不能提供足够的采样点导致较差的优化性能,反之则会增加计算量以致搜索时间过长;交叉概率控制交叉操作的频率,太大了会较快破坏高适配值的个体,反之则使搜索停止不前;变异用以加大种群多样性,其概率太小则难以产生新个体,太大则导致随机搜索。Grefenstette(1986)提出了用 GA 来优化 GA 参数的数值方法,其适用范围较广,但工作量大且难以作严格的理论分析;Davis(1991)提出了交叉和变异概率随遗传操作的在线性能而自适应取值的有效方法。此外,对二进制编码的 GA,Goldberg(1989)提出了串长可变的算法,取得了较好的效果。

影响 TS 性能的参数主要是禁忌表大小、邻域结构和数量、藐视准则和禁忌准则的设置。Glove(1989)对 TS 的结构、参数等各方面进行了详细描述和讨论;Lee 等(1996)研究了 SA,GA 和 TS 在资源受限的项目调度问题(RCPSP)中的应用,同时探讨了各算法参

数对优化性能的影响;Sexton 等(1998)针对神经网的训练问题提出了一种禁忌表大小可变的改进 TS 算法,对不同大小的禁忌表长度、邻域数量和搜索次数作了研究。

3. 算法操作的研究

算法操作是算法实施优化进程的关键步骤,设计优良的操作对改善算法性能和提高算法效率均有巨大作用。从搜索结构上考虑,邻域函数结构(状态产生方式)、状态更新方式和收敛或稳定判据最重要,关键参数的控制可归入参数选择研究内容。邻域大小和结构的选择应从计算效率考虑,保证候选解一定程度上遍布全解空间。状态更新方式的选择在使算法具有进化功能的同时,也应具有避免陷入局部极小的能力。由于 GA 的并行性以及遗传操作的复杂与多样性,这方面以 GA 的研究为多。由于基于马氏链的收敛条件在求解实际问题时难以实行,同时缺乏常规数学方法严格的收敛判据,广义邻域搜索算法通常在结合收敛性理论条件的同时采用注重搜索效率的启发式收敛判据。

SA 的接受函数采用概率方式保证了算法的突跳性,尽管邻域函数结构具有多样性,但算法每步仅随机尝试邻域中的一次采样。TS 通常采用贪婪思想与禁忌准则来接受邻域中的最优解,其局部搜索能力显然优于 SA,因此邻域的大小对 TS 性能影响较大。对组合优化问题,邻域函数通常可采用交换、插入、倒位等方式;对函数优化问题,邻域函数则可借鉴 GA 的变异操作。王凌等(1997)探讨了几种 SA 邻域函数在调度中的影响;Glover 等(1998)提出了自适应记忆的 TS 算法;Sexton 等(1998)研究了改变禁忌表结构的影响。对 GA 而言,新状态或种群的产生是通过并行化遗传操作实现的。复制操作用于避免基因的损失来提高全局收敛性和计算效率;交叉操作是 GA 区别于其他进化算法的重要特征,用于组合产生新个体来实施解空间的有效搜索并降低对有效模式的破坏概率;变异操作用于增加种群的多样性。在复制操作方面,DeJong(1975)和 Brindle(1981)分别提出了轮盘赌选择,因其选择误差较大的缺点又提出了无回放式随机采样操作;Brindle(1981)同时又提出了一种选择误差更小、操作简单的确定式采样和应用较广的无回放式余数随机采样。常用的交叉算子还有单点和多点交叉操作、部分匹配交叉操作、增强边缘重组算子、序号交叉算子和均匀排序交叉算子、循环交叉、双点交叉、置换交叉、算术交叉以及启发式交叉等(Goldberg, 1989)。组合优化中变异操作与 SA 状态产生函数研究较类似,在符号和实数编码中还有常规突变、均匀和非均匀突变等。此外,延长曾经适配值高而目前较差的基因块寿命的双倍体与显性遗传,保留最优个体的优先策略,用部分优质子串更新部分父串来保留优质父串的静态繁殖,分离与异位等高级操作也得到了研究。随着人们对算法机制与功能和自然更深入的理解与研究,模拟自然科学思想来实施优化的算法操作研究会得到进一步的发展。

4. 算法结构与混合算法的研究

为提高优化性能和效率,对不同搜索策略的特点进行取长补短,近年来发展了一些新颖的搜索机制和并行、混合搜索算法。刘岩等(1996)鉴于 SA 搜索序列不单调以及低温时难以在局部极小点突跳的缺点,提出了带有单调升温过程的 SA 算法,提高了算法的持

续搜索能力;Krishnakumar(1989)针对 GA 中种群数目大而造成适配值计算费时的瓶颈问题提出了称为 μ GA 的小群体方法,显示了较高的计算效率和动态系统优化潜力;Pothen 等(1994)为克服 GA 的早熟现象提出了基于迁移和人工选择的 GA(GAMAS);Zhou 等(1997)提出了 SA 与混沌相结合的搜索策略,提高了全局优化度和鲁棒性;Mahfoud 等(1995)和 Huntley 等(1996)分别提出了并行组合 SA 算法和结合局部搜索的并行 GA 算法;Bosenink 等提出了 Boltzmann-Darwin 优化策略;王凌等(1998)和张讲社等(1997)分别提出了 GASA 混合策略克服 SA 收敛缓慢和 GA 易早熟的缺点。此外,还有 GA 或 SA 与模糊集结合的混合算法(Wong et al, 1996),GA 与爬山法、梯度法等局部搜索算法的结合(Goldberg, 1989),TS 与基于线性规划的下降法的结合(Blue et al, 1998),SA 和 TS 的结合(Gil et al, 1998)。研究表明,新型的算法结构或混合算法对算法性能和效率有较大幅度的改善。此外,结合实际应用或理论问题对算法进行对比研究也是算法研究中值得关注的内容。它有助于分析算法的性能和适用域,且由比较可发现各算法独特的优点和不足,来改进结构或操作或参数,发展各种可能的高效混合算法。

6.5.2 应用研究概述

由于广义邻域搜索算法有很强的通用性,且无需问题的特殊信息,其应用领域非常广泛。目前主要的应用领域包括:

(1) 函数优化与组合优化:如 Yong 等(1995)提出一种函数优化的退火进化算法;Brindle(1981)研究了 GA 在函数优化中的理论与应用;Blue 等(1998)利用 TS 与基于线性规划的下降法结合搜索连续可微函数的极值点;Goldberg(1989)利用基于二进制编码的 GA 来优化函数并提出若干 Benchmark 问题;有约束 SAP 的 TS 求解(Nonobe et al, 1998)和 TSP 的求解(王凌等, 1998)等。

(2) 电路设计:如 VLSI 设计中元件布置问题的基于并行机的并行组合 SA 算法(Kurbel et al, 1998);电路分区问题的求解(Gil et al, 1998);电路的速度和容量优化(Sait et al, 1998)等。

(3) 生产调度:如 Flow-shop 和 Job-shop 问题的求解(Croce et al, 1995; Wang et al, 1999);操作时间可变且设置时间与排序相关的 JIT 调度问题研究(Kolahan et al, 1998);多目标的装配线平衡问题的求解(Kim et al, 1996);设置时间与排序相关的 E/T 调度问题研究(Wang et al, 1997);生产线中缓冲区设置和大小的确定问题(Lutz et al, 1998)等。

(4) 神经网络设计:结构设计和高效的学习算法是神经网络当今面临的主要问题,广义邻域搜索算法已成为解决这些问题的有力工具,尤其是遗传算法(Yao, 1993; 潘卫东, 1994)。王凌等(1998)提出了前向网络的两种混合训练算法;Angeline 等(1994)提出了回归网的构造进化算法;Sexton 等(1998)利用 TS 优化网络权值。

(5) 控制工程:如 PID 调节问题的研究(Potter et al, 1992);ARMAX 模型的辨识和预报研究(Yang et al, 1996)等。

(6) 机器学习和模式识别:如机器学习视觉识别问题的研究(Englander 1985);冗余机器人的逆运动问题的求解(Parker et al, 1989);模糊聚类问题的研究(Al-Sultan et al, 1997)等。

(7) 通信:如电信网络的布线问题(Davis, 1991);宽带通信网络的拓扑优化问题研究(Costamagna et al, 1998);通信网络呼叫允许问题研究(Rose et al, 1996)等。

此外, meta-heuristic 算法的研究还渗透到光学、电力、计算机科学、航空航天、交通、经济和管理等领域,在此不一一列举。

6.5.3 发展性研究

广义邻域搜索算法的研究正处于不断发展中,无论是理论方面,还是应用方面。然而,已有的研究成果相对分散,鉴于各种算法结构与研究的相似性,在注重个别问题的同时,应强调算法体系的研究,促使算法系统化的发展(王凌等, 2000)。分析目前的研究现状和发展方向,归纳起来,广义邻域搜索算法的发展性研究主要包括以下几方面:

(1) 归纳总结已有的分散成果,建立邻域算法的统一结构体系和理论研究体系。

(2) 寻求新的数学工具和分析方法,建立算法复杂性、收敛性和鲁棒性的分析研究理论,对算法收敛速度和优化度进行估计。

(3) 基于数学和计算机进行研究,尤其是算法的比较研究,提供控制参数选取的理论指导和规律性结论,提出搜索进入局部极小的合理判断准则。

(4) 针对搜索全局性、快速性和鲁棒性,发展新的优化机制和优化操作,尤其是基于算法统一框架发展高效的混合优化算法,包括:不同搜索机制的结合(如 SA 的概率突跳性, GA 的并行搜索能力, TS 较强的局部搜索能力及其记忆功能, CS 的遍历性搜索等),全局性与局部性搜索算法的结合,通用算法与问题特殊信息的结合等。

(5) 利用并行计算机技术,发展高效的并行或分布式优化算法。

(6) 分析并处理优化问题中的关键因素,降低问题的难度,譬如将 NP 难题在一定条件下转化为非 NP 难题,对大规模问题进行分解处理降低复杂度以提高优化效率。

(7) 拓宽算法的应用领域,支持与发展算法理论和算法本身,总结算法的适用域。

(8) 研制算法软件和硬件,注重开放性设计以便算法不断扩充与综合。

可以预言,随着算法理论研究的发展和应用领域的拓宽,广义邻域搜索算法具有广阔的研究与应用前景,而混合优化策略的研究将是其中的主要内容。

第 7 章

混合优化策略

近年来,混合优化策略得到了较广泛的应用,并取得了理想的效果,其设计与分析已成为算法研究的一个热点。基于前文介绍的广义邻域算法的统一结构,在设计单一算法的基础上,本章将阐述如何设计有效的混合优化策略,并着重介绍由遗传算法和模拟退火融合成的层次化 GASA 混合策略及其特点,同时利用非平稳马氏链理论对混合策略的收敛性进行分析,并对其全局搜索行为、搜索效率和初值鲁棒性进行定性分析。

7.1 引言

随着科技的发展和工程问题范围的拓宽,问题的规模和复杂度越来越大,传统算法的优化结果往往不够理想,同时算法理论研究的落后也导致了单一算法性能改进程度的局限性,而基于自然机理来提出新的优化思想是一件很困难的事。指导性搜索方法具有较强的通用性,无需利用问题的特殊信息,这也造成了对已知问题信息的浪费。尽管启发式算法对问题的依赖性较强,但对特殊问题却能利用问题信息较快地构造解,其时间性能较为理想。所以,如何合理结合两者优点来构造新算法,对于实时性和优化性同样重要的工程领域,具有很强的吸引力。基于这种现状,算法混合的思想已发展成为提高算法优化性能的一个重要且有效的途径,其出发点就是使各种单一算法相互取长补短,产生更好的优化效率。

近年来,基于混合思想的算法在工程中已有较多应用,但缺乏严格且丰富的理论研究

和效率分析,上一章介绍的算法统一框架为混合算法的系统化分析与研究提供了可行途径。在建立的统一算法框架基础上,有机地结合各算法的优点提出更高效的算法,进行较严格的收敛性和优化效率的理论分析,提出有关算法参数选择和算法适用域的规律性和指导性结论,拓宽算法在实际工程问题中的应用等,都是算法研究中值得重视的有价值课题。

本章将阐述基于统一结构如何设计有效的混合优化策略,并着重介绍由遗传算法和模拟退火融合成的混合策略,并对其收敛性和优化性能进行分析。

7.2 基于统一结构设计混合优化策略的关键问题

上一章我们通过综合广义邻域搜索算法及其混合算法的共性,建立了广义邻域搜索算法的一种统一结构,在结构上为算法的系统化研究和分析提供了一种有效的途径,并为开发新型高效的混合算法提供了指导。为了设计好适合问题的高效混合算法,不仅需要处理好上一章中所指出的单一广义邻域搜索算法优化流程中的六个环节,而且需要处理好统一结构中分解策略相关的一些关键问题。这些关键问题主要包括以下的几方面。

1. 问题分解与综合的处理

空间的分解策略有利于利用空间资源克服问题求解的复杂性,是提高优化效率的有效优化求解手段。分解的层次数与问题的规模和所采纳的算法有关。由于不同算法在适用域上存在差异,实际求解时要求子问题的规模适合于所采纳的子算法进行高效优化,同时应考虑到各子问题的分布能保证逆向综合时取得较好的优化度。例如,对平面大规模 TSP 问题,若以 SA 为子算法,研究表明,将子问题的规模设置在 50 点以内,并采用平面邻近分割或聚类的分解方法是比较有效的。

2. 子算法和邻域函数的选择

子算法和邻域函数的选择与问题的分解具有关联性。为提高整体优化能力,在对问题合理分解后,在进程层次上要求采用的各种子算法和邻域函数在机制和结构上具有互补性,使算法整体同时具备高效的全空间探索能力和局部趋化能力。例如,并行搜索和串行搜索机制相结合,全局遍历性与局部贪婪搜索相混合,大范围迁移和小范围摄动的邻域结构相结合等。

3. 进程层次上算法转换接口的处理

算法的接口问题,即在子算法确定后如何将它们在优化结构上融合,是提高优化效率和能力的主要环节。为此,首先要对各算法的机制和特点有所了解,对算法的优化行为和搜索效率进行深入的定性分析,并对问题的特性有一定的先验知识。当一种算法或邻域函数无助于明显改善整个算法的优化性能时,如优化质量长时间得不到显著提高,则可考虑切换到另一种搜索策略。例如,神经网络的 BP 训练进入平坦或多峰区时,可切换到

SA 搜索。但是,用严格的定量指标来准确衡量算法的动态优化能力和趋势具有一定的难度。并且,完全定量且一成不变的接口处理,将难以适应优化过程的动态演变。合理的处理手段应是基于规则自适应动态变化的。为了研究混合算法的整体性能,如收敛性等,在理论上将涉及到切换系统的研究内容,实际应用时也需要作广泛和深入的研究。

4. 优化过程中的数据处理

优化信息和控制参数在各算法间需要进行合理的切换,以适应优化进程的切换。特别是,要处理好不同搜索方式的算法间当前状态的转换和各子问题的优化信息交换与同步处理。原则上,这些问题属于技术层面的问题,应视所用算法、编程技术和计算机类型作出具体的设计。

总之,通过对上述关键问题的合理和多样化处理,可以构造出各种复合化结构的高效混合优化策略。

7.3 一类 GASA 混合优化策略

本节利用遗传算法和模拟退火算法作为子算法,基于广义邻域搜索算法的统一结构来构造一类 GASA 混合优化策略,以提高算法的优化性能。

7.3.1 GASA 混合优化策略的构造出发点

构造一类 GASA 混合优化策略的出发点主要在于以下几方面。

1. 优化机制的融合

理论上,GA 和 SA 两种算法均属于基于概率分布机制的优化算法。不同的是,SA 通过赋予搜索过程一种时变且最终趋于零的概率突跳性,从而可有效避免陷入局部极小并最终趋于全局最优;GA 则通过概率意义下的基于“优胜劣汰”思想的群体遗传操作来实现优化。对选择优化机制上如此差异的两种算法进行混合,有利于丰富优化过程中的搜索行为,增强全局和局部意义下的搜索能力和效率。

2. 优化结构的互补

SA 算法采用串行优化结构,而 GA 采用群体并行搜索。两者相结合,能够使 SA 成为并行 SA 算法,提高其优化性能;同时 SA 作为一种自适应变概率的变异操作,增强和补充了 GA 的进化能力。

3. 优化操作的结合

SA 算法的状态产生和接受操作每一时刻仅保留一个解,缺乏冗余和历史搜索信息;而 GA 的复制操作能够在下一代中保留种群中的优良个体,交叉操作能够使后代在一定程度上继承父代的优良模式,变异操作能够加强种群中个体的多样性。这些不同作用的

优化操作相结合,丰富了优化过程中的邻域搜索结构,增强了全空间的搜索能力。

4. 优化行为的互补

由于复制操作对当前种群外的解空间无探索能力,种群中各个体分布“畸形”时交叉操作的进化能力有限,小概率变异操作很难增加种群的多样性。所以,若算法收敛准则设计不好,则 GA 经常会出现进化缓慢或“早熟”收敛的现象。另一方面,SA 的优化行为对退温历程具有很强的依赖性,而理论上的全局收敛对退温历程的限制条件很苛刻,因此 SA 优化时间性能较差。两种算法结合,SA 的两准则可控制算法收敛性以避免出现“早熟”收敛现象,并行化的抽样过程可提高算法的优化时间性能。

5. 削弱参数选择的苛刻性

SA 和 GA 对算法参数具有很强的依赖性,参数选择不合适将严重影响优化性能。SA 的收敛条件导致算法参数选择较为苛刻,甚至不实用;而 GA 的参数又没有明确的选择指导,设计算法时均要通过大量的试验和经验来确定。GA 和 SA 的相混合,使算法各方面的搜索能力均有提高,因此对算法参数的选择不必过分严格。研究表明,混合算法在采用单一算法参数时,优化性能和鲁棒性均有大幅度提高,尤其对较大规模的复杂问题。

7.3.2 GASA 混合优化策略的流程和特点

基于上述出发点,构造一类高效 GASA 混合优化策略,其结构流程如图 7.3.1 所示。

对照统一结构,此 GASA 混合算法的特点可归纳如下:

(1) GASA 混合策略是标准 GA、SA 以及并行 SA 算法的一个统一结构。若在 GASA 混合策略中移去有关 SA 的操作,则混合策略转化为 GA 算法;若移去 GA 的进化操作,则转化为并行 SA 算法;进一步,若置种群数为 1,则转化为标准 SA 算法。

(2) GASA 混合策略是一个两层并行搜索结构。进程层次上,混合算法在各温度下串行地依次进行 GA 和 SA 搜索,是一种两层串行结构。其中,SA 的初始解来自 GA 的进化结果,SA 经 Metropolis 抽样过程得到的解又成为 GA 进一步进化的初始种群。空间层次上,GA 提供了并行搜索结构,使 SA 转化成为并行 SA 算法,因此混合算法始终进行群体并行优化。

(3) GASA 混合策略利用了不同的邻域搜索结构。混合算法结合了 GA 和 SA 搜索,优化过程中包含了 GA 的复制、交叉、变异和 SA 的状态产生函数等不同的邻域搜索结构。复制操作有利于优化过程中产生优良模态的冗余信息,交叉操作有利于后代继承父代的优良模式,高温下的 SA 操作有利于优化过程中状态的全局大范围迁移,变异和低温下的 SA 操作有利于优化过程中状态的局部小范围趋化性移动,从而增强了算法在解空间中的探索能力和效率。

(4) GASA 混合策略的搜索行为是可控的。混合策略的搜索行为,可通过退温历程(即初温、退温函数、抽样次数)加以控制。控制初温,可控制算法的初始搜索行为;控制温

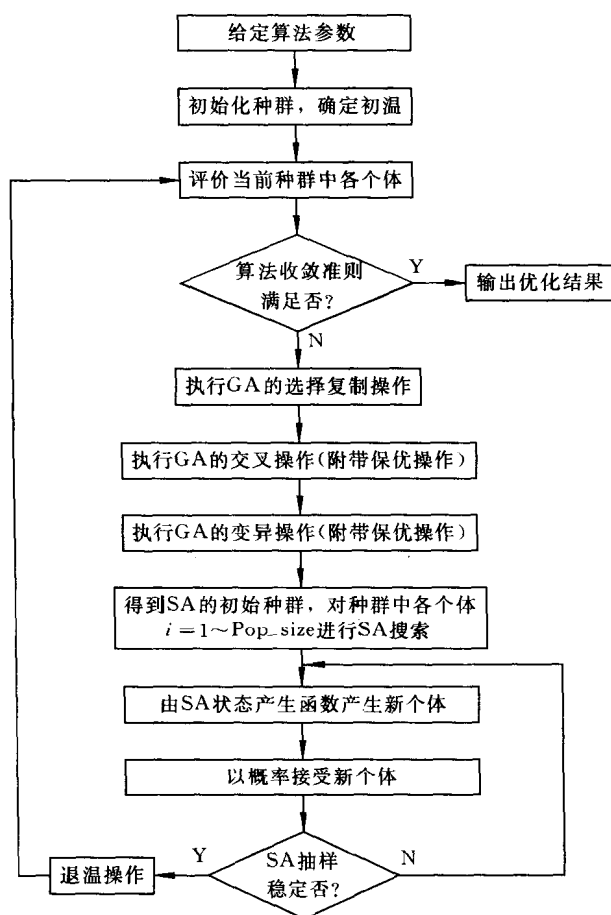


图 7.3.1 GASA 混合策略流程图

度的高低,可控制算法突跳能力的强弱,高温下的强突跳性有利于避免陷入局部极小,低温下的趋化性寻优有利于提高局部搜索能力;控制温度的下降速率,可控制突跳能力的下降幅度,影响搜索过程的平滑性;控制抽样次数,可控制各温度下搜索能力,影响搜索过程对应的齐次马氏链的平稳概率分布。这种可控性增强了克服 GA 易“早熟”收敛的能力。算法实施时,退温历程还可引入可变抽样次数、“重升温”等高级技术。

(5) GASA 混合策略利用了双重准则。理论上,抽样稳定和算法终止准则均由收敛条件决定。但是,这些条件往往不实用。在设计算法时,抽样稳定准则可用以判定各温度下算法的搜索行为和性能,也是混合算法中由 SA 切换到 GA 的条件;算法终止准则可用以判定算法优化性能的变化趋势和最终优化性能。两者结合可同时控制算法的优化性能和效率。

由此可见,在优化机制、结构和行为上,GASA 混合优化策略均结合了 GA 和 SA 的

特点,使两种算法的搜索能力得到相互补充,弥补了各自的弱点,它是一种优化能力、效率和可靠性较高的优化方法。对于存在多极小的复杂优化问题,混合策略的优化性能将体现出更明显的优越性。下面我们将从理论上对 GASA 混合策略的收敛性进行阐述。

7.3.3 GASA 混合优化策略的马氏链描述

在 GASA 混合优化策略中,各温度下的状态转移由 GA 的复制、交叉、变异和 SA 操作共同决定,且只与前一状态有关。同时,由于算法中温度是时变的,因此混合策略中状态的演化过程可用一个非平稳马氏链描述。记温度 T 下的状态转移矩阵为 $\mathbf{P}(T) = \mathbf{RCMQ}(T)$,其中 $\mathbf{R}, \mathbf{C}, \mathbf{M}, \mathbf{Q}(T)$ 分别表示 GA 的复制、交叉、变异和 SA 操作对应的状态转移矩阵。

对于有限状态的组合优化问题 $\min f(s_i)$,其中 s_i 为问题的一个状态,对应于算法的一个个体,状态集 $\Omega = \{s_i, i=1, 2, \dots, n\}$, $|\Omega| = n$ 。设种群集 Λ 的各种群均包含 m 个个体,种群总数 $|\Lambda| = \binom{n+m-1}{m}$,即为在 n 个不同元素中可重取 m 个元素的方案总数。那么,所有种群及其转换关系可由一个有向图 $G=(V, E)$ 来表示。其中, V 为所有种群组成的顶点集, $E = \{(i, j) \mid i, j \in \Lambda, j \in N(i)\}$ 为边集, $N(i)$ 表示种群 i 的邻域,且规定 $i \notin N(i), j \in N(i) \Leftrightarrow i \in N(j)$ 。进而,称有向图 G 是一个连通图,如果其任意两个种群是连通的,即存在一个正的概率序列使得一个种群经有限步状态转移能够达到另一种群。

下面,我们将对各状态转移矩阵及其相应操作间的关系作出说明。

1. 遗传操作

遗传操作仅与各状态的适配值有关,而与温度无关。因此,对应遗传操作的状态转移矩阵均为定常阵。具体地说,复制操作按与适配值有关的概率将种群映射到自身或其他种群,其对应的一步转移概率阵 $\mathbf{R}$ 为随机阵;交叉操作按交叉概率 $P_c \in [0, 1]$ 把一个种群通过邻域函数映射到另一种群,其对应的一步转移概率阵 $\mathbf{C}$ 为随机阵;变异操作按变异概率 $P_m \in (0, 1)$ 通过改变个体把一个种群由邻域函数映射到另一种群,其对应的一步转移概率阵 $\mathbf{M}$ 为严格正的随机阵。

2. 模拟退火操作

SA 中种群的转移由个体的转移引起,规定种群 L 属于种群 K 的邻域 $N(K)$ 当且仅当种群 L 和 K 仅存在一个不同个体。若 $s_{L_i} \neq s_{K_j}$,则要求个体 s_{K_j} 可由 s_{L_i} 通过状态产生函数得到,即 s_{K_j} 属于 s_{L_i} 的邻域 $N(s_{L_i})$ 。因此,SA 状态产生函数由种群 L 一步产生种群 K 的概率为

$$g_{LK} = g_{s_{L_i} s_{K_j}} / g(L) = \begin{cases} g(s_{L_i}, s_{K_j}) / [g(s_{L_i})g(L)] = g(s_i, s_j) / [g(s_i)g(L)], & K \in N(L) \text{ and } s_{K_j} \in N(s_{L_i}) \\ 0, & K \notin N(L) \end{cases} \quad (7.3.1)$$

其中 $g(s_i), g(L)$ 使得

$$\sum_{s_j \in N(s_i)} g(s_i, s_j) / g(s_i) = 1, \quad \sum_{s_i \in L} \sum_{s_j \in N(s_i)} g(s_i, s_j) / [g(s_i)g(L)] = 1$$

结合状态产生函数和接受函数,在温度 T 下 SA 操作由种群 L 一步转移到种群 K 的概率为

$$a_{LK}(T) = \begin{cases} 0, & K \notin N(L) \text{ and } K \neq L \\ g_{LK} \min\{1, \exp[-(f(s_{Kj}) - f(s_{Li}))/T]\} \\ \quad = g_{LK} \min\{1, \exp[-(f(K) - f(L))/T]\}, & K \in N(L) \\ 1 - \sum_{K \in N(L)} a_{LK}(T), & K = L \end{cases} \quad (7.3.2)$$

其中 $f(L) = \sum_{s_j \in L} f(s_j)$ 。

因此,在温度 T 下 SA 操作一步抽样决定的种群一步转移概率阵为 $A(T) = (a_{ij})$, $i, j = 1, 2, \dots, |\Delta|$, 多步抽样对应的状态转移矩阵为 $Q(T) = A^Z(T)$, 其中, Z 为该温度下的抽样次数。显然, A 和 Q 阵均是随机阵。

7.3.4 GASA 混合优化策略的收敛性

在考察算法收敛性之前,先介绍与非平稳马氏链相关的一些定义和定理 (Seneta, 1981)。

定义 7.3.1 令 $\{P_k\}$ 为以 $g(0)$ 为初始向量的非平稳马氏链的转移矩阵, 表 $g(k) = g(0)P_0P_1 \cdots P_{k-1}$, $P(m, k) = P_mP_{m+1} \cdots P_{k-1}$, $g(m, k) = g(0)P(m, k)$ 。

定义 7.3.2 定义行向量 $g = [g_1, g_2, \dots, g_n]$ 和 $n \times n$ 方阵 $A = (a_{ij})$ 的范数分别为

$$\|g\| = \sum_{i=1}^n |g_i|, \quad \|A\| = \sup_i \sum_{j=1}^n |a_{ij}|$$

引理 7.3.1 非平稳马氏链是弱遍历 (weak ergodic) 的, 若对任意 m , 有

$$\lim_{k \rightarrow \infty} \sup_{g_1(0), g_2(0)} \|g_1(m, k) - g_2(m, k)\| = 0$$

$g_1(0), g_2(0)$ 为任意初始状态概率向量。

引理 7.3.2 非平稳马氏链是强遍历 (strong ergodic) 的, 若存在一个非零向量 $q = [q_1, q_2, \dots, q_n]$, $q_i \geq 0, i = 1, 2, \dots, n$ 且 $\|q\| = 1$, 并对所有 m 和任意初始向量 $g(0)$, 有

$$\lim_{k \rightarrow \infty} \sup_{g(0)} \|g(m, k) - q\| = 0$$

注: 直观上, 弱遍历性意味着存在一种趋于 $P(m, k)$ 的行的倾向, 但这不足以说明全局收敛性。强遍历性不仅意味着对初始条件具有“遗忘”性, 同时还意味着极限的存在 (即概率 1 收敛)。

以下, 通过定义遍历系数来反映马氏链的遍历性。

定义 7.3.3 任给一个随机阵 P , 其遍历系数定义为

$$\tau_1(P) = \frac{1}{2} \max_{i,j} \sum_{k=1}^n |p_{ik} - p_{jk}| = 1 - \min_{i,j} \sum_{k=1}^n \min(p_{ik}, p_{jk})$$

引理 7.3.3 非平稳马氏链是弱遍历的, 当且仅当存在一个严格增的正整数序列 $\{k_i\}, i=0, 1, \dots$, 满足

$$\sum_{i=0}^{\infty} [1 - \tau_1(P(k_i, k_{i+1}))] = \infty$$

引理 7.3.4 若非时齐马氏链为弱遍历, 且对任意 m 存在行向量满足

$$\pi(m) = \pi(m)P(m), \quad \|\pi(m)\| = 1$$

且

$$\sum_{m=0}^{\infty} \|\pi(m) - \pi(m+1)\| < \infty$$

则此马氏链为强遍历的。若 $e^* = \lim_{n \rightarrow \infty} \pi(n)$, 则对任意 m , 进而有

$$\limsup_{n \rightarrow \infty} \|g(m, n) - e^*\| = 0$$

下面我们将讨论 GASA 混合优化策略的收敛性。

定义 7.3.4 有向图 G 的半径定义为

$$r = \min_{i \in (\Lambda - \Lambda_m)} \{\max_{j \in \Lambda} d(i, j)\}$$

其中 $\Lambda_m = \{i \in \Lambda \mid f(j) \leq f(i), \forall j \in N(i)\}$, $f(i) = \sum_{s_j \in i} f(s_j)$, $d(i, j)$ 为种群 i 到达种群 j 的最短路径, 即连通 i 和 j 的最少边数。并且, 称相应于最小半径的节点即 r 达到最小的节点为 G 的中心 I_c 。对于强连通有向图 G , 其任意节点 $j \in \Lambda$ 最多经 r 步转移就能达到中心 I_c 。

引理 7.3.5 若 P, Q 为随机阵, 则 PQ 仍为随机阵, 且不等式 $\tau_1(PQ) \leq \tau_1(P)\tau_1(Q) \leq \tau_1(P)$ 成立。

定理 7.3.1 若温度序列满足 $T_0 > T_1 > \dots > T_n, \lim_{n \rightarrow \infty} T_n = 0$, 且存在正整数 k_0 使得

$$\sum_{k=k_0}^{\infty} \exp(-rD/T_{k-1}) = \infty$$

其中,

$$D = \max_{L \in \Lambda} \max_{K \in N(L)} |f(K) - f(L)|$$

则 GASA 混合算法对应的马氏链为弱遍历。

证明 据引理 7.3.3, 要证明 GASA 混合策略的弱遍历性, 只需证明引理 7.3.3 的条件成立。

令 $w = \min_{L \in \Lambda} \min_{K \in N(L)} g_{LK}$, 则

$$\forall L \in \Lambda, K \in N(L), a_{LK}(T_h) \geq w \exp(-D/T_h) \quad h = 0, 1, \dots \quad (7.3.3)$$

由于状态产生概率与 T_h 无关, 则对于 $L \in (\Lambda - \Lambda_m)$, 有

(1) 若 $K \in N(L)$ 且 $f(K) > f(L)$, 则 $a_{LK}(T_h)$ 随 h 的增大而减小且 $\lim_{h \rightarrow \infty} a_{LK}(T_h) = 0$ 。

其中, K 的存在性由图的连通性和 L 的定义范围所保证。

(2) 若 $K \in N(L)$ 且 $f(K) \leq f(L)$, 则 $a_{LK}(T_h)$ 定常。

(3) 若 $K \notin N(L)$, 则 $a_{LK}(T_h) = 0$ 。

因此, $a_{LL}(T_h)$ 随 h 的增大而增大。同时, 据式 (7.3.3) 知, 必然存在整数 $k_0 < \infty$, 对所有 $L \in (\Lambda - \Lambda_m)$:

$$a_{LL}(T_h) \geq w \exp(-D/T_h), \quad h \geq (k_0 - 1)r \quad (7.3.4)$$

于是, 对任意 $L \in \Lambda, h \geq k_0 r$, 据 I_c 的定义得

$$a_{L_c}(T_{h-r}, T_h) \geq \prod_{i=h-r}^{h-1} [w \exp(-D/T_i)] \geq w^r \exp(-rD/T_{h-1}) \quad (7.3.5)$$

由于 R, C, M 和 $A(T)$ 均为随机阵, 由引理 7.3.3 知, 各温度下的状态转移阵 $P(T) = RCMQ(T) = RCMA^Z(T)$ 也是随机阵, 且 $\tau_1(P) \leq \tau_1(A)$ 。从而

$$\begin{aligned} \tau_1(P(T_{kr-r}, T_{kr})) &\leq \tau_1(A(T_{kr-r}, T_{kr})) \leq 1 - \min_{i,j} \min(a_{il_c}, a_{jl_c}) \\ &\leq 1 - w^r \exp(-rD/T_{kr-1}), \quad k \geq k_0 \end{aligned} \quad (7.3.6)$$

进而, 可得

$$\begin{aligned} \sum_{k=k_0}^{\infty} (1 - \tau_1(P(T_{kr-r}, T_{kr}))) &\geq \sum_{k=k_0}^{\infty} w^r \exp(-rD/T_{kr-1}) \\ &= w^r \sum_{k=k_0}^{\infty} \exp(-rD/T_{kr-1}) = \infty \end{aligned}$$

再令 $k_i = ir - r$, 则 $\sum_{i=0}^{\infty} [1 - \tau_1(P(k_i, k_{i+1}))] \geq \sum_{i=k_0}^{\infty} [1 - \tau_1(P(ir - r, ir))] = \infty$, 从而引理

7.3.3 条件成立。由此, 我们证明了 GASA 混合算法对应的非平稳马氏链为弱遍历。

定理 7.3.2 若定理 7.3.1 条件成立, 则 GASA 混合策略对应的马氏链为强遍历, 即算法最终以概率 1 收敛到由全局极小解构成的最优种群集。

证明 若定理 7.3.1 条件成立, 则 GASA 混合策略对应的马氏链为弱遍历。由引理 7.3.4 知, 证明强遍历只需证明引理 7.3.4 行向量的性质和极限分布的存在性。

由于各温度下 Metropolis 抽样稳定时均能得到准平稳概率分布, 即当对称条件 $g(s_{Li}, s_{Kj}) = g(s_{Kj}, s_{Li})$ 成立时 (算法中邻域函数的特性满足对称条件), 各个体对应的准平稳概率分布为 $\pi_{s_i}(T) = \delta(T) g(s_i) \exp(-f(s_i)/T)$, 其中 $\delta(T)$ 使得 $\|\pi(T)\| = 1$ 。混合算法中, GA 仅提供 SA 初始解, 在 SA 抽样次数足够多时, GA 并不影响准平稳分布。当温度趋于零时, 平稳分布 π 趋于最优极限向量 e^* 。

$$e_{s_i}^* = \begin{cases} g(s_i)/g^*, & s_i \in S^* \\ 0, & s_i \notin S^* \end{cases} \quad (7.3.7)$$

其中, 最优状态集 $S^* = \{s_i \in \Omega \mid f(s_i) \leq f(s_j), \forall s_j \in \Omega\}$, $g^* = \sum_{s_j \in S^*} g(s_j)$ 。

考察准平稳概率分布知,存在惟一非负整数 $Z, 0 \leq Z < \infty$,使得

$$\begin{aligned} \forall s_i \in S^*, \quad \pi_{s_i}(T_{h+1}) &> \pi_{s_i}(T_h), \quad h \geq Z \\ \forall s_i \notin S^*, \quad \pi_{s_i}(T_{h+1}) &< \pi_{s_i}(T_h), \quad h \geq Z \end{aligned} \quad (7.3.8)$$

不失一般性,设种群仅含两个个体($m=2$),则种群集可描述为

$$\Lambda = \{(s_1, s_1), \dots, (s_1, s_n), (s_2, s_2), \dots, (s_2, s_n), (s_3, s_3), \dots, (s_n, s_n)\}$$

令各个种群依次对应的序号为

$$1, \dots, n, n+1, \dots, 2n-1, 2n, \dots, |\Lambda|, \quad |\Lambda| = N = n(n+1)/2$$

根据 SA 为准平稳分布又知,混合策略在各温度下的准平稳概率分布为

$$\xi = [\xi_1, \dots, \xi_N], \quad \xi_1 = \pi_{s_1}^2, \quad \xi_2 = 2\pi_{s_1}\pi_{s_2}, \dots,$$

$$\xi_n = 2\pi_{s_1}\pi_{s_n}, \dots, \xi_N = \pi_{s_n}^2, \quad \sum_{i=1}^N \xi_i = 1$$

