

中北大学

硕士学位论文

Hopfield神经网络在TSP问题中的应用

姓名：兰兆青

申请学位级别：硕士

专业：应用数学

指导教师：白艳萍

20080101

Hopfield 神经网络在 TSP 问题中的应用

摘要

旅行商问题是组合优化领域中的一个典型问题，该问题的核心就是要求出一个包含所有 n 个城市的具有最短路程的环路。虽然它陈述起来很简单，但求解却很困难，并且已经被证明是 NP 完全问题。但它确实广泛存在，且是诸多领域内出现的多种复杂问题的集中概括和简化形式。因此提出一种有效地解决 TSP 问题的算法有着较高的理论意义和实际应用价值。

本文首先从现有求解 TSP 的算法入手，通过研究大量的参考文献，了解了各种算法的主要思想，并对各种算法进行了整理和分类。研究发现遗传算法、模拟退火算法和蚁群算法这三种算法在解决 TSP 问题中表现出了一定的优势，并且在实际的问题的求解中得到了广泛的应用。接下来本文就对这三种算法进行了深入的研究，并进行了编程实现。文中分别用这三种算法解决了 48 个城市的 TSP 问题。从结果中作者发现模拟退火算法的优化过程较长；蚁群算法同样是搜索时间比较长，也容易陷于局部最优解，使搜索停滞。遗传算法实际应用时易出现早熟收敛和收敛性差等缺点。如何快速准确的解决 TSP 问题成为了现在 TSP 算法研究中的一个难点。作者提出了一种基于 Hopfield 神经网络的算法，由于神经网络是并行计算的，其计算量不随维数的增加而发生指数性“爆炸”，因而对于优化问题的高速计算特别有效。在实际的实验过程中作者发现该算法从在一个致命的缺点就是网络极不稳定，经常得不到结果。为此，作者在对现有算法进行了改进。经过研究发现最终结果的准确性很大程度上取决于初始参数的设置。在认识到这一点后，对每个参数对结果的影响进行了分析，最后给出了参数的合理设置方法。本文还对能量函数进行了改进，使得问题的求解更加快速准确。对‘出现重复解的问题’进行了解决采用了一种从固定起点出发的办法。最后应用该算法解决了西安旅游问题，首先结合西安旅游地图，根据具体的旅游问题给出了网络的能量函数，进而构建了一个 Hopfield 神经网络。选取了西安的著名旅游景点，对景点进行了变换和归一化，对算法编程进行了实验。实验结果表明，该方法对 10 个景点和 15 个景点的迭代次数大都集中在 250~350

之间，说明该方法对于处理旅游路线的选择问题是行之有效的。

关键词：旅行商，组合优化，遗传算法，模拟退火算法，蚁群算法

The application of Hopfield neural network in the TSP problem

Lan Zhaoqing

Tutor :Bai Yanping

Abstract

TSP is a typical problem in the field of combinatorial optimization, the core of this problem is to find the shortcut including all the cities. Although it is very simple to state, it is not so easy to solve, what is more, it has been proved to be the NP-complete problem. But it does exist extensively, and it is the central generalize and simplified form of many complex issues. Therefore propose an effective solution to the problem of the TSP algorithms have a higher theoretical and practical value.

This paper chose the existing algorithms of TSP to start. By studying a large number of references; to understand the main ideas of a variety of algorithms and organize a variety of algorithms, then make the classification. It is found that genetic algorithm, simulated annealing algorithm and ants algorithm showed a certain advantages in solving the TSP, and they are widely used in the solution of practical problems. Then, to this paper, the three algorithms is deeply researched and programmed. The three algorithms were separately used to solve the 48 cities TSP problems. From the results of the simulated, the author found that annealing's process of optimization is longer; ant algorithm is also relatively long, but also easy in a local optimal solution to search stagnation; At the practical application of premature, GA appears the shortcomings of easy to premature convergence and convergence of poor. How fast and accurate to solve the problem has now become a difficult point in the problem of TSP algorithm. The author presents a Hopfield neural network algorithm, the neural net work is the parallel computing, and its calculation is not the dimension of the increase in the index of "explosion" and therefore, the optimization of high-speed computing particularly effective. In the actual process of the algorithm from the author found in a fatal drawback is the network

extremely unstable, often not the results. To this end, the author of the existing algorithms made improved. After the final results of the study found that a large extent depends on the accuracy of the initial parameters set. In recognition of this point, the results of an analysis of the parameters are given a reasonable set of methods. This article's energy function was also improved, making the solution more quickly and accurately. Finally, the author brought up a method of fixed starting point of departure for resolving the "duplication of the problem" from the approach. Finally application of the method to solve the Xi'an tourism, first, based on the Xi'an tourism map, according to the specific issue of tourism, the network's energy function is given; thereby building a Hopfield neural networks. Xi'an has been selected the well-known tourist attractions, the attractions for the transformation and normalization of the algorithm programming experiment. Experimental results show that the method on 10 scenic spots and attractions of the 15 mostly concentrated in the number of iterative 250 to 350, and the method to deal with the problem of the route of choice is effective.