不失一般性,令最优状态集 $S^* = \{s_1, s_2\}$,那么由 π_{s_i} 的性质知,当 $h \geq Z$ 时,

$$\begin{aligned} \|\xi(T_{h+1}) - \xi(T_h)\| &= \sum_{i=1}^N |\xi_i(T_{h+1}) - \xi_i(T_h)| \\ &= \sum_{i=1}^{2n-1} |\xi_i(T_{h+1}) - \xi_i(T_h)| + \sum_{i=2n}^N (\xi_i(T_h) - \xi_i(T_{h+1})) \\ &= \sum_{i=1}^n |\xi_i(T_{h+1}) - \xi_i(T_h)| + \sum_{i=n+1}^{2n-1} |\xi_i(T_{h+1}) - \xi_i(T_h)| \\ &\quad + \sum_{i=1}^{2n-1} (\xi_i(T_{h+1}) - \xi_i(T_h)) \\ &= \sum_{i=1}^2 (\xi_i(T_{h+1}) - \xi_i(T_h)) + 2 \sum_{i=3}^n |\pi_{s_1}(T_{h+1})\pi_{s_i}(T_{h+1}) \\ &\quad - \pi_{s_1}(T_h)\pi_{s_i}(T_h)| + \xi_{n+1}(T_{h+1}) - \xi_{n+1}(T_h) \\ &\quad + 2 \sum_{i=3}^n |\pi_{s_2}(T_{h+1})\pi_{s_i}(T_{h+1}) - \pi_{s_2}(T_h)\pi_{s_i}(T_h)| \\ &\quad + \sum_{i=1}^{2n-1} (\xi_i(T_{h+1}) - \xi_i(T_h)) \end{aligned}$$

从而

$$\begin{aligned} \sum_{h=Z}^{\infty} \|\xi(T_{h+1}) - \xi(T_h)\| &= \sum_{h=Z}^{\infty} \sum_{i=1}^2 (\xi_i(T_{h+1}) - \xi_i(T_h)) \\ &\quad + 2 \sum_{h=Z}^{\infty} \sum_{i=3}^n |\pi_{s_1}(T_{h+1})\pi_{s_i}(T_{h+1}) - \pi_{s_1}(T_h)\pi_{s_i}(T_h)| \\ &\quad + \sum_{h=Z}^{\infty} (\xi_{n+1}(T_{h+1}) - \xi_{n+1}(T_h)) + 2 \sum_{h=Z}^{\infty} \sum_{i=3}^n |\pi_{s_2}(T_{h+1})\pi_{s_i}(T_{h+1}) \end{aligned}$$

$$- \pi_{s2}(T_h) \pi_{si}(T_h) | + \sum_{h=Z}^{\infty} \sum_{i=1}^{2n-1} (\xi_i(T_{h+1}) - \xi_i(T_h))$$

分别对上式各项进行估计,得

$$\begin{aligned}
 (1) \quad & \sum_{h=Z}^{\infty} \sum_{i=1}^2 (\xi_i(T_{h+1}) - \xi_i(T_h)) \leq \xi_1(T_{\infty}) + \xi_2(T_{\infty}) \leq 1 \\
 (2) \quad & \sum_{h=Z}^{\infty} (\xi_{n+1}(T_{h+1}) - \xi_{n+1}(T_h)) \leq \xi_{n+1}(T_{\infty}) \leq 1 \\
 (3) \quad & \sum_{h=Z}^{\infty} \sum_{i=1}^{2n-1} (\xi_i(T_{h+1}) - \xi_i(T_h)) \leq \sum_{i=1}^{2n-1} \xi_i(T_{\infty}) \leq 1 \\
 (4) \quad & 2 \sum_{h=Z}^{\infty} \sum_{i=3}^n | \pi_{s1}(T_{h+1}) \pi_{si}(T_{h+1}) - \pi_{s1}(T_h) \pi_{si}(T_h) | \\
 &= 2 \sum_{h=Z}^{\infty} \sum_{i=3}^n | \pi_{s1}(T_{h+1}) \pi_{si}(T_{h+1}) - \pi_{s1}(T_{h+1}) \pi_{si}(T_h) \\
 &\quad + \pi_{s1}(T_{h+1}) \pi_{si}(T_h) - \pi_{s1}(T_h) \pi_{si}(T_h) | \\
 &\leq 2 \sum_{h=Z}^{\infty} \sum_{i=3}^n [\pi_{s1}(T_{h+1}) (\pi_{si}(T_h) - \pi_{si}(T_{h+1})) + \pi_{si}(T_h) (\pi_{s1}(T_{h+1}) - \pi_{s1}(T_h))] \\
 &\leq 2 \sum_{i=3}^n [\pi_{s1}(T_{\infty}) (\pi_{si}(T_Z) - \pi_{si}(T_{\infty})) + \pi_{si}(T_Z) (\pi_{s1}(T_{\infty}) - \pi_{s1}(T_Z))] \\
 &\leq 2 \sum_{i=3}^n 2 = 4(n-2)
 \end{aligned}$$

同理,可证

$$2 \sum_{h=Z}^{\infty} \sum_{i=3}^n | \pi_{s2}(T_{h+1}) \pi_{si}(T_{h+1}) - \pi_{s2}(T_h) \pi_{si}(T_h) | \leq 4(n-2)$$

故可导出 $\sum_{k=1}^{\infty} \| \xi(T_{k+1}) - \xi(T_k) \| < \infty$ 。结合对称条件,表明引理 7.3.4 中行向量的性质满足。

此外,由式(7.3.7)可知,极限分布存在。从而,GASA 混合策略对应的马氏链为强遍历。

注:上述推导容易推广到 $m > 2$ 和多全局极小解的情况,而前文介绍的非齐次单一模拟退火的收敛定理只不过是混合算法的一个特例。

推论 7.3.1 若温度固定为零,则 GASA 混合策略对应的马氏链仍以概率 1 收敛到全局最优。

注:当温度固定为零时,SA 算法成为趋化性算法,其状态转移阵蜕化为定常随机阵。GA 对应的状态转移阵均为定常随机阵,从而混合算法对应齐次马氏链。由于算法中包含保优操作,因此整个算法模型将转化为 Rudolph(1994)讨论的情况,参考其结论便可证明此推论。

由定理 7.3.1 知,退温函数是实现收敛性的一个关键因素。对应定理条件,以下的推论给出了一种具体的退温函数。

推论 7.3.2 若 GASA 混合优化策略中退温函数的形式为 $T_h = \lambda / \log(h + h_0)$, $h = 0, 1, \dots$, 其中 h_0 为给定参数且 $2 \leq h_0 < \infty$, $\lambda \geq rD$, 则混合算法以概率 1 收敛到全局最优解。

7.3.5 GASA 混合优化策略的效率定性分析

理论上,已经证明 SA 算法和带有保优操作的 GA 具有概率 1 的全局收敛性。实际上,算法难以实施理论上的苛刻收敛条件,因此两者的优化性能和效率均不太理想,也导致算法参数选取的困难性。为实现良好的优化性能,GA 需要以较大种群数和较长遗传步数进行优化,SA 则需要缓慢的退温历程加以保证。通过下面的定性分析说明,GASA 混合算法在优化性能、优化效率和鲁棒性方面具有明显的优越性。

1. 优化性能(避免陷入局部极小的能力)提高

一方面,一旦种群中各个体完全相同或分布“畸形”时,单一 GA 的复制和交叉操作通常不能产生新个体来增加种群的多样性。另一方面,缺少 SA 操作的小概率变异搜索,将使算法长时间“徘徊”在若干旧状态上,进而出现“早熟收敛”的现象;而大概率变异操作将使算法成为随机寻优。

此外,进化计算方法中为提高全局收敛效率,复制操作时自然希望适配值高的状态得以较大的生存概率。但是,不难理解,过强的复制操作将使种群过分地吸引到局部极小解。

因此,混合算法中 SA 的嵌入,是对 GA 变异操作的一种有效补充,赋予优化过程在各状态具有可控的概率突跳特性,尤其在高温时使得算法具有较大的突跳性,是避免陷入局部极小的有力手段,加大了打破上述僵局的可能,也减弱了 GA 对算法参数的过分依赖性。其次,在温度较低时,SA 算法演变为几乎是概率 1 的保优变异操作,Metropolis 抽样过程将实现很强的趋化性局部搜索。而相对 SA 而言,混合策略又实行群体并行优化。所以,混合策略的优化性能,尤其是避免陷入局部极小的能力,必然提高。

2. 优化效率提高

混合策略中引入 GA,使得 SA 串行搜索转化为多点并行搜索。而 GA 和 SA 各自不同的邻域搜索结构相结合,使得算法在解空间中的探索能力和范围均有所提高,因此优化效率必然比 SA 高。

3. 鲁棒性提高

相对 SA 而言,GASA 混合策略的多点搜索必然削弱算法对初值的依赖性;相对 GA 而言,混合策略的搜索行为可通过控制温度参数而加以控制,且理论上 GA 并不影响平稳分布,因此鲁棒性也将提高。

但是,初值对算法效率有一定的影响,同时考虑到通用 GA 和 SA 算法对问题信息的无依赖性,虽然随机选取初值的方法比较简单且很常用,但在处理不同问题时建议不妨利用问题信息或启发式方法构造一些解作为混合策略的初始种群,以取得良好的搜索效率。

归纳而言,GASA 混合策略可描述为:GA 利用 SA 得到的解作为初始种群,通过并行化遗传操作使种群得以进化;SA 对 GA 得到的进化种群进行进一步优化,温度较高时表现出较强的概率突跳性,体现为对种群的“粗搜索”,温度较低时演化为趋化性局部搜索算法,体现为对种群的“细搜索”。这种混合不仅是算法结构上的,而且是搜索机制和进化思想上的相互补充,为较好解决复杂优化问题提供了有力的途径。

第 8 章

混合优化策略的应用

本章着重介绍基于前文统一结构开发的混合优化策略的应用,内容涉及函数优化、模型参数估计、控制器参数整定、TSP 问题、生产调度、神经网络设计、光学仪器设计等,通过数值仿真和比较对混合策略的优化行为与性能得到进一步了解。

8.1 基于模拟退火-单纯形算法的函数优化

工程中的许多优化问题存在大规模、高维、非线性、非凸等复杂特性,而且存在大量局部极小。求解这类问题时,许多传统的确定性优化算法易于陷入局部极小,而且对初值非常敏感,甚至需要导数信息,如牛顿法、单纯形等。尽管一些具有全局优化特性的随机算法较大程度上克服了这些困难,然而基于单一结构和机制的算法一般难以实现高效优化,因此如何有效求解高维复杂函数的全局最优解仍旧是一个开放问题,开发具有通用性的高效算法也一直是该领域的重要研究课题。在此有机结合模拟退火和单纯形搜索法,提出了一类通用、简单易实现的并行化混合优化策略,并基此优化高维复杂函数。

8.1.1 单纯形算法简介

单纯形搜索法(simplex method, SM),也称可变多面体搜索法,是一种传统的处理无约束最优化问题的直接算法(Nelder et al, 1965; Lagarias, 1998),算法首先在 n 维欧氏空间 E_n 中构造一个包含 $n+1$ 个顶点的凸多面体,求出各顶点的函数值,并确定其中的最

大值、次大值和最小值,然后通过反射、扩张、内缩、缩边等策略求出一个较好解,用之取代最大(差)点,从而构成新的多面体,如此多次迭代则可逼近一个性能较好的极小点。

SM 的具体步骤如下:

(1) 确定初始点。

(2) 反射操作:求出 $n+1$ 个顶点的函数值,确定其中的最大值 f_G 、次大值 f_H 和最小值 f_L ,除去最大值点 $X^{(G)}$ 并计算剩余的 n 个点的形心 $\bar{X}$,然后求出 $X^{(G)}$ 关于 $\bar{X}$ 的对称点 $X^{(n+2)}$,计算 $X^{(n+2)}$ 函数值。若 $f(X^{(n+2)}) < f_L$ 则令 $X^{(n+3)} = \bar{X} + \gamma(X^{(n+2)} - \bar{X})$,其中 $\gamma > 1$,其初始值可取为 2,并计算 $X^{(n+3)}$ 函数值。若 $f(X^{(n+3)}) < f(X^{(n+2)})$ 则用 $X^{(n+3)}$ 取代 $X^{(G)}$ 并转步骤(5);否则用 $X^{(n+2)}$ 取代 $X^{(G)}$ 并转步骤(5)。

(3) 若 $f_L \leq f(X^{(n+2)}) \leq f_H$,则用 $X^{(n+2)}$ 取代 $X^{(G)}$ 并转步骤(5)。

(4) 若 $f(X^{(n+2)}) \geq f_H$ 则说明步长过大,需要内缩。 $f(X') = \min\{f(X^{(n+2)}), f(X^{(H)})\}$,令 $X^{(n+4)} = \bar{X} + \beta(X' - \bar{X})$,其中 $\beta = 0.5$,计算 $f(X^{(n+4)})$ 。若 $f(X^{(n+4)}) \leq f(X')$ 则用 $X^{(n+4)}$ 取代 $X^{(G)}$ 并转步骤(5);若 $f(X^{(n+4)}) > f(X')$ 则进行缩边,即 $X^{(i)} = (X^{(i)} + X^{(L)})/2, i=1, 2, \dots, n+1$,并转步骤(5)。

(5) 算法收敛否? 若 $\left\{ \frac{1}{n+1} \sum_{i=1}^{n+1} [f(X^{(i)}) - f(\bar{X})]^2 \right\}^{1/2} < \epsilon$ 则停止计算,否则转步骤(2)。

从算法流程可见,SM 简单、计算量小、优化快速,且不要求函数可导,因而适用范围较广。但它对初始解依赖性较强,容易陷入局部极小点,而且优化效果随函数维数的增加明显下降。Lagarias(1998)研究了 SM 方法求解低维函数时的收敛特性,但结论难以推广到高维问题,也即单一 SM 难以保证对高维复杂函数具有较好的优化效果。Yen 等(1998)利用 SM 与遗传算法进行混合,取得了较满意的结果。在此将探讨 SM 与 SA 的混合及其优化性能。

8.1.2 SMSA 混合优化策略

SM 与 SA 的混合策略,以下简称 SMSA,其算法流程如图 8.1.1 所示。

选择这两种算法进行混合的出发点可归纳如下:

(1) 机制的融合:SM 是确定性的下降方法,SA 是基于随机分布的算法。SM 利用 n 维空间中的 $n+1$ 维多面体的反射、内缩、缩边等性质进行优化,可以迅速得到局部最优解。SA 通过赋予搜索过程一种时变且最终趋于零的概率突跳性,从而可有效避免陷入局部极小并最终趋于全局最优解。选择机制上存在如此差异的两种算法进行混合,有利于丰富优化中的搜索行为,增强全局和局部意义下的搜索能力和效率。

(2) 结构互补:SM 始终由 $n+1$ 个点进行搜索,SA 则是串行单链的,两者混合可使

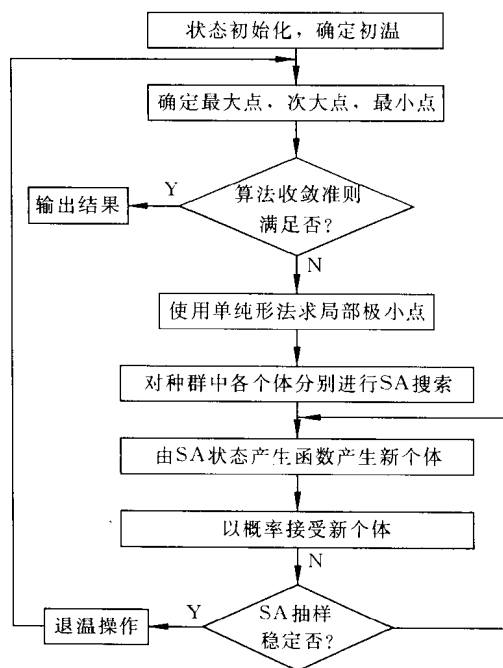


图 8.1.1 SMSA 混合优化策略

SA 成为并行算法。

(3) 行为互补: 基于可变多面体结构的确定性 SM 收敛速度快, 但易陷入局部极小点。基于概率分布机制的 SA 具有突跳性, 不易陷入局部极小点, 但收敛速度慢。两种算法结合, 可以互相补充不足, 大大提高算法的效率。具体算法流程是利用 SM 搜索到局部极小点, 然后利用 SA 的突跳性搜索得到 SM 新的初始解, 使它能跳出局部极小点, 伴随退温操作通过循环而趋近于全局最优解。

(4) 削弱参数选择的苛刻性: SA 对参数具有很强的依赖性, 参数选择不合适将严重影响优化性能。SA 的收敛条件导致参数选择较为苛刻, 甚至不实用, 设计时均要通过大量的试验和经验来确定。SM 和 SA 相混合, 使算法各方面的搜索能力均有提高, 因而对参数的选择不必过分严格。研究表明, 混合算法在采用单一算法相同参数时优化性能和鲁棒性均有大幅度提高, 尤其对较大规模的复杂问题。

下面就混合策略优化高维复杂函数的操作与参数作设计。

8.1.3 算法操作与参数设计

为了使基于 SMSA 混合策略的函数优化取得高效的优化性能, 在此对其操作和参数

作如下设计。

(1) 由于经典单纯形搜索法是一种无约束的优化方法,无边界约束处理,而优化问题通常对变量的变化区间有要求,因而需要添加边界约束处理环节。文献中大多采用撞壁法,即当反射、扩张等操作使变量越出可行域时,就取自变量为边界值。然而在多极小函数优化时,函数边界本身可能就是局部极小,或者在边界点的小邻域内存在多个局部极小点,使用撞壁法将使陷入局部极小的概率大大增加,优化效果明显下降。为了防止单纯形搜索法在进行反射操作时越出可行域,并克服撞壁法的缺陷,我们作如下处理:通常的反射操作为 $X^{(n+2)} = \bar{X} + \gamma(\bar{X} - X^{(H)})$, $\gamma=1$;若 $x_i^{(n+2)}$ 越出可行域,则令 $\gamma=0.9\gamma$,并重新计算 $X^{(n+2)}$,直到 $X^{(n+2)}$ 在可行域内。算法中 $X^{(n+3)}$ 的处理方法也如此。混合算法中单纯型搜索法的终止准则采用固定步数法,下节仿真中设定为 150 步。这种处理方式既可满足单纯形搜索法的搜索条件,又将其基于可变多面体的搜索优点尽可能发挥。

(2) SA 状态产生函数与接受函数:SA 状态产生函数采用附加扰动方式 $x' = x + \eta\xi$,其中 ξ 为满足柯西分布的随机扰动,如此既可较大概率产生小幅度扰动以实现局部搜索,又可适当产生大幅度扰动以实现大步长迁移来走出局部极小区域。状态接受函数采用 $\min\{1, \exp(-\Delta/t)\} > \text{random}[0,1]$ 作为接受新状态的条件,其中 Δ 为新旧状态的目标值差, t 为“温度”。同时,及时更新“Best So Far”的最优状态以免遗失最优解。

(3) 初温:算法中设置“最佳个体在初温下接受最差个体的概率”为 $p_r \in (0,1)$,当初始种群(即为单纯形法而随机产生的 $n+1$ 个状态)产生后,可利用上述接受函数确定初温,即 $t_0 = -(f_w - f_b)/\ln(p_r)$,其中 f_b 和 f_w 分别为种群中最佳和最差个体的目标值。由于考虑了初始种群的相对性能和分散度,初温与种群性能存在一定的关系,且通过调整 p_r 可容易得到调整。因而此策略具有一定的普遍性和指导性,一定程度上避免了初温低导致突跳不充分和初温高导致过多迂回搜索的缺点。

(4) 退温函数:采用指数退温函数,即 $t_k = \alpha t_{k-1}$, α 为退温速率。

(5) 抽样稳定和算法终止准则:SA 的 Metropolis 抽样过程采用定步长抽样法,即在各温度下均以一定步数 L_1 进行抽样,达到阈值就进行退温操作。算法终止准则兼顾优化性能和效率,避免过多无谓的搜索和优化度的严重下降,采用阈值判断法,即若最优解在连续 L_2 步数的退温期间均不变则近似认为收敛。

8.1.4 数值仿真与分析

选取如下典型函数(Yao et al, 1999),各函数的维数均取为 30,由于大部分函数存在分布不均的多极小,且有的具有非凸特性,有的具有强振荡特性,再加上高维特性,从而造成问题求解的复杂性。研究表明,SA 和 SM 能够较好地解决低维函数的优化,但对高维复杂函数的优化能力值得怀疑。因此,我们以典型的高维复杂函数探讨混合策略的优化性能,并与单一算法作比较。

(1) Rosenbrock 函数

$$f_1(X) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2], \quad |x_i| \leq 30$$

(2) Ackley 函数

$$f_2(x) = -20 \exp \left[-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right] - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i \right) + 20 + e, \quad |x_i| \leq 32$$

(3) Schwefel 函数

$$f_3(X) = \sum_{i=1}^n |x_i| + \prod_{i=1}^n |x_i|, \quad |x_i| \leq 10$$

(4) Griewank 函数

$$f_4(X) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1, \quad |x_i| \leq 600$$

(5) Generalized Penalized 函数

$$f_5(x) = \frac{\pi}{n} \left\{ 10 \sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(3\pi y_{i+1})] + (y_n - 1)^2 \right\} \\ + \sum_{i=1}^n u(x_i, 10, 100, 4), \quad y_i = 1 + (x_i + 1)/4, \quad |x_i| \leq 600$$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

(6) Generalized Schwefel 函数

$$f_6(x) = \sum_{i=1}^n -x_i \sin(\sqrt{|x_i|}), \quad |x_i| \leq 500$$

(7) Generalized Rastrigin 函数

$$f_7(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10), \quad |x_i| \leq 5.12$$

其中, $f_1 \sim f_5, f_7$ 的最优解为 0, f_6 的最优解为 -12569.5。

以 Visual C++ 为仿真环境, 采用 P III 550/128M 计算机对上述各问题分别用 SA, SM 和 SMSA 进行优化, 算法参数选取为: $p_r = 0.1, \alpha = 0.9, L_1 = 70, L_2 = 100, \eta$ 初始值取 1.0, 每隔 100 代置 $\eta = \beta\eta$, 其中 β 为 $[0.01, 0.9]$ 之间的随机数。各算法对各算例均随机执行 50 次, 仿真统计结果如表 8.1.1 所示。其中, 若搜索最终解与问题最优的绝对误差大于 0.5, 则认为算法陷入局部极小。

由仿真结果可见:

(1) 单纯形搜索法对高维复杂函数的优化性能与初值鲁棒性均很差, 它虽然是一种确定性方法, 而且利用了可变多面体的空间结构, 但对初始解和问题的依赖性仍较强, 对

表 8.1.1 仿真结果统计与比较

函数	算法	最佳优化值	平均优化值	最差优化值	优化值方差	陷入局部极小次数
$f_1(X)$	SMSA	0.003579	0.004124	0.007942	0.000701	0
	SA	0.091159	1.709216	14.913612	3.417524	10
	SM	3413127.7	4471424.8	106917575	63395272	50
$f_2(X)$	SMSA	0.000011	0.000019	0.000025	0.000003	0
	SA	0.000329	10.916361	19.444722	8.02062	16
	SM	12.07561	16.310453	19.578658	2.154887	50
$f_3(X)$	SMSA	0.000019	0.000024	0.000026	0.000003	0
	SA	1.989359	2.465039	2.779682	0.184694	50
	SM	16.95377	2.582e11	1.279e13	1.791e12	50
$f_4(X)$	SMSA	0	0	0	0	0
	SA	0	0.00997	0.041831	0.011841	39
	SM	28.76466	175.430593	398.804239	136.0389	50
$f_5(X)$	SMSA	0	0	0	0	0
	SA	0.000001	9.888872	286.710455	48.983816	38
	SM	20.999782	27830976	148819861	46783718	50
$f_6(X)$	SMSA	-12569.5	-12569.5	-12569.5	0	0
	SA	-12451.0	-12098.1	-11858.9	146.96	50
	SM	141.889	279.409	363.85	52.986	50
$f_7(X)$	SMSA	0	0	0	0	0
	SA	0.058582	0.173107	5.829279	0.805271	35
	SM	141.889282	279.4093	363.85039	52.98591	50

高维复杂函数的优化过程很容易陷入局部极小点。

(2) 单一模拟退火算法尽管具有搜索行为概率可控的特点,但在缺乏单纯形搜索法的定向搜索能力配合时,在极小点附近的趋化性搜索效果不好,初值鲁棒性也不理想。虽然性能较 SM 法有所改善,但仍不适合于高维复杂函数的优化。

(3) SMSA 融合 SM 和 SA 两者的优点,包含了随机性搜索和确定性搜索,既有 SA 退火历程对搜索行为的控制,又有单纯形法可变多面体结构的几何指导,对高维复杂函数的优化表现出高精度的优化性能和很好的初值鲁棒性,是一种很有潜力的函数优化算法。

此外,基于问题维数对算法性能影响的研究表明,SM 和 SA 的优化质量和初值鲁棒性随问题维数的增加明显下降,而 SMSA 的优化性能对问题维数的增加具有较强的适应性。因此,对于低维函数,由于各类算法均能取得较满意的优化性能,建议采用 SA 或 SM 求解,以节省优化时间;但对高维复杂函数,建议采纳 SMSA 混合策略,以实现高精度和强初值鲁棒性的优化性能。

8.2 基于混合策略的控制器参数整定和模型参数估计研究

8.2.1 引言

模型参数辨识和控制器参数整定是控制工程中的重要研究课题。传统的最小二乘法及其推广算法通常是在已知系统模型结构的基础上进行的,难以确定系统的时滞、阶次等信息,因而应用具有局限性,尤其对于非线性时滞系统,而且其搜索策略本质上是基于误差指标的梯度下降,显然对多峰型目标函数很易陷入局部极小。同时,控制器控制效果的好坏完全取决于其参数的整定,如比例积分微分(PID)控制器,而传统整定方法,如 Ziegler-Nichols 法(Z-N 法)、响应曲线法、临界比例法、继电型自整定法、单纯形法等,或是依赖于对象模型,或是易于陷入局部极小,同样存在一定的应用局限性,且难以实现高性能的整定效果,常常超调较大、调整时间较长、误差指标过大等。近年来,遗传算法作为一种新兴的优化和自学习算法在控制工程中逐渐受到重视(Visio 1999; Varsek et al, 1993),它是一种基于生物进化论的并行搜索算法,搜索空间大,能够避免陷入局部极小,但存在难以确定算法参数和易早熟收敛的弱点。在此利用 GASA 混合策略,通过设计合适的算法操作来使其适应不同类型模型的参数估计和 PID 控制器参数的整定。

8.2.2 模型参数估计和 PID 参数整定

要做到对模型参数的合理估计,通常要求部分模型的先验知识,输出量的可测性,信噪比足够大,并确定要估计的参数。

通常,系统模型可描述为

$$y(t) = f(r, \theta) \quad (8.2.1)$$

其中, $y(t)$ 为系统输出, r 为系统输入, $\theta=(\theta_1, \theta_2, \dots, \theta_k)$ 为待估计参数, f 的形式可以是传递函数、状态空间或 ARMA 模型等。

所谓参数估计,即在一定的系统输入下,根据系统输出与实际采样数据 $y_n(t), t=1, 2, \dots, n$ 获得参数的估计值,从而确定过程的数学模型。基于混合策略进行参数估计的原理如图 8.2.1 所示。

PID 控制器具有结构简单、容易实现、控制效果好、鲁棒性强等特点,同时它原理简明,参数物理意义明确,理论分析体系完整,并为工程界所熟悉,因而在工业过程控制中至今仍得到广泛应用(Astrom et al, 1995)。通常,PID 控制器算式和其离散化后的增量式分别如下:

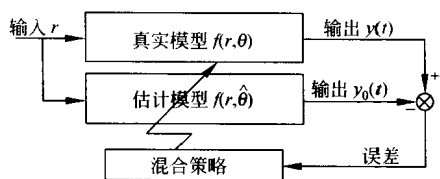


图 8.2.1 基于混合策略进行参数估计的原理图

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right] \quad (8.2.2)$$

$$\Delta u(k) = K_p \left\{ \Delta e(k) + \frac{T_0}{T_i} e(k) + \frac{T_d}{T_0} [\Delta e(k) - \Delta e(k-1)] \right\} \quad (8.2.3)$$

其中, K_p , T_i 和 T_d 分别为比例、积分和微分参数, T_0 为采样周期, $e(t)$, $u(t)$ 分别为误差变量和对象输入。若记 $K_i = K_p T_0 / T_i$, $K_d = K_p T_d / T_0$, 则上式改写为

$$\Delta u(k) = K_p \Delta e(k) + K_i e(k) + K_d [\Delta e(k) - \Delta e(k-1)] \quad (8.2.4)$$

所谓 PID 参数整定, 即确定实现最佳控制目标的上述三参数。基于混合策略进行控制器参数整定的原理如图 8.2.2 所示。

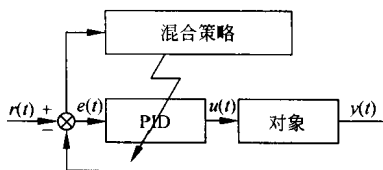


图 8.2.2 基于混合策略进行控制器参数整定的原理图

可以说, 参数估计和整定本质上均是基于一定目标函数的参数寻优问题。

8.2.3 混合策略的操作与参数设计

(1) 编码: 模型参数估计和控制器参数整定实质上是多维函数优化问题, 由于二进制编码通常会导致很大的计算量和存储量, 且串长影响算法精度。在此采用双精度实数编码, 将待估计模型参数和待整定的控制器参数直接用实数型向量形式表示来进行优化搜索。

(2) 目标函数: 对于整定问题, 在此采用能反映系统调节品质的绝对误差矩积分 (ITAE), 即 $\int_{\tau}^{\infty} t |e(t)| dt$ 为目标函数, 它强调抑制超调和调整时间。对于估计问题, 令采样获得的系统真实输出为 $y_0(t)$, 在搜索参数下的模型输出为 $y(t)$, 参数估计目标就是使两者差距最小, 在此采用下式为适应值函数, 同时为防止分母为 0 或者溢出, 在分母加上 0.01。

$$f = \frac{1}{\sqrt{\sum_t (\mathbf{y}(t) - \mathbf{y}_0(t))^T (\mathbf{y}(t) - \mathbf{y}_0(t)) + 0.01}} \quad (8.2.5)$$

(3) 初始种群和初温: 鉴于算法的随机性, 初始种群通常采用随机方式产生。初温的确定同前文, 即 $t_0 = -(c_w - c_b) / \ln p_r$ 。

(4) 选择和交叉操作: 对于估计问题, 采用比例选择以概率 $f_i / \sum f_i$ 来选取 P_s 个用于交叉的个体; 对于整定问题, 首先将种群中各个体按目标值由大到小进行排列, 以概率 $2k / [P_s(1 + P_s)]$ 选择第 k 号个体与种群中的最优个体进行交叉概率 P_c 下的算术交叉, 并重复 $P_s/2$ 次, P_s 为种群数, 然后从新旧种群集中择优产生后代种群, 即 $(P_s, 2P_s)$ 策略。

(5) 变异操作和 SA 状态产生函数: 由于 SA 可进行状态劣向转移以克服陷入局部极小, 因此将变异的性质改为概率 1 的趋化性搜索, SA 状态产生函数视为变异概率可控的变异操作, 在高温下实现较强的变异功能, 在低温下实现较强的趋化功能。实数编码下的变异操作和 SA 状态产生函数采用附加扰动方式 $x' = x + \eta\xi$, 其中 ξ 为满足正态分布的随机扰动, η 为尺度参数。

(6) SA 接受函数和退温函数: 采用最常用方式 $\min\{1, \exp(-\Delta/t)\} > \text{random}[0, 1]$, 式中 t 为温度参数, Δ 为新旧状态目标值的差, 并及时更新“Best So Far”以免遗失最优解。退温策略采用常用的指数退温策略, 即 $t_k = \lambda t_{k-1}$, $0 < \lambda < 1$ 。

(7) 终止准则: 采用固定 L_1 步抽样策略, 且若最优解在连续 L_2 步退温期间均不变则近似认为收敛。如此可兼顾优化性能和效率, 避免过多无谓搜索和优化度的严重下降。

8.2.4 数值仿真与分析

1. 模型参数估计

为了充分考察混合策略的性能, 采用以下多种系统模型进行仿真实验。

模型 1: 二阶惯性带延迟环节的传递函数形式, 待估计参数为比例系数 k , 惯性系数 T_1 , T_2 和延迟系数 τ 。

$$\frac{y(s)}{u(s)} = \frac{k}{T_1 s^2 + T_2 s + 1} e^{-\tau s} \quad (8.2.6)$$

模型 2: 状态空间模型, 待估计参数为 $\theta_1, \theta_2, \theta_3$ 和 θ_4 。

$$\begin{cases} \begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} \theta_1 x_1(t) x_2(t) \\ \theta_2 x_1^2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ u(t) \end{bmatrix} \\ y(t) = \theta_3 x_2(t) - \theta_4 x_1^2(t) \\ x_1(0) = x_2(0) = 1 \\ t = 0, 1, \dots, 50 \end{cases} \quad (8.2.7)$$

模型 3: Hammerstein 模型, 待估计参数为 a_1, a_2, b_0, b_1 和 d 。

$$\begin{cases} A(z^{-1})y(k) = z^{-d}B(z^{-1})\varphi[u(k)] \\ A(z^{-1}) = 1 + a_1z^{-1} + a_2z^{-2} \\ B(z^{-1}) = b_0 + b_1z^{-1} \\ \varphi[u] = \begin{cases} \sqrt{u+1/2} - \sqrt{1/2}, & 5 \geq u \geq -1/2 \\ -\sqrt{|u+1/2|} - \sqrt{1/2}, & -5 \leq u < -1/2 \end{cases} \end{cases} \quad (8.2.8)$$

基于 Matlab 与 C++ 的混合编程环境进行仿真,各参数取值如下: $P_s=20$, $p_r=0.1$, $\lambda=0.85$, $L_1=30$, $L_2=20$, 采样时间 $T_0=0.1$ 。实验发现 η 对搜索效果的影响较大,在此取 $\eta=(p_a-p_b)/10$, 其中 p_a, p_b 分别为搜索范围的上限和下限。由于变异采用基于正态分布随机数的扰动方式,而在 $(-5, 5)$ 以外的范围概率分布几乎为 0, 因此这种做法可以使搜索邻域集中在限定范围内,提高搜索效率。此外,对于离散参数的寻优,例如模型 3 中的参数 d , 对其交叉后得到的个体 x'_1, x'_2 分别进行上界和下界取整,对其变异后的个体和 SA 产生的新个体均进行四舍五入取整。

各算法对各模型均进行 20 次随机仿真,统计得到的平均结果如表 8.2.1~表 8.2.3 所示。

表 8.2.1 模型 1 参数估计结果

参数	k	T_1	T_2	τ
真实值	1	1	2	1
估计值	1.0000	0.9990	1.9997	1

表 8.2.2 模型 2 参数估计结果

参数	θ_1	θ_2	θ_3	θ_4
真实值	0.5	0.3	1.8	0.9
估计值	0.5069	0.3048	1.8095	0.9077
结果(姜波等,2000)	0.4916	0.3014	1.8432	0.9267

表 8.2.3 模型 3 参数估计结果

参数	a_1	a_2	b_0	b_1	d
真实值	-1.5	0.7	1	0.5	2
估计值	-1.5004	0.6984	0.9861	0.4516	2
结果(姜波等,2000)	-1.4982	0.697	1.3654	-0.0371	2

由此可见,混合策略对各模型参数的估计结果是非常令人满意的。需要指出的是,对 Hammerstein 模型参数的估计还存在一定的误差,分析这类模型,某些参数对模型输出效果的影响远小于其他参数,例如 b_1 , 而混合策略对参数估计仅利用模型的输出数据,在这种情况下,这些参数的估计值就很可能偏离真实值较大。图 8.2.3 显示了真实 Hammerstein 模型与估计模型在相同输入下的输出结果比较,两条曲线极为接近的,因此我们认为参数估计结果是可以接受的。此外,与文献(姜波等,2000)相比,混合策略得到的结果大大优于单纯的遗传算法,这也正是由于混合策略结合了遗传算法和模拟退火两者的优点,提高了全局优化性能。

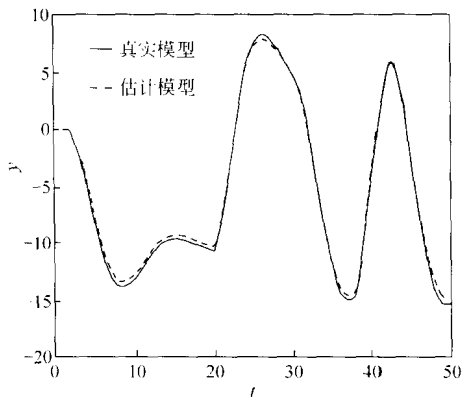


图 8.2.3 模型 3 与估计模型在相同输入下的输出结果比较

2. 控制器参数整定

首先,以一阶惯性加纯滞后环节 $\frac{e^{-0.5s}}{2s+1}$ 为控制对象,考察混合策略整定 PID 控制器的统计性能,并与 GA 和 Z-N 方法作比较。算法参数设定为: $\eta=0.6$,其他参数同上。20 次随机仿真的统计结果如表 8.2.4 所示,而 Z-N 方法整定所得的目标值为 12.6951,三种方法的典型结果如图 8.2.4,混合策略与 GA 的典型目标下降曲线如图 8.2.5 所示。

表 8.2.4 混合策略和 GA 的性能比较

算法	最优目标值	平均目标值	最差目标值	目标值均方差	平均进化代数
混合策略	3.9902	6.0101	9.9262	1.5974	39.80
GA	4.3980	7.1619	13.3277	2.3001	49.15

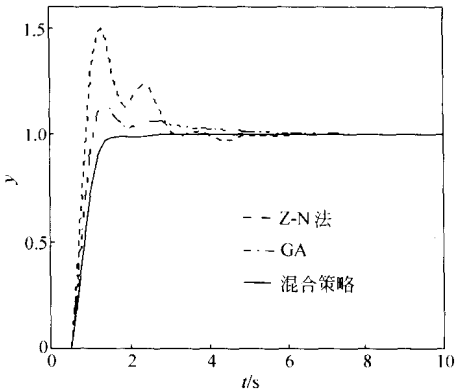


图 8.2.4 典型整定结果比较

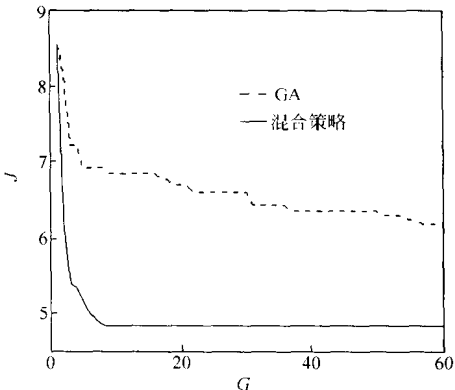


图 8.2.5 混合策略和 GA 的目标下降曲线比较

其次,对水轮机系统 $\frac{1}{1+0.2s} \frac{1-0.8s}{1+0.4s} \frac{1}{0.2+0.96s}$ 的控制器(Liu et al, 1997)进行整

定研究,取 $\eta=0.1$, $T_0=0.04$ 。采用混合策略、GA 和单纯形法整定(Liu et al, 1997)而得的阶跃响应曲线如图 8.2.6 所示。

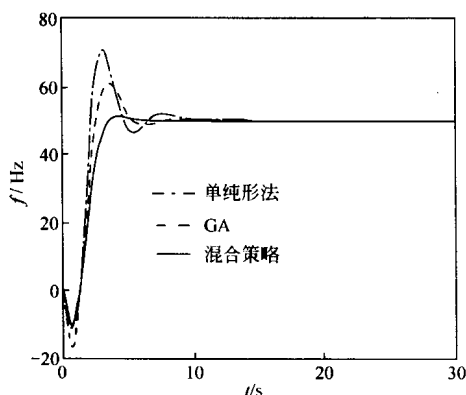


图 8.2.6 对水轮机系统的整定结果比较

仿真结果表明:

(1) 混合策略具有良好的优化性能、优化速度和初值鲁棒性,且明显优于 GA。

(2) 混合策略整定而得控制器具有良好的控制品质,且大大优于 GA 和 Z-N 方法,具体表现在超调小、调整时间短、误差指标小等。

此外,基于混合策略的优化过程简单易行,原理很容易理解,并且对问题信息和对象的依赖程度较小,因此混合策略完全可以很好地解决系统辨识和控制器整定这些复杂的多极小优化问题,同时其通用性的一面又使得算法具有较广阔的应用前景。

8.3 基于混合策略的 TSP 优化研究

8.3.1 TSP 的混合优化策略设计

本节我们针对 TSP 问题,提出和设计了一个 GASA 混合优化策略,其优化流程如图 8.3.1 所示。其中,2opt 启发式操作(Lin et al, 1973)的引入是为了考察基于问题信息的搜索对随机算法性能的影响。

下面,首先具体来设计混合策略的编码和优化操作。

(1) 编码选择:前文在介绍 SA 算法时简单阐述了 TSP 的几种编码方案的特点,鉴于路径编码策略的优点,我们以此来作为编码策略,即直接采用城市在路径中的位置来构造用于优化的状态,如此可在优化过程中加入启发式信息,也有利于优化操作的设计。

(2) 适配值函数和选择操作的设计:适配值函数用于对各状态的目标值作适当变换,

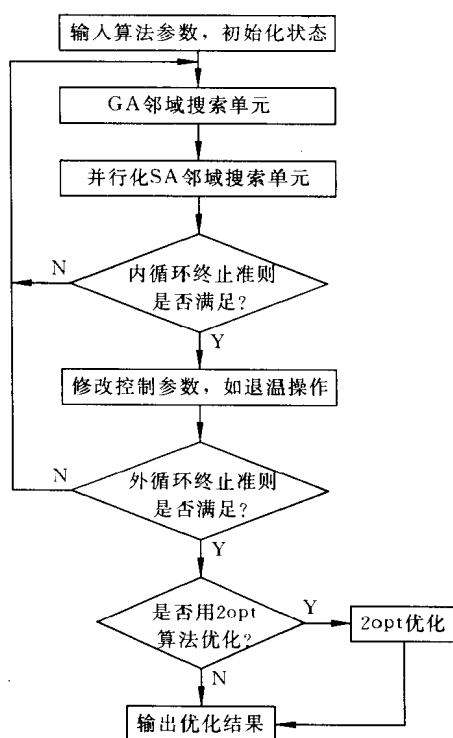


图 8.3.1 TSP 的 GASA 混合优化流程

以体现各状态性能的差异。取 $f_x = \exp(-d_x)$ 为适配值函数, 其中 d_x 为状态 x 的回路长度。为使赋予适配值高的个体有较高的生存概率, 采用比例选择策略, 即产生随机数 $\xi \in [0, 1]$, 若

$$\sum_{j=1}^{i-1} f_j / \sum_{j=1}^{\text{Pop_size}} f_j < \xi \leq \sum_{j=1}^i f_j / \sum_{j=1}^{\text{Pop_size}} f_j$$

则选择状态 i 进行复制。

(3) 交叉操作的设计: 交叉操作的目的是组合出继承父代有效模式的新个体, 进行解空间中的有效搜索。但是, Non-ABEL 群置换操作产生后代方式简单, 过分打乱了父串, 不利于保留有效模式; 次序交叉和循环交叉对父串的修改幅度也较大。PMX 算子一定程度上满足 Holland 图式定理的基本性质, 子串能够继承父串的有效模式。基此, 我们利用 PMX 作为交叉算子。

(4) 变异操作和 SA 状态产生函数的设计: 对于基于路径编码的变异和 SA 状态产生函数操作, 可将其设计为: ① 互换操作 (SWAP); ② 逆序操作 (INV); ③ 插入操作 (INS)。前文曾对三种操作的性能作了简单分析, 由于表示 TSP 解的串很长, SWAP 算子更有利于算法的大范围探索, INV 算子则更有利于算法的小范围迁移。基此, 我们选择 INV 操

作作为变异操作,在染色体中引入小幅度变化,增大一定的种群多样性。同时,为配合 INV 的变异操作,体现算法中邻域函数的“混合”,利用 SWAP 操作作为 SA 的状态产生函数。

(5) SA 状态接受函数的设计:采用 $\min\{1, \exp(-\Delta/t)\} > \text{random}[0,1]$ 作为接受新状态的条件,其中 Δ 为新旧状态的目标值差, t 为“温度”。

(6) 退温函数的设计:采用指数退温函数,即 $t_k = \lambda t_{k-1}$, λ 为退温速率。

(7) 温度修改准则和算法终止准则的设计:为适应算法性能的动态变化,较好地兼顾算法的优化性能和时间性能,采用阈值法设计的“温度修改”和“算法终止”两准则。也即,若优化过程中得到的最佳优化值连续 20 代进化保持不变,则进行退温;若最佳优化值连续 20 次退温仍保持不变,则终止搜索过程,以此优化值为算法的优化结果。

8.3.2 基于典型算例的仿真研究

TSP 研究中常用的典型算例是 Eilon 等 1969 年提出的 30, 50 和 75 城市 TSP 问题,至今为许多文献所采纳。其中, Fogel(1993)利用进化规划进行了深入研究,算法利用 100 个个体构成种群,优化结果存在较大的波动性,其所得的最佳优化结果如下:

(1) 30 城市问题:最佳优化值为 423.741,经产生 4×10^4 个后代得到。

(2) 50 城市问题:最佳优化值为 427.855,经产生 1×10^5 个后代得到。

(3) 75 城市问题:最佳优化值为 549.18,经产生 3.25×10^5 个后代得到。

下面以 Fogel 给出的结果作为参考最优值,利用 16MB RAM 的 Pentium 586/133 计算机,以 Borland C++ 为仿真环境,以此三个算例研究 GASA 混合优化策略的性能,并与 GA、SA、并行 SA(PSA)、贪婪法、2opt、SA+2opt 和 PSA+2opt 等算法作比较,约定各算法均随机运行 20 次。

算法参数选择如下:种群数取 10;交叉和变异概率分别取 0.99 和 0.9;退温速率取 0.9;初温取为初态目标值的均方值;GA 和 SA 在各代进化中均循环 100 步。

图 8.3.2 和图 8.3.3 显示了 GASA, GA, SA, PSA 对 75 城市问题的优化进程,表 8.3.1 归纳了各算法对不同算例的统计优化性能。

根据上述仿真结果,可对混合优化策略导出如下的结论:

(1) GASA 混合优化策略的全局优化度最高。对于中小规模问题,不仅能得到最优解,甚至其平均优化度优于一些对比性算法的最佳优化度。

(2) GASA 混合优化策略的鲁棒性能最高。波动率小,优化结果可靠。

(3) GASA 混合优化策略的时间性能较 SA 和 PSA 有较大的改善。相对于 GA,由于 GASA 自适应增加了一定的优化步数,克服了 GA 易早熟收敛的弱点。

表 8.3.1 基于典型 TSP 算例的算法性能比较

算法		性能指标	TSP 典型算例		
			30 城市问题	50 城市问题	75 城市问题
混合算法	GASA	E_m	0	0	-0.0125*
		E_f	0	1.688	2.411
		E_g	554	738	870
对比算法	贪婪法	E_m	11.703	14.335	10.567
	2opt	E_m	0	0.847	1.912
		E_f	3.261	5.412	5.421
	SA	E_m	0	5.172	10.587
		E_f	4.619	9.716	15.098
		E_g	1018	1345	1693
	SA+2opt	E_m	0	0.865	2.032
		E_f	2.995	4.411	5.563
	PSA	E_m	0	2.215	5.301
		E_f	0.274	5.306	9.325
		E_g	1012	1343	1718
	PSA+2opt	E_m	0	0.943	2.125
		E_f	0.274	3.508	4.684
	GA	E_m	0.049	3.576	3.112
		E_f	4.073	6.824	8.136
		E_g	404	409	425

注： E_m 为最小相对误差， E_f 为波动率， E_g 为平均进化代数；对于 75 城市算例，混合算法发现了优化解 542.3093，优于 Fogel 的解 549.18。

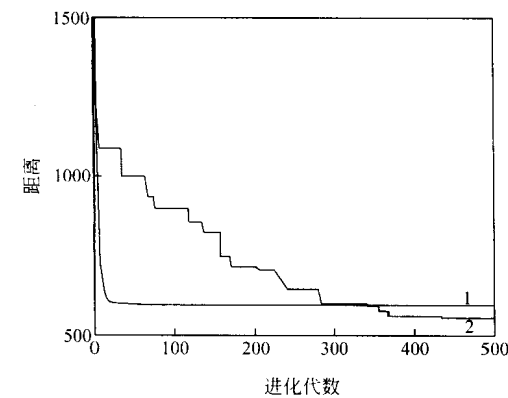


图 8.3.2 优化性能变化：1—GA, 2—GASA

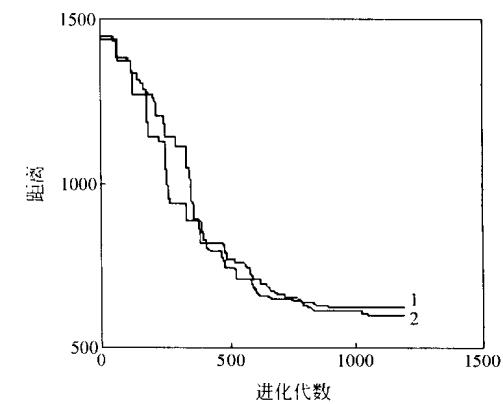


图 8.3.3 优化性能变化：1—SA, 2—PSA

对于结论(3),图 8.3.2 和图 8.3.3 针对一个复杂的问题给出了直观的图示。图中显示,去除算法终止准则的影响,GA 经 100 代进化就达到最终优化值,但优化度较差,呈现出明显的早熟收敛;SA 和 PSA 需 1000 代左右进化才达到最终优化值,搜索过程冗长;GASA 经 500 代进化达到最终优化值,尽管进化代数多于 GA,然而优化质量较两者有明显改善。

此外,由仿真结果还可得到如下一些结论:

(1) SA 和 PSA 的优化解可通过 2opt 操作进行进一步改进。因此,单一 SWAP 邻域搜索,即使在并行搜索的情况下,仍难以消除交叉路径,对大规模问题尤其如此。

(2) GA 中 PMX 和 INV 的混合能够消除交叉路径,但无法突破较大的能量障碍,仍旧易发生早熟收敛现象。

(3) GASA 中熔合了 PMX,INV 和 SWAP 算子。一方面,算法自身能够完全消除交叉路径,无需采用 2opt 操作进行改进;另一方面,算法自身具有避免陷入局部极小的能力,全局优化度较高。

(4) 贪婪法和 2opt 法对问题结构(即点的分布)具有很强的依赖性。

因此,对于复杂优化问题,单一机制的优化算法很难实现全局优化,且效率较低。多种优化机制和邻域搜索结构相混合,是能较大程度提高全局优化度和鲁棒性的有力途径,并可一定程度上放松对单一算法参数选择的苛刻性。

8.3.3 对 TSP 的进一步讨论

1. 大规模 TSP 研究

许多实际工程问题往往属于大规模 TSP,譬如 VLSI 芯片加工问题(对应于 1.2×10^6 城市 TSP, Korte, 1988)、X-ray 衍射问题(对应于 14000 城市 TSP, Bland et al, 1989)、电路板设计中的钻孔问题(对应于 17000 城市 TSP, Litke, 1984),以及对应于 442 城市 TSP 的著名 PCB442 问题(Grotschel et al, 1991)等。对于不少实际问题,由于结构的复杂性,其最优解附近往往存在大量局部极小值,同时鉴于 TSP 的 NP 复杂性,任何单一算法在处理这类大规模问题时都会面临优化性能和时间性能的双重挑战。

邻域搜索算法统一结构中的空间分解体现了“分而治之”的思想(Cesari, 1996),结合“混合”优化策略,为解决大规模问题提供了一种有效的途径。基此,我们把整个问题分解成若干区和层,各层中的每个区作为一个小规模问题,并用混合算法求解,再把各层视为以每个区作为一个点的又一个小规模问题,结合一定连接方式再利用混合算法逐区逐层求解,以快速地得到大规模问题的一个满意解。这种处理方法,利用分解策略降低问题的求解复杂性,利用子算法高效求解子问题,可实现对大规模问题较高的优化效率和较满意的优化质量。由于工程中并不过于追求最优解,但需满足一定的优化效率,因此上述求解策略是可接受的。譬如,将 $m \times n$ 个城市等分为 m 个区,则空间容量就由 $(m \times n)!$ 大大

缩小为 $m \times n!$ 。

就以 2400 城市 TSP 为例(各点在 200×200 正方形中随机产生),取 80 个城市为一子区,利用 SA 作为子算法,并采用就近连接原则,在 10 次随机运行中得到的最优值为 16080.7,平均值为 16217.3,寻优时间 763.3 秒(PC 486/66 计算机),波动率为 0.84%。显见,结果的各方面优化性能都是比较满意的。若采用 GASA 作为子算法,优化性能可进一步提高。

此外,我们还对不同规模的问题进行了仿真研究(王凌等,1998),研究表明:

(1) 对规模大的问题,分区法的寻优度、波动性和时间性能均比单一算法要好。并且,在分解和合并原则固定时,寻优时间基本与问题规模呈线性关系。

(2) 对小规模问题(如 $n < 80$),就优化度而言宜采用单一算法,但分区法在时间性能上具有优越性,且基本无波动性。

2. 动态和开环 TSP 研究

在实际工程问题中,往往包含若干时变和非对称因素,或者不要求闭环结构,因此对这些问题的研究是很有意义的。譬如,网络的连接可采用光纤,也可采用普通导线;道路交通可能是单行的,或会出现堵塞;信息高速网络要求动态选择路由;PERT 图则对应于一类开环问题。

处理这类问题的最简单方法是对网络路径加权,以权值的变化来反映网络动态的变化,甚至可以是随机变化的。此外,通过对权值的预测,结合适当的修正,也可实现动态网络的优化。这些思路对解决离散时变调度问题也具有指导意义。对开环问题的处理,则只需适当地修改目标函数即可。在上述推广中,由于算法结构和优化流程无需修改,混合优化策略依然是解决这些复杂问题的优选方法。

8.4 基于混合策略的加工调度研究

8.4.1 基于混合策略的 Job-shop 优化研究

8.4.1.1 引言

Job-shop 问题(JSP)是一类较 TSP 更为复杂的典型调度问题,是许多实际问题的简化模型。一个 JSP 可描述为: n 个工件在 m 台机器上加工, O_{ij} 表示第 i 个工件在第 j 台机器上的操作,相应的操作时间 T_{ij} 为已知,事先给定各工件在各机器上的加工次序(称为技术约束条件),要求确定与技术约束条件相容的各机器上所有工件的加工次序,使加工性能指标达到最优。在 Job-shop 问题中,除技术约束外,通常还假定每一时刻每台机器只能加工一个工件,且每个工件只能被一台机器所加工,同时加工过程为不间断,机器间缓

冲区容量为无限。若各工件的技术约束条件相同,一个 JSP 就转化为较为简单的 Flow-shop 问题。进而,若各机器上各工件的加工次序也相同,则问题可进一步转化为置换 Flow-shop 问题。

一般,JSP 的求解要比 TSP 更困难,其原因在于:

- (1) 编码复杂且多样化;
- (2) 解空间容量巨大, n 个工件、 m 台机器的问题包含 $(n!)^m$ 种排列;
- (3) 存在约束条件的限制,必须考虑解的可行性;
- (4) 将搜索状态解码为有效调度的操作复杂且多样化。

此外,JSP 对应的优化超曲面也要复杂得多,存在多个分布无规则甚至彼此相邻的极小点,对算法的优化造成很大困难。

JSP 研究中常用典型算例* 包括:MT06(6×6)、MT10(10×10)、MT20(20×5)和 LA01(10×5)、LA06(15×5)、LA11(20×5)、LA16(10×10)、LA21(15×10)、LA26(20×10)、LA31(30×10)、LA36(15×15)等,其中括号内数字分别表示工件数和机器数,MT 和 LA 分别代表问题构造者 Muth,Thompson 和 Lawrence。此外,还可对上述问题分类为“简单问题”和“复杂问题”,MT06,LA01,LA06 和 LA11 因几乎所有算法均能得到最优解而被称为简单问题,另外 7 个问题因其求解困难而被称为复杂问题。下面,对引用最多的 MT10 算例的一些代表性研究结果进行归纳,如表 8.4.1 所示。

表 8.4.1 MT10 的代表性研究结果归纳

作者和年份	算法	优化值	时间性能	计算机类型
Lageweg**1984	列举	930	1h,47min 遇到,持续 9h,6min 无改进	CYBER 170-750
Carlier 等 1989	分支定界	930	产生 22021 个节点,耗时约 5h	Prime 2655
Matsuo 等 1988	Controlled SA	946		
Adams 等 1988	列举型 SB	930	851s	VAX780/11
Nakano 等 1991	GA	965	种群 1000 进化 150 代	
Van Laarhoven 等 1992	改进型迭代法 SA	1006 930	57772s	VAX-785

* 可以从 Inter 网站点 <ftp://mscmga.ms.ic.ac.uk/pub/jobshopl.txt> (或 jobshop2.txt) 下载。

** From the private communication of Lageweg B J with Van Laarhoven et al discussing the achievement of a makespan of 930 for MT10, this is also cited from Van Laarhoven et al 1992.

续表

作者和年份	算法	优化值	时间性能	计算机类型
Dell'Amico 等 1993	TS	935		
Croce 等 1995	GA	946	628s	PC 486/25
Dorndoft 等 1995	12 种单一规则	1191		
	12 种规则混合	1088		
	基于规则的 GA	960		DEC station
	移动瓶颈法 SB	1031		3100
	部分列举型 SB	951		
	基于 SB 的 GA	938		

由此可见,对 JSP 问题的求解相当困难,即使是 10×10 规模的问题也是如此。同时,大量研究表明:

- (1) 列举型和分支定界策略的计算量巨大,难以应用于大规模问题;
- (2) 基于优先规则的方法,能快速构造解,但优化质量很差;
- (3) SB 优化度较好,但求解过程和算法实施很复杂,且难以移植;
- (4) 单一 SA 和 GA 算法的性能对参数有较强依赖性,而结合优先规则的混合方法对优化性能改善程度很有限。

因此,目前有关 JSP 的研究主要集中在如何改善优化算法的性能。为此,我们下面将提出对 JSP 的一个 GASA 混合优化策略。

8.4.1.2 JSP 的析取图描述和编码

首先,对 JSP 研究中的若干重要概念和编码问题,进行简要的介绍。

1. JSP 的若干重要概念

活动调度、半活动调度和非延迟调度是 JSP 研究中的重要概念,三者的包含关系如图 8.4.1 所示,其定义如下:

定义 1 称一个调度为活动调度,如果在不推迟其他操作或破坏优先顺序的条件下,其中没有一个操作可以提前加工。

定义 2 称一个调度为半活动调度,如果在不改变机器上加工顺序的条件下,其中没有操作可以提前。

定义 3 称一个调度为非延迟调度,如果至少存在一个工件等待加工时,对应地不存在相应的处于空闲的机器。

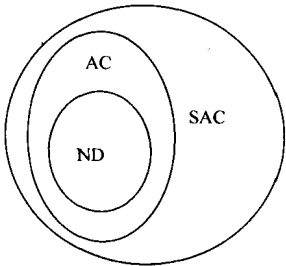


图 8.4.1 三种类型调度的关系(ND 表示非延迟调度集,AC 表示活动调度集,SAC 表示半活动调度集)

对于正规调度指标,如最大完成时间,业已证明最优调度必为活动调度。因此,如果将搜索空间限于活动调度集,不仅能保证最优调度的存在,而且能够提高优化效率。

2. JSP 的析取图描述

析取图是描述 JSP 的常用工具。对 n 工件、 m 台机器(共 N 个操作)的 JSP,对应的析取图 $G=(V, A, E)$ 如图 8.4.2 所示。其中, V 为所有操作构成的顶点集,包括 0 和 $N+1$ 两个虚拟操作(分别表示加工开始和结束); A 为 n 条子弧构成的弧集,子弧(实线)表示某工件按约束条件在所有机器上从开始到结束的加工路径; E 为 m 条子边构成的边集,子边(虚线)表示同一机器上加工各操作的连接。

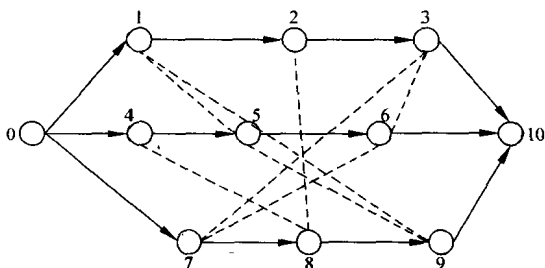


图 8.4.2 3 工件、3 机器 JSP 的析取图表示

现取最大完成时间为指标,那么对 JSP 的求解就归结为找到各边(即机器)上作为优先决策的各操作的一组顺序(即走向),当同一机器上有多个操作出现冲突时,上述顺序用于决定各操作的先后,最终得到各操作间没有冲突的一个有向非循环图,而其关键路径长度即为最大完成时间。

3. JSP 的编码研究

编码问题是算法实施优化的首要和关键问题。鉴于 JSP 的约束性,编码技术必须考虑码的合法性和可行性。目前,编码研究已成为利用 GA 求解 JSP 的关键内容之一。JSP 的编码可归纳为直接和间接两种编码。

(1) 直接编码:将各调度(排列)作为状态,通过状态演化达到寻优目的。如基于操作的编码、基于工件的编码等。

(2) 间接编码:将一组工件的分配处理规则作为状态,算法优化的结果是一组最佳的分配规则序列,再由分配规则序列构造调度。如基于优先规则的编码、基于析取图的编码等。

下面,对 JSP(n 工件、 m 机器)的若干编码方式及其性质进行归纳。

(1) 基于操作的编码:每个状态由 $n \times m$ 个代表操作的基因组成,是所有操作的一个排列,并称 $n \times m$ 为标准长度。其优点是,任意基因串的排列均能表示可行调度。其缺点是,只具有半 Lamarckian 性(Whitley et al, 1994),即 GA 利用此编码使后代继承父代优良模式的能力较弱。解码过程是先将状态转化为有序调度表,然后产生调度方案。

(2) 基于工件的编码:每个状态由 n 个代表工件的基因组成,是所有工件的一个排列。其优点是,任意工件的排列均能表示可行调度,且具有 Lamarkian 性,即 GA 能够利用此编码使后代继承父代的优良模式。其缺点是,仅能表征部分解空间,不能保证全局最优解的存在性。解码过程是先加工第一号工件的所有操作,然后依次以其最早允许时间加工后面各工件的所有操作。

(3) 基于优先表的编码:每个状态由分别对应于 m 台不同机器的 m 个子串构成,各子串是一个长度为 n 的符号串,用于表示一种优先表,各符号表示相应机器上的加工操作。其优点是,码长为标准长度,且各状态均能表示可行调度。其缺点是,不具有 Lamarkian 性,难以用遗传操作实行进化。解码过程通过分析机器上当前等待队列的状态,结合优先表决定相应的操作。

(4) 基于工件对关系的编码:Nakano 等(1991)利用二元矩阵表示调度,矩阵决定相应机器上工件对的优先关系。这种编码应用较少,其缺点是,码长大于标准长度,存在较大的冗余,必须考虑合法性,且 Lamarkian 性较差。

(5) 基于优先规则的编码:每个状态由一个优先分配规则序列构成。算法的优化结果是一个满意的规则序列,然后以此产生调度方案。其优点是,具有 Lamarkian 性,码长为标准长度,且能保证调度的可行性。其缺点是,优化性能较差。

(6) 基于析取图的编码:采用每个状态(由各边的操作顺序组成)作为优先决策,以决定同台机器上发生操作冲突时各操作的顺序。其优点是,码长为标准长度,能保证调度的可行性。其缺点是,只具有半 Lamarkian 性,解码过程较复杂。

(7) 基于完成时间的编码:此编码策略利用各操作完成时间的有序表表示状态。其优点是,无需解码过程。其缺点是,不具有 Lamarkian 性,必须考虑状态的合法性,且需要设计特殊的状态转移算子。

(8) 基于机器的编码:每个状态为所有机器的排列,并以此通过移动瓶颈方法构造调度。其特点是,只具有半 Lamarkian 性,码长小于标准长度,解码过程较复杂。需要指出的是,这种编码仅能表征部分解空间,不能保证全局最优解的存在性。

8.4.1.3 JSP 的混合优化策略设计

下面,约定优化指标为最大完成时间(makespan)。为保证编码策略不遗漏问题的全局最优解,使优化操作容易处理状态的可行性和合法性问题,并使算法具有高效的性能,首先来设计一种利用技术约束条件矩阵实施基于操作的编码策略的算法,然后再来设计解码操作将码转化为活动调度,最后设计进行优化的混合策略和优化操作。

1. 编码算法

首先对若干符号进行介绍,然后给出编码算法。

(1) 机器顺序阵 J_M ,其中 $J_M(i, j)$ 表示加工 i 工件的第 j 个操作的机器号, $J_M(i, \cdot)$ 表示 i 工件的所有操作按优先顺序加工的各机器号的排列。

(2) 加工时间阵 T , 其中 $T(i, j)$ 为 i 工件在 j 机器上的加工时间。

(3) 工件排列阵 M_j , 其中 $M_j(i, j)$ 为 i 机器上第 j 次加工的工件号, $M_j(i, \cdot)$ 表示 i 机器上依次加工的各工件的排列。

机器顺序阵即为技术约束矩阵, 和加工时间阵一样, 也是事先已知的。工件排列阵则是调度的一种表示形式, 可用于构造 Gantt 图。

[编码算法]

Repeat

(1) 随机产生整数 I 表示某一工件, 满足 $1 \leq I \leq n$ 。

(2) 由 J_M 得到 I 工件目前尚未加工的优先权最高的操作对应的机器 $J_M(I, 1)$ 。

(3) 将 J_M 的第 I 行各元依次左移一位, 尾部空出的位置填 0。

Until J_M 的所有元均为 0。

上述算法利用技术约束条件(即工件加工的优先次序), 是一种基于操作的编码策略, 对应状态空间容量为 $(m \times n)! / (n!)^m$ 。

2. 解码算法

下面, 设计将由上述算法产生的码 $s[i], i = 1, \dots, n \times m$ 及其任意置换方式解码成可行的活动调度策略的算法。

[活动化解码算法]

(1) 令 $k[i] = 1, i = 1, \dots, n$ 。

(2) For $i = 1$ to $n \times m$

(2.1) 得到加工工件 $s[i]$ 的机器号 $J_M(s[i], k[s[i]])$;

(2.2) 令 $k[s[i]] = k[s[i]] + 1$;

(2.3) 将工件 $s[i]$ 在机器 $J_M(s[i], k[s[i]])$ 上的操作以最早允许加工时间加工, 即从零时刻到当前时刻对该机器上的各加工空闲依次判断能否将此工件插入加工。若能, 则在空闲中插入加工, 并修改该机器上的加工队列; 否则, 以当前时刻加工该工件, 将此工件排在当前队列的末尾。

算法中, 工件 I 能在空闲时间段 $[a, b]$ 插入加工的条件为: $\max(t(I), a) + t_{proc} \leq b$ 。其中, a 和 b 分别为空闲起始和终止时刻, $t(I)$ 为工件 I 目前的最早允许加工时间, t_{proc} 为工件在机器上的加工时间。

若算法中(2.3)步骤采用以当前时刻加工工件的策略, 则算法将产生半活动调度, 称之为半活动化解码算法。结合编码算法可知, 任何可行的半活动调度均能以相同概率产生。因此, 问题的最优解必然包含在解空间中。

3. 举例(3 工件、2 机器)

设机器顺序阵 $J_M = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 2 \end{bmatrix}$, 加工时间阵 $T = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 2 & 1 \end{bmatrix}$ 。

若编码算法产生的随机数序列为 $[1 \ 3 \ 2 \ 2 \ 1 \ 3]$,则相应的半活动调度对应的 Gantt 图如图 8.4.3 所示,工件排列阵 $M_j = \begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 3 \end{bmatrix}$,最大完成时间为 7。若作活动化解码处理,所得活动调度对应的 Gantt 图如图 8.4.4 所示,工件排列阵为 $M_j = \begin{bmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \end{bmatrix}$,最大完成时间为 6(最优解)。

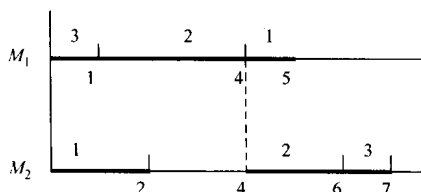


图 8.4.3 半活动调度对应的 Gantt 图

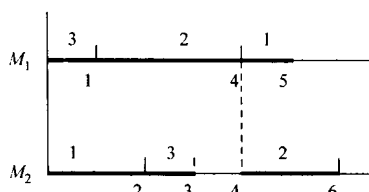


图 8.4.4 活动调度对应的 Gantt 图

4. GASA 混合优化策略[JSP 的 GASA 混合优化策略]

- (1) 初始化算法参数(种群数目 P_{size} , 退温速率 λ , 仿真次数等)。
- (2) 由调度生成算法随机产生初始种群。
- (3) 经活动化解码计算各个体对应调度的目标值, 令最优值 c^* 为当前种群中最优个体的评价, $k=0$ 。
- (4) 判断算法收敛准则是否满足? 若是, 转第(13)步; 否则, 转步骤(5)。
- (5) 令 $l=0$ 。
- (6) 随机选择个体与种群中的最优个体进行交叉运算, 产生两个新个体。若新个体目标值优于 c^* , 则更新 c^* 和最优调度; 否则, 保持 c^* 和最优调度。令 $l=l+1$ 。
- (7) 若 $l < P_{size}/2$, 则返回步骤(6)。
- (8) 保留原种群和新产生的所有个体中 P_{size} 个最佳个体。
- (9) 对所有个体进行变异操作, 保留 P_{size} 个最佳个体作为 SA 的初始种群, 并及时更新 c^* 和最优调度。
- (10) 对种群中的各个体均进行定步长抽样的模拟退火操作, 以概率 $\min[1, \exp(-\Delta/t_k)]$ 接受后代, 及时更新 c^* 和最优调度。
- (11) 进行退温操作 $t_k = \lambda \cdot t_{k-1}$, $\lambda \in (0, 1)$, 并返回第(4)步。
- (12) 输出该次仿真所得最优解和寻优时间。
- (13) 再次进行优化否? 若是, 则转第(2)步; 否则, 转步骤(14)。
- (14) 输出多次优化的统计结果, 并结束算法。

5. 优化操作和参数的设计

JSP 的 GASA 混合优化策略的许多优化操作与 TSP 的 GASA 混合优化策略相同, 如采用 INV 变异操作、SA 的 SWAP 产生函数和指数退温策略等。但是, 为了使初温的

选取具有更强的指导性并使交叉操作适合于基于操作的编码策略,我们对初温和交叉操作进行了特殊设计。

(1) 初温:初温通过式 $t_0 = -(c_w - c_b) / \ln(p_r)$ 确定。

(2) 交叉操作:基于操作的 JSP 编码策略使个体中存在许多相同“基因”,因此难以应用 TSP 优化中的交叉算子。为了使交叉操作具有快速进化的能力,并保证子代个体的可行性,我们设计了基于互补集的顺序选择法。具体步骤如下:

① 把 $1 \sim n$ 自然数集随机分成两个互余集合 S_1 和 S_2 ;

② 从前到后顺序选择父代个体 A_1 中属于 S_1 的基因,以及父代个体 A_2 中属于 S_2 的基因,构成子代个体 B_1 ;

③ 类似步骤②,从前到后顺序选择父代个体 A_1 中属于 S_2 的基因,以及父代个体 A_2 中属于 S_1 的基因,构成子代个体 B_2 。

8.4.1.4 基于典型算例的仿真研究

为了考察 GASA 混合策略的性能,以 Borland C++ 为仿真环境,采用 64MB RAM 的 Pentium II/350 计算机,对 11 个典型算例作如下的研究:

(1) 对各算例均作 20 次随机仿真,对 GASA 的各项优化指标与 SA 和 GA 作比较,统计结果如表 8.4.2 所示;

表 8.4.2 GASA,GA 和 SA 的性能比较

Problem	n, m	c^{**}	GASA			SA			GA		
			c^*	%	$\bar{t}$	c^*	%	$\bar{t}$	c^*	%	$\bar{t}$
MT06	6, 6	55	55	0	2.12	55	0	0.16	55	3.273	0.12
LA01	10, 5	666	666	0	5.72	666	0	0.48	666	3.048	0.3
LA06	15, 5	926	926	0	12.29	926	0	0.63	926	0	0.37
LA11	20, 5	1222	1222	0	25.94	1222	0	1.47	1222	0	0.6
MT10	10, 10	930	930	2.548	44.37	939	6.645	2.42	997	11.871	0.5
MT20	20, 5	1165	1165	1.221	90.17	1227	8.326	4.14	1247	12.403	1.19
LA16	10, 10	945	945	1.005	29.43	979	4.169	1.84	979	6.032	0.61
LA21	15, 10	1046	1058	2.590	184.46	1083	7.486	8.92	1156	14.56	1.07
LA26	20, 10	1218	1218	0.441	417.88	1253	5.099	19.85	1328	12.783	2.48
LA31	30, 10	1784	1784	0	546.21	1784	0.392	36.92	1836	4.826	3.62
LA36	15, 15	1268	1292	3.059	348.11	1321	5.804	16.7	1384	11.908	2.4

(2) 对各算法均能得到最优解的算例,对最优解的首达时间作比较,结果如表 8.4.3(a)所示;

(3) 以表 8.4.2 中 GASA 相同的时间测试 SA 和 GA 求解复杂问题的性能,结果见表 8.4.3(b)所示;

表 8.4.3(a) 对简单问题的最优解首达时间(秒)比较

Problem	n, m	t_c (GASA)	t_c (SA)	t_c (GA)
MT06	6, 6	0.034	0.133	0.108
LA01	10, 5	0.049	0.353	0.247
LA06	15, 5	0.015	0.049	0.092
LA11	20, 5	0.034	0.122	0.233

表 8.4.3(b) SA 和 GA 以 GASA 相同时间(表 8.4.2)求解复杂问题的性能

Problem	n, m	SA		GA	
		c^*	%	c^*	%
MT10	10, 10	937	3.355	980	10.215
MT20	20, 5	1178	3.271	1247	11.442
LA16	10, 10	946	1.101	956	4.44
LA21	15, 10	1079	4.73	1142	11.577
LA26	20, 10	1218	0.562	1299	10.197
LA31	30, 10	1784	0	1785	1.822
LA36	15, 15	1293	3.121	1359	9.621

(4) 将 GASA 算法的最优性能与 JSP 算法研究的若干权威文献的结果作比较,结果如表 8.4.4 所示。

表 8.4.4 GASA 混合算法与文献结果的比较

Problem	n, m	c^{**}	GASA Best	GA Best	TS Best	SA Best	SB Best
MT06	6, 6	55	55	55	55	55	55
LA01	10, 5	666	666	666	666	666	666
LA06	15, 5	926	926	926	926	926	926
LA11	20, 5	1222	1222	1222	1222	1222	1222
MT10	10, 10	930	930	946	935	930	930
MT20	20, 5	1165	1165	1178	1165	1165	1178
LA16	10, 10	945	945	979	945	956	978
LA21	15, 10	1046	1058	1097	1048	1063	1084
LA26	20, 10	1218	1218	1231	1218	1218	1224
LA31	30, 10	1784	1784	1784	1784	1784	1784
LA36	15, 15	1268	1292	1305	1278	1293	1305