Key words: traveling saleman, combinatorial optimization, genetic algorithm, simulated annealing algorithm, ants algorithm

第一章 绪论

1.1 问题的提出

TSP 问题的原始定义是：有一个推销员，要到 n 个城市推销商品，他要找出一个包含所有 n 个城市的具有最短路程的环路。TSP 的历史很久，最早的描述是 1759 年欧拉研究的骑士周游问题，即对于国际象棋棋盘中的 64 个方格，走访 64 个方格一次且仅一次，并且最终返回到起始点。TSP 由美国 RAND 公司于 1948 年引入，该公司的声誉以及线性规划这一新方法的出现使得 TSP 成为一个知名且流行的问题。1856 年，哈密尔顿推出了二十城游戏(Lcosian Game)，第一次使用旅行商问题(Traveling Salesman Problem)一词可能是在 1931 或 1932 年，当时 A.W Tucke 从普林斯顿大学 Hassler Whitney 那里听到了 Traveling Salesman Problem(旅行商问题)一词。不管是被称为骑士周游问题(Knights Tour)、投递问题(Messenger Problem)或是旅行商问题(Traveling Salesman Problem)，这一问题已让很多著名的数学家为之苦苦思索了几个世纪。同样的问题在中国还有另一个描述方法：一个邮递员从邮局出发，到所辖街道投递邮件，最后返回邮局，如果他必须走遍所辖的每条街道至少一次，那么他应如何选择投递路线，使所走的路程最短？这个描述之所以称为中国邮递员问题，因为是我国学者管梅古教授于 1962 年提出的这个问题并且给出了一个解法。

TSP 问题是组合优化领域中的一个典型问题，涉及求多个变量的函数的最小值。虽然它陈述起来很简单，但求解却很困难，它一直是运筹学中最富挑战性的问题之一，并且已经被证明是 NP 完全问题。对于具有 n 个城市的 TSP 问题，其可能的路径数目为 $(n-1)! / 2$ ，至今尚未找到有效的求解方法，在理论上枚举法可以解这一问题，但是当 n 较大时，解题的时间消耗会使枚举法显得没有任何实际价值。因此寻求一种求解时间短，能满足实际问题精度要求的解，成为解决该问题的主要途径^[1]。

1.2 TSP 的应用和价值^[2]

旅行商问题（TSP）是组合优化中的古典问题，涉及求多个变量的函数的最小值。它是对给定的 N 个城市找出一条最短的路径，这条路径对每个城市都访问一次，最后还回到出发的那个城市。这与我们外出旅游十分相似，TSP 问题中对每个城市都访问一次对应于旅游问题中对每个景点都游览一遍。因此，可以说 TSP 问题代表了一类组合优化问题，因而探讨 TSP 问题的有效求解方法有着广泛的应用前景。

旅行商问题在现实生活中应用十分广泛，可被用来模拟生活中旅行商所要旅行的地区问题。顶点表示旅行商所要旅行的城市（包括起点），边的耗费给出了在两个城市旅行所需的时间（或花费）旅行路径表示当旅行商周游了所有的城市再回到出发点时所走的路线。

旅行商问题还可用来模拟其他问题。例如要在一个金属薄片或印刷电路板上钻许多孔。孔的位置已知，这些孔由一个机器钻头来钻，它从起始位置开始，移动到每一个钻孔位置钻孔，然后回到起始位置。总共花的时间是钻所有孔的时间与钻头移动的时间。钻所有孔所需的时间独立于钻孔顺序，然而，钻头移动时间是钻头移动距离的函数。因此，我们在进行工作的过程中希望在各种合理路径中找到最短的移动路径。TSP 很自然地作为大量的运输和后勤应用的一个子问题提出。例如：在校区内如何安排校车路线接送学生的问题，因为它对 Merrill Flood（二十世纪四十年代初一位研究 TSP 的先驱者）的研究提供了源动力，所以这个运输应用对 TSP 有着重要的历史意义。从 1940 开始 TSP 的第二个应用包括从某地运输农耕设备到另一个地方去检测土壤，这吸引了孟加拉的 P.C.Mahalanobis 和爱荷华州的 R.J.Jesson 的数学研究。更多新近的应用包括电报公司中呼叫服务的时序安排、货舱中栈式起重机的路线安排、收邮包的卡车行车路线安排等等。

虽然运输问题是 TSP 最自然的应用，但由于其模型的简单性，使得 TSP 在其他领域都有着有趣的应用。一个经典的例子是如何安排机器在一块电路板或其他物体上钻孔，其中需要钻的孔可以看成是各个城市，而旅行的费用就是钻头从一个孔移到下一个孔所花的时间，虽然钻孔的技术不断发展，但无论何时，只要钻机设备的移动时间在制