表中,各符号的意义如下:
 n, m 分别为工件数和机器数;
 c^{**} 为问题的最优值;
 c^* 为算法 20 次仿真得到的最优值;
 $\bar{t}$ 为 20 次随机仿真的平均 CPU 时间;

%为平均优化值相对 c^* 的偏差;

t_0 为最优解的首达时间。

算法参数选取如下:

种群数 P_{size} 取 20;

初温下的最差接受概率 p_r 取 0.1;

退温速率取 0.9;

抽样步数取 $n \times m$;

以最优值连续 30 次退温均保持不变为算法终止条件。

通过对上述典型算例的仿真研究,可对 JSP 的混合优化策略得到如下的一些结论:

(1) 对于复杂问题,GASA 的优化性能较 SA 和 GA 有大幅度的提高,且对大部分算例均能得到最优值。

(2) 对于复杂问题,GASA 的鲁棒性能较 SA 和 GA 有大幅度的提高。

(3) 对于简单问题,各算法均能得到最优值,但 GASA 搜索到最优解的首达时间最短,即混合策略发现最优解的效率最高。

(4) 与文献结果相比较,混合算法对各算例的最优性能均优于 SA,GA 和 SB 算法;与 TS 相比,GASA 仅对 LA21 和 LA36 的最优性能略次,但对 MT10 的性能要好。

需要说明的是,各文献算法对参数具有很强的依赖性,且波动性能较差,而混合算法在这方面体现出了较大的优越性。同时,就时间性能而言,尽管 GASA 需要花费较多的优化时间,但这一方面是算法终止条件判断的结果,另一方面也是为避免早熟收敛而付出的代价。然而,单一算法即使以混合策略相同时间优化,也达不到混合策略的优化效果。或者说,单一算法要实现混合策略的优化效果,需要付出更多的搜索时间,甚至根本达不到混合策略的优化能力。由于单一算法的性能需要合适的参数作保证,因此算法的混合是放松对参数苛刻选择条件的有效途径。

总之,对 JSP 的有效求解说明,混合优化策略尤其是 GASA 混合策略,是解决复杂调度问题的有效而可靠的优选方法。

附:下面给出 MT10 的加工时间阵、技术约束阵,以及本文算法所得的最优工件排序阵及其对应的最优加工 Gantt 图。同时,Gantt 图也明显地反映了此问题的求解复杂性。

$$T = \begin{bmatrix} 29 & 78 & 9 & 36 & 49 & 11 & 62 & 56 & 44 & 21 \\ 43 & 28 & 90 & 69 & 75 & 46 & 46 & 72 & 30 & 11 \\ 85 & 91 & 74 & 39 & 33 & 10 & 89 & 12 & 90 & 45 \\ 71 & 81 & 95 & 98 & 99 & 43 & 9 & 85 & 52 & 22 \\ 6 & 22 & 14 & 26 & 69 & 61 & 53 & 49 & 21 & 72 \\ 47 & 2 & 84 & 95 & 6 & 52 & 65 & 25 & 48 & 72 \\ 37 & 46 & 13 & 61 & 55 & 21 & 32 & 30 & 89 & 32 \\ 86 & 46 & 31 & 79 & 32 & 74 & 88 & 36 & 19 & 48 \\ 76 & 69 & 85 & 76 & 26 & 51 & 40 & 89 & 74 & 11 \\ 13 & 85 & 61 & 52 & 90 & 47 & 7 & 45 & 64 & 76 \end{bmatrix}$$

$$J_M = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 3 & 5 & 10 & 4 & 2 & 7 & 6 & 8 & 9 \\ 2 & 1 & 4 & 3 & 9 & 6 & 8 & 7 & 10 & 5 \\ 2 & 3 & 1 & 5 & 7 & 9 & 8 & 4 & 10 & 6 \\ 3 & 1 & 2 & 6 & 4 & 5 & 9 & 8 & 10 & 7 \\ 3 & 2 & 6 & 4 & 9 & 10 & 1 & 7 & 5 & 8 \\ 2 & 1 & 4 & 3 & 7 & 6 & 10 & 9 & 8 & 5 \\ 3 & 1 & 2 & 6 & 5 & 7 & 9 & 10 & 8 & 4 \\ 1 & 2 & 4 & 6 & 3 & 10 & 7 & 8 & 5 & 9 \\ 2 & 1 & 3 & 7 & 9 & 10 & 6 & 4 & 5 & 8 \end{bmatrix}$$

最优工件排序阵 $M_j^* =$

$$\begin{bmatrix} 2 & 9 & 1 & 7 & 4 & 5 & 10 & 8 & 6 & 3 \\ 4 & 6 & 7 & 10 & 9 & 5 & 3 & 8 & 1 & 2 \\ 6 & 4 & 5 & 8 & 2 & 7 & 10 & 9 & 1 & 3 \\ 6 & 7 & 9 & 5 & 3 & 1 & 2 & 4 & 10 & 8 \\ 4 & 2 & 5 & 6 & 8 & 1 & 7 & 9 & 10 & 3 \\ 6 & 5 & 9 & 7 & 8 & 1 & 10 & 3 & 2 & 4 \\ 7 & 4 & 10 & 6 & 9 & 8 & 1 & 2 & 3 & 5 \\ 4 & 6 & 5 & 9 & 7 & 3 & 1 & 8 & 2 & 10 \\ 6 & 4 & 10 & 5 & 7 & 3 & 8 & 9 & 1 & 2 \\ 6 & 7 & 2 & 9 & 10 & 5 & 8 & 4 & 3 & 1 \end{bmatrix}$$

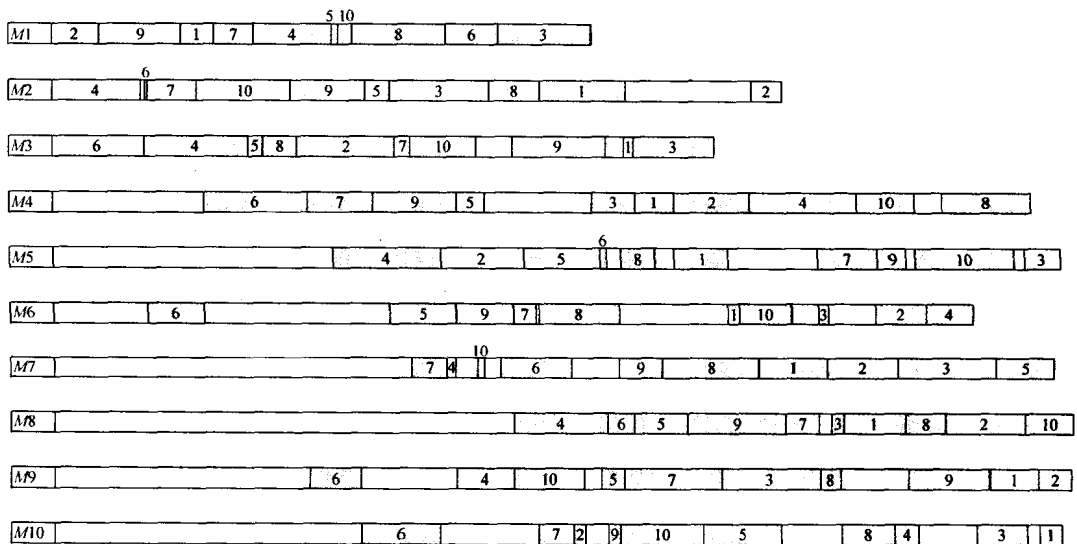


图 8.4.5 MT10 算例最优调度对应的加工 Gantt 图

8.4.2 基于混合策略的置换 Flow-shop 优化研究

置换 Flow-shop 调度问题(Permutation Flow-shop)是一类经典的加工调度问题,它是 Job-shop 问题的一个特例,具有很强的工程背景。若指标为最终加工完成时间,即 makespan,则此问题通常记为 $n/m/P/c_{\max}$ 。该问题虽然描述很简单,但已被证明为 NP 难题,它通常存在大量局部极小,而且问题的特性很难分析,从而很难高效地得到问题的全局最优解。鉴于其广泛的工程背景和理论难度,对该问题的研究具有重要的理论意义和工程价值,同时开发有效的优化算法也一直是该领域的热点课题。

置换 Flow-shop 问题的求解方法通常可分为精确方法、构造型方法、改进型方法和神经网络等。精确方法,如列举法、分支定界、动态规划等,计算量和存储量大,仅适合于小规模问题。构造型方法通过一定的规则来构造问题的解,如 Gupta 法、Johnson 法、Palmer 法、CDS 法、RA 法、NEH 法、SL 法、WSH 法等,其中 NEH 法被认为是至今最好的多项式构造型算法,这类算法能快速构造解,但通常解的质量和算法通用性较差。改进型算法从若干解出发,通过对其邻域的不断搜索和当前解的替换来改进解的质量,如遗传算法、模拟退火、进化规划、禁忌搜索等,它们通常可取得较好的优化效果,但往往需要大量迭代,且算法操作、参数和结构需要合适选取。自 Hopfield 用神经网络求解 TSP 问题之后,神经网络也成为求解调度问题的有效方法,它通过神经网络的动态演化来达到对应优化解的稳定态,但网络参数和能量函数要精心设计,并且算法复杂性较大。

本节结合启发式和随机方法产生初始解,对种群进行分解并用多种交叉操作进行进化,在整体替换后用模拟退火的 Metropolis 抽样过程代替变异操作,进而提出了混合优化策略,并通过仿真和比较来验证算法的有效性。

8.4.2.1 混合优化策略

作为一种通用的全局优化算法,GA 近年来在各领域得到广泛应用,但大量研究表明 GA 存在易早熟、算法参数敏感等缺点,取得良好的性能需要依赖较大的种群并对算法作精心设计。为了改善算法性能,下面我们提出一种混合优化策略。

首先,理论上 GA 的全局收敛性保证了算法对初值的鲁棒性,但在实际应用时由于收敛条件难以保证,从而导致算法的优化性能和效率对初始种群的依赖性。因此,我们采用启发式方法(如 NEH 法)和随机方法共同产生初始种群,进而保证了初始种群一定的质量,并不失初始种群的多样性,同时启发式方法的快速性保证了这种初始化的速度。

其次,GA 的优化过程中交叉操作通常是作用在整个种群上,并且其结构是一成不变的。在此我们将种群分解为若干子种群(类似于小生群概念),各子种群分别用不同机制的交叉操作来实现进化,用复合多交叉方式来继承父代的优良模式,并一定程度上保持种群的多样性,从而在多种邻域结构下丰富了交叉环节的搜索行为和机制,同时增强全空间

和局部范围的搜索能力。

再则,我们采用种群整体替换策略,将作用在不同子种群上的交叉操作产生的所有新个体与父代种群进行整体择优筛选,从而加速种群的进化过程。

尤其是,由于 GA 的复制对当前种群外的解空间无探索能力,个体分布“畸形”时交叉的进化能力有限,小概率变异很难增加种群的多样性,若收敛准则设计不好,GA 经常会出现进化缓慢或“早熟”收敛现象。因此,我们将 SA 基于概率突跳的 Metropolis 抽样过程融入 GA,即将可控性概率劣向转移的避免陷入局部极小的机制融入 GA,并且抽样方式又能够体现为另一种邻域搜索。同时, Metropolis 抽样过程起到概率可控的变异操作,控制初温可控制初始搜索行为,控制温度的高低可控制突跳能力的强弱,高温下的强突跳性有利于避免陷入局部极小,低温下的趋化性有利于提高局部搜索能力,控制降温速率可控制突跳能力的下降幅度,控制抽样次数可控制各温度下搜索能力,如此不仅丰富了 GA 的搜索行为,而且避免了变异概率难以选取的困难。

由此,我们构造如图 8.4.6 所示的一种混合策略,图中 P_s 为种群大小。

值得一提的是,该改进算法很容易采用并行机或分布式方式处理,并且保持了遗传算

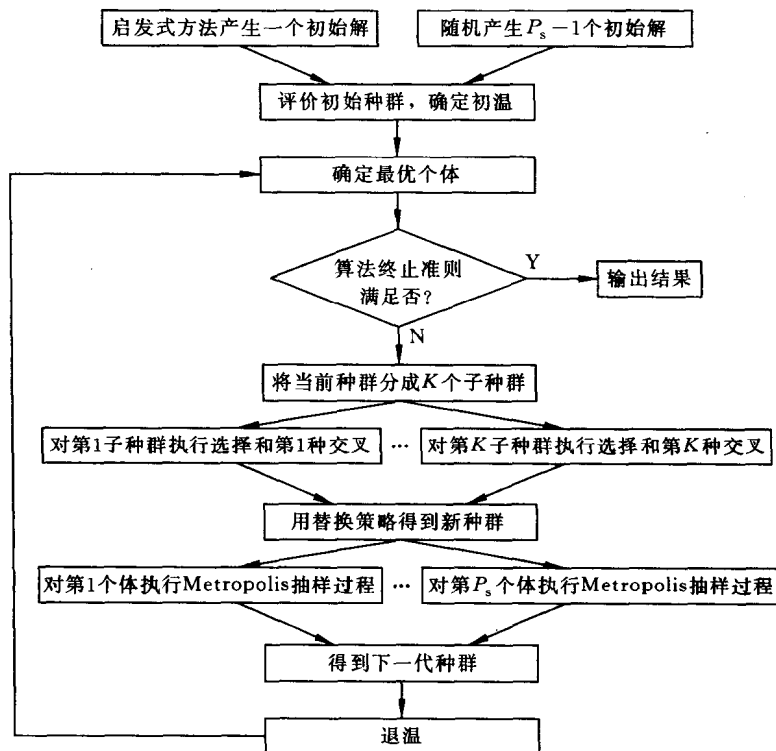


图 8.4.6 混合优化策略的流程图

法的通用性,可以在多种领域加以应用。下面针对置换 Flow-shop 对算法操作和参数作设计。

8.4.2.2 算法操作与参数设计

(1) 初始种群。为了保证初始种群一定的质量,并不失种群的多样性,我们采用 NEH 法产生一个初始解,同时随机地产生其他个体,来共同组成初始种群,仿真时种群数取 40。

(2) 初温。当初始种群产生后,算法确定其中的最优和最差状态(目标值分别为 c_b 和 c_w),并令最差状态相对最优状态的接受概率为 p_r ,由 $t_0 = -(c_w - c_b) / \ln p_r$ 可确定初温。折衷考虑优化质量和效率,仿真时取 $p_r = 0.1$ 。

(3) 交叉操作。在执行交叉操作前,我们将整个种群分成 K 个子种群(仿真时均分为 4 份),在各子种群中按一定的概率随机选取一个个体与整个种群中的最佳个体以不同的方式进行交叉,直至产生 K 个新子种群,然后进行替换操作。仿真时采用 LOX, C1, PMX, NABEL 这四种不同方式的交叉操作来继承父代优良模式,其中 LOX 能够尽量保留基因间的相对位置和相对染色体顶端的绝对位置, C1 能够在不过分打乱染色体的基础上提供足够的修改范围, PMX 能够一定程度上满足模式定理使最佳模式得以最大可能保留, NABLE 采用群的置换操作来快速产生新个体并使旧个体发生很大的修改,如此复合交叉操作使得搜索行为具有明显的多样性。

(4) 种群替换。采用种群整体替换策略,将作用在不同子种群上的交叉操作产生的所有新个体与父代种群进行整体择优筛选,从而加速种群的进化过程。

(5) Metropolis 抽样过程。抽样过程是针对每个个体进行的,我们对旧个体采用互换 SWAP 方式产生新个体,并通过判断 $\min\{1, \exp(-\Delta/t)\} > \text{random}[0, 1]$ 来接受新状态,式中 t 为温度, Δ 为新旧个体的目标值差,如此起到概率可控的变异操作,而且可以做到增加种群多样性并避免搜索陷入局部极小。考虑到搜索空间大小 $n!$ 受工件数 n 的影响,仿真时抽样过程执行 n 步,并及时更新“Best So Far”的个体,以免遗失最优解。

(6) 退温。退温策略采用工程中常用的指数退温,即 $t_k = \lambda t_{k-1}$,仿真时取 $\lambda = 0.95$,它通常能在优化度和优化效率间起到较好的折衷效果。

(7) 终止准则。为了与传统 GA 进行比较,我们简单地设置最大进化代数作为终止准则,同时考虑到问题规模对算法的影响,仿真时最大进化代数设置为工件数与机器数的积,即 $n \times m$ 。

8.4.2.3 数值仿真与分析

我们选取被广泛应用的 29 个 Benchmark 问题进行混合算法的性能研究,并与传统 GA 和 NEH 作比较,即 Carlier(1978)提出的 8 个算例 Car1, Car2, ..., Car8 和 Taillard(1993)提出的 21 个算例 Rec01, Rec03, ..., Rec41,具体数据见书后附录,它们可以从网址

<http://mscmga.ms.ic.ac.uk> 下载。混合策略和 GA 对各算例均随机运行 20 次(GA 采用 LOX 交叉, 变异概率 0.05, 初始种群随机产生), 仿真统计结果如表 8.4.5 所示。表中 c^* 为问题的最优解, RE 表示与 c^* 的相对误差 $(c_{\max}^{\text{Heu}} - c^*)/c^* \times 100\%$, BRE 为最优相对误差, ARE 为平均相对误差, WRE 为最差相对误差。为了更形象地表现混合策略、GA 与 NEH 的相对性能, 图 8.4.7 显示了混合策略、GA 结果相对于 NEH 结果的偏差, 即 $(c_{\max}^{\text{GA}} - c_{\max}^{\text{NEH}})/c_{\max}^{\text{NEH}} \times 100\%$, 横轴为算例序号。

表 8.4.5 仿真统计结果比较

Problem	n, m	c^*	混合策略			NEH	GA	
			BRE	ARE	WRE	RE	BRE	ARE
Car1	11,5	7038	0	0	0	0	0	0.27
Car2	13,4	7166	0	0	0	2.93	0	4.07
Car3	12,5	7312	0	0	0	1.79	1.19	2.95
Car4	14,4	8003	0	0	0	0.39	0	2.36
Car5	10,6	7720	0	0.08	0.61	4.24	0	1.46
Car6	8,9	8505	0	0.24	0.76	3.62	0	1.86
Car7	7,7	6590	0	0	0	6.34	0	1.57
Car8	8,8	8366	0	0	0	1.09	0	2.59
Rec01	20,5	1247	0	0.14	0.16	8.42	2.81	6.96
Rec03	20,5	1109	0	0.14	0.18	6.58	1.89	4.45
Rec05	20,5	1242	0	0.31	1.53	4.83	1.93	3.82
Rec07	20,10	1566	0	0.59	1.15	5.36	1.15	5.31
Rec09	20,10	1537	0	0.79	2.41	6.77	3.12	4.73
Rec11	20,10	1431	0	1.48	2.37	8.25	3.91	7.39
Rec13	20,15	1930	0.62	1.52	3.16	7.62	3.68	5.97
Rec15	20,15	1950	0.46	1.28	2.87	4.92	2.21	4.29
Rec17	20,15	1902	1.73	2.69	3.68	7.47	3.15	6.08
Rec19	30,10	2093	1.09	1.58	2.39	6.64	4.01	6.07
Rec21	30,10	2017	1.44	1.52	1.64	4.56	3.42	6.07
Rec23	30,10	2011	0.45	0.99	1.74	10.0	3.83	7.46
Rec25	30,15	2513	1.63	2.74	3.94	6.96	4.42	7.20
Rec27	30,15	2373	0.80	2.11	3.33	8.51	4.93	6.85
Rec29	30,15	2287	1.53	2.59	3.72	5.42	6.21	8.48
Rec31	50,10	3045	0.49	1.62	2.82	10.28	6.17	8.02
Rec33	50,10	3114	0.13	0.75	0.83	4.75	3.08	5.12
Rec35	50,10	3277	0	0	0	5.01	1.46	3.30
Rec37	75,20	4951	2.26	3.49	4.34	7.80	6.56	8.72
Rec39	75,20	5087	1.14	1.93	3.58	7.71	6.39	7.57
Rec41	75,20	4960	3.27	3.78	4.69	9.58	7.42	8.92

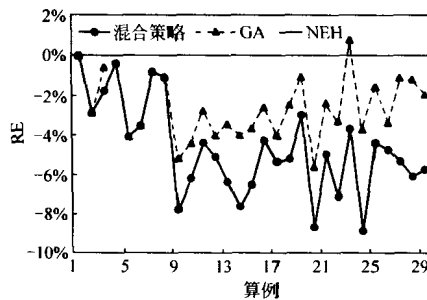


图 8.4.7 混合策略和 GA 对上述算例的优化结果
相对 NEH 结果的偏差

由仿真结果可见:

(1) 混合策略具有很好的优化质量,尤其对 20×10 以下规模的问题均能够得到最优解,甚至是 100% 的最优解,对较大规模的问题能够得到良好的近似最优解,且质量大大优于传统 GA 和 NEH 方法。

(2) 混合策略的 ARE 指标很小,这反映了算法较好的初值鲁棒性。

(3) 混合策略的自身波动性很小,其最差的优化质量与最佳优化质量差距不太大,而且最差性能也全面地大大优于 NEH 方法,并对较大规模问题还大幅度地优于传统 GA,而传统 GA 对 NEH 的性能改进幅度较小,甚至会产生差于 NEH 的解。

此外,韦有双等(2000)的改进启发式方法、Osman 等(1989)的模拟退火、Widmer 等(1989)的禁忌搜索的优化结果均还不能一致地优于 NEH 方法,但混合策略能够 100% 地大幅度优于 NEH 方法,甚至混合策略的最差结果也远远优于 NEH 方法。由此可见混合策略在优化质量方面相对于 NEH 以及传统 GA 和 SA 的优越性。

因此,我们认为混合策略之所以表现出良好优化能力的原因是,结合启发式方法和随机方法有利于产生较好质量并保持一定多样性的初始种群,采用复合化多交叉操作不仅能够很好地继承父代优良模式,进行全空间的有效搜索,而且也能增加种群的多样性,有利于进化过程的发展,种群的整体替换有利于加快种群的更新,Metropolis 抽样过程在增加局部搜索能力的同时赋予搜索过程避免局部极小的能力,而且这种能力是可控的,同时这种多点搜索很容易并行化处理。需要指出的是,由于计算机技术的高速发展,混合策略的优化过程完全可以加快,并且相对 NEH 方法其优化质量上的改进完全弥补了时间性能上一定程度的增加,当然进一步提高优化效率(如采用并行计算技术等)也是需深入的工作。

8.4.3 基于混合策略的一类批量可变流水线调度问题的优化研究

8.4.3.1 问题描述及其性质

柔性流水线调度中各类工种的加工批量通常是可变的,因此本节考虑一类批量可变

流水线调度问题。问题的描述如下:

假定所有机器呈流水线排列,各工件的加工顺序相同;在每一时刻,各机器只能加工一类工种的一个工件,且各工件只能在某一台机器上加工;给定加工时间、运输时间和工件加工设置时间(与加工顺序有关),要求确定使最终加工完毕时间最小的各类工种的批量分割,以及所有子批量的最优排序。

在这种问题中,批量和排序是相互制约的。Karmarkar 等(1985)利用排队论模型证明,确定批量对流水时间和等待时间的影响实质上是寻找最优批量的路径优化;同时指出,批量与流水时间存在 U 型关系,即过大或过小的批量都会导致较长的流水时间。

假定有 N 类工种, M 台机器,每类工种的总批量均为 L ,且机器间缓冲区容量为无限。若以每类工种的每个工件为子批量,则批量总数为 $N \times L$,解空间容量为 $(N \times L)!$,不仅搜索效率低,而且由于过多的子批量数将导致设置和运输时间的增加,搜索空间中得到的最优解不一定是问题的全局最优解。若以每类工种视为一整体,则解空间容量为 $N!$,此时较大批量占有机器会使前方机器长时间处于“饥渴”状态,从而降低工作效率,并加大后方工件的等待时间。

由于设置时间与排序有关,最优排序的搜索空间又与批量分割有关,确定批量分割和排序的最优组合就成为解决问题的关键。本节中,我们要来设计求解这类问题的混合策略。

8.4.3.2 混合优化策略的设计

1. 编码策略

借鉴压缩技术,可设计一种批量大小和排序的动态联合策略,以动态缩小搜索范围,即把同类工件合并为一个子批量(以下简称 JLS 操作)。譬如,3 类工种 A、B、C,各含 4 个工件,假定初始排序为 $A(1)B(1)C(1)C(1)B(1)B(1)C(1)C(1)B(1)A(1)A(1)A(1)$,其子批量数为 12,经 JLS 操作后的排序为 $ABC(2)B(2)C(2)BA(3)$,子批量数为 7,上述排序中括号内的数字表示各子批量所含的工件数。

JLS 操作本质上是合并操作。因此,在算法初始阶段以每个工件作为一个子批量,并在相应的解空间中产生初始种群。在进化过程中,一旦确定相应子批量数下的最优排列,就利用 JLS 操作来减小批量数,搜索空间也作相应变化。

2. 保优技术

JLS 操作对解空间的动态缩小,势必导致一些状态的不可返回。因此,可采用保优技术,随时保留搜索进程中发现的最佳状态。

3. 重升温技术

由于传统 SA 只有退温过程,高温时能够避免陷入局部极小,但低温时其突跳概率几乎为零。为了配合 JLS 操作,改善各阶段的搜索能力,可采用动态调温技术,在保留常规 SA 降温操作的同时,在每次 JLS 操作后适当提高温度。

4. 交叉操作

交叉操作采用 ABEL 操作,以快速产生后代。假定父串分别为 a, b ,则子串 c, d 分别为 $c[i]=a[b[i]], d[i]=b[a[i]], i=1, 2, \dots, n$ 。

5. 变异和 SA 状态产生函数

均采用互换操作。

6. 双阈值准则

为了使算法具有较好的优化性能,设置如下双阈值准则:

(1) 利用阈值 STEP1 确定各子批量数下算法性能的改变情况,即若连续 STEP1 代进化不能改善性能,则认为已得到该子批量数下的最优解。

(2) 利用阈值 STEP2 确定 JLS 操作引起算法性能的变化,即若连续 STEP2 次 JLS 操作没有改善优化性能,则认为已得到问题的最优解,终止算法。

7. 混合优化算法[一类批量可变调度问题的混合优化策略,简称 GSJLS]

(1) 算法初始化:给定初温 t_0 、子批量数、种群数 P_{size} 、阈值 STEP1 和 STEP2,令 $t=t_0, q=0, opt1=opt2=\infty$ 。

(2) 随机初始化种群,产生 P_{size} 个 $1 \sim K$ 的排列(K 为子批量数),令 $p=0$ 。

(3) 选择种群中的两个最优个体以交叉概率进行交叉,产生两个子个体,保留四者中的两个优良个体。

(4) 对种群中的所有个体以变异概率进行变异。

(5) 对种群中的各个体进行 SA 操作:

(5.1) 由状态产生函数产生新个体并确定其目标值;

(5.2) 计算新、旧个体的目标函数差值 Δ ;

(5.3) 以概率 $\min[1, \exp(-\Delta/t)]$ 接受新个体作为当前个体。

(6) 确定种群中的最优解,将其存入 $opt3$ 。

(7) 若 $opt1 \leq opt3$,则令 $p=p+1$;否则,令 $opt1=opt3, p=0$ 。

(8) 若 $p < \text{STEP1}$,则 $t=\lambda \cdot t$ 并转(3);否则,转(9)。

(9) 若 $opt2 \leq opt1$,则令 $q=q+1$;否则,令 $opt2=opt1, q=0$ 。

(10) 若 $q < \text{STEP2}$,进行 JLS 操作,并确定新子批量数,令 $t=t_0 \cdot \lambda^q$ 进行升温,然后转(2);否则,转(11)。

(11) 输出最优解 $opt2$,计算时间,子批量数以及各子批量的排列。

通过分析可知,上述算法具有如下的优点:

(1) JLS 操作降低子批量总数,可动态缩小搜索范围,提高搜索效率;

(2) GA 群体并行搜索和 SA 概率突跳相结合,可加强搜索能力;

(3) 保优操作避免遗失优化中已搜索到的优良解;

(4) 重升温操作增大算法在批量数发生变化后的概率突跳能力;

(5) 双阈值策略使算法具有较好的优化性能。

8.4.3.3 仿真结果和分析

针对 5 类工种、5 台机器的问题,假定各工种均包含 10 个工件,加工时间为 $[0, 1, 1.1]$ 的随机数,设置时间为 $[5, 25]$ 的随机数且与排序有关,运输时间取为 $[5, 10]$ 的随机数。为了研究问题对批量和排列的联合依赖性,将本文给出的算法与以下两种方案作比较:

方案 1(简称 GSNJLS1): 每类工种视为一批量,搜索空间为 $N!$;

方案 2(简称 GSNJLS2): 各工件视为一批量,搜索空间为 $(N \times L)!$ 。

算法参数选取如下:种群数 12;交叉概率 0.95;变异概率 0.01;初温 10;退温速率 0.9;STEP1 取 100;STEP2 取 20;GSNJLS1 和 GSNJLS2 策略中阈值分别取为 100 和 500。

利用 Pentium586/120 计算机,以 Borland C++ 为环境,在上述参数下对各算法作 20 次随机仿真,仿真统计结果见表 8.4.6。

表 8.4.6 各算法仿真结果统计

算法	最优值	波动率(%)	平均搜索时间(s)	最优子批量数
GSJLS	140.32	5.92	7.11	13
GSNJLS1	153.25	0	0.11	5
GSNJLS2	193.79	27.55	18.98	50

由仿真结果,可得到如下的一些结论:

GSNJLS1 策略在较小批量数对应的解空间中优化,尽管能够快速搜索到该批量数下的最优排列,但优化性能很差。原因是,子批量过大的工种占有机器,既导致前方机器长时间处于“饥渴”状态,也导致后方工件的长时间等待。因此,所得到的解并不是原问题的最优解。

GSNJLS2 策略在较大批量数对应的解空间中优化,搜索时间长,且优化性能差。原因是,子批量数过多导致过多的加工设置和运输时间。因此,得到的解也不是原问题的最优解。

混合算法(GSJLS)联合考虑问题中的批量和排列,具有最佳优化性能,且时间性能和波动性能也较满意,可以认为是处理这类对批量分割与排序同时存在依赖性的问题的可行和有效算法。

8.5 基于混合策略的神经网络权值学习研究

鉴于 GA,SA 和 TS 的全局优化特性和通用性,即优化过程无需导数信息,我们将其采用作为子算法。进而,基于实数编码构造 BPSA,GASA 和 GATS 混合学习策略,以提高前向网络学习的速度、精度和初值鲁棒性,特别是避免陷入局部极小的能力。

8.5.1 BPSA 混合学习策略

对于 BP 算法,学习缓慢的原因是优化曲面上存在局部极小和平坦区。SA 在搜索过程中基于概率突跳性能够避免局部极小,可最终趋于全局最优。基此,在 BPSA 混合学习策略中,采用以 BP 为主框架,并在学习过程中引入 SA 策略。这样做,既利用了基于梯度下降的有指导学习来提高局部搜索性能,也利用了 SA 的概率突跳性来实现最终的全局收敛性,从而可提高学习速度和精度。

BPSA 混合学习策略的算法步骤如下:

- (1) 随机产生初始权值 $w(0)$, 确定初温 t_1 , 令 $k=1$ 。
- (2) “细调”, 即利用 BP 计算 $w(k)$, $w(k)=w(k-1)-\alpha\partial E/\partial w$ 。
- (3) “粗调”, 即利用 SA 进行搜索:
 - (3.1) 利用 SA 状态产生函数产生新权值 $w'(k)$, $w'(k)=w(k)+\eta$, 其中 $\eta\in(-1,1)$ 为随机扰动。
 - (3.2) 计算 $w'(k)$ 的目标函数值与 $w(k)$ 的目标函数值之差 ΔC 。
 - (3.3) 计算接受概率 $P_r=\min[1,\exp(-\Delta C/t_k)]$ 。
 - (3.4) 若 $P_r>\text{random}[0,1)$, 则取 $w(k)=w'(k)$; 否则, $w(k)$ 保持不变。
 - (4) 利用退温函数 $t_{k+1}=vt_k$ 进行退温, 其中 $v\in(0,1)$ 为退温速率。
 - (5) 若 $w(k)$ 对应的目标值满足要求精度 ϵ , 则终止算法并输出结果; 否则, 令 $k=k+1$, 转步骤(2)。

8.5.2 GASA 混合学习策略

上述 BPSA 混合策略利用 SA 来实现全局优化,但优化过程是串行的,需要大量复杂繁琐的梯度计算。因此,这里我们利用 GA 提供并行搜索主框架,再结合遗传群体进化和 SA 概率突跳搜索,以多点并行化无导数搜索来实现全局优化,以此来解决传统算法中导数依赖性的弱点。

算法步骤如下:

- (1) 随机产生初始种群 P_0 , 确定初温 t_0 , 令 $k=0$ 。
- (2) 对 P_k 中各个体进行 SA 搜索:
 - (2.1) 利用 SA 状态产生函数产生新个体(同 BPSA)。
 - (2.2) 计算新、旧个体的目标函数值之差 ΔC 。
 - (2.3) 计算接受概率 $P_r=\min[1,\exp(-\Delta C/t_k)]$ 。
 - (2.4) 若 $P_r>\text{random}[0,1)$, 用新个体取代旧个体; 否则, 旧个体不变。
- (3) 以交叉概率对候选种群中目标值最小的两个体进行交叉, 产生两个新个体, 采用

2/4 优选原则确定后代。

(4) 以变异概率对候选种群中各个体进行变异,采用择优原则确定后代。至此产生下一代种群 P_{k+1} 。

(5) 利用退温函数进行退温(同 BPSA)。

(6) 若目标值满足要求精度 ϵ ,则终止算法并输出结果;否则,令 $k = k + 1$,转步骤(2)。

8.5.3 GATS 混合学习策略

TS 是区别于 SA 的另一种串行优化算法。其机制是,通过设置近期操作的存储结构(禁忌表,Tabu list),以当前解邻域中未在禁忌表中出现或满足藐视准则(aspiration criterion)的最佳状态来替换当前状态,而并不在乎其性能是否优于当前解,按此原则产生一定的突跳性以避免局部极小,同时也避免迂回搜索的出现。在此,我们利用 GA 提供并行搜索主框架,结合遗传群体进化和 TS 较强的具有避免迂回搜索能力的邻域搜索,实现快速全局优化。

算法步骤如下:

(1) 确定算法参数,初始化种群,并确定最优状态。

(2) 判断目标值满足要求精度 ϵ 否? 若满足,则终止算法并输出结果;否则,继续以下步骤。

(3) 基于当前种群进行遗传选择操作。

(4) 进行交叉操作,保留优良个体并及时更新最优状态。

(5) 对各个体进行禁忌搜索:

(5.1) 设置一个性能极差的临时状态。

(5.2) 判断 TS 邻域搜索次数是否满足? 若满足,则以当前临时状态替换当前个体;否则,继续以下步骤。

(5.3) 由当前个体在其邻域中产生新状态,计算其评价值。

(5.4) 判断新状态满足藐视准则否? 若成立,则更新当前最优状态,并转(5.6);否则,继续以下步骤。

(5.5) 判断新状态满足禁忌准则否? 若禁忌表中已存在该状态,则转(5.2);否则,转入步骤(5.6)。

(5.6) 对禁忌表进行 FIFO 处理,移去最先进入表中的状态并将新状态加入禁忌表,然后判断新状态是否优于临时状态? 若是,则以新状态替换临时状态;否则,保持临时状态不变。

(6) 以新的种群返回步骤(2)。

8.5.4 编码和优化操作设计

FNN 的学习中,变量多、搜索空间大、优化曲面复杂,与传统的优化问题存在较大的差异。为了实现上述算法的高效优化性能,需要对算法操作和编码进行设计。

(1) 编码:对多变量优化问题二进制编码会导致很大的计算量和存储量,且串长影响算法精度。因此,在给定网络结构下,我们以一组权值表征 FNN,其中权值采用双精度实数编码。进而,作为优化过程中的个体,由一个表征 FNN 的权值矢量来表示,而种群又由若干个体所构成。

(2) 交叉操作:为了配合上述实数编码策略,采用算术交叉算子以快速产生后代个体。

(3) 邻域搜索(变异、SA 状态产生函数和 TS 禁忌搜索):变异、SA 状态产生函数和 TS 禁忌搜索,目的均是基于邻域搜索增加种群的多样性,实现状态转移(包括局部趋化和劣向移动)。在上述实数编码的基础上,邻域搜索采用附加扰动的方式,即为 $x' = x + \eta$, $\eta \in (-1, 1)$ 。

(4) 禁忌准则:禁忌准则是使 TS 避免迂回搜索现象的关键环节。在有限状态空间的组合优化中,对禁忌准则的判断归结为对新状态与禁忌表中各状态严格相同性的判断。显然,这种做法难以应用到高维无限实数空间的搜索中。为此,采用如下二重准则,作为状态禁忌的近似判断准则:

① 对新状态的目标值与禁忌表各状态的目标值进行判断:若相对偏差均大于禁忌阈值,则对禁忌表作 FIFO 处理,把新状态加入禁忌表;否则,进行步骤(2)的判断。

② 对新状态的各状态分量与禁忌表中所有状态的各状态分量进行判断:若禁忌表中所有状态均至少存在某个分量与新状态相应分量的相对偏差大于禁忌阈值,则对禁忌表作 FIFO 处理,把新状态加入禁忌表;否则,认为新状态在禁忌表中已出现,禁止作为当前状态。

(5) 藐视准则:上述禁忌准则的近似性会造成若干优良状态(指优于至今所搜索到的最优状态)被禁忌。为避免上述现象的发生,在算法中先于禁忌准则判断进行如下的藐视准则判断:若新状态的目标值优于搜索过程至今所得的最优状态的目标值,则跳过禁忌判断,直接对禁忌表作 FIFO 处理,把新状态加入禁忌表;否则,进行禁忌准则判断。