造业的过程中占据显著的地位，TSP 在减少费用上就扮演了一个非常重要的角色。

以下给出一些当前 TSP 的应用^[3]：

半导体制造厂家使用 CONCORDE（一种主要用于计算 TSP 的程序代码）中链式 Lin-Kernighan 启发式样算法来优化整个电路的扫描链路。用来检测的扫描链路包含在一个芯片当中，它可以最小化时间或者能量的消耗。

TSP 的另一个应用是在一个给定的区域中，安排收集投币式公用电话中的硬币。 CONCORDE 中链式 Lin-Kernighan 启发式算法的改进版可以解决各种各样的硬币收集问题，因为单行道的存在以及城市交通中的其他问题使得从 X 到 Y 的旅行费用和从 Y 到 X 的旅行费用相等的假设变的不切实际，所以需要改进来解决这些问题。

1.3 TSP 问题的研究现状^[4]

在组合优化问题中最有代表性经典难题就是 TSP 问题(Traveling Salesman Problem,TSP),即旅行商问题。该问题是运筹学里的著名命题，根据各城市间的距离 d_{ij} 所形成的距离矩阵的对称性可分为对称 TSP 和非对称 TSP, 其中前者的求解难度远大于后者，在问题复杂性^[5]方面属于 NP-hard 类问题。TSP 问题是目前研究最为广泛的组合优化问题之一。回顾 TSP 问题的研究历史,可谓是“百花齐放，百家争鸣”！对于 TSP 问题，最早也最自然的想法就是采用穷举法。然而，所谓最优解的必然存在性和必能找到的特点是建立在问题的规模较小的情况下。当可行解集合中的点个数较少时，通过最直观的穷举法很容易得到最优解；但倘若可行解集合中的有限点数目逐渐增多，此时如果仍采用完全枚举、判别、比较、选择的步骤则无疑是十分困难和不可取的,所需要的时间和空间是庞大得惊人的,如城市数为 50 时,计算时间已达 5×10^{48} 年。对于 n 个城市,各种可能的路径及其距离之和的计算量将正比于 $n!/2$, 随 n 的不断增长, 将出现“指数爆炸”现象。故 TSP 在算法复杂性^[5]方面是属于指数时间算法的，所以穷举法不能适应大规模的 TSP 的问题。

穷举法是一种最优算法,但在实际问题中,各种最优算法的计算时间随问题的规模增加而以指数速度增加使人难以忍受,也将超过目前计算机的运算能力,所以一种相对于最优算法的启发式算法应运而生。文献^[6]给出了启发式算法的定义:启发式算法是一种技术,这种技术使得在可接受的计算费用内去寻找最好的解,但不一定能保证所得解的可行性和最优性,甚至在多数情况下,无法阐述所得解与最优解的近似程度。

最早的启发式算法可以追溯到 40 年代末期,这些算法能快捷有效地解决工程中的一些实际组合问题。然而到了 60-70 年代间,由于对数学模型及最优解算法的重视程度日益提高,先前提出的一些启发式算法都被称为“method of quick and dirty”。随着 70 年代计算复杂性理论的逐渐完善,人们对最优解的认识有了新的转变,所以促使了不考虑算法所得解与最有解的偏离程度的启发式算法再次受到重视,这是一个重要的转折点。早期的启发式算法^[7]主要有步算法、2-opt 算法^[8]、数学规划算法和解空间松弛算法等。

现代智能优化算法是 80 年代初兴起的启发式算法。1985 年, Hopfield 和 Tank 用连续型 Hopfield 网络求解 TSP,同年发表了著名论文《“Neural” computation of decisions in optimization problems》,开创了基于神经网络求解 TSP 问题的新纪元。文献^[9]对什么是 TSP 问题做了详细的阐述,为后来者取得突破性成果打下了扎实的基础。Hopfield 等人所提出的将组合优化问题映射到神经网络中的思想具有改革性和开创性,但所提出的 TSP 能量函数模型以及参数选取上的不足也是存在的。文献^[10]就 Hopfield 网络求解 TSP 问题的算法稳定性进行了研究,而文献^[11]则从 TSP 能量函数入手,对 Hopfield 网络的动力学系统本质做了研究,进而给出了 Hopfield 网络无法收敛到有效解答的根本原因并提出了改进后的 Aiyer 能量函数和参数选取原则。此间,文献^[12]也同样提出了又一种改进后的 Abe 能量函数和参数选取原则。1991 年 G.C.Fox^[13]首次给出了 physical computation 的定义以及它的五大分支,即模拟退火(simulated annealing, SA)、人工神经网络(artificial neural network, ANN)、确定性退火(deterministic annealing, DA)、弹性网络(elastic network, EN)和遗传算法(genetic algorithm, GA)。对诸如 TSP 等组合优化问题中的难解类问题,