8.5.5 仿真结果与分析

异或(XOR)问题已成为神经网络学习算法研究的典型算例。所谓异或问题,即以 $\{(0,0), (0,1), (1,0), (1,1)\}$ 为样本的线性不可分的最简单算例,相应的优化曲面不规则且存在多个局部极小。这一节中,我们以此作为仿真算例。

在以下的研究中,约定网络结构是固定的,即为 2-3-1 网络结构(输入层、隐层和输出层节点数分别为 2,3,1),阈值单元、激励函数和目标函数与 BP 算法相同。

我们采用 Borland C++ 语言编制仿真程序,在 16MB RAM 的 Pentium 586/120 计算机上,对如下的几个方面进行了研究:

(1) 对混合策略 BPSA, GASA 和 GATS,以及对比性算法 BP 和 BPM(带动量项的改进 BP)的研究,其中以 $(-1,1)$ 为初始权值范围:

- 算法实现不同规定精度 10^{-n} (n 称为精度指数)的进化代数和实际学习误差比较研究,表 8.5.1 给出了 50 次随机实验的统计性能。

表 8.5.1(a) GATS, GASA 和 BPSA 的性能统计,初值范围 $(-1,1)$, 50 次随机实验

精度指数	GATS		GASA		BPSA	
	平均	平均	平均	平均	平均	平均
	迭代代数	实际误差	迭代代数	实际误差	迭代代数	实际误差
1	14.28	$8.455177e-2$	122.76	$8.732457e-2$	71.17	$9.370065e-2$
2	21.36	$7.679245e-3$	190.94	$8.600268e-3$	125.22	$8.954079e-3$
3	26.98	$7.536829e-4$	280.1	$8.350415e-4$	190.18	$7.809259e-4$
4	30.88	$7.377861e-5$	335.88	$8.295012e-5$	236.74	$7.806090e-5$
5	37.26	$7.162009e-6$	381.7	$8.333022e-6$	293.65	$7.473826e-6$
6	42.32	$7.648741e-7$	437.98	$8.492219e-7$	370.93	$7.389303e-7$
10	57.86	$6.765379e-11$	632.34	$8.000656e-11$	537.24	$7.150198e-11$
15	80.7	$7.281993e-16$	834.7	$8.075591e-16$	655.60	$7.919644e-16$
20	97.5	$7.454220e-21$	1007.78	$8.327293e-21$	778.65	$7.541186e-21$
25	117.38	$6.900844e-26$	1168.74	$8.164772e-26$	952.47	$6.891083e-26$

表 8.5.1(b) BP 和 BPM 的性能统计,初值范围取 $(-1,1)$, 50 次随机实验

精度指数	BP		BPM	
	平均迭代代数	平均实际误差	平均迭代代数	平均实际误差
1	616.46	$9.935016e-2$	461.72	$9.791899e-2$
2	1043	$9.986718e-3$	583.74	$9.957603e-3$
3	4008.46	$9.998483e-4$	1471.94	$9.993885e-4$
4	30232.96	$9.999810e-5$	9338.84	$9.999379e-5$
5	279769.4	$9.999978e-6$	84019.72	$9.999942e-6$
6			802667.1	$9.999993e-7$

- 算法实现不同规定精度的时间性能比较研究,如图 8.5.1 所示。
- 优化过程中实际误差下降情况的比较研究,如图 8.5.2 所示。

(2) 对算法相对于初始权值的鲁棒性的研究

- 初值范围取 $(-0.1, 0.1)$ 时,各算法实现规定精度的最大、平均和最小迭代代数比较研究(50次随机仿真),如图8.5.3、图8.5.5和图8.5.7所示。
- 分别在 $(-0.1, 0.1)$ 与 $(-1, 1)$ 各取50组不同初值,各算法实现规定精度的平均迭代步数比较研究,如图8.5.4、图8.5.6和图8.5.8所示。

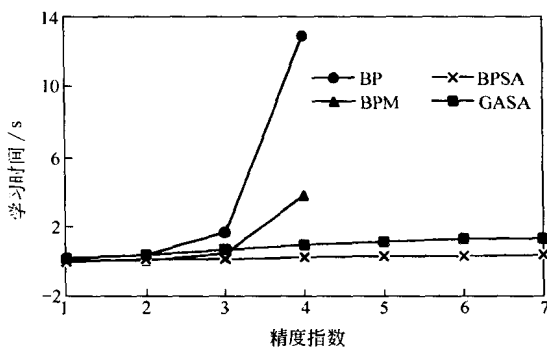


图 8.5.1 精度对算法学习时间的影响

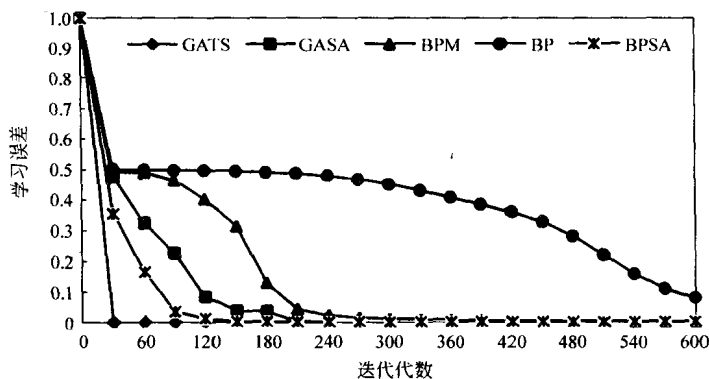


图 8.5.2 各算法实际学习误差的下降曲线

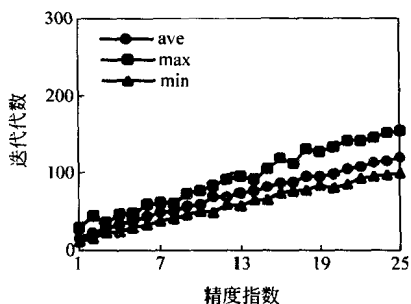


图 8.5.3 相同初值范围中不同初值和精度对 GATS 迭代代数的影响

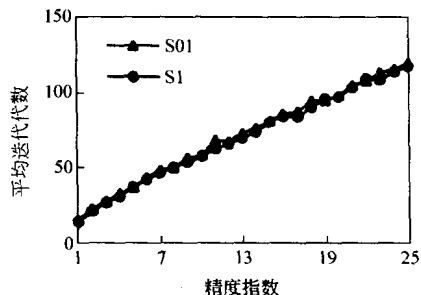


图 8.5.4 不同初值范围和精度对 GATS 平均迭代代数的影响

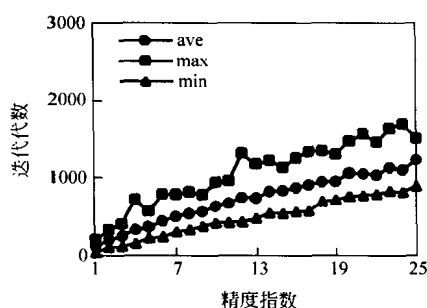


图 8.5.5 相同初值范围中不同初值和精度对 GASA 迭代代数的影响

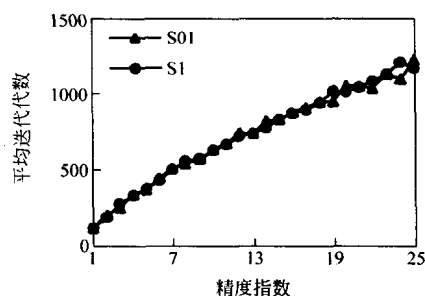


图 8.5.6 不同初值范围和精度对 GASA 平均迭代代数的影响

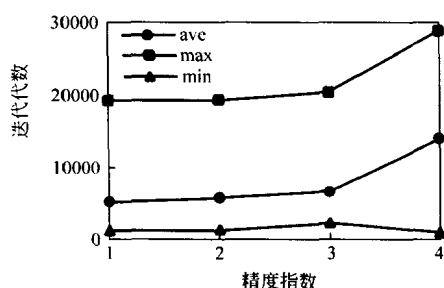


图 8.5.7 相同初值范围中不同初值和精度对 BPM 迭代代数的影响

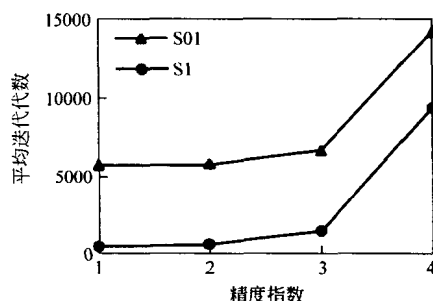


图 8.5.8 不同初值范围和精度对 BPM 平均迭代代数的影响

注：图 8.5.3、图 8.5.5 和图 8.5.7 中初值范围均为 $(-0.1, 0.1)$ ；

图 8.5.4、图 8.5.6 和图 8.5.8 中 S01 和 S1 分别表示初值范围 $(-0.1, 0.1)$ 和 $(-1, 1)$ 。

算法的参数选取如下：学习率取 0.8，BPM 的惯性系数取 0.7，初温取 10，退温速率取 0.85，种群数取 6，交叉概率取 0.95，变异概率取 0.01，禁忌表长取 5，邻域搜索次数取 20，禁忌阈值取 0.01%，邻域扰动采用均匀分布机制。

上述仿真研究表明，BP 学习精度和速度很差，在实现有限精度后学习时间随精度提高而大幅度增加；BPM 尽管性能较 BP 有改善，但仍难以实现学习的高精度和快速性，特别是两者学习性能对初值有很强的依赖性。对混合学习策略，其性能优越性可归纳如下：

- (1) 混合策略的学习精度大幅度提高，在相同要求精度下实际学习误差明显减小。
- (2) 混合学习策略的学习速度大幅度提高，进化代数随精度提高而增加的幅度不大。
- (3) 混合学习策略对初始权值的鲁棒性大幅度提高。具体表现为，算法性能对相同和不同范围内的不同初值均具有较好的鲁棒性。

(4) GASA 和 GATS 混合策略无需导数信息，从而避免了 BP 反传运算的复杂性，节省了大量的计算量和存储量，并可将优化指标函数推广到复杂且不可微情况。

此外，若把隐层数和隐节点数也作为优化变量，则上述算法可实现网络结构和权值的

同步学习。此外,为了取得更佳的学习效果,算法中的邻域扰动机制可采用更高效的方式,如柯西分布、高斯分布等。

8.6 基于混合策略的神经网络结构学习研究

由于神经网络的多模型性和结构变化的不连续性,网络的结构设计一直是神经网络研究中的一个开问题。本节针对一类径向基函数(RBF)网络的结构设计问题,通过将其转化为一类组合优化问题,进而设计有效的兼有结构与权值双重学习的混合优化策略,以及相应的优化操作。

8.6.1 RBF 网络简介

RBF 网络是一类典型的三层前向神经网络,其拓扑结构如图 8.6.1 所示。

假定网络有 N 个隐单元(不包含阈值单元)和 M 个输出单元,则网络能够实现如下输入输出映射关系:

$$y_i(\mathbf{X}) = w_{0,i} + \sum_{j=1}^N w_{j,i} \Phi(\|\mathbf{X} - c_j\|/\sigma_j), \quad i = 1, 2, \dots, M \quad (8.6.1)$$

其中, $\mathbf{X}$ 为输入向量, y_i 为第 i 个输出单元对应 $\mathbf{X}$ 的输出值, $w_{0,i}$ 为第 i 个输出单元的阈值, $w_{j,i}$ 为第 j 个隐单元到第 i 个输出单元的权重, $\|\cdot\|$ 为欧氏范数, $\Phi(\cdot)$ 为径向基函数, c_j 为第 j 个隐单元的中心, σ_j 为规划因子。

由于高斯函数的可分解性及其与视神经感受野形状的相似性,通常将径向基函数取为高斯函数 $\Phi(x) = \exp(-x^2/\sigma)$ 。研究表明(Park et al, 1991),规划因子的相同性并不影响网络的逼近能力,而中心的选择对网络逼近能力有很大影响。因此,本节主要研究隐层的中心选择问题。

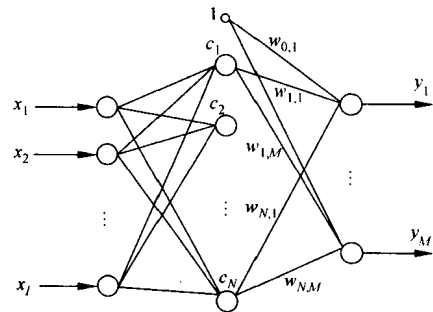


图 8.6.1 径向基函数网络结构图

8.6.2 RBF 网络结构优化的编码和操作设计

目前,RBF 网络的学习算法主要有 Moody 法(Moody et al, 1989)、正交最小二乘法(Chen et al, 1991)、Givens 法(Chen et al, 1992)等。这些方法的缺点,除难以确定隐单元数外,或是中心选择不好,或是算法限制条件苛刻,或是网络推广性较差。在此,我们利

用混合策略来设计网络结构。首先,给出编码和优化操作等环节的设计策略。

(1) 编码策略设计。通常网络的中心选择限于样本集,即选取样本的输入向量作为隐单元的中心,从而缩小搜索空间。因此,我们提出如下编码策略:

首先,将中心集定义为训练样本和检验样本的并集,并将所有样本由 1 至 K 依次编号,其中 K 为样本总数。

其次,令种群中个体的长度为隐单元总数(不包括阈值单元),其中各基因表示一个样本编号,相应样本的输入向量作为该隐单元的中心。为避免节点冗余,编码要求个体中各基因互异。

例如,若中心集中样本总数为 10,则个体 $p=[1,8,10,3,7,4]$ 表示网络采用 6 个隐单元,其中心分别为中心集中的第 1,8,10,3,7 和 4 号样本的输入向量。

可见,这种编码策略将结构优化转化为组合优化问题,从而适合于采用 GASA 混合策略进行优化。

(2) 交叉操作设计。鉴于上述编码的特殊性,我们设计如下的简单且易实现的交叉操作:

首先,对两个父代个体 p_1 和 p_2 随机产生交叉始位置 b_1, b_2 和交叉末位置 e_1, e_2 。

其后,令后代保留父代个体交叉始位置前和交叉末位置后的基因串,用“ $p_2(p_1)$ 的两交叉位置间且异于 $p_1(p_2)$ 各基因的”基因替代 $p_1(p_2)$ 的两交叉位置间的基因串并传给相应的后代。

例如,若父代个体为 $p_1=[1,8,10,3,7,4]$, $p_2=[6,1,9,5,4,2,3]$,交叉位置为 $b_1=2, b_2=3, e_1=4, e_2=5$,则后代为 $q_1=[1,9,5,7,4]$, $q_2=[6,1,8,10,2,3]$ 。

显然,这种交叉算子对父代基因进行重组时能保证后代的合法性,且能动态改变染色体长度来改变隐单元数目。

(3) 变异和 SA 操作设计。变异和 SA 状态产生函数可设计如下:

在补元集(即异于个体中所有基因的编号集合)中,随机选择一个元素替代个体中的一个随机位置上的基因。

例如,若个体为 $p=[1,8,10,3,7,4]$,则补元集为 $\{2,5,6,9\}$,若随机选择的补元元素为 2,随机替代位置为 4,则新个体为 $q=[1,8,10,2,7,4]$ 。

这种简单操作能够引入新的模式,且能保证后代的合法性。此外,SA 以概率 $\min\{1, \exp[(E-E')/t]\}$ 接受新状态,并采用指数退温。

(4) 增添和删除算子设计。为增加结构优化时的灵活性,我们在算法中添加了增添和删除操作,具体设计如下:

① 增添算子:随机选择 $[0, K-L_p]$ 中的整数 L_a 作为增添长度,其中 L_p 为个体长度,在个体尾部随机增添 L_a 个互异补元,以产生新个体。

显然,增添算子能够引入新的中心模式,是一种增长长度的变异操作。

② 删除算子:随机选择 $(0, L_p)$ 中的整数 L_d 作为删除长度,在个体尾部删除 L_d 个基

因,以产生新个体。

考虑到在具有相同逼近精度的网络中小型网络的推广性能较好,因此,算法中设置删除概率比增添概率大。

此外,为避免遗失优良解,对上述所有算子均实施保优操作。

(5) 目标函数选取。熟知,基于训练误差的单目标学习常会出现过拟合现象,影响网络的推广性。因此,我们采用如下的目标函数选取策略:

首先,用训练样本确定网络,并通过检验样本进行检验。

其次,采用以两种误差的综合指标均方值作为网络性能的评价目标。

最后,在每代进化后交换训练样本集和检验样本集。

这种策略的特点是,不仅能产生较好地逼近训练样本,而且能使网络具有较好的推广性,可在一定程度上消除两组样本中随机噪声的影响。

(6) 权重学习算法设计。鉴于最小二乘法中矩阵伪逆运算的复杂性,我们采用梯度下降法按给定步数优化权值,训练误差和权值迭代式如下:

$$e_{tr} = \frac{1}{2} \sum_{i=1}^P \sum_{j=1}^M (y_{i,j} - \tilde{y}_{i,j})^2, \quad \tilde{y}_{i,j} = \sum_{l=0}^N h_{i,l} w_{l,j} \quad (8.6.2)$$

$$\frac{\partial e_{tr}}{\partial w_{l,j}} = - \sum_{i=1}^P (y_{i,j} - \tilde{y}_{i,j}) h_{i,l} w_{l,j} = w_{l,j} - \alpha \frac{\partial e_{tr}}{\partial w_{l,j}} \quad (8.6.3)$$

其中, P 为训练样本数, M 为输出维数, N 为隐单元数, α 为学习率, $\tilde{y}$ 和 y 分别为实际和期望输出, $h_{i,j}$ 为第 i 层第 j 单元的输。

(7) 算法终止准则设计。算法终止准则要兼顾算法的学习精度和速度,在此采用以最优指标连续 S 代学习仍保持不变作为准则,即

算法初始化时,置最优指标为群体中最优个体的综合指标,并置终止计数变量 m 为0。

若每代进化中最优指标发生改变,则及时更新最优指标,并置 m 为0;否则,令 $m = m + 1$,一旦 $m = S$ 就结束算法。

8.6.3 RBF网络结构的混合优化策略

通过对编码策略和优化操作的设计,下面我们来给出网络结构混合优化的具体步骤。

(1) 确定输入和输出层节点数,对中心集样本编号,确定算法参数。

(2) 初始化种群:从中心集中随机选择一定数目的样本构成个体,样本数作为隐单元数,对应输入向量作为隐单元中心。

(3) 以不同的随机初值重复执行下列步骤若干次:利用梯度法以训练样本对各个体进行规定步数的学习,确定网络权值,并计算训练误差;采用检验样本对网络进行检验,得到检验误差;确定各个体的综合误差和种群中的最优个体。

(4) 判断是否满足算法终止准则? 若是, 则转步骤(11); 否则, 转步骤(5)。

(5) 在种群中随机选择个体与最优个体以交叉概率进行交叉, 得到两个新个体, 并用步骤(3)方法进行评价, 然后用 2/4 原则进行保优。

(6) 对各个体以变异概率进行变异操作。

(7) 对各个体在当前温度下进行模拟退火操作。

(8) 对各个体以增添概率增添随机数目的隐单元。

(9) 对各个体以删除概率删除随机数目的隐单元。

(10) 确定最优个体并进行降温, 交换训练集和检验集, 然后转步骤(4)。

(11) 输出最佳网络结构(包括隐单元数、中心、权值和性能指标)。

通过归纳可知, 上述算法具有以下的特点:

(1) 编码方式特殊, 有利于随机算法搜索。

(2) 梯度法计算权值简单, 通过多次随机运行可在一定程度上消除初值的影响。

(3) GA 和 SA 相结合, 能够改善优化效率, 并增强避免陷入局部极小的能力。

(4) 增添和删除操作的引入, 可增加结构搜索的灵活性。

(5) 综合指标的优化保证了网络具有较好的逼近能力和推广性。

(6) 训练集和检验集的交替学习, 能在一定程度上克服过拟合现象和消除样本中随机噪声的影响。

此外, 若主要关注给定精度下的最优网络结构, 即综合隐单元数最少的网络, 则只需以节点数为指标, 按精度要求进行权值计算, 并优选节点数和中心。上述算法中, 结构和算子无需多大变化。

8.6.4 计算机仿真与分析

为了考察所提出的算法的性能, 我们采用下式函数产生无噪声样本, 然后对采样数据叠加一定的噪声, 有

$$f(x) = 0.01 \times \{1/[(x-0.3)^2 + 0.01] + 1/[(x-0.9)^2 + 0.04] - 6\} \quad (8.6.4)$$

中心集采用如下的两个集合的并集:

(1) A 样本集: 21 个点, 样本输入值由 0 至 2 每隔 0.01 取一点, 各样本输出均附加 $\pm 5\%$ 内的随机噪声。

(2) B 样本集: 20 个点, 样本输入值由 0.05 至 1.95 每隔 0.01 取一点, 各样本输出均附加 $\pm 5\%$ 内的随机噪声。

在考察上述算法性能的同时, 取以下的四种方案进行对比。

方案 1: 分别以 A, B 集进行训练和检验, 仅以训练误差作为指标, 训练和检验集在学习过程中不交换。

方案 2: 分别以 B, A 集进行训练和检验, 仅以训练误差作为指标, 训练和检验集在学习过程中不交换。

方案 3: A, B 集共同作为训练集。

方案 4 (即上述混合算法): 采用训练和检验误差的综合指标, 训练集和检验集在学习过程中交换。

作为第 i 次仿真所得网络对带噪声样本的综合误差 e_i 和对无噪声样本的综合误差 E_i , 分别定义为

$$e_i = 0.5 \times \sqrt{\left\{ \sum_{x \in Tr} [f_i(x) - y(x)]^2 \right\}^2 + \left\{ \sum_{x \in Ts} [f_i(x) - y(x)]^2 \right\}^2} \quad (8.6.5)$$

$$E_i = 0.5 \times \sqrt{\left\{ \sum_{x \in Tr} [f_i(x) - \hat{f}(x)]^2 \right\}^2 + \left\{ \sum_{x \in Ts} [f_i(x) - \hat{f}(x)]^2 \right\}^2} \quad (8.6.6)$$

其中, n_i 为第 i 次仿真所得隐节点数, Tr 和 Ts 分别为训练样本集和检验样本集, $f_i(x)$ 为第 i 次所得网络对应输入 x 的输出, $\hat{f}(x)$ 为对应 x 的无噪声样本输出, $y(x)$ 为对应 x 的有噪声样本输出。

采用 16MB RAM 的 Pentium 586/133 计算机, 以 Borland C++ 为仿真环境, 对上述各方案均随机运行 10 次。算法参数选取如下: 种群数 20, 交叉和变异概率分别取 0.99 和 0.9, 增添和删除概率分别取 0.1 和 0.2, 退温速率 0.8, 初温 1.0, 学习率 0.05, 规划因子 0.02, 梯度计算步数 500 并连续随机运行 4 次, 算法终止步数取为 5。仿真统计性能见表 8.6.1 和图 8.6.2。

表 8.6.1 各方案性能比较

方案	$\bar{N}$	$\bar{e}$	$\bar{E}$	$\bar{t}/s$
方案 1	36.8	0.02777	0.01392	247.72
方案 2	30	0.03409	0.02642	267.22
方案 3	36	0.02026	0.01098	743.88
方案 4	17.4	0.01833	0.01082	394.79

注: 表中 $\bar{N}$ 为平均隐节点数, $\bar{e}$ 为有噪声样本的平均综合逼近误差, $\bar{E}$ 为无噪声样本的平均综合逼近误差, $\bar{t}$ 为平均计算时间。

仿真结果表明, 本节中所提出的混合算法即方案 4, 具有以下优点:

- (1) 所得网络的隐节点少, 对硬件设计有利, 同时推广性好;
- (2) 多目标优化, 使得网络对学习样本逼近总误差小;
- (3) 两组样本集的交替使网络具有一定的抗噪声能力, 对无噪声样本的逼近总误差小。

仿真结果也表明, 方案 1 和 2 基本上是对训练样本的逼近; 方案 3 利用大样本集训

练,计算量大,优化过程缓慢。因此,本节所提出的混合优化策略是可行和有效的。

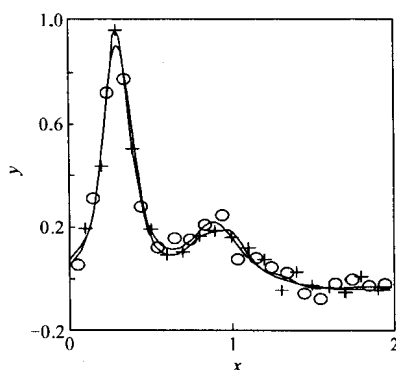


图 8.6.2 混合算法的逼近效果(“+”表示 A 集点,“o”表示 B 集点)

8.7 基于混合策略的光学仪器设计研究

8.7.1 引言

在激光驱动的惯性约束核聚变(ICF)中,要求激光束具有极高的均匀性,即所谓的“平顶”光束。而大功率激光器的输出光束往往存在波面畸变,其振幅与位相都带有噪声。为提高聚焦系统的抗近场噪声能力以实现均匀照明,一般采用阵列型结构的聚焦系统,如图 8.7.1 所示。入射光束被分成多个子束分别处理,在每个小单元中近似认为入射波面是均匀的,最后由主聚焦透镜将各子束在焦面叠加。典型的技术方案,如随机位相板、透镜阵列等,虽提高了焦斑的均匀性,但由于硬边衍射的影响,并不能得到令人满意的强度包络。为消除硬边衍射对焦斑均匀性的影响,阵列单元需具有特殊的非球面位相分布,而用传统光学器件很难实现这种特殊功能(郑学哲, 1997)。

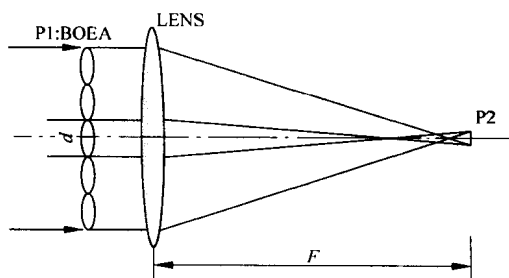


图 8.7.1 阵列型 ICF 均匀照明系统原理图

近年来,二元光学技术发展迅速,尤其在波面整形方面表现出极大的优越性,被认为是很有希望实现 ICF 均匀照明的新技术途径。鉴于均匀照明与位相恢复的相似性,通常采用迭代优化方法来设计二元光学器件的位相结构,如较早提出的 KPP, DPP 以及 GS 及其改进法,可近似达到 8 阶超高斯分布。考虑离散计算时采样造成的影响,基于 Y-G 算法的设计可进一步提高焦斑边缘的陡度,但其顶部的均匀性还不能满足物理实验的要求。此外,这些优化算法都由目标函数极值点处导数为零而得到基本迭代公式,优化过程只接受目标函数下降的自变量取值,因而优化结果依赖于初值且易陷于局部极小是普遍存在的问题。尽管结合几何变换等方法提供的初始解可改善优化结果,但尚未能找到产生足够均匀焦斑的器件位相分布。

模拟退火和遗传算法是全局概率型优化算法,搜索过程不依赖问题的特殊信息,可以避免上述算法的弱点。为了尽可能得到近“平顶”的焦斑强度包络,本节以 GASA 混合算法设计 ICF 均匀照明系统中二元阵列单元的位相分布,探讨混合算法在工程中的应用潜力和有效性(郑学哲等, 1998; Zhai et al, 1998)。

8.7.2 模型设计

为简化分析,取图 8.7.1 的中心阵列单元。考虑圆对称情况,由菲涅尔积分公式可得焦面的输出光场为

$$U_2(r) = \frac{ik_0}{f} \exp\left[-ik_0\left(f + \frac{r^2}{2f}\right)\right] \int_0^R J_0\left(\frac{k_0\rho r}{f}\right) \exp(i\varphi(\rho)) \rho d\rho \quad (8.7.1)$$

其中, f 为透镜焦距, $k_0 = 2\pi/\lambda$, R 为阵列单元口径, J_0 为零阶 Bessel 函数, φ 为阵列单元位相。

由于数值计算必须进行离散采样处理,因而最后设计得到的是由不同常数位相环带组成的器件。上述积分可改写为各个环带积分的和,即

$$U_2(r) = \frac{ik_0}{f} \exp\left[-ik_0\left(f + \frac{r^2}{2f}\right)\right] \sum_{n=1}^N e^{i\varphi_n} \int_{\rho_{n-1}}^{\rho_n} J_0\left(\frac{k_0\rho r}{f}\right) \rho d\rho \quad (8.7.2)$$

其中, N 为器件环带数, φ_n 为第 n 个环带的位相, ρ_{n-1} 为第 n 个环带的最小半径, ρ_n 为第 n 个环带的最大半径。

对 ICF 均匀照明问题只要求输出光强呈“平顶”分布,而对焦面位相分布无特殊要求,所以设计目标是求使下式中 F 最小的位相分布 φ_n :

$$F = \|U_2 - U_I\|^2 \quad (8.7.3)$$

其中, F 为实际输出焦斑复振幅分布 U_2 和理想焦斑分布 U_I 间的距离。即希望通过优化设计寻求特殊的位相结构,重新分配焦斑的强度分布,来抑制由阵列单元硬边衍射引起的焦斑强度调制。

8.7.3 仿真研究和设计结果

根据上述设计模型,利用提出的 GASA 混合算法来设计二元阵列单元的位相分布。设计参数为:单元口径 $d=30\text{mm}$,聚焦透镜焦距 $f=800\text{mm}$,波长 $\lambda=0.6328\mu\text{m}$,焦斑半径 $r_0=200\mu\text{m}$ 。由式(8.7.2)知,迭代过程中的变量只有 φ_n ,因而可以得到一个由不变量形成的系数矩阵,它与更新后位相的乘积就是新的输出光场分布。这可以大大缩短求适配值的时间,从而提高算法的效率。

定义顶部均匀性 rms 如下:

$$rms = \sqrt{\sum_{i=1}^n \left(\frac{I_i - \bar{I}}{\bar{I}} \right)^2 / (n-1)}, \quad \bar{I} = \sum_{i=1}^n I_i / n \quad (8.7.4)$$

其中, I_i 表示焦斑顶部各点光强。

考虑设计条件及计算效率,种群数取 2,两个初始状态分别取几何变换的结果和常数位相分布。优化过程得到的器件连续位相分布如图 8.7.2,焦斑强度分布如图 8.7.3 中曲线 2 所示。焦斑光强分布接近 16 阶超高斯分布 $I_0 \exp\left[-2\left(\frac{x}{w}\right)^{16}\right]$,顶部均匀性为 $rms=3.2\%$,能量利用率大于 90%。可见混合算法在二元光学器件设计中的有效性。

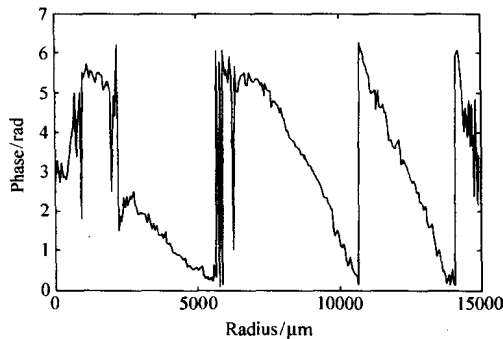


图 8.7.2 混合算法所得阵列单元位相分布单元

同样条件下,Y-G 算法优化所得的焦斑光场分布结果如图 8.7.3 中曲线 3 所示。虽然焦斑边缘更陡,但顶部均匀性明显不如混合算法结果。

此外,混合算法与模拟退火算法的性能比较也表明混合算法所得的焦斑比 SA 结果有更高的顶部均匀性和更小的旁瓣,如图 8.7.4 所示。

上述研究表明,基于混合算法的优化设计不仅很有效,且性能优于传统算法和单一 SA 算法。这个实际工程设计问题的成功解决,再一次表明了不同算法在优化结构和优化机制上的混合和相互补充能够表现出更有效的全局优化特性,同时也体现出混合算法

在工程中的应用潜力。

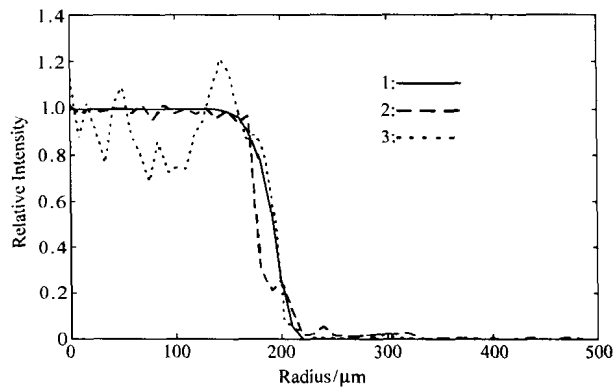


图 8.7.3 混合算法与 Y-G 算法的性能比较。曲线 1 表示理想 16 阶超高斯分布；2 表示混合算法优化所得焦斑光强分布；3 表示 Y-G 算法优化所得焦斑光强分布。

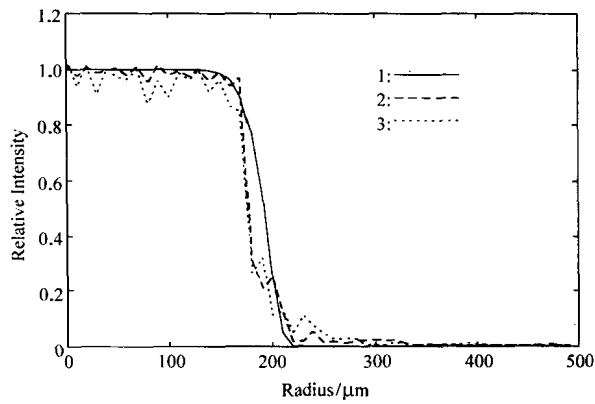


图 8.7.4 混合算法与 SA 的性能比较。曲线 1 表示理想 16 阶超高斯分布；2 表示混合算法优化所得焦斑光强分布；3 表示 SA 算法优化所得焦斑光强分布。

附录

Benchmark 问题

A: TSP Benchmark 问题

(1) 30 城市 TSP 问题 ($d^* = 423.741$ by D B Fogel)

41 94;37 84;54 67;25 62;7 64;2 99;68 58;71 44;54 62;83 69;64 60;18 54;22 60;
83 46;91 38;25 38;24 42;58 69;71 71;74 78;87 76;18 40;13 40;82 7;62 32;58 35;
45 21;41 26;44 35;4 50

(2) 50 城市 TSP 问题 ($d^* = 427.855$ by D B Fogel)

31 32;32 39;40 30;37 69;27 68;37 52;38 46;31 62;30 48;21 47;25 55;16 57;
17 63;42 41;17 33;25 32;5 64;8 52;12 42;7 38;5 25; 10 17;45 35;42 57;32 22;
27 23;56 37;52 41;49 49;58 48;57 58;39 10;46 10;59 15;51 21;48 28;52 33;
58 27;61 33;62 63;20 26;5 6;13 13;21 10;30 15;36 16;62 42;63 69;52 64;43 67

(3) 75 城市 TSP 问题 ($d = 549.18$ by D B Fogel)

48 21;52 26;55 50;50 50;41 46;51 42;55 45;38 33;33 34;45 35;40 37;50 30;
55 34;54 38;26 13;15 5;21 48;29 39;33 44;15 14;16 19;12 17;50 40;22 53;21 36;
20 30;26 29;40 20;36 26;62 48;67 41;62 35;65 27;62 24;55 20;35 51;30 50;
45 42;21 45;36 6;6 25;11 28;26 59;30 60;22 22;27 24;30 20;35 16;54 10;50 15;
44 13;35 60;40 60;40 66;31 76;47 66;50 70;57 72;55 65;2 38;7 43;9 56;15 56;
10 70;17 64;55 57;62 57;70 64;64 4;59 5;50 4;60 15;66 14;66 8;43 26

(4) Hopfield-10 城市 TSP 问题 ($d^* = 2.691$)

0.4 0.4439; 0.2439 0.1463; 0.1707 0.2293; 0.2293 0.761; 0.5171 0.9414;
 0.8732 0.6536; 0.6878 0.5219; 0.8488 0.3609; 0.6683 0.2536; 0.6195 0.2634

(5) Grotschel-442 城市 TSP 问题

20 40; 20 50; 20 60; 20 70; 20 80; 20 90; 20 100; 20 110; 20 120; 20 130; 20 140;
 20 150; 20 160; 20 170; 20 180; 20 190; 20 200; 20 210; 20 220; 20 230; 20 240;
 20 250; 20 260; 20 270; 20 280; 20 290; 20 300; 20 310; 30 320; 20 330; 20 340;
 20 350; 20 360; 30 40; 30 50; 30 60; 30 70; 30 80; 30 90; 30 100; 30 110; 30 120;
 30 130; 30 140; 30 150; 30 160; 30 170; 30 180; 30 190; 30 200; 30 210; 30 220;
 30 230; 30 240; 30 250; 30 260; 30 270; 30 280; 30 290; 30 300; 30 310; 30 320;
 30 330; 30 340; 30 350; 40 40; 40 50; 40 60; 40 70; 40 80; 40 90; 40 100; 40 110;
 40 120; 40 130; 40 140; 40 150; 40 160; 40 170; 40 180; 40 190; 40 200; 40 210;
 40 220; 40 230; 40 240; 40 250; 40 260; 40 270; 40 280; 40 290; 40 300; 40 310;
 40 320; 40 330; 40 340; 40 350; 40 360; 50 150; 50 183; 50 310; 60 40; 70 30; 70 60;
 70 150; 70 160; 70 180; 70 210; 70 240; 70 270; 70 300; 70 330; 70 360; 80 30; 80 60;
 80 103; 80 150; 80 180; 80 210; 80 240; 80 260; 80 270; 80 300; 80 330; 80 360; 90 30;
 90 60; 90 150; 90 180; 90 210; 90 240; 90 270; 90 300; 90 330; 90 360; 100 30; 100 60;
 100 110; 100 150; 100 163; 100 180; 100 210; 100 240; 100 260; 100 270; 100 300;
 100 330; 100 360; 110 30; 110 60; 110 70; 110 90; 110 150; 110 180; 110 210; 110 240;
 110 270; 110 300; 110 330; 110 360; 120 30; 120 60; 120 150; 120 170; 120 180;
 120 210; 120 240; 120 270; 120 300; 120 330; 120 360; 130 30; 130 60; 130 70;
 130 113; 130 150; 130 180; 130 210; 130 220; 130 240; 130 270; 130 300; 130 330;
 130 360; 140 30; 140 60; 140 93; 140 150; 140 180; 140 200; 140 210; 140 240;
 140 250; 140 270; 140 282; 140 290; 140 300; 140 330; 140 360; 150 150; 150 180;
 150 190; 150 210; 150 240; 150 270; 150 280; 150 286; 150 300; 150 330; 150 360;
 160 110; 160 130; 160 150; 160 180; 160 210; 160 240; 160 270; 160 300; 160 330;
 160 360; 170 120; 170 150; 170 180; 170 210; 170 240; 170 360; 180 30; 180 60;
 180 123; 180 150; 180 180; 180 210; 180 240; 190 30; 190 60; 190 300; 190 352;
 200 30; 200 37; 200 60; 200 80; 200 90; 200 100; 200 110; 200 120; 200 130; 200 140;
 200 150; 200 160; 200 170; 200 180; 200 190; 200 200; 200 210; 200 220; 200 230;
 200 240; 200 250; 200 260; 200 270; 200 280; 200 290; 200 300; 200 310; 200 350;
 210 30; 210 60; 210 320; 220 30; 220 47; 220 60; 220 320; 230 30; 230 60; 230 340;
 240 30; 240 60; 240 210; 250 30; 250 80; 260 40; 260 50; 260 80; 260 90; 260 100;
 260 110; 260 120; 260 130; 260 140; 260 150; 260 160; 260 170; 260 180; 260 190;
 260 200; 260 210; 260 220; 260 230; 260 240; 260 250; 260 260; 260 270; 260 280;
 260 290; 260 300; 260 310; 260 340; 270 70; 270 80; 270 90; 270 100; 270 110;

270 120;270 130;270 140;270 150;270 160;270 170;270 180;270 190;270 200;
270 210;270 220;270 230;270 250;270 260;270 270;270 280;270 290;270 300;
270 310;270 320;270 330;270 340;270 350;270 360;270 370;270 380;280 90;
280 113;290 40;290 50;290 140;290 240;290 300;300 70;300 80;300 90;300 100;
300 110;300 120;300 130;300 150;300 160;300 170;300 180;300 190;300 200;
300 210;300 220;300 230;300 250;300 260;300 270;300 280;300 290;300 300;
300 310;300 320;300 330;300 340;300 350;300 360;300 370;300 380;15 350;
15 355;47 255;47 335;47 345;54 233;54 243;62 365;62 371;75 255;85 52;85 70;
85 228;94 74;95 222;91 260;105 105;115 135;117 228;122 221;135 75;135 170;
135 214;145 77;155 30;155 50;155 185;165 105;169 268;171 31;171 51;175 75;
179 258;172 261;179 333;172 341;183 270;183 280;183 345;206 165;205 315;
217 190;211 200;212 275;215 325;229 140;222 282;228 325;239 130;232 150;
245 71;262 365;275 52;276 236;285 220;285 270;285 335;293 95;295 175;
295 205;52 320;230 350;232 315;53 210;255 71;75 49;0 0

B: 置换 Flow-shop Benchmark 问题

(1) Car1 Flow-shop 问题 (11×5 , $J^* = 7038$)

375 632 12 460 528 796 532 14 257 896 532;
12 452 876 542 101 245 230 124 527 896 302;
142 758 124 523 789 632 543 214 753 214 501;
245 278 534 120 124 375 896 543 210 258 765;
412 398 765 499 999 123 452 785 463 259 988

(2) Car2 Flow-shop 问题 (13×4 , $J^* = 7166$)

654 321 12 345 678 963 25 874 114 785 203 696 302;
147 520 147 586 532 145 24 517 896 543 210 784 512;
345 789 630 214 275 302 142 24 520 336 699 855 221;
447 702 255 866 332 225 589 996 541 234 784 512 345

(3) Car3 Flow-shop 问题 (12×5 , $J^* = 7312$)

456 789 876 543 210 123 456 789 876 543 210 124;
537 854 632 145 785 214 752 143 698 532 145 274;
123 225 588 669 966 332 144 755 322 100 114 753;
214 528 896 325 147 856 321 427 546 321 401 214;
234 123 456 789 876 543 210 123 456 789 876 543

(4) Car4 Flow-shop 问题 (14×4 , $J^* = 8003$)

456 789 630 214 573 218 653 214 204 785 696 532 12 457;
856 930 214 257 896 532 142 547 865 321 124 12 345 678;
963 21 475 320 124 752 147 532 145 763 214 257 854 123;
696 320 142 753 214 528 653 214 527 536 214 528 888 999

(5) Car5 Flow-shop 问题 (10×6 , $J^* = 7720$)

333 333 252 222 255 555 558 888 889 999;
991 111 222 204 477 566 899 965 588 889;
996 663 222 114 123 456 789 876 543 210;
123 456 789 876 543 210 124 537 854 632;
145 785 214 752 143 698 532 145 247 451;
234 532 586 532 142 573 12 14 527 856

(6) Car6 Flow-shop 问题 (8×9 , $J^* = 8505$)

887 799 999 666 663 333 222 114;
447 779 999 666 25 558 886 541;
234 567 852 140 222 558 965 412;
159 267 483 753 420 159 25 863;
201 478 520 145 699 875 633 222;
555 444 120 142 578 965 412 25;
463 123 456 789 876 543 210 123;
456 789 630 258 741 36 985 214;
753 21 427 520 142 534 157 896

(7) Car7 Flow-shop 问题 (7×7 , $J^* = 6590$)

692 581 475 23 158 796 542;
310 582 475 196 325 874 205;
832 14 785 696 530 214 578;
630 214 578 214 785 236 963;
258 147 852 586 325 896 325;
147 753 2 356 565 898 800;
255 806 699 877 412 302 120

(8) Car8 Flow-shop 问题 (8×8 , $J^* = 8366$)

456 789 654 321 456 789 654 789;
654 123 123 456 789 654 321 147;
852 369 632 581 472 586 320 120;
145 678 965 421 365 824 758 639;

632 581 475 32 536 325 863 21;
425 396 325 147 852 12 452 863;
214 123 456 789 654 321 456 789;
654 789 654 123 123 456 789 654

(9) Rec01 Flow-shop 问题 (20×5 , $J^* = 1247$)

5 74 67 97 87 1 69 69 11 87 25 50 58 79 35 70 79 34 37 50;
76 21 48 36 86 42 32 12 63 52 59 42 76 48 57 76 22 99 24 88;
74 83 6 71 64 20 99 54 24 43 88 72 71 20 78 53 77 49 32 46;
99 52 66 68 11 90 26 80 16 10 87 77 82 63 99 2 74 3 35 63;
26 90 38 81 31 23 57 16 89 26 40 29 94 97 80 19 95 61 4 76

(10) Rec03 Flow-shop 问题 (20×5 , $J^* = 1109$)

34 11 63 22 76 20 44 29 54 62 50 89 50 32 12 59 41 94 28 66;
6 65 67 46 4 22 30 89 12 96 13 69 56 24 88 78 20 48 59 33;
63 4 3 88 34 7 55 12 21 61 48 57 8 23 64 95 83 26 10 34;
85 1 73 1 9 3 68 96 74 79 40 1 67 87 14 59 65 93 81 8;
60 73 100 66 76 28 92 71 2 53 37 70 46 62 13 48 20 3 20 5

(11) Rec05 Flow-shop 问题 (20×5 , $J^* = 1242$)

59 89 18 65 1 49 99 35 8 39 60 71 76 30 77 5 98 79 38 69;
37 41 56 67 79 100 9 46 98 73 18 1 30 98 36 82 84 28 36 19;
67 42 75 50 71 30 34 58 97 20 97 4 51 25 76 64 4 84 47 54;
39 59 95 57 78 76 44 26 20 55 61 88 77 43 16 13 34 69 86 83;
30 43 75 13 88 36 62 73 73 30 22 52 22 5 45 14 26 36 1 97

(12) Rec07 Flow-shop 问题 (20×10 , $J^* = 1566$)

28 50 75 65 84 33 91 51 58 44 70 60 39 72 100 79 14 3 68 94;
18 30 50 42 68 72 66 23 61 27 94 38 33 35 71 23 23 32 33 17;
38 75 33 66 42 79 87 100 17 40 7 34 53 45 80 57 36 81 94 54;
11 82 58 29 41 85 88 93 54 19 19 55 87 20 89 54 79 26 37 27;
97 38 56 36 86 81 97 48 25 34 31 63 2 84 47 70 4 19 33 55;
23 39 41 29 23 51 36 84 71 33 48 28 6 23 15 99 65 59 74 34;
90 28 51 10 95 72 21 74 52 3 38 70 51 10 90 85 78 80 64 7;
52 84 29 84 30 19 59 7 47 89 48 35 42 34 33 5 51 90 50 56;
79 48 75 14 73 48 61 98 49 39 73 68 93 8 97 9 95 44 22 10;
63 57 97 67 97 48 4 55 86 66 34 88 67 48 26 4 79 33 17 41

(13) Rec09 Flow-shop 问题 (20×10 , $J^* = 1537$)

77 99 74 4 49 53 88 92 2 97 28 64 25 19 81 9 5 8 34 13;

95 28 25 21 95 59 47 99 49 18 93 22 96 41 1 29 74 66 7 56;
 41 42 92 40 96 75 28 84 86 28 93 91 98 87 74 49 19 40 58 45;
 97 4 29 80 74 19 11 13 46 77 7 56 51 15 56 48 27 71 94 27;
 47 7 4 66 96 13 86 73 58 92 25 46 21 31 8 72 71 17 22 40;
 45 30 21 85 63 50 90 55 42 54 89 27 20 78 55 38 35 61 27 26;
 10 65 47 1 59 82 93 19 24 49 49 32 93 54 3 26 52 84 40 90;
 41 45 36 33 84 60 38 93 79 24 11 70 64 74 92 3 76 49 19 28;
 72 51 61 1 70 9 33 74 12 19 93 94 86 71 28 49 79 52 26 27;
 8 94 9 4 29 13 59 25 17 71 45 5 11 6 5 80 47 56 77 88

(14) Rec11 Flow-shop 问题 (20×10 , $J^* = 1431$)

96 25 76 41 65 81 49 93 39 52 72 55 8 27 34 15 59 93 20 59;
 80 44 62 6 91 69 9 89 83 46 4 43 91 11 86 5 55 4 91 13;
 56 10 48 3 75 65 12 31 55 64 29 47 81 94 87 39 43 43 73 93;
 48 41 54 33 30 93 54 14 32 13 94 32 93 38 10 23 49 5 41 26;
 14 64 47 77 47 61 75 37 18 54 85 87 14 33 64 16 23 84 100 11;
 88 52 35 44 55 3 66 57 9 62 51 97 86 67 30 1 25 55 38 7;
 50 19 72 43 51 44 34 33 93 45 29 41 64 8 47 57 51 22 75 66;
 15 28 54 50 1 17 12 96 65 80 65 86 42 55 51 55 72 78 9 42;
 67 72 27 19 36 6 32 32 75 19 50 17 70 99 69 62 9 31 76 54;
 65 27 56 43 73 14 6 45 73 65 16 30 3 18 26 35 1 11 71 99

(15) Rec13 Flow-shop 问题 (20×15 , $J^* = 1930$)

78 13 47 26 68 47 63 52 91 100 63 96 54 50 17 46 91 99 20 18;
 37 81 9 61 41 60 21 100 21 54 58 60 55 33 7 76 42 93 84 4;
 79 100 31 5 46 23 10 38 59 77 93 79 89 51 12 32 30 57 25 21;
 98 77 40 22 58 23 50 79 55 53 24 84 77 100 66 32 39 23 1 43;
 100 45 86 26 37 57 12 90 39 90 18 86 40 46 98 28 97 52 72 55;
 21 39 27 52 59 60 84 52 75 91 78 31 71 27 25 48 72 89 63 86;
 82 60 69 57 22 35 91 81 3 53 74 41 21 30 76 63 96 4 79 99;
 27 87 50 97 43 56 7 54 11 68 57 79 43 80 25 92 78 74 63 38;
 7 38 87 10 49 54 44 51 14 18 100 63 60 56 76 85 71 21 17 81;
 22 91 34 68 21 73 75 11 24 86 92 17 58 50 25 69 75 10 38 74;
 9 17 13 8 42 81 60 76 36 28 51 31 13 97 69 24 17 53 21 34;
 57 85 15 49 70 61 72 50 72 61 73 65 3 70 69 38 26 94 36 40;
 84 43 95 41 13 15 28 24 48 86 6 41 73 9 73 57 94 59 1 61;
 91 81 96 16 2 70 83 12 69 36 12 77 44 22 45 21 3 100 73 76;

5 33 72 35 76 51 52 23 55 15 83 74 85 92 74 66 39 68 58 100

(16) Rec15 Flow-shop 问题 (20×15 , $J^* = 1950$)

60 79 94 23 38 27 23 41 92 69 16 37 4 55 17 76 81 63 98 47;
 70 53 57 3 32 14 55 91 44 69 38 19 57 65 70 94 49 70 39 48;
 51 74 19 82 68 77 67 90 12 71 36 55 26 21 33 54 83 14 91 89;
 74 87 77 87 7 20 6 43 37 54 48 2 29 99 81 34 72 85 5 33;
 6 18 70 45 49 92 64 37 75 36 92 96 91 27 24 12 31 66 55 99;
 4 16 90 54 49 74 68 96 20 96 20 10 94 53 35 28 72 41 28 10;
 27 44 36 76 72 33 19 99 14 3 55 81 21 42 77 24 91 64 43 54;
 5 95 73 60 31 44 73 46 43 27 89 29 30 78 21 5 8 54 97 94;
 30 41 37 61 70 43 92 41 26 16 75 2 30 63 83 59 40 10 35 52;
 79 3 24 98 8 52 96 51 64 61 57 34 51 25 19 4 93 65 23 44;
 58 20 28 68 26 10 3 33 85 94 89 83 31 86 44 46 84 26 50 73;
 49 59 23 14 81 81 21 49 14 96 62 1 62 60 70 93 63 58 99 67;
 91 74 16 10 86 30 51 17 54 85 36 45 53 64 90 32 67 20 59 44;
 20 33 97 17 60 93 3 64 54 34 36 23 29 76 34 78 36 64 63 36;
 63 42 75 9 74 71 25 68 97 34 69 64 51 69 90 48 84 41 95 98

(17) Rec17 Flow-shop 问题 (20×15 , $J^* = 1902$)

59 72 92 49 42 21 80 4 46 23 53 72 5 80 7 68 77 17 61 29;
 11 54 26 27 12 61 99 54 23 84 91 17 44 19 16 73 2 86 14 6;
 4 36 15 99 55 52 88 43 54 92 80 53 96 73 46 57 1 32 70 6;
 79 3 81 64 62 49 83 63 77 94 8 36 37 95 82 4 82 35 58 12;
 94 59 86 30 37 44 11 44 60 8 41 9 21 78 97 66 2 6 24 78;
 31 23 56 51 24 98 93 78 53 10 89 24 44 78 82 71 49 50 36 28;
 74 40 92 26 24 26 47 44 42 77 3 80 49 31 41 87 89 53 70 40;
 82 59 47 89 42 68 80 39 72 58 87 9 13 13 21 43 27 39 57 13;
 53 89 93 40 41 61 100 76 90 64 57 28 86 50 11 60 34 94 31 61;
 51 37 21 64 88 25 87 99 11 95 75 60 74 93 50 56 52 89 100 19;
 19 85 40 60 14 6 84 29 22 55 37 94 89 98 5 30 85 22 21 39;
 31 67 77 67 33 46 17 38 68 15 8 99 3 80 28 21 26 75 76 98;
 46 39 84 67 85 75 43 14 94 19 23 67 82 46 95 14 80 59 54 69;
 47 65 10 100 4 37 93 75 24 62 88 10 85 9 84 37 87 74 94 14;
 10 60 91 3 20 5 58 25 14 67 65 44 61 37 45 61 58 27 57 3

(18) Rec19 Flow-shop 问题 (30×10 , $J^* = 2093$)

40 39 31 65 11 48 47 67 32 27 81 58 38 14 70 75 69 47 52 58 61 63 63 42 51 100 42

83 5 69;
 16 3 56 77 5 56 60 100 47 97 12 56 30 13 98 30 41 92 7 11 69 22 25 86 100 90 41 24
 84 35;
 50 35 88 58 75 41 16 3 40 84 1 53 81 1 83 45 37 28 97 18 45 5 83 8 42 68 76 14
 57 72;
 59 70 71 66 12 55 67 85 47 30 88 88 51 84 22 64 12 4 56 100 17 77 78 83 53 91 76 100
 6 62;
 100 65 83 86 56 3 61 70 66 68 63 100 70 97 27 13 3 28 90 47 4 99 89 39 10 46 61 26
 60 90;
 78 80 69 93 64 94 36 19 85 28 32 8 28 69 44 47 81 3 60 24 31 19 66 26 66 5 52 41
 91 8;
 38 40 48 69 20 11 36 58 99 26 38 57 10 20 93 6 92 32 37 41 83 99 8 99 19 59 44 19
 18 44;
 76 49 98 49 6 87 62 87 50 98 82 92 93 68 46 49 25 85 42 48 32 37 57 75 2 11 78 18
 83 67;
 9 52 88 85 83 78 13 61 19 88 68 39 53 19 91 57 24 8 19 51 68 92 89 60 24 10 40 21
 44 4;
 68 50 96 51 49 48 74 51 45 96 61 45 45 83 45 21 36 94 15 65 5 19 56 67 41 44 57 12
 87 77.

(19) Rec21 Flow-shop 问题 (30×10 , $J^* = 2017$)

12 55 34 100 41 24 29 89 23 81 72 10 47 13 49 50 99 7 48 96 68 53 57 37 39 36 63 60
 2 45;
 89 2 78 14 92 62 34 62 34 57 47 15 54 27 57 11 87 8 100 86 73 38 13 79 49 54 35 28
 51 94;
 69 94 27 23 88 19 38 32 89 35 75 26 58 29 24 93 85 90 74 19 55 4 12 92 57 59 26 1
 79 42;
 94 19 40 42 52 35 72 54 66 78 27 12 73 85 44 53 9 95 60 15 13 43 12 35 23 69 38 51
 74 9;
 22 43 21 52 99 24 29 93 10 23 66 98 44 29 18 52 87 29 74 45 28 11 21 63 53 62 47 94
 51 70;
 31 74 17 18 41 49 83 59 48 66 64 12 87 64 97 13 98 79 19 1 16 49 68 13 80 12 82 86
 28 4;
 22 72 15 29 68 100 44 8 27 1 30 53 87 62 59 51 34 70 21 90 57 12 2 58 42 77 89 42
 78 52;
 95 10 62 70 22 65 91 24 11 3 49 81 98 62 75 76 22 79 6 49 20 91 80 36 29 37 34 75

87 54;

36 99 96 21 57 13 65 86 94 77 42 46 34 79 17 87 66 6 77 98 76 47 9 65 52 87 1 76

81 16;

38 16 63 47 66 41 100 66 45 14 7 3 15 74 22 95 11 58 84 80 71 3 28 94 33 47 93 77

35 27

(20) Rec23 Flow-shop 问题 (30×10 , $J^* = 2011$)

54 51 44 66 18 100 51 88 41 92 12 98 11 16 13 83 17 59 44 31 44 9 90 29 81 53 40 66

16 78;

2 69 18 95 53 39 86 95 24 78 95 31 53 22 80 41 78 8 79 63 37 74 31 33 66 58 72 90

3 97;

2 31 6 42 28 72 28 45 93 9 32 39 83 32 5 11 85 26 34 50 88 81 8 42 79 82 13 75

46 19;

75 100 6 1 61 19 56 83 33 47 60 91 29 21 57 64 87 77 85 39 43 55 79 54 36 91 79 45

2 22;

39 25 9 21 84 44 13 5 70 39 53 7 85 96 15 16 16 4 86 20 50 50 55 5 75 21 9 59 79 5;

97 58 96 65 52 66 22 30 84 48 44 38 31 42 99 26 29 84 25 32 17 13 59 93 32 65 39 46

11 79;

70 30 52 24 33 8 51 81 66 65 63 10 55 88 76 98 88 64 76 34 52 82 52 5 36 28 90 98

26 90;

5 3 31 78 31 59 99 49 50 79 28 2 100 100 56 77 16 62 21 9 38 100 59 38 2 53 9 99

47 57;

40 18 52 20 14 80 51 86 11 49 66 4 16 12 85 59 20 56 94 39 8 69 83 32 68 39 37 26

88 54;

16 64 14 81 35 94 8 46 87 4 53 60 10 31 87 60 41 14 49 85 17 89 75 70 77 95 3 55

58 68

(21) Rec25 Flow-shop 问题 (30×15 , $J^* = 2513$)

68 34 57 22 28 79 98 69 65 9 92 27 10 56 93 2 2 86 51 38 4 23 81 54 56 81 55 2 6 88;

91 21 69 24 6 80 55 84 21 92 22 67 65 74 96 99 72 54 99 25 29 95 43 33 58 24 33 67

88 65;

17 4 12 24 47 27 71 63 79 84 73 89 77 66 79 100 49 79 68 18 84 66 11 39 10 37 74 56

52 62;

68 48 2 3 71 21 80 15 50 88 4 10 85 61 50 13 44 52 2 97 11 57 95 23 92 21 43 82

32 51;

46 67 51 76 81 56 23 27 67 48 38 35 5 27 35 15 84 2 66 35 61 91 78 44 95 97 66 49

54 52;

86 24 68 65 93 36 45 66 68 71 31 67 25 41 28 35 48 67 44 7 47 15 1 32 95 60 65 74 20 89;
 24 63 34 94 94 24 44 73 56 71 55 88 14 100 100 3 90 69 98 45 17 90 87 16 73 25 32
 97 10 23;
 43 68 8 69 21 94 22 98 53 90 49 43 78 26 84 58 48 78 83 98 2 84 11 96 83 21 96 11
 7 55;
 58 96 17 73 32 53 40 64 71 24 66 51 32 92 78 39 27 38 50 81 68 25 26 29 57 53 29 75
 64 63;
 86 45 55 33 23 50 93 38 29 54 83 22 23 79 81 56 45 92 53 18 85 88 80 87 83 34 7 76
 14 79;
 40 91 80 86 73 55 38 3 63 77 75 23 21 100 65 57 49 13 13 60 93 65 29 34 24 57 33 65
 35 63;
 27 97 61 36 48 7 4 69 36 96 82 60 11 39 69 48 26 25 57 73 64 24 100 25 54 12 78 41
 81 94;
 38 96 51 48 35 78 96 46 62 66 87 54 65 11 17 82 36 40 39 86 98 80 28 80 48 34 30 76
 92 79;
 82 3 32 42 67 14 42 27 77 49 82 22 60 59 96 86 20 37 50 34 34 98 40 14 81 28 36 9 22
 39;
 86 46 36 85 59 53 53 34 35 29 89 76 23 97 19 53 33 80 92 3 62 76 37 83 20 87 45
 80 81 50

(22) Rec27 Flow-shop 问题 (30×15 , $J^* = 2373$)

94 73 55 53 89 41 80 63 84 10 36 66 79 90 64 34 25 40 6 17 55 84 44 92 71 20 48 48
 91 57;
 24 46 6 9 63 42 35 83 78 94 32 98 42 52 40 69 6 62 38 79 15 59 76 97 90 94 56 46
 5 78;
 34 90 28 67 14 38 48 93 16 30 59 49 65 18 19 36 13 98 74 95 11 92 34 2 69 66 18 80
 10 60;
 45 94 38 19 77 67 72 11 19 17 18 11 49 17 49 47 9 3 83 70 12 29 98 59 8 75 96 26
 66 30;
 45 20 93 46 14 92 77 87 31 37 56 61 77 66 100 71 46 23 21 70 85 89 87 47 47 18 53
 42 64 20;
 93 9 74 41 47 21 19 67 56 50 39 85 12 7 74 44 80 69 14 40 40 89 99 69 89 14 5 68
 57 3;
 28 2 81 68 3 100 75 38 56 14 65 65 38 83 12 71 37 1 84 53 11 56 19 47 53 32 29 81
 13 68;
 83 89 89 33 35 39 37 33 18 50 77 8 75 24 71 20 52 3 41 5 26 48 89 85 86 66 22 11

38 83;
 12 35 61 68 98 85 81 43 77 34 91 46 80 37 80 5 76 43 100 86 8 32 43 66 97 2 41 1
 26 45;
 73 38 38 46 96 42 48 40 65 80 21 56 76 48 9 96 95 68 88 57 97 41 8 69 93 1 83 41
 52 42;
 34 87 9 51 24 37 64 46 82 88 58 39 65 15 31 15 56 91 58 59 38 29 55 19 56 10 57 34
 89 94;
 64 14 61 96 45 10 2 3 14 52 90 29 50 92 1 83 84 37 59 37 28 9 55 85 58 9 14 59
 38 26;
 57 15 90 74 68 41 1 86 96 100 65 63 8 72 55 93 17 92 36 40 3 58 87 32 65 8 61 33
 93 32;
 58 28 24 16 18 16 65 1 20 37 33 42 9 20 18 36 9 84 99 33 59 84 22 86 37 28 90 43
 93 47;
 49 50 65 71 23 67 4 68 6 65 65 4 33 2 9 95 51 48 68 89 30 58 97 65 94 66 4 78 50 94

(23) Rec29 Flow-shop 问题 (30×15 , $J^* = 2287$)

19 84 85 14 57 43 24 9 46 99 77 11 75 41 4 22 93 46 5 94 97 83 24 78 58 2 82 57
 17 79;
 84 3 77 44 76 47 47 55 84 35 41 52 31 27 1 51 54 31 47 5 15 28 15 9 28 65 92 60
 7 93;
 11 50 46 9 4 21 29 99 94 34 4 9 5 97 2 78 86 7 33 39 19 40 53 17 3 45 22 51 9 83;
 52 48 60 71 70 8 62 52 86 26 25 71 90 6 3 74 78 95 99 96 30 81 49 74 30 41 44 44
 54 8;
 100 50 22 29 38 36 41 83 72 44 26 36 80 82 33 96 59 99 83 60 19 25 21 90 40 1 92 30
 31 81;
 59 5 21 10 58 90 28 65 23 15 15 78 61 80 6 1 10 48 54 56 75 99 32 21 64 96 51 87
 17 61;
 66 53 53 9 83 84 35 31 20 98 69 82 30 6 35 58 59 87 79 26 32 3 46 85 36 36 4 24
 2 69;
 18 55 61 79 39 82 60 67 63 99 17 95 14 59 18 51 31 17 7 19 85 91 95 69 23 39 35
 42 66 99;
 72 57 43 82 36 99 4 85 81 23 86 88 30 40 79 17 37 9 63 49 24 99 57 2 4 55 67 29
 47 34;
 65 39 1 33 75 60 23 75 17 74 39 88 81 38 78 40 67 26 5 48 79 94 96 49 7 14 80 34
 73 68;
 96 29 96 63 85 51 29 51 60 93 34 3 83 73 62 37 92 20 67 50 1 7 65 40 17 17 32 79

1 51;
 71 17 16 64 40 84 82 36 56 66 97 58 95 22 23 58 74 20 10 31 53 51 61 98 37 20 91 68
 16 46;
 9 23 26 85 50 100 63 12 10 85 43 63 93 98 24 80 28 96 1 68 81 44 82 90 81 3 49 15
 32 91;
 23 61 100 39 34 27 53 19 15 56 43 66 7 44 15 64 28 20 27 11 57 10 62 51 96 21 17 78
 86 52;
 61 99 4 67 35 59 31 44 85 24 28 19 17 23 5 61 40 27 1 83 8 100 92 20 20 67 20 98
 85 50

(24) Rec31 Flow-shop 问题 (50×10 , $J^* = 3045$)

59 30 22 70 38 51 63 48 92 91 62 67 73 27 48 7 55 4 19 30 71 83 45 40 100 97 74 73
 14 100 53 68 69 34 7 47 54 9 64 38 72 26 88 73 77 20 99 13 40 77;
 47 1 58 81 17 15 84 62 29 21 6 83 18 94 28 51 46 4 97 95 39 40 33 67 6 75 89 46 24
 29 11 77 40 41 26 31 96 13 22 25 6 33 39 70 4 36 64 7 87 8;
 20 90 68 2 48 72 75 71 91 56 3 70 55 71 46 48 65 31 37 86 93 37 12 83 82 81 14 62 27
 8 74 88 46 10 79 83 8 81 51 16 40 37 63 57 85 9 7 80 34 14;
 43 97 3 32 53 8 75 70 99 49 89 49 49 33 73 4 48 49 30 22 87 25 63 11 78 22 33 27 62
 55 66 47 62 17 76 28 28 1 33 24 56 84 43 5 50 89 68 57 96 76;
 49 5 33 72 57 34 71 6 54 43 48 50 66 31 89 29 61 28 37 17 38 68 32 62 5 38 11 63 72
 88 94 51 23 25 35 92 94 94 9 62 23 61 98 100 9 4 67 22 27 19;
 74 70 48 57 17 90 13 94 64 20 97 50 56 68 35 62 36 78 16 16 7 47 40 69 43 2 43 34 85
 96 66 73 31 33 92 83 50 82 25 4 39 86 27 31 45 32 85 78 78 82;
 38 59 27 32 25 40 10 10 89 27 79 60 90 45 98 37 69 73 15 61 24 81 60 46 18 53 70 58
 98 23 98 16 45 17 77 96 20 29 22 39 38 22 32 34 35 76 60 75 53 86;
 46 63 12 25 50 44 97 71 89 68 21 28 29 52 97 15 14 26 89 79 1 62 54 93 73 44 58 91
 99 98 87 17 21 28 15 18 28 82 64 77 5 94 20 11 3 20 23 17 40 21;
 18 15 65 13 72 47 81 29 38 99 96 15 87 95 67 10 78 29 11 24 91 96 66 80 86 73 47 11
 25 19 5 87 15 45 27 84 99 27 78 36 75 93 25 98 41 84 55 70 72 10;
 12 93 21 87 72 77 31 99 87 73 77 50 4 40 9 66 100 26 16 9 34 19 92 50 62 74 8 80 7
 79 85 96 40 68 69 45 65 45 88 60 44 17 25 76 80 6 52 55 91 51

(25) Rec33 Flow-shop 问题 (50×10 , $J^* = 3114$)

58 38 16 32 10 96 8 25 42 37 92 78 35 47 47 2 93 38 70 14 60 32 25 77 68 92 46 74 83
 72 13 80 54 7 100 5 23 22 17 78 12 71 37 54 35 91 40 1 57 75;
 62 49 53 59 74 70 72 19 93 62 78 73 34 3 63 100 15 62 1 38 26 13 52 2 65 57 66 85 56
 90 8 56 59 20 61 42 74 74 8 1 58 3 62 75 58 14 47 23 69 44;

99 31 33 13 78 92 78 53 72 60 25 64 89 73 36 92 96 29 63 71 23 38 62 38 85 17 7 55
31 55 2 40 82 10 16 57 65 74 16 55 9 68 60 14 3 89 1 77 76 90;
63 36 21 99 58 53 8 65 85 66 97 47 92 85 23 8 40 19 47 55 50 22 40 83 60 25 11 42 98
34 30 18 81 45 67 26 36 80 97 3 83 81 85 19 47 62 6 76 55 73;
18 13 85 61 39 1 7 83 35 39 4 68 97 27 8 67 84 32 20 100 32 37 76 33 97 68 13 98 24
90 94 87 33 96 63 24 88 40 67 68 57 61 31 32 23 52 19 71 56 4;
68 39 99 98 23 52 94 57 68 71 79 47 74 28 3 27 59 13 79 41 93 59 15 60 25 26 66 51
75 21 24 98 71 70 96 35 68 65 95 26 27 3 84 66 98 59 1 99 83 74;
14 64 20 30 10 58 66 95 93 5 19 79 91 45 57 15 29 21 38 93 79 85 36 64 57 86 51 40
59 76 77 100 4 93 67 53 100 53 70 33 78 14 80 3 18 44 86 40 12 51;
45 58 85 14 78 71 32 71 4 27 67 97 21 34 68 34 40 96 3 11 85 80 7 2 71 49 26 69 53
60 71 41 87 77 12 67 69 86 45 5 14 75 75 69 26 62 37 14 26 25;
98 72 96 22 75 15 90 68 94 65 90 37 41 29 43 63 15 70 60 20 48 94 18 33 94 74 41 20
50 82 100 2 21 48 20 94 84 19 17 9 61 49 59 77 64 76 79 27 12 20;
37 31 99 66 42 50 92 65 45 87 53 10 18 85 68 20 10 76 48 3 20 19 28 9 59 35 94 64 45
76 80 86 18 22 64 72 93 70 97 66 55 94 6 90 22 33 95 46 53 95

(26) Rec35 Flow-shop 问题 (50×10 , $J^* = 3277$)

100 19 70 50 80 47 40 84 85 60 78 98 4 45 73 77 22 90 12 9 29 83 83 83 11 10 73 85
56 59 70 4 51 18 61 25 84 74 33 38 43 15 75 29 14 29 73 87 86 2;
72 3 56 93 97 59 63 80 46 60 64 31 57 45 94 84 66 51 92 76 12 15 95 46 32 74 40 30
63 67 66 66 42 30 9 91 8 2 7 43 98 19 98 60 49 20 19 79 99 41;
76 19 39 100 3 29 69 68 59 2 21 42 30 25 83 11 5 87 43 62 71 73 87 82 15 73 4 58 26
73 80 79 90 61 3 31 95 24 62 2 28 50 42 80 78 27 33 52 31 41;
100 68 71 71 10 3 21 82 35 50 5 73 11 45 59 82 77 27 21 46 70 32 29 2 27 99 20 89 87
65 32 43 85 67 2 2 61 42 68 4 51 30 67 86 93 66 36 85 25 88;
16 29 29 84 14 26 56 4 68 90 85 83 67 7 1 10 97 65 92 71 46 51 46 55 2 54 51 48 53
60 93 39 84 57 61 14 85 33 42 62 43 75 24 70 45 70 93 24 78 6;
9 22 91 64 32 20 73 45 84 20 55 48 4 59 72 9 28 76 64 65 96 6 67 54 43 89 18 15 8 61
56 83 29 60 18 97 41 24 41 95 84 90 63 13 94 95 21 89 10 77;
5 16 100 67 92 50 56 100 89 78 15 71 82 88 65 67 61 67 94 76 12 3 89 85 23 83 37 82
80 94 26 55 73 25 44 91 88 62 78 76 13 94 15 100 35 7 44 50 41 89;
87 13 86 29 67 26 10 96 18 56 23 49 77 42 62 27 82 20 67 65 70 29 73 3 79 5 18 77 46
86 41 25 8 10 78 84 4 13 67 91 71 35 45 86 46 11 51 4 66 80;
34 87 88 28 72 1 46 29 97 62 36 72 98 57 45 43 62 75 60 30 76 3 69 20 28 28 61 2 5
38 21 62 95 20 38 88 86 62 99 67 64 51 92 88 18 75 4 37 35 21;

15 70 99 81 68 70 40 67 58 27 87 30 21 81 76 8 96 67 46 38 19 24 33 57 29 90 75 3 62
 1 9 13 57 95 74 26 51 10 6 78 81 83 44 6 85 52 24 50 1 54

(27) Rec37 Flow-shop 问题 (75×20 , $J^* = 4890 \sim 4951$)

3 71 4 33 13 23 68 77 6 34 65 29 82 2 94 89 40 28 57 56 82 12 52 32 6 53 71 50 30 24
 66 17 64 17 12 11 98 86 10 31 35 32 60 78 73 55 61 51 12 54 13 51 53 30 18 78 95 82
 74 31 40 11 81 10 85 16 40 17 33 98 50 49 81 57 13;
 64 63 39 3 90 62 4 27 68 87 70 28 97 38 37 5 78 16 6 59 10 21 77 90 94 52 46 5 33 98
 93 34 64 51 85 84 91 73 22 55 14 66 4 45 31 30 32 56 44 63 5 88 53 67 24 39 56 42 88
 11 55 86 97 4 49 1 51 16 39 86 51 49 24 42 96;
 92 89 31 86 44 25 69 44 42 81 2 73 86 58 31 72 75 55 55 42 9 72 74 82 88 10 90 30 34
 2 57 47 45 51 87 3 70 89 82 98 10 16 41 32 63 32 64 5 30 38 50 17 12 32 1 69 7 15 16
 59 17 88 95 89 40 49 39 57 78 100 46 65 30 39 17;
 79 2 64 65 27 100 56 95 39 37 63 14 15 14 83 23 55 29 18 20 59 83 66 7 27 77 47 86
 5 10 28 16 53 44 18 30 59 11 15 26 23 11 20 4 44 44 33 66 23 9 53 10 60 7 82 38 55
 39 18 48 8 2 62 40 18 99 14 16 86 29 73 75 12 91 41;
 4 95 41 46 42 81 45 37 87 51 7 60 41 3 26 16 2 38 4 5 71 56 40 27 48 56 12 22 65 29
 24 87 78 82 12 4 17 37 77 72 82 94 19 92 74 52 37 63 93 12 31 86 38 73 2 61 10 36 50
 87 9 1 74 65 34 67 75 94 85 34 52 75 63 3 6;
 67 21 85 44 10 17 29 29 29 66 99 15 25 65 96 3 98 16 19 92 47 19 69 83 73 73 7 100
 65 43 3 36 59 14 69 17 73 80 6 43 7 42 13 14 62 98 91 31 37 50 38 93 32 98 57 1 48
 40 45 54 84 78 4 52 41 12 33 95 17 40 32 84 14 64 1;
 69 27 30 23 72 68 30 100 44 56 51 72 87 88 92 40 1 66 53 3 84 89 34 61 38 57 51 10
 30 83 69 10 55 77 75 73 18 85 71 1 21 7 56 92 63 43 30 48 44 93 93 67 64 58 45 51 24
 36 87 99 22 28 17 11 59 35 36 87 35 55 45 27 51 90 42;
 89 56 58 69 88 36 64 45 76 81 9 34 53 90 94 37 19 63 17 54 74 100 79 86 19 1 38 43
 42 14 66 26 48 14 27 17 38 71 20 15 11 86 47 22 66 59 6 36 4 90 98 58 25 79 23 88 29
 25 81 26 16 29 71 72 85 91 94 25 37 11 51 86 72 88 40;
 50 31 81 81 34 70 44 36 51 17 90 50 1 72 19 21 83 35 31 51 99 6 25 38 47 84 19 57 86
 69 92 64 22 27 65 47 9 12 41 7 48 22 18 44 38 60 96 57 80 24 76 47 73 15 58 14 18 32
 41 57 14 75 26 76 62 46 12 72 62 43 100 45 13 66 89;
 9 17 92 95 13 55 59 85 18 70 77 100 25 62 47 42 33 30 69 11 42 74 42 6 54 95 47 80
 12 58 62 91 36 16 72 71 6 94 77 22 22 93 77 43 82 11 73 35 88 51 64 96 39 10 51 29
 36 90 32 31 78 90 9 4 65 10 82 47 97 51 49 60 75 12 12;
 2 24 19 5 23 41 50 69 39 27 26 53 50 84 48 58 57 12 15 83 59 10 27 29 56 78 46 11 57
 32 9 87 89 64 86 51 85 36 76 5 46 47 81 48 70 96 100 80 59 85 33 1 11 13 80 10 28 80