对其近似算法的研究一直是一个世界性的重要课题。现代智能优化算法中的这些算法虽然思想各异却有着共同的目标—求 NP-hard 组合优化问题的全局最优解。虽然 NP-hard 理论^[5]限制了这些算法只能以启发式的思想去求解问题，但是相对于早期的启发式算法而言，解答的有效性和全局最优的概率都明显高于早期启发式算法，因而是值得探索和研究的。

传统的一些非启发式方法，从计算复杂度理论的角度考虑，其不足显而易见，因而一种称为启发式的算法应运而生，其基本思想简而言之即：在可以接受的时间和空间复杂度的限制下去寻求最优的解。这种思想对解决复杂组合优化问题而言无疑是一种新的探索。它为传统的主要基于数学的方法之外寻找到了一条更宽更佳的新途径。而从实际效果来看，以禁忌算法^[14]、模拟退火算法、遗传算法、人工神经网络算法为代表的现代智能优化算法发展至今，成果斐然。

目前世界上研究 TSP 最主要遇到的瓶颈问题就是如何避免陷入局部最优以及计算效率的问题。为了解决这些难题，20 世纪 90 年代起，一些更新的思想和算法逐渐形成。由群居性昆虫行为特性获得灵感的蚂蚁算法（又称蚁群算法）是目前研究的热点；又如由混沌现象受启发的一系列新尝试：将混沌机制和启发式搜索方法、人工神经网络、模拟退火等相交叉结合，建立更多优质高效的算法来提高算法的计算效率和对全局最优点的获取能力。下面将从以下五方面来阐述以 TSP 为代表的组合优化问题的研究现状。

1.3.1 模拟退火算法(SA)

模拟退火算法^[15,16]是局部搜索算法的扩展。理论上来说它是一个全局最优算法。与贪婪策略不同的是模拟退火算法摆脱了局部最优的束缚，它的解逼近全局最优。

模拟退火算法是从物理和化学的退火过程类推得来的，由 Metropolis 算法和退火过程组成。Metropolis 算法^[17]于 1953 年提出，是一种改进的 Monte Carlo 方法。1983 年，文献^[18]将退火的思想成功应用在组合优化问题中，建立了模拟退火算法。该算法为解决

凹平面最优问题提供了一个强有力的工具。文献^[18]提出了一个基于模拟退火算法求解旅行商问题的改进算法，并在并行设计环境 Multi-pascal 中加以实现。

1.3.2 遗传算法(GA)

仿生物界的遗传变异的原理和方法, Holland^[19]等人于 1975 年提出了遗传算法思想。这种算法是以达尔文的生物进化论为启发而创建的, 是基于生物进化中自然选择、适者生存和物种遗传思想的搜索算法。文献^[20]采用交换算子操作和模拟退火思想对遗传算法进行改进, 求解中国 TSP 问题, 显著提高了算法的优化效率。文献^[21]利用遗传算法的机理提出了求解 TSP 问题的一整套进化策略, 并对算法的有效性进行了分析。文献^[22]在求解 TSP 问题时引入了小生境技术, 提高了遗传算法处理多峰函数优化问题的能力。文献^[23]提出了新倒位算子, 提出了新的遗传算法求解 TSP 问题。文献^[24]提出了 TSP 的一种改进遗传算法, 通过在传统 GA 中引入“幼代”及其成长过程, 解除了两种能力间的制约关系, 求解质量显著提高。文献^[25]将时间窗约束转化为目标约束, 采用序列编码设计了有时间约束旅行商问题的启发式遗传算法。

1.3.3 人工神经网络算法(ANN)

1985 年 Hopfield 等人用 Hopfield 网络^[9]为解决 TSP 难题开辟了一条崭新的途径, 获得了巨大的成功。CHNN 是一种重要优化方法且易于硬件实现但却难以克服陷入局部极小。Wilson^[10]等人通过基于 CHNN 的 TSP 求解, 表明其计算量大、鲁棒性差、优化性能和可靠性差。而多层前向网络 BP 由于学习算法存在不足, 往往会使网络稳定在误差

局部极小点即所得到的结果不是最优解。

为了解决这些问题,许多国家的学者都在尝试用新的方法和思想建立新的人工神经网络以改善 Hopfield 网络的动力学特性。在我国,将神经网络算法应用在优化问题上的研究也在不断深入推进。文献^[26]采用了玻尔兹曼网络求解 TSP 问题,网络收敛速度较快。文献^[27]对多路旅行商问题(MTSP)根据出发城市的不同和返回情况分成了四个子问题,建立了各问题的神经网络计算能量函数和迭代公式,提出了智能化的优化方法。文献^[28]提出了采用 Hopfield 网络解 TSP 的改进算法。此外,文献^[29]提出了把混沌机制和神经网络相融合形成一类有学习能力的系统,提出了混沌神经网络模型。文献^[30]又提出了以混沌模拟退火为退火机的瞬态混沌神经网络。文献^[31]提出了一种基于退火策略的混沌神经网络优化算法较大程度提高了优化、时间和对初值的鲁棒性能。文献^[32]提出了适应性混沌模拟退火算法,改进了原先的混沌模拟退火算法。应用于 TSP 问题中,求解效果显著。