41 9 8 86 16 98 53 31 6 75 62 15 9 41 35 39 26;
 7 5 59 16 51 62 72 30 9 8 50 83 81 43 91 75 67 67 97 6 68 80 67 76 82 82 100 13 59
 92 94 67 43 76 45 55 36 42 54 75 36 91 20 28 44 23 45 14 60 38 27 32 84 35 67 84 15
 16 65 68 54 79 21 85 2 25 34 35 57 54 6 18 19 65 35;
 42 56 78 63 91 67 19 3 99 10 68 26 64 87 24 31 87 96 88 19 67 38 61 90 6 13 41 26 40
 65 63 18 42 71 66 12 21 29 96 8 77 38 52 5 9 98 35 4 17 10 44 4 2 74 80 6 64 89 42
 10 2 56 31 15 17 74 47 21 8 44 89 30 76 66 58;
 87 22 7 77 95 46 40 15 30 71 50 3 90 98 37 43 78 27 100 35 92 57 40 78 33 88 57 35
 37 81 84 88 61 95 6 37 56 54 48 81 21 58 11 21 56 8 94 56 61 3 97 86 37 43 20 11 83
 29 64 32 100 60 28 48 87 36 13 60 44 57 17 66 71 24 43;
 4 71 62 70 12 30 92 90 40 86 93 55 89 36 64 66 58 82 6 22 47 23 18 23 56 59 100 66
 23 30 35 11 57 95 75 71 79 24 55 44 91 18 73 32 91 75 31 34 81 96 28 49 59 81 63 79
 27 31 38 23 84 44 52 81 30 22 5 53 65 63 97 39 89 69 28;
 93 83 74 88 76 10 88 35 98 51 92 45 58 5 85 19 58 74 66 58 97 57 37 60 75 54 97 98
 33 95 38 43 14 81 98 46 28 75 41 6 75 77 67 25 55 87 75 1 67 4 67 34 12 29 41 26 90
 62 98 76 1 3 37 83 45 52 60 48 21 9 65 71 30 73 53;
 7 71 7 46 41 72 45 88 57 31 15 38 56 48 18 96 38 22 54 62 3 54 22 36 50 45 24 23 96
 62 58 45 9 60 35 94 23 34 62 69 39 85 42 70 57 12 72 98 76 16 55 53 97 97 89 34 19
 54 74 42 43 87 11 76 68 97 100 37 95 84 40 25 23 47 32;
 21 31 99 35 77 84 43 64 84 64 88 70 43 70 36 52 54 3 84 10 16 62 35 45 29 48 87 77
 30 76 8 43 22 14 11 29 10 23 59 35 94 2 52 62 72 51 91 9 46 71 75 76 15 61 5 63 80
 21 79 71 40 52 99 100 95 53 64 48 45 24 75 17 91 65 31;
 14 80 47 82 78 52 46 27 87 44 40 42 42 57 30 27 41 90 14 83 86 43 95 8 77 16 70 31
 62 87 30 40 90 78 28 88 32 58 90 76 89 28 16 2 59 86 6 67 44 33 60 72 42 70 78 15 55
 95 60 98 82 10 7 19 22 14 17 80 77 6 91 80 46 88 30;
 29 47 35 93 45 37 45 66 24 67 68 27 67 35 6 78 74 43 58 17 45 89 24 46 81 75 3 26 67
 99 52 80 63 6 39 16 17 47 35 36 29 20 38 38 99 26 49 23 27 65 68 91 12 37 95 42 55
 42 30 75 56 79 72 91 15 49 23 90 32 52 81 72 6 88 47

(28) Rec39 Flow-shop 问题 (75×20 , $J^* = 5043 \sim 5087$)

55 24 81 84 10 64 77 3 22 37 80 5 91 44 92 90 19 83 11 58 4 54 38 9 48 59 72 42 49
 96 15 58 8 66 65 58 24 69 23 39 99 5 89 4 31 74 63 52 21 75 44 82 69 10 33 95 75 77
 31 91 38 77 36 70 60 9 48 32 39 75 69 76 76 73 97;
 40 12 90 10 34 47 60 8 54 83 6 58 1 21 28 59 75 60 28 12 75 1 3 16 1 44 56 72 54 88
 84 79 34 34 90 38 40 81 72 49 73 30 30 11 32 80 66 25 18 32 77 90 31 86 66 7 70 41
 30 75 35 97 72 95 27 55 90 66 46 15 84 89 91 76 55;

64 19 30 89 14 46 53 27 84 68 46 9 75 86 9 37 86 99 43 10 17 7 47 29 69 11 21 92 41
10 66 1 33 4 81 56 46 94 66 55 79 78 22 100 32 59 47 91 5 34 44 26 99 5 10 7 13 71
47 33 48 32 54 5 3 89 31 22 78 41 79 84 73 88 97;
57 81 12 73 13 6 50 68 9 45 77 27 43 99 93 49 16 66 98 36 28 48 93 87 80 43 76 2 70
53 49 81 99 76 10 25 68 14 61 32 76 66 100 3 22 31 29 38 8 11 47 38 33 5 54 1 97 13
49 29 99 35 34 20 41 71 8 76 22 21 71 96 43 86 24;
84 46 61 60 97 51 28 57 93 15 28 62 72 36 61 75 33 15 26 95 78 52 25 24 5 47 56 60
15 100 20 4 8 10 16 56 83 78 46 16 42 5 8 83 83 88 4 58 7 30 32 41 14 30 92 32 4 87
86 31 76 2 12 37 32 75 72 52 70 6 64 74 91 94 89;
85 1 31 86 2 40 86 51 70 6 43 92 90 98 1 10 62 16 60 5 85 62 22 76 59 34 85 23 85 32
49 47 63 57 15 48 92 9 83 26 73 87 52 79 6 91 22 10 90 23 95 17 25 80 68 100 12 65
44 61 51 76 21 97 41 13 99 10 83 77 79 100 76 68 61;
40 88 69 49 21 2 35 86 48 37 96 93 19 92 53 70 36 67 5 18 98 62 52 36 6 78 95 25 58
14 33 83 57 70 66 12 100 47 95 12 89 44 19 81 23 61 38 21 2 99 53 31 27 57 62 12 97
35 68 63 12 33 68 84 73 52 64 75 79 91 55 37 30 51 41;
96 66 13 36 59 29 21 32 86 32 43 17 78 74 79 91 34 55 29 54 53 75 94 6 37 35 72 48
85 33 64 78 97 70 71 97 73 46 91 5 13 62 26 3 17 16 74 89 85 10 37 65 17 42 14 30 14
14 4 83 74 25 9 26 98 75 28 58 87 63 61 79 39 91 8;
68 16 6 44 59 5 63 38 57 78 79 64 64 33 51 23 78 71 82 15 100 92 71 40 30 96 17 98
68 83 21 88 14 52 83 12 67 4 3 68 28 39 81 57 97 57 16 79 74 58 9 33 59 41 60 20 51
8 69 41 52 57 36 94 60 89 60 79 76 3 43 98 99 1 83;
37 20 51 14 21 7 35 89 81 46 46 66 18 36 83 45 31 78 47 25 21 71 38 6 87 16 55 89 30
13 99 74 49 61 69 60 46 60 66 48 18 12 89 38 67 14 39 29 93 69 16 21 95 74 19 15 74
32 49 11 30 41 5 8 62 1 88 7 70 13 97 7 55 77 33;
70 41 46 26 51 11 24 32 2 26 22 32 23 87 41 28 14 93 80 19 76 44 40 5 8 41 19 87 11
82 20 64 83 70 40 36 26 2 81 4 40 80 42 67 23 3 23 25 77 71 25 79 2 85 47 28 96 44
89 72 61 49 51 12 97 98 34 46 2 93 37 77 78 34 61;
48 75 75 21 12 71 47 12 66 32 58 17 9 56 7 36 41 45 96 69 62 97 24 16 71 14 91 77 70
94 97 52 58 39 42 42 36 48 26 45 2 21 50 62 57 57 12 94 1 58 13 39 35 23 54 99 57 52
33 72 69 46 52 60 69 3 9 62 59 24 6 77 38 9 42;
85 85 43 37 72 78 9 46 92 78 58 12 45 86 90 37 31 2 5 19 50 26 3 59 1 57 83 57 85 97
46 13 2 44 92 27 52 99 21 98 35 62 32 27 50 32 27 14 99 11 4 59 80 97 64 62 94 91 7
22 41 93 33 40 20 2 41 16 33 20 32 50 78 45 6;
67 16 70 89 61 95 27 19 56 50 36 75 54 82 97 58 52 91 9 67 53 76 72 3 55 37 88 66 58
76 54 55 63 91 73 70 70 82 43 13 78 50 64 81 88 94 37 77 99 15 62 98 12 13 38 47 26

87 65 22 42 38 45 23 60 65 76 51 79 69 58 6 100 50 92;
 43 41 99 36 92 35 58 78 36 100 63 40 76 36 12 23 10 41 7 26 41 17 48 50 66 63 33 94
 87 80 64 58 3 66 40 42 14 24 62 47 21 51 29 72 23 49 41 18 20 17 99 83 13 55 1 69 1
 64 50 76 84 43 43 2 63 92 82 49 37 12 66 17 6 2 95;
 47 18 81 86 7 40 10 66 80 51 3 84 68 52 61 42 70 42 82 39 85 20 67 74 15 88 17 49 59
 45 30 63 17 91 8 16 89 26 49 8 75 50 55 39 75 63 25 9 90 65 58 18 18 56 56 66 60 36
 18 54 17 6 44 80 87 64 34 45 72 11 88 50 70 48 6;
 35 31 23 66 38 45 17 8 99 92 19 34 17 72 29 43 93 84 46 34 27 29 72 8 18 7 5 65 68
 18 45 5 75 62 94 27 40 38 51 32 63 47 15 21 44 90 87 92 82 64 67 45 79 72 55 82 33
 13 50 1 45 86 54 4 15 92 65 89 42 31 84 56 28 3 11;
 69 57 33 33 17 72 8 61 92 88 58 40 94 5 86 71 22 24 90 53 76 45 70 87 58 67 27 73 65
 91 61 10 100 16 71 98 61 43 57 39 48 72 21 30 62 2 60 74 3 34 77 69 70 67 70 15 64
 1 39 13 84 38 85 63 31 76 8 64 4 15 7 17 4 56 83;
 72 77 94 15 58 65 3 64 65 48 82 57 91 27 31 96 38 1 51 80 6 70 41 9 93 35 1 84 1 94
 31 4 29 50 32 69 78 100 3 80 47 46 45 36 36 87 57 37 45 23 39 84 75 55 91 75 25 82
 14 13 95 4 7 95 86 22 70 23 97 35 4 96 28 43 8;
 51 26 39 14 99 68 19 76 86 25 88 2 13 95 47 92 78 80 94 46 83 78 31 64 26 91 56 46
 9 18 71 92 30 58 75 3 91 13 67 77 91 82 49 16 30 65 20 18 8 62 80 85 25 73 29 15 81
 98 67 10 76 38 94 69 10 91 10 92 83 58 16 79 95 40 10

(29) Rec41 Flow-shop 问题 (75×20 , $J^* = 4910 \sim 4960$)

88 50 49 66 6 97 95 80 55 52 24 51 38 15 35 80 54 85 30 67 2 64 7 12 83 50 67 15 97
 44 10 75 44 46 36 93 23 55 44 69 36 73 35 3 81 98 10 95 59 31 66 44 70 72 6 52 13 21
 29 17 90 77 34 40 99 42 22 20 41 14 93 59 16 94 47;
 49 16 45 95 54 31 63 11 61 23 33 53 69 91 55 22 78 35 28 17 90 56 52 39 3 54 28 87
 17 28 56 62 33 79 35 64 51 6 66 92 70 10 49 47 90 90 83 28 13 47 68 36 36 81 5 70 62
 91 94 31 89 78 34 19 17 41 26 33 47 81 69 73 79 97 64;
 15 58 92 36 31 45 69 21 42 2 90 25 49 29 57 50 51 27 69 96 49 95 27 61 41 33 30 47
 29 42 78 54 12 21 71 80 91 70 10 27 65 90 94 95 34 80 23 82 99 40 98 90 59 70 18 18
 14 71 83 67 64 42 94 36 10 84 50 25 51 96 51 34 87 78 98;
 75 11 46 29 33 56 23 82 5 91 65 73 91 15 70 5 91 28 40 75 5 31 74 84 34 22 72 32 76
 2 34 19 32 71 60 46 3 30 50 11 87 77 4 15 86 14 22 59 99 39 4 42 25 54 50 63 66 57
 61 24 21 50 35 24 90 32 73 63 73 3 76 40 70 63 43;
 21 21 99 8 28 65 47 30 93 41 4 36 16 51 68 24 97 93 40 95 36 68 20 49 32 75 8 85 86
 16 4 5 98 89 7 7 68 95 18 92 96 34 37 60 81 6 64 10 24 10 99 44 25 48 41 60 84 18 91
 59 1 21 50 41 75 57 15 6 8 25 89 53 98 27 71;

6 82 1 100 40 7 11 89 1 34 98 29 89 63 61 21 3 93 1 64 43 24 17 59 11 57 10 6 4 92
80 4 25 28 26 53 71 66 49 55 45 6 48 7 45 47 24 12 71 89 76 67 25 33 5 75 4 79 10 34
10 26 92 44 43 75 51 6 3 25 86 20 88 77 98;
79 23 9 32 7 78 10 97 56 28 81 8 25 74 23 64 14 15 35 65 23 51 17 12 53 83 69 4 3 10
92 41 73 33 85 23 64 50 48 51 75 63 82 24 7 71 86 43 58 75 60 6 30 74 5 15 22 68 20
70 35 58 5 69 68 43 61 14 21 23 86 5 95 98 83;
74 33 37 8 96 74 40 18 57 29 33 93 57 98 15 31 95 81 86 34 73 78 46 100 90 89 95 31
46 14 33 26 41 53 60 84 76 7 76 7 49 74 81 94 61 55 76 45 12 51 59 60 3 60 31 39 32
84 93 82 1 22 43 90 8 73 83 61 55 39 12 96 41 19 30;
45 2 99 68 62 61 70 18 61 77 73 76 43 52 8 42 7 84 77 16 55 74 78 97 85 22 13 87 98
28 80 84 94 91 80 89 73 13 12 1 35 92 78 22 79 29 76 50 6 24 7 87 16 76 71 31 9 87
68 82 85 55 98 42 43 81 60 10 85 65 55 85 9 7 83;
63 95 79 25 70 74 24 31 29 81 72 13 40 14 20 71 2 73 71 6 15 59 23 18 80 71 76 6 97
10 31 71 66 76 88 12 85 68 46 30 57 87 6 25 80 32 67 96 55 4 30 44 81 2 94 68 58 27
60 69 88 79 28 46 69 57 97 100 1 62 68 30 85 93 40;
20 19 38 44 29 11 32 59 84 24 22 44 83 57 7 100 28 10 87 26 19 84 62 22 54 77 33 88
35 92 99 25 62 26 60 32 33 81 17 10 50 56 69 57 12 51 17 75 20 67 59 64 73 3 97 22
71 59 30 51 14 36 59 51 3 33 83 20 37 50 2 60 16 88 17;
18 36 68 14 97 65 92 76 40 34 29 26 47 41 81 66 19 88 47 95 54 74 27 12 53 98 21 19
17 62 29 15 44 65 51 100 36 7 87 74 92 88 37 37 11 8 1 89 95 67 80 30 43 66 72 61 55
3 78 80 31 79 99 88 71 95 33 86 75 49 82 72 43 43 45;
13 18 53 23 57 11 63 20 9 86 32 49 69 45 58 11 36 86 41 7 11 10 83 35 94 17 83 95 85
7 40 89 84 79 55 16 91 35 28 75 5 73 59 67 27 94 47 92 57 98 94 49 95 84 98 40 40 54
60 63 94 6 80 19 55 41 57 8 24 59 93 30 1 59 86;
64 76 73 40 88 54 98 6 89 83 49 73 66 64 27 42 20 96 32 13 50 80 63 85 57 16 21 59
41 3 5 34 1 41 16 11 38 32 54 90 51 73 18 70 41 67 8 52 7 12 25 35 50 8 69 19 70 56
39 6 66 56 36 94 38 7 60 83 88 10 10 74 13 64 86;
3 87 92 61 61 85 48 94 90 28 25 23 90 17 22 47 59 51 39 66 99 30 77 81 9 55 35 91 90
25 13 96 41 40 11 96 62 1 30 92 53 7 69 83 15 58 29 69 47 55 9 51 50 97 9 28 22 84
29 1 66 16 70 68 79 94 31 31 63 17 12 89 17 86 14;
13 1 98 80 43 13 36 86 63 88 12 7 71 28 27 41 69 33 82 90 55 1 75 53 31 33 8 29 95
29 80 20 49 89 90 25 92 14 40 44 23 2 29 77 44 55 57 24 54 71 24 85 53 52 30 60 41
97 75 49 18 6 59 53 85 78 82 70 40 15 45 1 84 16 46;
1 21 86 74 91 67 55 36 15 20 8 56 71 82 43 34 60 51 60 69 82 27 61 62 36 12 45 13 24
4 11 22 44 5 60 89 97 13 92 14 83 29 7 96 1 52 4 36 53 84 84 34 37 23 48 90 54 70 94

3 4 25 22 75 87 88 34 4 2 59 28 91 50 37 6;
 80 71 3 40 65 90 51 10 51 86 51 7 39 59 2 30 28 8 5 80 19 34 59 57 12 36 65 60 6 15
 22 89 2 70 31 46 99 2 92 28 98 11 4 38 98 96 68 74 100 58 83 36 4 96 50 25 9 11 68
 41 15 2 38 83 23 70 59 52 33 15 32 47 44 36 57;
 15 89 11 98 92 62 98 18 65 5 30 59 32 18 8 20 95 2 31 1 53 56 100 3 7 80 13 25 56 74
 72 67 100 39 77 29 40 75 26 12 75 80 21 16 5 56 62 39 30 2 47 83 51 19 78 47 67 5 55
 31 78 63 76 36 85 90 43 61 87 88 55 7 85 56 13;
 93 36 10 63 68 24 55 8 57 1 63 92 61 67 36 15 78 23 73 77 9 67 74 79 55 100 39 19 14
 23 68 99 88 34 61 17 76 35 18 75 77 24 40 71 93 3 25 7 23 71 74 82 67 28 88 96 25 74
 65 49 97 9 7 41 83 33 13 72 83 46 82 42 97 48 16

注:以上 Flow-shop 问题自站点 <http://mscmga.ms.ic.ac.uk/> 下载

C: Job-shop Benchmark 问题

(1) FT06 or MT06 Job-shop 问题 (6×6 , $J^* = 55$)

机器约束矩阵	加工时间矩阵
3 1 2 4 6 5;	3 10 9 5 3 10;
2 3 5 6 1 4;	6 8 1 5 3 3;
3 4 6 1 2 5;	1 5 5 9 1;
2 1 3 4 5 6;	7 4 4 3 1 3;
3 2 5 6 1 4;	6 10 7 8 5 4;
2 4 6 1 5 3;	3 10 8 9 4 9;

(2) FT10 or MT10 Job-shop 问题 (10×10 , $J^* = 930$)

机器约束矩阵	加工时间矩阵
1 2 3 4 5 6 7 8 9 10;	29 43 85 71 6 47 37 86 76 13;
1 3 5 10 4 2 7 6 8 9;	78 28 91 81 22 2 46 46 69 85;
2 1 4 3 9 6 8 7 10 5;	9 90 74 95 14 84 13 31 85 61;
2 3 1 5 7 9 8 4 10 6;	36 69 39 98 26 95 61 79 76 52;
3 1 2 6 4 5 9 8 10 7;	49 75 33 99 69 6 55 32 26 90;
3 2 6 4 9 10 1 7 5 8;	11 46 10 43 61 52 21 74 51 47;
2 1 4 3 7 6 10 9 8 5;	62 46 89 9 53 65 32 88 40 7;
3 1 2 6 5 7 9 10 8 4;	56 72 12 85 49 25 30 36 89 45;
1 2 4 6 3 10 7 8 5 9;	44 30 90 52 21 48 89 19 74 64;
2 1 3 7 9 10 6 4 5 8;	21 11 45 22 72 72 32 48 11 76;

(3) FT20 or MT20 Job-shop 问题 (20×5 , $J^* = 1165$)

机器约束矩阵	加工时间矩阵
1 2 3 4 5;	29 43 39 71 26 47 61 32 76 64 11 11 85 99 6 95 37 86 11 13;
1 2 4 3 5;	9 75 91 81 22 52 46 46 40 85 78 28 10 52 61 2 21 74 69 7;
2 1 3 5 4;	49 46 90 85 14 84 32 31 85 61 21 90 74 95 49 25 13 48 51 76;
2 1 5 3 4;	62 69 45 22 21 6 32 19 76 47 36 46 89 98 53 72 89 79 89 52;
3 2 1 4 5;	44 72 12 9 72 48 30 36 26 90 56 30 33 43 69 65 55 88 74 45;
3 2 5 1 4;	
2 1 3 4 5;	
3 2 1 4 5;	
1 4 3 2 5;	
2 3 1 4 5;	
2 4 1 5 3;	
3 1 2 4 5;	
1 3 2 4 5;	
3 1 2 4 5;	
1 2 5 3 4;	
2 1 4 5 3;	
1 3 2 4 5;	
1 2 5 3 4;	
2 3 1 4 5;	
1 2 3 4 5;	

(4) LA01 Job-shop 问题 (10×5 , $J^* = 666$)

机器约束矩阵	加工时间矩阵
2 1 5 4 3;	53 21 12 55 83 92 93 60 44 96;
1 4 5 3 2;	21 71 42 77 19 54 87 41 49 75;
4 5 2 3 1;	34 26 31 66 64 43 87 38 98 43;
2 1 5 3 4;	55 52 39 77 34 62 69 24 17 79;
1 4 3 2 5;	95 16 98 79 37 79 77 83 25 77;
2 3 5 1 4;	
4 5 2 3 1;	
3 1 2 4 5;	
4 2 5 1 3;	
5 4 3 2 1;	

(5) LA06 Job-shop 问题 (15×5 , $J^* = 926$)

机器约束矩阵	加工时间矩阵
2 3 5 1 4;	53 21 12 55 83 92 93 60 44 96 95 61 46 28 87;
4 5 2 3 1;	21 71 42 77 19 54 87 41 49 75 7 9 91 52 45;
3 1 2 4 5;	34 26 31 66 64 43 87 38 98 43 35 95 59 27 39;
4 2 5 1 3;	55 52 39 77 34 62 69 24 17 79 76 35 16 50 41;
5 4 3 2 1;	95 16 98 79 37 79 77 83 25 77 28 10 59 43 9;
3 2 1 4 5;	
1 4 2 5 3;	
1 2 3 5 4;	
3 4 5 1 2;	
1 5 4 2 3;	
5 3 1 4 2;	
1 5 3 2 4;	
5 4 2 3 1;	
5 2 1 3 4;	
1 2 3 5 4;	

(6) LA11 Job-shop 问题 (20×5 , $J^* = 1222$)

机器约束矩阵	加工时间矩阵
3 2 1 4 5;	53 21 12 55 83 92 93 60 44 96 95 61 46 28 87 20 78 33 69 96;
1 4 2 5 3;	21 71 42 77 19 54 87 41 49 75 7 9 91 52 45 54 28 89 74 78;
1 2 3 5 4;	34 26 31 66 64 43 87 38 98 43 35 95 59 27 39 71 37 8 94 45;
3 4 5 1 2;	55 52 39 77 34 62 69 24 17 79 76 35 16 50 41 14 26 66 27 69;
1 5 4 2 3;	95 16 98 79 37 79 77 83 25 77 28 10 59 43 9 43 33 42 84 81;
5 3 1 4 2;	
1 5 3 2 4;	
5 4 2 3 1;	
5 2 1 3 4;	
1 2 3 5 4;	
1 4 2 5 3;	
5 3 1 2 4;	
2 3 5 1 4;	
3 2 5 1 4;	
5 1 4 3 2;	
2 1 5 4 3;	
5 2 4 1 3;	
2 1 3 4 5;	
5 1 3 2 4;	
5 3 2 4 1;	

(7) LA16 Job-shop 问题 (10×10 , $J^* = 945$)

机器约束矩阵	加工时间矩阵
2 7 10 9 8 3 1 5 4 6;	53 12 83 93 44 95 46 87 78 69;
5 3 6 10 1 8 2 9 7 4;	21 42 19 87 49 7 91 45 28 74;
4 3 9 2 5 10 8 7 1 6;	34 31 64 87 98 35 59 39 37 94;
2 4 3 8 9 10 7 1 6 5;	55 39 34 69 17 76 16 41 26 27;
3 1 6 7 8 2 5 10 4 9;	21 55 92 60 96 61 28 20 33 96;
3 4 6 10 5 7 1 9 2 8;	95 98 37 77 25 28 59 9 33 84;
4 3 1 2 10 9 7 6 5 8;	71 77 54 41 75 9 52 54 89 78;
2 1 4 5 7 10 9 6 3 8;	26 66 43 38 43 95 27 71 8 45;
5 3 9 6 4 8 2 7 10 1;	52 77 62 24 79 35 50 14 66 69;
9 10 3 5 4 1 8 7 2 6;	16 79 79 83 77 10 43 43 42 81;

(8) LA21 Job-shop 问题 (15×10 , $J^* = 1046$)

机器约束矩阵	加工时间矩阵
3 4 6 10 5 7 1 9 2 8;	53 12 83 93 44 95 46 87 78 69 20 58 42 80 75; 21 42 19 87 49 7 91 45 28 74 31 72 46 12 41; 34 31 64 87 98 35 59 39 37 94 17 23 17 50 50; 55 39 34 69 17 76 16 41 26 27 87 99 48 19 55; 21 55 92 60 96 61 28 20 33 96 24 45 98 94 61; 95 98 37 77 25 28 59 9 33 84 76 28 27 28 14; 71 77 54 41 75 9 52 54 89 78 18 76 67 63 37; 26 66 43 38 43 95 27 71 8 45 32 86 62 98 18; 52 77 62 24 79 35 50 14 66 69 81 90 27 50 79; 16 79 79 83 77 10 43 43 42 81 25 97 48 80 72;
4 3 1 2 10 9 7 6 5 8;	
2 1 4 5 7 10 9 6 3 8;	
5 3 9 6 4 8 2 7 10 1;	
9 10 3 5 4 1 8 7 2 6;	
9 8 7 10 3 2 6 5 1 4;	
5 6 4 10 1 9 7 8 3 2;	
6 5 3 7 2 8 1 4 10 9;	
2 6 1 4 3 8 9 7 10 5;	
3 6 7 10 2 4 9 1 8 5;	
2 5 1 3 10 9 6 4 8 7;	
6 10 1 5 7 4 3 2 9 8;	
6 10 9 8 5 7 4 1 2 3;	
2 9 1 3 10 4 6 7 5 8;	
5 4 7 6 3 9 2 10 8 1;	

(9) LA26 Job-shop 问题 (20×10 , $J^* = 1218$)

机器约束矩阵	加工时间矩阵
9 8 7 10 3 2 6 5 1 4;	53 12 83 93 44 95 46 87 78 69 20 58 42 80 75 47 33 62 57 84;
5 6 4 10 1 9 7 8 3 2;	21 42 19 87 49 7 91 45 28 74 31 72 46 12 41 71 31 44 39 85;
6 5 3 7 2 8 1 4 10 9;	34 31 64 87 98 35 59 39 37 94 17 23 17 50 50 14 69 17 58 90;

续表

机器约束矩阵	加工时间矩阵
2 6 1 4 3 8 9 7 10 5;	55 39 34 69 17 76 16 41 26 27 87 99 48 19 55 96 32 29 50 61;
3 6 7 10 2 4 9 1 8 5;	21 55 92 60 96 61 28 20 33 96 24 45 98 94 61 75 44 97 63 20;
2 5 1 3 10 9 6 4 8 7;	95 98 37 77 25 28 59 9 33 84 76 28 27 28 14 57 58 8 20 56;
6 10 1 5 7 4 3 2 9 8;	71 77 54 41 75 9 52 54 89 78 18 76 67 63 37 60 34 15 21 67;
6 10 9 8 5 7 4 1 2 3;	26 66 43 38 43 95 27 71 8 45 32 86 62 98 18 65 47 40 57 70;
2 9 1 3 10 4 6 7 5 8;	52 77 62 24 79 35 50 14 66 69 81 90 27 50 79 79 58 66 32 30;
5 4 7 6 3 9 2 10 8 1;	16 79 79 83 77 10 43 43 42 81 25 97 48 80 72 22 51 38 87 15;
5 8 10 3 4 9 6 7 2 1;	
9 6 2 8 3 4 7 10 5 1;	
3 5 4 2 9 7 8 1 10 6;	
1 9 4 8 6 3 5 7 2 10;	
10 1 5 9 7 3 6 4 8 2;	
4 3 6 1 8 5 9 2 7 10;	
2 8 9 4 5 6 7 1 3 10;	
2 8 3 1 9 7 4 10 6 5;	
3 4 5 10 1 7 8 9 2 6;	
2 1 6 4 10 8 9 3 7 5;	

(10) LA31 Job-shop 问题 (30×10 , $J^* = 1784$)

机器约束矩阵	加工时间矩阵
5 8 10 3 4 9 6 7 2 1;	53 12 83 93 44 95 46 87 78 69 20 58 42 80 75.47 33 62 57 84 82 52 81 91 7 63
9 6 2 8 3 4 7 10 5 1;	26 12 63 89;
3 5 4 2 9 7 8 1 10 6;	21 42 19 87 49 7 91 45 28 74 31 72 46 12 41 71 31 44 39 85 23 52 61 66 39 6
1 9 4 8 6 3 5 7 2 10;	80 15 36 96;
10 1 5 9 7 3 6 4 8 2;	34 31 64 87 98 35 59 39 37 94 17 23 17 50 50 14 69 17 58 90 45 38 30 20 64 64
4 3 6 1 8 5 9 2 7 10;	87 20 62 64;
2 8 9 4 5 6 7 1 3 10;	55 39 34 69 17 76 16 41 26 27 87 99 48 19 55 96 32 29 50 61 38 54 68 33 54 91
2 8 3 1 9 7 4 10 6 5;	75 79 96 95;
3 4 5 10 1 7 8 9 2 6;	21 55 92 60 96 61 28 20 33 96 24 45 98 94 61 75 44 97 63 20 18 54 79 32 19 74
2 1 6 4 10 8 9 3 7 5;	75 80 22 8;
6 8 7 1 4 3 10 5 2 9;	95 98 37 77 25 28 59 9 33 84 76 28 27 28 14 57 58 8 20 56 84 16 57 20 56 40
10 9 6 8 1 2 3 7 4 5;	6 8 40 64;
10 6 7 8 5 1 2 9 4 3;	71 77 54 41 75 9 52 54 89 78 18 76 67 63 37 60 34 15 21 67 29 62 89 84 83 42
10 4 6 2 5 7 8 9 1 3;	44 61 33 38;
3 2 5 9 6 10 8 4 7 1;	26 66 43 38 43 95 27 71 8 45 32 86 62 98 18 65 47 40 57 70 21 74 89 55 8 98
10 6 5 3 8 4 2 1 9 7;	39 43 10 18;

续表

机器约束矩阵	加工时间矩阵
4 3 5 2 10 1 7 6 8 9;	52 77 62 24 79 35 50 14 66 69 81 90 27 50 79 79 58 66 32 30 50 57 11 8 7 61
9 8 3 1 10 6 7 4 2 5;	24 26 18 15;
4 3 7 5 8 9 6 10 1 2;	16 79 79 83 77 10 43 43 42 81 25 97 48 80 72 22 51 38 87 15 41 37 81 24 40 15
5 7 2 3 8 1 9 6 4 10;	22 22 5 23;
7 1 5 4 8 9 2 6 3 10;	
4 10 7 6 1 9 5 3 8 2;	
5 2 9 1 8 7 6 4 10 3;	
10 2 5 4 9 3 7 1 8 6;	
4 3 7 10 8 1 5 6 2 9;	
2 5 1 3 10 7 8 9 6 4;	
2 4 1 3 10 8 9 5 7 6;	
6 4 7 2 1 8 9 10 3 5;	
2 1 8 5 4 6 10 9 7 3;	
5 9 3 4 2 7 8 10 6 1;	

(11) LA36 Job-shop 问题 (15×15, $J^* = 1268$)

机器约束矩阵	加工时间矩阵
5 4 7 15 11 3 10 2 1 8 9 6 13 12 14;	53 55 93 96 46 20 69 24 42 94 47 44 57 20 52;
12 5 2 8 9 15 13 1 4 7 10 6 11 14 3;	21 77 87 75 91 54 74 18 46 63 71 34 39 67 52;
10 6 3 8 5 13 1 14 7 11 4 12 9 2 15;	34 66 87 43 59 71 94 32 17 98 14 47 58 70 38;
6 1 10 7 5 14 8 9 12 13 3 2 11 15 4;	55 77 69 79 16 14 27 81 48 50 96 58 50 30 54;
11 5 9 3 1 12 15 10 7 8 2 14 4 6 13;	21 83 60 95 28 78 96 58 98 75 75 62 63 82 54;
1 3 5 14 4 13 15 7 2 10 12 9 8 11 6;	95 79 77 77 59 43 84 25 27 80 57 51 20 15 16;
9 5 13 1 8 12 7 11 4 14 2 6 15 3 10;	71 19 41 7 52 28 78 72 67 41 60 44 21 23 62;
5 14 12 4 8 10 2 3 13 9 15 1 11 7 6;	26 64 38 35 27 37 45 23 62 50 65 17 57 45 74;
6 2 7 9 14 11 3 4 8 12 15 5 1 10 13;	52 34 24 76 50 26 69 99 27 55 79 29 32 38 57;
12 6 5 9 8 1 10 7 15 4 11 14 3 13 2;	16 37 83 28 43 33 81 28 48 14 22 8 87 84 37;
8 4 1 5 13 15 11 2 10 14 6 9 3 12 7;	12 92 44 61 87 33 20 45 80 61 33 97 84 18 81;
2 3 4 6 5 7 10 8 11 9 12 14 13 1 15;	42 54 49 9 45 89 31 76 12 37 31 15 85 29 61;
4 8 14 6 12 13 3 5 11 2 10 7 15 9 1;	31 43 98 95 39 8 17 86 50 18 69 40 90 21 30;
10 8 6 15 11 5 12 3 2 4 14 7 1 13 9;	39 62 17 35 41 66 87 90 19 79 32 66 61 50 68;
10 11 12 15 9 1 8 7 13 2 3 14 5 4 6;	98 79 25 10 9 42 76 97 28 72 58 38 56 41 57;

注：以上 Job-shop 问题自站点 <ftp://mscmga.ms.ic.ac.uk/pub/jobshop1.txt> (或 jobshop2.txt) 下载

参 考 文 献

- Aarts E H L and Van Laarhoven P J M. 1985. Statistical cooling: a general approach to combinatorial optimization problem. *Philips J of Research*, 40: 193-226
- Adams J, Balas E and Zawack D. 1988. The shifting bottleneck procedure for job shop scheduling. *Mgmt Sci*, 34: 391-401
- Aihara K, Takabe T and Toyoda M. 1990. Chaotic neural networks. *Phys Lett A*, 144(6/7): 333-340
- Al-Sultan K S and Fedjki C Á. 1997. A tabu search-based algorithm for the fuzzy clustering problem. *Pattern Recognition*, 30(12): 2023-2030
- Amari S. 1978. Mathematical theory on formation of category detecting nerve cells. *Biol Cybern*, 29: 127-136
- Androulakis I P, Venkatasubramanian V. 1991. A genetic algorithm framework for process design and optimization. *Computer Chem Engng*, 15(4): 217-228
- Angeline P J, Saunders G M and Pollack J B. 1994. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans on Neural Networks*, 5(1): 54-65
- Asai H, Onodera K, Kamio T and Ninomiya H. 1995. A study of Hopfield neural networks with external noises. In: *IEEE International Conference on Neural Networks Proceedings*. NJ: IEEE Piscataway, 1584-1589
- Astrom K, Hagglund T. 1995. PID controllers: theory, design and tuning. New York: ISA
- Athreya K B, Doss H, Sethuraman J. 1996. On the convergence of the Markov chain simulation method. *Ann. Statist.*, 24: 69-100
- Azimi M R, Sheedvash S, Trujillo F O. 1993. Recursive dynamic node creation in multilayer neural networks. *IEEE Trans on Neural Networks*, 4(2): 242-256
- Bac F Q, Perov V L. 1993. New evolutionary genetic algorithms for NP-complete combinatorial optimization problems. *Biol. Cybern.*, 69: 229-234
- Back T. 1992. The interaction of mutation rate, selection and self-adaptation within genetic algorithms. In: Manner R and Manderick B eds. *Parallel Problem Solving from Nature II*. Amsterdam, North Holland: Elsevier, 15-25
- Baker K R. 1974. Introduction to sequencing and scheduling. New York: Wiley
- Balabanian N and Bickart T A. 1969. Electrical network theory. NY: John Wiley
- Bartlett E B. 1994. Dynamic node architecture learning: an information theoretic approach. *Neural Networks*, 7(1): 129-140
- Belew R K. 1992. Evolving networks: using the genetic algorithms with connectionist learning. *Artificial Life II*
- Ben-Daya M, Al-Fawzan M. 1998. A tabu search approach for the flow shop scheduling problem.