1.3.4 蚁群算法(ACA)

蚁群算法是继禁忌算法、模拟退火算法、遗传算法、人工神经网络算法之后的又一种应用于组合优化问题的现代优化启发式算法。蚁群算法就是利用群集智能解决组合优化问题的典型例子^[33]。20世纪90年代,意大利学者 Dorigo^[34]等人首先提出了蚁群算法(ant colony algorithm),并将蚁群算法应用于 TSP 问题,取得了较好的效果。随后,不少研究人员对该算法提出了改进算法,不断提高蚁群算法的适应性。文献^[35]对蚁群算法给出了改进思路及算法描述,并将改进后的算法应用于景点群最优路线问题。文献^[36]提出了一种基于蚁群算法的 TSP 问题分段求解算法。文献^[37]进一步提出了基于蚁群算法的多层次前馈神经网络,并通过仿真实验表明了该网络兼有神经网络广泛映射能力和蚁群算法

快速全局收敛的性能。文献^[38]对蚁群算法进行了改进,提出了智能蚂蚁算法,可以在减少计算量的同时取得更好的搜索结果。而文献^[39]则创新性地提出了一种带聚类处理的并行蚂蚁系统,极大地提高了蚂蚁系统的收敛速度。目前又有一种更加新的基于种群行为的优化算法被提出^[40]。该算法称为自由搜索(Free Search, FS),它具有稳定性和鲁棒性,适合于解决对现实世界中的“黑箱”实现优化工作。可见,目前采用蚁群算法及其相应的改进算法来求解 TSP 为代表的 NP 问题已成为了一种热点研究方向。

1.3.5 其他算法

除了上述主要算法外,目前还有各种其他不同的新算法、新思路来解决 TSP,以期不断提高组合优化问题的求解效率、全局最优解的获取能力。比如文献^[41]发展了 Fox^[13]的观点,给出了以建立具有“非线性、不确定、追求满意解”的超图灵模型为最终目标;以利用自然规律为手段的“按自然法则计算”的思想求解 TSP。文献^[42]采用了基于多项式时间的进化算法来求解 TSP 问题。文献^[43]采用了免疫算法来解决 TSP 问题,并证明免疫算法是一种收敛速度快、收敛性好的算法。文献^[44]提出了采用混沌神经动力学与 2-opt 算法相结合的新算法来求解 TSP 问题。

1.3.6 未来可研究的问题

展望未来的技术难点,解决 TSP 问题今后的研究热点从攻克的问题角度来看可以概括为以下三方面:

- 1) 收敛效率问题(计算的迭代次数)
- 2) 解的可行性问题(包括全局最优点和局部最优点)

3) 解的最优化问题(全局最优点)

对于采用人工神经网络算法来求解 TSP 问题时可研究的问题主要体现在两个方面：

1) 如何提高神经网络的求解能力(计算速度、解的可行性和最优化)

2) 如何丰富网络的动力学性能, 提高神经网络的稳定性

此外, 值得一提的是文献^[45]提出了动态突触联想记忆概念, 并指出动态突触网络模型能够再现记忆模式, 并在两种记忆模式之间的切换表现为间歇性的特征。这一现象可以反映出实际神经系统的灵活性, 并且能够反映出实际网络在接收和响应新的外部激励时的稳定性。目前国内对动态突触神经网络(Dynamic Synapses Neural Network, 简写为 DSNN)的研究和应用极少, 而该网络从理论上分析可知: 它将更大程度的丰富神经网络的动态特性, 使得网络更加稳定、具有更强的最优解的搜索能力, 因此是一个新的可研究方向。

1.4 本文的主要安排

本文主要对 TSP 求解问题进行了研究, 并对各种算法进行了详细的介绍。选取了比较有代表性的算法进行了编程实现, 在比较了这三种算法的基础上提出了一种基于 Hopfield 神经网络的 TSP 问题的求解方法。并对现有的算法进行了改进得到了比较满意的结果。最后应用改进后的算法对西安旅游问题进行了求解并得到了最优的旅游路线。具体的安排如下

第一章首先介绍了 TSP 问题的由来和其应用价值, 接着详细介绍了 TSP 问题的研究现状。

第二章首先对 TSP 的问题进行了分析, 介绍了其作为 NP 完整问题的求解难点。接着建立了数学模型并对 TSP 问题进行了分类。最后详细介绍了各种求解 TSP 问题的算法。

第三章选择了遗传算法、模拟退火算法和蚁群算法这三个比较有代表性的算法进行了编程实现。针对每个算法分别介绍了主要算法思想和实现步骤, 最后给出了其求解 TSP 问题的结果。

第四章提出了一种基于 Hopfield 神经网络的求解算法。在介绍了 Hopfield 神经网络模型之后，详细介绍了如何实现其对 TSP 问题的求解。在分析该方法的缺陷之后分别就参数设置和能量函数对现有算法进行了改进，并针对有重复解的现象对算法提出了进一步的改进。最后给出了改进算法求解的结果。

第五章对西安旅游这样一个实际的问题进行了应用，并给出了最佳旅游路线。

第六章对本文的主要工作进行了总结并提出了进一步的工作展望。

第二章 对旅行商问题的分析和一些经典算法的介绍

TSP (Traveling Salesman Problem) 问题属于 NP 完全问题，若用穷举搜索算法则需考虑所有可能的情况、找出所有的路径，再对其进行比较来找到最佳的路径。这种方法随着城市数 n 的上升算法时间随 n 按指数规律增长，即存在所谓的指数爆炸问题。事实上，在 n 个城市的 TSP 问题中，一条有效的路径可以看成 n 个城市的一种排列。 n 个城市有 $n!$ 种排列，注意到两个顺序完全相反的方案其行程相同，而对一种排列从哪个城市出发都可以，所以有效路径的方案数目为 $R_n = \frac{n!}{2n}$ ，例如： $R_4=3$ ， $R_5=12$ ， $R_6=120$ ， $R_{10}=181440$ ， $R_{30}=4.4 \times 1030$ ， $R_{60}=6.93 \times 1078$ ， $R_{100}=4.67 \times 10155$ 。可见路径总数随 n 增大而急剧增长，当城市数目增加到一定的程度，计算量增加到无法进行的地步。

2.1 TSP 问题的描述及数学模型^[2,47]

旅行商问题(Traveling Salesman Problem, 简记 TSP)，自 1932 年 K · Menger 提出以来，已引起各领域许多研究者的兴趣。它是组合数学中的一个古老而又困难的问题，也是组合优化中研究最多的问题之一，但至今尚未彻底解。其描述为：一推销员要到若干城市推销货物，从城市 1 出发，经过其余各城市一次且仅仅一次，然后回到出发点，求其最短行程，即寻找一条巡回路径 $T = (T_1, T_2, \dots, T_n)$ ，使得下列目标函数最小：

$$f(T) = \sum_{i=1}^{n-1} d(t_i, t_{i+1}) + d(t_n, t_1) \quad (2.1)$$

上式中 t_i 为城市号，取值在 1 到 n 之间的自然数， $d(t_i, t_j)$ 表示城市 i 和城市 j 之间的距离，对于对称式 TSP，有 $d(t_i, t_j) = d(t_j, t_i)$ 。用图论的语言描述为：在一个赋权完全图中，找出一个最小权的哈密顿圈。

令 $G = (V, E)$ 为赋权完全图， $V = \{1, 2, \dots, n\}$ 为顶点集， E 为边集，各顶点间距离 d_{ij} 已

知。 $(d_{ij} > 0, d_{ii} = 0, i, j \in V)$

设

$$x_{ij} = \begin{cases} 1 & \text{边}(i, j) \text{在最优路线上} \\ 0 & \text{其} \quad \text{它} \end{cases} \quad (2.2)$$

则 TSP 的数学模型可写成如下的线性规划形式：

$$\text{Min} Z = \sum_{i \neq j} d_{ij} x_{ij} \quad (2.3)$$

$$\text{S.t.} \left\{ \begin{array}{ll} \sum_{j \neq i} x_{ij} = 1 & i \in V \\ \sum_{i \neq j} x_{ij} = 1 & j \in V \\ \sum_{i, j \in S} x_{ij} \leq |S| - 1 & S \subseteq V \\ x_{ij} \in \{0, 1\} & i, j \in V \end{array} \right. \quad (2.4)$$

这里， $|S|$ 为集合 S 中所含图 G 的顶点个数。前面两个约束意味着对每个顶点而言，仅有一条边进和一条边出，后一条约束则保证了没有任何子回路解的产生。于是，满足上述约束条件的解构成了一条遍历所有顶点的哈密顿回路。

通常一个组合优化问题的解可能不是唯一的，即可以同时存在着多个，满足条件的赋值使得目标函数达到最大（或最小）值，但目标函数所达到的最大（或最小）值总是唯一的。

2.2 TSP 问题的分类^[47]

从问题对应到图的类型，TSP 可以分为两类：

I、城市间的距离都是对称的，它对应的是图论中的无向图； II、两个城市间的距离是非对称的，它所对应的是图论中的有向图。

从问题本身的限制条件的强弱，主要有三类：

I、不做任何限制（但是一般都要求城市间的费用不为负数），只是给出距离矩阵，求最小回路； II、要求距离间要满足三角不等式； III、定义在欧氏平面上的 TSP，即