- European Journal of Operational Research, 109; 88-95
- Bland R G and Shallcross D F. 1989. Large traveling salesman problem arising from experiments in X-ray crystallography: a preliminary report on computation. *Ops Res Lett*, 8; 125-128
- Blue J A and Bennett K P. 1998. Hybrid extreme point tabu search. *European J. of Operational Research*, 106(2/3); 676-688
- Bosenink T, Ebeling W and Engel A. 1987. Boltzmann and Darwin strategies in complex optimization. *Physics Letters A*, 125(6/7); 307-310
- Brandt R, Wang Y, et al. 1988. Alternative networks for solving the traveling salesman problem and list matching problem. In: *Proc IEEE Int Conf Neural Network. II*; 333-340
- Brindle A. 1981. Genetic algorithms for function optimization. Ph D Dissertation. Alberta: Univ of Alberta.
- Burdett R L, Kozan E. 2000. Evolutionary algorithms for flowshop sequencing with non-unique jobs. *International Transactions in Operational Research*, 7; 401-418
- Carlier J and Pinson E. 1989. An algorithm for solving the job-shop problem. *Mgmt Sci*, 35; 164-176
- Carlier J. 1978. Ordonnancements a contraintes disjonctives. *Operations Research*, 12; 333-351
- Cesari G. 1996. Divide and conquer strategies for parallel TSP heuristics. *Computers & Operations Research*, 23(7); 681-694
- Chang C S and Huang J S. 1996. Optimal SVC placement for voltage stability reinforcement. *ICEE*, 1; 253-257
- Chelouah R, Siarry P. 2000. Tabu search applied to global optimization. *European Journal of Operational Research*, 123; 256-270
- Chen L, Aihara K. 1995. Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks*, 8(6); 915-930
- Chen R. The basis of immunology. 1980. Beijing; Health Press
- Chen S, Billings S A, Grant P M. 1992. Recursive hybrid algorithm for nonlinear system identification using radial basis function networks. *Int J Control*, 55(5); 1051-1070
- Chen S, Cowan C F N, Grant P M. 1991. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans on Neural Networks*, 2(2); 302-309
- Chen, L, Aihara, K. 1999. Global search ability of chaotic neural networks. *IEEE Transactions on Circuits & Systems I-Fundamental Theory & Applications*, 46(8); 974-993
- Choi C, Lee J. 1998. Chaotic local search algorithm. *Artificial Life & Robotics*, 2(1); 41-47
- Choi S H, Ko J W, Manousiouthakis V. 1999. A stochastic approach to global optimization of chemical processes. *Computers & Chemical Engineering*, 23; 1351-1356
- Coit D W, Smith A E. 1996. Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. *Computers & Operations Research*, 23(6); 515-526
- Costamagna E, Fanni A and Giacinto G. 1998. A tabu search algorithm for the optimization of telecommunication networks. *European J of Operational Research*, 106(2/3); 357-372
- Costamagna E, Fanni A, Giacinto G. 1998. A tabu search for the optimisation of telecommunication

- networks. *European Journal of Operational Research*, 106: 357-372
- Croce F D, Tadei R and Volta G. 1995. A genetic algorithm for the job shop problem. *Computers & Operations Research*, 22(1): 15-24
- Cvijovic D, Klinowski. 1995. Taboo search: An approach to the multiple minima problem. *Science*, 667: 664-666
- Davis L. 1987. *Genetic algorithms and simulated annealing*. Los Altos: Morgan Kaufmann Publishers
- Davis L. 1991. *Handbook of genetic algorithm*. New York: Van Nostrand Reinhold
- De Jong K A. 1975. *An analysis of the behavior of a class of genetic adaptive systems*. Ph D Dissertation. Ann Arbor: Univ Microfilms
- Dell'Amico M and Trubian M. 1993. Applying tabu search to the job shop scheduling problem. *Ann Ops Res*, 41: 231-252
- Dorigo M, Gambardella L M. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans Evolutionary Computation*, 1(1): 53-66
- Dorndoft U and Pesch E. 1995. Evolution based learning in a job shop scheduling environment. *Computers & Operations Research*, 22(1): 25-40
- Eberhart R C, Dobbins R W. 1991. Designing neural network explanation facilities using genetic algorithms. In: *IEEE IJCNN'91*. Seattle, 1758-1763
- Eiben A E, Aarts E H and Van Hee K M. 1991. Global convergence of genetic algorithms: an infinite markov chain analysis. In: Schwefel H P and Manner R eds. *Parallel Problem Solving from Nature*. Heidelberg, Berlin: Springer-Verlag, 4-12
- Eilon S and Christofides N. 1969. Distribution management: mathematical modeling and practical analysis. *Operational Research Quarterly*, 20: 309-319
- Englander A C. 1985. Machine learning of visual recognition using genetic algorithms. In: *Proc of the 1st Int Conf on Genetic Algorithms and their Application*. Pittsburgh, PA, 197-201
- Faigle U and Kern W. 1992. Some convergence results for probabilistic tabu search. *ORSA J on Computing*, 4(1): 2-37
- Faigle U, Kern W. 1992. Some convergence results for probabilistic tabu search. *ORSA Journal of Computing*, 4: 32-37
- Fang Y, Kincaid T G. 1996. Stability analysis of dynamical neural networks. *IEEE Trans on Neural Networks*, 7(4): 996-1006
- Fogel D B. 1994. An introduction to simulated evolutionary optimization. *IEEE Trans on Neural Networks*, 5(1): 3-14
- Fogel D B. 1993. Applying evolutionary programming to selected traveling salesman problems. *Cybernetics and System*, 24: 27-36
- Foo S Y, Takefuji Y, Szu H. 1995. Scaling properties of neural networks for job-shop scheduling. *Neurocomputing*, 8(1): 79-91
- Forti M, Manetti S and Marini M. 1994. Necessary and sufficient condition for absolute stability of neural networks. *IEEE Trans on Circuits Syst I*, 41: 491-494

- Fredman M L, Johnson D S, McGeoch L A, Ostheimer G. 1995. Data structures for traveling salesman. *J of Algorithms*, 18(3): 432-479
- Fukushima K. 1980. Neocognitron; a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern*, 36(4): 193-220
- Garey M R and Johnson D S. 1979. *Computers and intractability; A guide to the theory of NP-completeness*. Freeman
- Geman S and Geman D. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans on Patt Anal and Mach Intell*, 6(6): 721-741
- Gil C and Ortega J. 1998. Meta-heuristic for circuit partition in parallel test generation. *Parallel and Distributed Processing*, 315-323
- Glover F, Kochenberger G A and Alidaee B. 1998. Adaptive memory tabu search for binary quadratic programs. *Management Science*, 44(3): 336-345
- Glover F, Kelly J P, Laguna M. 1995. Genetic algorithms and tabu search; hybrids for optimization. *Computers and Operations Research*, 22(1): 111-134
- Glover F, Laguna M. 1992. Tabu search. In: Reeves C R (Ed.), *Modern heuristic techniques for combinatorial problems*. Blackwell Scientific Publication, Oxford, 70-150
- Glover F, Taillard E, Werra D de. 1993. A user's guide to tabu search. *Annals of Operations Research*, 41: 3-28
- Glover F. 1989. Tabu search-part I. *ORSA Journal on Computing*, 1: 190-206
- Glover F. 1989. Tabu search-part II. *ORSA Journal on Computing*, 2: 4-32
- Glover F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13: 533-549
- Glover F. 1989. Tabu search; Part I. *ORSA Journal on Computing*. 1(3): 190-206
- Glover F. 1990. Tabu search; Part II. *ORSA Journal on Computing*. 2(1): 4-32
- Goldberg D E and Segrest P. 1987. Finite markov chain analysis of genetic algorithm. In: *Proc of the Second Int Conf on Genetic Algorithms*. 1-8
- Goldberg D E. 1989. *Genetic algorithms in search, optimization, and machine learning*. MA: Addison-Wesley
- Grefenstette J J. 1986. Optimization of control parameters for genetic algorithms. *IEEE Trans on SMC*, 16(1): 122-128
- Grefenstette J J. 1981. *Parallel Adaptive Algorithms for Function Optimization*. Technical Report No. CS-81-19, Nashville, Vanderbilt University, Computer Science Department
- Grossberg S. 1969. Some network that can learn, remember and reproduce any number of complicated spacetime pattern. *J of Mathematics and Mechanics*, 19: 53-91
- Grotschel M and Holland O. 1991. Solution of large-scale symmetric traveling salesman problems. *Mathematical programming*, 51(2): 141-202
- Hajek B. 1988. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2): 311-329

- Hale J K. 1977. Theory of functional differential equations. New York: Springer-Verlag
- Hanafi S. 2000. On the convergence of tabu search. *Journal of Heuristic*, 7: 47-58
- Happle B L M, Murre J M. 1994. Design and evolution of modular neural network architectures. *Neural Networks*, 7(6/7): 985-1004
- Harp S A. 1989. Towards the genetic synthesis of neural networks. In: *Proc of 3rd Conf on GA*. 360-369
- Hasegawa M, Ikeguchi T, Matozaki T and Aihara K. 1997. An analysis on additive effects of nonlinear dynamics for combinatorial optimization. *IEICE Transaction Fundamentals*, E80-A (1): 206-213
- Haupt R. 1989. A survey of priority rule based scheduling. *OR Spektrum*, 11: 3-16
- Hayakawa Y, Marumoto A and Sawada Y. 1995. Effects of the chaotic noise on the performance of neural-network model for optimization problems. *Physical Review E*, 51(4): R2693-R2696
- Hebb D O. The organization of behavior. New York: Wiley, 1949
- He J, Kang L. 1999. On the convergence rates of genetic algorithms. *Theoretical Computer Science*, 229: 23-39
- Hirsch M W. 1989. Convergent activation dynamics in continuous time networks. *Neural Networks*, 2: 331-349
- Holland J H. 1975. Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press
- Hopfield J J and Tank D W. 1985. Neural computation of decision in optimization problem. *Biol Cybern*, 52: 141-152
- Hopfield J J. 1982. Neural networks and physical systems with emergent collective computational abilities. In: *Proc Natl Acad Sci. USA*, 79: 2554-2558
- Hopfield J J. 1984. Neurons with graded response have collective computational properties like those of two-state neuron. In: *Proc Natl Acad Sci. USA*, 81: 3088-3092
- Huntley C L and Brown D E. 1996. Parallel genetic algorithms with local search. *Computers & Operations Research*, 23(6): 559-571
- Hush D R, Horne B and Salas J M. 1992. Error surfaces for multilayer perceptrons. *IEEE Trans on SMC*, 22(5): 1152-1161
- Jain A S and Meeran S. 1999. Deterministic job-shop scheduling: past present and future. *European J of Operational Research*, 113: 390-434
- James R J W, Buchanan J T. Performance enhancements to tabu search for the early/tardy scheduling problem. *European Journal of Operational Research*, 106: 254-265
- Jeong I K and Lee J J. 1996. Adaptive simulated annealing genetic algorithm for control application. *Int J of Systems Science*, 27(2): 241-253
- Jiao L, Wang L. 2000. A novel genetic algorithm based on immunity. *IEEE Trans on Systems, Man, and Cybernetics*, 30(5): 552-561
- Karmarkar U S and Kekre S. 1985. Lot-sizing in multi-item multi-machine job shops. *IIE Trans*, 290-298

- Kaszkurewicz E and Bhaya A. 1994. On a class of globally stable neural circuits. *IEEE Trans on Circuits Syst I: Fundamental Thoery Appl*, 41(2): 171-174
- Keane A J. 1995. Genetic algorithm optimization of multi-peak problems; studies in convergence and robustness. *Artificial Intelligence in Engineering*, 9(2): 75-83
- Kelly D G. 1990. Stability in contractive nonlinear neural networks. *IEEE Trans on Biomed Eng*, 3(3): 231-242
- Kelly J D, Davis L. 1991. Hybridizing the genetic algorithm and the K nearest neighbors classification algorithm. In: *Proc of 4th Conf on GA*. Los Altos, CA; Morgan Kaufmann Publishers, 377-383
- Khalil H K. 1992. *Nonlinear system*. New York; Macmillan
- Kim Y K, Kim Y J and Kim Y. 1996. Genetic algorithms for assembly line balancing with various objectives. *Computers Ind Enging*, 30(3): 397-409
- Kirkpatrick S, Gelatt C D and Vecchi M P. 1983. Optimization by simulated annealing. *Science*, 220: 671-680
- Kirkpatrick S, Toulouse G. 1985. Configuration space analysis of traveling salesman problem. *J Phys*, 46: 1277-1292
- Kohonen T. 1984. *Self-organization and associative memory*. Berlin; Springer
- Kolahan F and Liang M. 1998. An adaptive TS approach to JIT sequencing with variable processing times and sequence-dependent setups. *European J of Operational Research*, 109(1): 142-159
- Korte B. 1988. Applications of combinatorial optimization. In: *13rd Int mathematical programming symp*
- Kramer A H, Sangiovanni-Vincentelli A. 1988. Efficient parallel learning algorithms for neural networks. In: *Advances in Neural Information Processing Systems*. San Mateo; Morgan Kaufmann, 40-48
- Krishnakumar K. 1989. Micro-genetic algorithms for stationary and non-stationary function optimization. *SPIE Intelligent Control and Adaptive Systems*, 1196: 289-296
- Kruschke J K. 1989. Improving generalization in back-propagation networks with distributed bottleneck. In: *IEEE IJCNNs*. Washington, DC, 443-448
- Kurbel K, Schneider B and Singh K. 1998. Solving optimization problems by parallel recombinative simulated annealing on a parallel computer — an application to standard cell placement in VLSI design. *IEEE Trans on SMC*, 28(3): 454-461
- Kwok T, Smith K. 1999. A unified framework for chaotic neural-network approaches to combinatorial optimization. *IEEE Trans on Neural Networks*, 10(4): 978-981
- Lagarias J C. 1998. Convergence properties of the Nelder-Mead simplex method in low dimension. *SIAM J on Optimization*, 9(1): 112-158
- Lawrence S. 1984. *Resource constrained project scheduling; an experimental investigation of heuristic scheduling techniques*. GSIA, Carnegie-Mellon univeristy
- Lee J and Kim Y. 1996. Search heuristics for resource constrained project scheduling. *J of Operational Research Society*, 47: 678-689

- Lee Y, Oh S H and Kim M. 1993. An analysis of premature saturation in back propagation learning. *Neural Networks*, 6: 719-728
- Leung Y, Gao Y, Xu Z B. 1997. Degree of population diversity- a perspective on premature convergence in genetic algorithms and its Markov-chain analysis. *IEEE Trans on Neural Networks*, 8(5): 1165-1176
- Lin S and Kernighan B W. 1973. An effective heuristic for traveling-salesman problem. *Ops Res*, 21: 498-516
- Lin W, Delgado J G, Gause D C, et al. 1995. Hybrid Newton-Raphson genetic algorithm for the traveling salesman problem. *Cybernetics and Systems*, 26: 387-412
- Litke J D. 1984. An improved solution to the traveling salesman problem with thousands of nodes. *ACM commmun*, 27: 1227-1236
- Liu L, Mao Z. 1997. Water turbines PID controller based on genetic algorithm. *Automation of Electric Power Systems*, 21(12): 41-43
- Lutz C M, Davis K R and Sun M. 1998. Determining buffer location and size in production lines using tabu search. *European Journal of Operational Research*, (2/3): 301-316
- Lutz C M, Davis K R and Sun M. 1998. Determining buffer location and size in production lines using tabu search. *European Journal of Operational Research*, 106: 301-316
- Mahfoud S W and Goldberg D E. 1995. Parallel recombinative simulated annealing: a genetic algorithm. *Parallel Computing*, 21: 1-28
- Majhi A K, Patnaik L M and Raman S. 1995. A genetic algorithm-based circuit partitioner for MCMs. *Microprocessing & Microprogramming*, 41(1): 83-96
- Martins C L, Pato M V. 1998. Search strategies for the feeder bus network design problem. *European Journal Operational Research*, 106: 425-440
- Matsuo H, Suh C J and Sullivan R S. 1988. A controlled search simulated annealing method for the general jobshop scheduling problem. Department of Management, Texas University, Austin
- Matsuoka K. 1992. Stability conditions for nonlinear continuous neural networks with asymmetric connection weights. *Neural Networks*, 5: 495-500
- McCulloch W W, Pitts W. 1943. A logic calculus of the ideas imminent in neurons activity. *Bulletin of Mathematical Biophysics*, 5: 115-133
- Metropolis N, Rosenbluth A W and Rosenbluth M N, et al. 1953. Equations of state calculations by fast computing machines. *J Chem Phys*, 21: 1087-1091
- Michalewicz Z. 1994. Genetic algorithms + data structures = evolution programs. Berlin: Springer-Verlag
- Michel A N, Farrel J A and Porod W. 1989. Qualitative analysis of neural networks. *IEEE Trans on Circuits Syst*, 36(2): 229-243
- Minsky M and Papert S. 1969. *Perceptrons*. Cambridge, MA: MIT Press
- Mitra D, Romeo F and Sangiovanni-Vincentelli A. 1986. Convergence and finite-time behavior of simulated annealing. *Adv Appl Prob*, 18: 747-771

- Mladenovic N, Hansen P. 1997. Variable neighborhood search. *Computers and Operations Research*, 24: 1097-1100
- Moody J O, Antsaklis P J. 1996. The dependence identification neural network construction algorithm. *IEEE Trans on Neural Networks*, 7(1): 3-15
- Moody J, Darken C. 1989. Fast learning in networks of locally-tuned processing units. *Neural Computation*, (1): 281-294
- Mublenbein H. 1991. Evolution in Time and Space; the Parallel Genetic Algorithm. In: Raulins G eds. *Foundations of Genetic Algorithms*. Morgan Kaufmann
- Muth J F and Thompson G L. 1963. *Industrial scheduling*. Prentice-Hall: Englewood Cliffs
- Nakano R and Yamada T. 1991. Conventional genetic algorithm for job shop problems. In: *Proc 4th Int Conf Genetic algorithms and their Applications*. San Diego, 474-479
- Nawaz M, Enscoe E Jr, Ham I. 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1): 91-95
- Nelder J A, Mead A. 1965. A simplex method for function minimization. *Computer Journal*, 7: 308-313
- Nishimori H and Inoue J. 1998. Convergence of simulated annealing using the generalized transition probability. *J Phys A: Math Gen*, 31: 5661-5672
- Nonobe K and Ibaraki T. 1998. A tabu search approach to the constraint satisfaction problem as a general problem solver. *European J of Operational Research*, 106(2/3): 599-623
- Nowicki E, Smutnicki C. 1996. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, 91: 160-175
- Nozawa H. 1992. A neural-network model as a globally coupled map and applications based on chaos. *Chaos*, 2(3): 3140-3145
- Ogbu F A and Smith D K. 1990. The application of the simulated annealing algorithm to the solution of the $n/m/C_{\max}$ flow-shop problem. *Computers & Operations Research*, 17(3): 243-253
- Omidvar O M, Wilson C L. 1992. Optimization of neural network topology and information content using Boltzmann methods. In: *IEEE IJCNNs*. Baltimore, 594-599
- Osman I H, Potts C N. 1989. Simulated annealing for permutation flow-shop scheduling. *Omega* 17 (6): 551-557
- Ott E, Grebogi C and Yorke J A. 1990. Controlling chaos. *Physical Review Letters*, 64(11): 1196-1199
- Park J, Stanberg I W. 1991. Universal approximation using radial basis function networks. *Neural Computation*, 3(3): 246-257
- Park M and Kim Y. 1998. A systematic procedure for setting parameters in simulated annealing algorithms. *Computers & Operations Research*, 25(3): 207-217
- Parker D B. 1987. Optimal algorithms for adaptive networks; second order back propagation, second order direct propagation and second order Hebbian learning. In: *Proc IEEE 1st Int Conf Neural Networks*. San Diego, CA, 593-600
- Parker J K and Goldberg D E. 1989. Inverse kinematics of redundant robots using genetic algorithms. In: *IEEE Int Conf Robotics and Automation*. 271-275

- Pecora L, Carroll T. 1990. Synchronization in chaotic systems. *Physical Review Letters*, 64(8): 821-824
- Ponnambalam S G, Aravindan P, Rajesh S V. 2000. A tabu search algorithm for job shop scheduling. *International Journal of Advanced Manufacturing Technology*, 16: 765-771
- Poths J C, Giddens T D and Yadaw S B. 1994. The development and evaluation for an improved genetic algorithm based on migration and artificial selection. *IEEE Trans on SMC*, 24(1): 73-86
- Potter B and Jones A H. 1992. Genetic tuning of digital PID controllers. *Electronic letters*, 28(9): 834-844
- Press W H, Teukolsky S A. 1991. Simulated annealing optimization over continuous spaces. *Computers in Physics*, 5(4): 426-429
- Qi X and Palmieri F. 1994. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, Part I: Basic properties of selection and mutation. *IEEE Trans on Neural Networks*, 5(1): 102-119
- Reeves C R. 1995. A genetic algorithm for flowshop sequencing. *Computers & Operations Research*, 22(1): 5-13
- Rego C. 1998. Relaxed tours and path ejections for the traveling salesman problem. *European Journal of Operational Research*, 106: 522-538
- Rose C and Yates R D. 1996. Genetic algorithms and call admission to telecommunications networks. *Computers & Operations Research*, 23(5): 485-499
- Rosenthal J S. 1995. Minorization conditions and convergence rates for Markov chain Monte Carlo. *J. Amer. Statist. Assoc.*, 90: 558-566
- Rosenblatt F. 1962. *Principles of neurodynamics*. New York: Spartan Book
- Rosenblatt F. 1959. The perceptron: a perceiving and recognizing automation. *Cornell Aeronautical Laboratory Report*, 85-406-1
- RoyChowdhury P, Singh Y P, Chansarkar R A. 2000. Hybridization of gradient decent algorithms with dynamic tunneling methods for global optimization. *IEEE Trans SMC*, 30(3): 384-390
- Rudolph G. 1994. Convergence properties of canonical genetic algorithms. *IEEE Trans on Neural Network*, 5(1): 96-101
- Rumelhart D E and McClelland J L. 1986. *Parallel distributed processing*. Cambridge, MA: MIT Press
- Sait S M and Youssef H. 1998. Cmos/BiCMos mixed design using tabu search. *Electronics Letters*, 34(14): 1395-1396
- Sarkar D. 1996. Randomness in generalization ability: a source to improve it. *IEEE Trans on Neural Networks*, 7(3): 676-685
- Schraudolph N N, Belew R K. 1992. Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9: 9-21
- Seneta E. 1981. *Non-negative matrices and markov chains*. 2nd ed., New York: Springer-Verlag
- Sexton R S, Alidaee B, Dorsey R E, et al. 1998. Global optimization for artificial neural networks: a tabu search application. *European J of Operational Research*, 106(2/3): 570-584
- Sexton R S, Alidaee B, Dorsey R E, Johnson J D. 1998. Global optimization for artificial neural

- networks; a tabu search application. *European Journal of Operational Research*, 106: 570-584
- Shi L, Olafsson S, Chen Q. 1999. A new hybrid optimization algorithm. *Computers & Industrial Engineering*, 36: 409-426
- Smith D. 1985. Bin packing with adaptive search. In: *Proc of Int Conf on Genetic algorithms and their Applications*, 202-206
- Starkweather T. 1991. A comparison of genetic sequencing operators. In: *Proc 4th Int Conf Genetic Algorithms*. Los Altos, CA; Morgan Kaufmann Publishers, 69-76
- Syswerda G. 1989. Uniform crossover in genetic algorithms. In: *3rd Int Conf on Genetic Algorithms*, 2-9
- Szu H. 1988. Fast TSP algorithm based on binary neuron output and analog input using the zero-diagonal interconnect matrix and necessary and sufficient constraints of the permutation matrix. In: *Proc IEEE Int Conf Neural Network. II*; 259-266
- Taillard E. 1993. Benchmarks for basic scheduling problems. *European J of Operational Research*, 64: 278-285
- Talbi E G, Hafidi Z, Geib J M. 1998. A parallel adaptive tabu search approach. *Parallel Computing*, 24: 2003-2019
- Tan-Has T and Chang W. 1998. A hybrid strategy for Gilbert's channel characterization using gradient and annealing techniques. *Int J of system science*, 29(6): 579-585
- Tarek M N, Albert Y Z. 1994. Toward generating neural networks structures for function approximation. *Neural Networks*, 7(1): 89-99
- Tokuda I, Aihara K and Nagashima T. 1998. Adaptive annealing for chaotic optimization. *Physical Review E*, 58(4): 5157-5160
- Tokuda I, Nagashima T and Aihara K. 1997. Global bifurcation structure of chaotic neural networks and its application to traveling salesman problems. *Neural Networks*, 10(9): 1675-1690
- Van Laarhoven P, Aarts E and Lenstra J. 1992. Job shop scheduling by simulated annealing. *Ops Res*, 40: 113-125
- Varanelli J M and Cohoon J P. 1999. A fast method for generalized starting temperature determination in homogeneous two-stage simulated annealing systems. *Computers & Operations Research*, 26: 481-503
- Varsek A, Urbancic T, Filipic B. 1993. Genetic algorithms in controller design and tuning. *IEEE Trans Systems, Man and Cybernetics*, 23(5): 1330-1339
- Verhoeven M G A. 1998. Tabu search for resource - constrained scheduling. *European Journal of Operational Research*, 106: 266-276
- Vigo D and Maniezzo V. 1997. A genetic/tabu thresholding hybrid algorithm for the process allocation problem. *J of Heuristics*, 3(2): 91-110
- Visioli A. 1999. Fuzzy logic based set-point weight tuning of PID controllers. *IEEE Trans Systems, Man and Cybernetics*, A29(6): 587-592
- Wang L, Li W F, Zheng D Z. 2001. A class of hybrid strategy for adaptive IIR filter design. In: *The 8th I CONIP'2001, Shanghai*

- Wang L, Li W F, Zheng D Z. 2001. Design high-order digital differentiator with simulated annealing. In: The 5th ICEMI'2001, Guilin
- Wang L and Wang M. 1997. A hybrid algorithm for earliness-tardiness scheduling problem with sequence dependent setup time. In: Proc. Of the 36th IEEE Conf on Decision and Control, 2; 1219-1222
- Wang L, Smith K. 1998. On chaotic simulated annealing. IEEE Transactions on Neural Networks, 9 (4); 716-718
- Wang L, Zheng D Z. 2000. A class of hybrid optimization strategy for parameter estimation and PID tuning. Submitted to Cybernetics and Systems
- Wang L, Zheng D Z. 2001. A modified genetic algorithm for job shop scheduling. International Journal of Advanced Manufacturing Technology (accepted)
- Wang L, Zheng D Z. 2000. An effective modified genetic algorithm for flow-shop scheduling problems. Submitted to Information Sciences
- Wang L, Zheng D Z. 2001. An effective optimization strategy for job-shop scheduling problems. Computers & Operations Research, 28(6); 585-596
- Wang L, Zheng D Z. 2000. Global derivative-free training for feed-forward neural networks. In: The 3rd Asian Control Conference, Shanghai, 1570-1575
- Wang L, Zheng D Z. 2000. Optimizing flow-shop scheduling problems by a class of hybrid strategy. Submitted to European J of Operational Research
- Wang S, Tsai C M, et al. 1991. Hopfield nets with time-varying energy function for solving the traveling salesman problem. In: Proc IJCNN91. Baltimore, 807-812
- Wang Z, Massimo C D, Tham M T, Morris A J. 1994. A procedure for determining the topology of multilayer feedforward neural networks. Neural Networks, 7(2); 291-300
- Wang Z, Tham M T and Morris A J. 1992. Multilayer feedforward neural networks; a canonical form approximation of nonlinearity. Int J Control, 56(3); 655-672
- Watrous R L. 1987. Learning algorithms for connectionist networks; applied gradient method of nonlinear optimization. In: Proc IEEE 1st Int Conf Neural Networks. San Diego, CA, 619-627
- Whitley D, Gordan V and Mathias K. 1994. Lamarckian evolution, the baldwin effect and function optimization. In: Davidor Y, et al eds. Parallel solving from nature—PPSN III, Berlin: Springer, 6-15
- Widmer M, Hertz A. 1989. A new heuristic method for the flow shop sequencing problem. European Journal of Operational Research, 41; 186-193
- Widrow B. 1960. An adaptive adaline neuron using chemical memistors. Stanford Electronics Laboratory Technical Report
- Wilson V and Pawlay G S. 1988. On the stability of the TSP problem algorithm of Hopfield and Tank. Biol Cybern, 58; 63-70
- Wolpert D H, Macready W G. 1997. No free lunch theorems for optimization. IEEE Trans Evolutionary Computation, 1(1); 67-82

- Wong K P and Wong Y W. 1996. Combined genetic algorithm/simulated annealing/fuzzy set approach to short-term generation scheduling with take-or-pay fuel contract. *IEEE Trans on Power Systems*, 11 (1): 128-136
- Xu L, Klasa S and Yuille A. 1992. Recent advances on techniques of static feedforward networks with supervised learning. *Int J of Neural Systems*, 3: 253-290
- Xu X and Tsai W T. 1991. Effective neural algorithms for the traveling salesman problem. *Neural Network*, 4: 193-205
- Yang G Z, Gu B Y and Dong B Z. 1993. Theory of the amplitude-phase retrieval in any linear transform system and its applications. *Int J Mod Phys B*, 7: 3152-3224
- Yang H T, Huang C M and Huang C L. 1996. Identification of ARMAX model for short term load forecasting; an evolutionary programming approach. *IEEE Trans on Power Systems*, 11(1): 403-408
- Yao X, Liu Y and Lin G. 1999. Evolutionary programming made faster. *IEEE Trans Evolutionary Computation*, 3(2): 82-102
- Yao X. 1993. A review of evolutionary artificial neural networks. *Int J Intelligent Systems*, 8: 539-567
- Yen J, Liao J C, Bogju L, et al. 1998. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Trans on SMC*, 28(2): 173-191
- Yong L, Lishan K and Evans D J. 1995. The annealing evolution algorithm as function optimizer. *Parallel Computing*, 21(3): 389-400
- Zhai J H, Yan Y B, Jin G F, et al. 1998. Global/local united search algorithm for global optimization. *Optik*, 108(4): 161-164
- Zhao M. 1997. A hybrid algorithm for identifying nonlinear dynamical systems based on genetic algorithm and steepest decent algorithm. *High Technology letters*, 7(6): 31-36
- Zhou C S and Chen T L. 1997. Chaotic annealing for optimization. *Physical Review E*, 55(3): 2580-2587
- Zhou D, Yasuda K and Yokoyama R. 1997. A method to combine chaos and neural-network based on the fixed point theory. *Transactions of IECE of Japan*, 117-C (5): 599-665
- 方剑, 席裕庚. 1996. 神经网络结构设计的准则和方法. *信息与控制*, 25(3): 156-164
- 王子才, 张彤, 王宏伟. 1999. 基于混沌变量的模拟退火优化方法. *控制与决策*, 14(4): 381-384
- 王凌, 王雄. 2000. 间歇化工过程最优化的研究进展. *清华大学学报*, 40(S2): 265-269
- 王凌, 李文峰, 郑大钟. 非最小相位系统的控制器的优化设计. *自动化学报*, Vol. 28
- 王凌, 李文峰, 郑大钟. 2001. 基于一类混合策略的模型参数估计和控制器参数整定研究. *控制与决策*, Vol. 16
- 王凌, 闫铭等. 2001. 高维复杂函数的一类有效混合优化策略. *清华大学学报*, Vol. 41, No. 9
- 王凌, 郑大钟. 2001. 求解同顺序加工调度问题的一类改进遗传算法. *系统工程理论与实践*.
- 王凌, 郑大钟. 2000. Meta-heuristic 算法研究进展. *控制与决策*, 15(3): 257-262
- 王凌, 郑大钟. 1999. TSP 及其基于 Hopfield 神经网络优化的研究. *控制与决策*, 14(6): 669-674
- 王凌, 郑大钟. 1998. TSP 问题次优化求解方法的比较. *控制与决策*, 13(1): 79-83

- 王凌,郑大钟. 2001. 一种 GASA 混合优化策略. 控制理论与应用, Vol. 18, No. 4
- 王凌,郑大钟. 2000. 一种基于退火策略的混沌神经网络优化算法. 控制理论与应用, 17(1): 139-142
- 王凌,郑大钟. 1998. 一类 GASA 混合策略及其收敛性研究. 控制与决策, 13(6): 699-672
- 王凌,郑大钟. 1998. 一类批量可变流水线调度问题的研究, 见: 98'中国控制会议. 长沙: 国防大学出版社, 491-494
- 王凌,郑大钟. 2000. 邻域搜索算法的统一结构和混合优化策略. 清华大学学报, 40(9): 125-128
- 王凌,郑大钟. 1999. 径向基函数网络结构的混合优化策略. 清华大学学报, 39(7): 50-53
- 王凌,郑大钟. 1998. 前向网络的两种混合学习策略. 清华大学学报, 38(9): 95-97
- 王凌,郑大钟. 2000. 基于 Cauchy 和 Gaussian 分布状态发生器的模拟退火算法. 清华大学学报, 40(9): 109-112
- 王凌,郑大钟. 2000. 基于一类非线性特性的 FNN 训练算法. 控制与决策, 15(1): 19-22
- 王凌,郑大钟. 2000. 基于不同状态发生器的模拟退火算法性能研究. 见: 中国控制会议, 香港, 430-434
- 王凌,郑大钟. 2001. 基于遗传算法的 Job Shop 调度研究进展. 控制与决策, Vol. 16
- 王凌,郑大钟. 2001. 混合优化策略统一结构的探讨. 控制与决策, Vol. 16
- 王凌,郑大钟. 2001. 混沌优化方法的研究进展. 计算技术与自动化, No. 1
- 王凌,郑大钟. 1997. 模拟退火算法求解 Flow-shop 问题的研究. 见: 中国控制与决策学术年会. 沈阳: 东北大学出版社, 390-394
- 王凌. 1999. 混合优化策略和神经网络中若干问题的研究. 北京: 清华大学博士学位论文
- 王凌. 1997. 随机优化算法和混合优化策略. 北京: 清华大学硕士学位论文
- 韦有双,杨湘龙,冯允成. 2000. 一种新的求解 Flow Shop 问题的启发式算法. 系统工程理论与实践, 20(9): 41-47
- 刘岩,韩承德,王义和等. 1996. 模拟退火的背景与单调升温的模拟退火算法. 计算机研究与发展, 33(1): 4-10
- 刘勇,康立山,陈毓屏. 1998. 非数值并行算法——遗传算法. 北京: 科学出版社
- 庄镇泉,王煦法,王东生. 1992. 神经网络与神经计算机. 北京: 科学出版社
- 张讲社,徐宗本,梁怡. 1997. 整体退火遗传算法及其收敛充要条件. 中国科学, E 辑, 27(2): 154-164
- 张彤,王宏伟,王子才. 1999. 变尺度混沌优化方法及其应用. 控制与决策, 14(3): 285-288
- 李兵,蒋慰孙. 1997. 混沌优化方法及其应用. 控制理论与应用, 14(4): 613-615
- 杨行峻,郑君里. 1992. 人工神经网络. 北京: 高等教育出版社
- 孟庆春. 1995. 基因算法及其应用. 济南: 山东大学出版社
- 郑大钟. 1990. 线性系统理论. 北京: 清华大学出版社
- 郑学哲,王凌等. 1998. 实现 ICF 均匀照明的二元光学器件的混合优化设计. 中国激光 A, 25(3): 265-269
- 郑学哲. 1997. 二元光学技术在惯性约束核聚变靶面均匀照明中的应用研究. 北京: 清华大学博士学位论文
- 姚新,陈国良. 1990. 模拟退火算法及其应用. 计算机研究与发展, 7: 1-6
- 姜波,汪秉文. 2000. 基于遗传算法的非线性系统模型参数估计. 控制理论与应用, 17(1): 150-152

- 席裕庚,柴天佑,恽为民.1996.遗传算法综述.控制理论与应用,13(6):697-708
- 徐心和.1990.旅行商问题的一种新解法.东北大学学报,11(1):68-73
- 康立山,谢云,尤矢勇等.1998.非数值并行算法——模拟退火算法.北京:科学出版社
- 梁化楼,戴贵亮.1995.人工神经网络与遗传算法的结合:进展与展望.电子学报,23(10):194-200
- 黄炯,邬永革.1996.基于遗传算法的系统在线辨识.信息与控制,25(3):171-176
- 彭宏,王兴华.1997.具有 Elitist 选择的遗传算法的收敛速度估计.科学通报,42(2):144-147
- 焦李成.1992.神经网络系统理论.西安:电子科技大学出版社
- 潘卫东.1994.利用遗传技术辅助设计人工神经网络.模式识别与人工智能,7(1):72-77