Euclid TSP，它给出每个点在欧氏平面上的坐标，而城市间的距离就是以他们的欧式距离来定义。

从问题的多项式可解性上分，TSP 可以分成两类：

I、目前已经知道有多项式时间算法可解的，比如其距离矩阵满足特定的条件（Demidenko 条件；Kalmanson 条件；Supnick 条件等）；II、目前尚没有发现多项式时间算法可解的，而研究热点是如何寻找更多的多项式时间可解的情形。

TSP 的研究经过几十年的发展，还引申出其他扩展形式：多旅行商问题（Muti-Salesman Problem），多目标旅行商问题(Muti-Objective TSP)^[48]。

2.3 解 TSP 的算法综述

作为组合优化的一个经典问题，TSP 问题吸引了广大学者对它进行研究。这一章中我们总结了相关的一些方法^[47]，主要分为两类：一类是精确算法（完全算法）；另一类算法是近似算法或启发式算法（不完全算法）。

完全算法能保证完全搜索问题的整个解空间，从而找到最优巡回，但需要消耗 $O(n!)$ 级的运算时间；有些完全算法虽然运用一些精巧的技术来减少搜索空间，但本质上还是进行全局搜索，并没有降低运算时间复杂度。

不完全算法不能保证搜索空间的全部解空间，所以也不能保证能够找到最优解，甚至在某些实例上连解都得不到，但它们与完全算法相比却具有运算时间上的优势，即它们的运算时间复杂度只是多项式，并不随着输入规模的扩大产生“组合爆炸”。这类算法采用的是启发式策略来指导搜索，普遍比完全算法要快。下面将分别对它们做简要介绍。

2.4 精确算法

2.4.1 可解情形（特例）

对于 TSP 的一些特殊情况，业已研究出一系列非常优美的结果，如：机器排序问题（Gilmore 等，1964）、二分图情形（Lawler，1971）、平面 TSP 中的一些特例（Burkard，1989）等等。由于可解情形的结果都是成熟的定理，因此严格来说已不属于算法研究的范畴。

2.4.2 穷举搜索法（Exhaustive Search Method）

这是一种最简单但最费时的方法，因为它要把所有可能的路径方案的长度都列出来，从中选择一个最短的。该方法的优点是找到的最短路径一定是全局最优解，但其致命的缺点为惊人的计算开销。用该方法计算 n 个城市问题，所要计算的路径为 $(n-1)!/2$ 条。

2.4.3 贪婪算法^[49]（Greedy Method）

这种方法就像一个贪心的小孩在框中挑苹果那样，总是以先挑最大的为原则。在组合算法中，将每一步都取局部最优的求解方法称为贪婪法。由于贪婪法的局限性，它只能求解“单峰”的情况，对于“多峰”的复杂情形则不能得到全局最优解。因此，贪婪法用在 TSP 问题上一般可以得到局部最优解，而所得结果的好坏与城市间距离的具体情况和从哪个城市开始有关。

2.4.4 线性规划算法 (Linear Programming algorithm)

这是求解 TSP 的最早的一种算法，主要是采用整数线性规划中的割平面法，即先求解模型中由前两个约束构成的松弛 LP 问题，然后通过增加不等式约束产生割平面，逐渐收敛到最优解。

Dzntzig 等人早在 1954 年就求解过 $n=42$ 的 TSP 最优解。70 年代中期对于 TS 多面体理论的研究，产生了一些比较有效的不等式约束，如子回路消去不等式 (sub-tour elimination)、梳子不等式 (comb inequalities)、团树不等式 (clique tree inequalities) 等等。曾经报导过用割平面法在中小型机上求解 $n=318$ 规模的例子，但是，由于该方法在寻找割平面时常常要凭借经验，因此；后来很少作为一般方法使用。

2.4.5 动态规划算法 (Dynamic Programming algorithm)

记 S 为集合 $\{2,3,\dots,n\}$ 的子集， $k \in S$ ， $C(S, k)$ 为从 1 出发遍历 S 中的点并终止在 k 的最优行程。当 $|S|=1$ 时， $C(\{k\}, k) = d_{1k} (k=2,3,\dots,n)$ 。当 $|S|>1$ 时，根据最优化原理，可将 TSP 的动态规划方程写成 $C(S,k) = \min_{j \in S-\{k\}} [C(S-\{k\}, j) + d_{jk}]$ 。按方程规则可逐步迭代求解。

其主要特点是将一个问题分为若干相互联系的阶段，每个阶段进行决策优化。在这种多阶段决策优化过程中，无论其初始状态和初始决策如何，以后的最优策略只取决于由最初决策所形成的当前状态。换言之，以后诸决策对于以第一个决策所形成的状态为初始状态的过程而言，必须构成最优策略。动态规划算法的时间复杂度为 $O(n^2 \cdot 2^n)$ 、空间复杂度为 $O(n^2 \cdot 2^n)$ ，其计算复杂性显然比穷举法要显著降低，但是要找出一条最短路径仍然有巨大的计算工作量。故一般除了很小规模的问题外，几乎不予采用。

2.4.6 分支定界算法 (Branch delimitation algorithm)

分支定界算法是一种应用范围很广的搜索算法，它通过有效的约束界限来控制搜索过程，使之能向着状态空间树上有最优解的分支推进，以便尽快找出一个最优解。该方法的关键在于约束界限的选取，不同的约束界限，可形成不同的分支定界法。

(1) 以分派问题为界

通过求解相应的分派问题，得到 TSP 的一个下界，以此进行分支定界搜索。这是一种使用较多的分支定界算法。

(2) 以匹配问题为界

通过求解相应的匹配问题，得到 TSP 的一个下界，以此进行分支定界搜索。
该方法适用于对称型 TSP。

(3) 以最小 1 树问题为界

通过求解相应的最小 1 树问题，得到 TSP 的一个下界，以此进行分支定界搜索。在此基础上，Held 和 Karp (1970) 曾将问题加以转换，得到更紧的下界，有时甚至能将搜索树整个显示出来。

虽说分支定界法对于较大规模的问题并不十分有效可有时却被用来求解近似解。而且，将分支定界法与一些启发式算法相结合，常常能获得一些意外的成功。

2.5 近似算法

由于精确式算法所能求解的问题规模十分有限，实际中使用的往往都是多项式阶数的近似算法或启发式算法 (heuristics)。算法的好坏用 $C/C^* \leq \varepsilon$ 来衡量， C 为近似算法所得到的总行程， C^* 为最优总行程， ε 为最坏情况(worst case)下近似解与最优解的总行程之比所不超过的上界值。

2.5.1 插入算法 (insertion heuristics)

插入型算法可按插入规则的不同而分为若干类，其一般思想为：

Step1 通过某种插入方式选择插入边 (i, j) 和插入点 k ，将 k 插入 i 和 j 之间，形成 $\{\dots, i, k, j, \dots\}$

Step2 依次进行直至形成回路解。

适用范围：对称型 ΔTSP 。

具体实施中，可以将出发点取遍 V 中各点而得到多个解，从中选择最好的一个，但此时的时间复杂度增加了 n 倍。常见的插入型算法有：

(1) 最近插入 (nearest insertion) 法

最坏情况： $\varepsilon = 2$ ；时间复杂度： $O(n^2)$ 。

(2) 最小插入 (cheapest insertion) 法

最坏情况： $\varepsilon = 2$ ；时间复杂度： $O(n^2 \lg n)$ 。

(3) 任意插入 (arbitrary insertion) 法

最坏情况： $\varepsilon = 2 \lg n + 0.16$ ；时间复杂度： $O(n^2)$ 。

(4) 最远插入 (farthest insertion) 法

最坏情况： $\varepsilon = 2 \lg n + 0.16$ ；时间复杂度： $O(n^2)$ 。

(5) 凸核插入 (convex hull insertion) 法

最坏情况： ε = 未知；时间复杂度： $O(n^2 \lg n)$ 。

2.5.2 最近邻算法 (nearest neighbor heuristics)

Step1 任取一出发点；

Step2 依次取最近的点，加入当前解中直至形成回路解。

适用范围：对称型 ΔTSP 。

具体实施中，可以将出发点取遍 V 中各点而得到多个解，从中选择最好的一个，

但此时的时间复杂度增加了 n 倍。

最坏情况: $\varepsilon = (\lg n + 1)/2$; 时间复杂度: $O(n^2)$ 。

2.5.3 Clark & Wright 算法

Step1 任取一出发点 P , 计算 $S_{ij} = d_{pi} + d_{pj} - d_{ij}$;

Step2 将各 S_{ij} 由大到小排列;

Step3 将排列好后的各 (i,j) 依次适当联结, 形成回路解。

适用范围: 对称型 ΔTSP 。

具体实施中, 可以将出发点 P 取遍 V 中各点而得到多个解, 从中选择最好的一个, 但此时的时间复杂度增加了 n 倍。

最坏情况: $\varepsilon = 2\lg n / 7 + 5 / 21$; 时间复杂度: $O(n^2)$ 。

2.5.4 双生成树算法 (double-spanning-tree heuristics)

Step1 首先求出最小生成树;

Step2 将树中各边都添一重复边并求出其 Euler 回路;

Step3 在 Euler 回路点序列中去除重复点, 形成回路解。

适用范围: 对称型 ΔTSP 。

最坏情况: $\varepsilon = 2$; 时间复杂度: $O(n^2)$ 。

2.5.5 Christofides 算法

Step1 首先求出最小生成树；

Step2 对树中所有奇顶点解最小权匹配问题；

Step3 将匹配边添入生成树并求出其 Euler 回路；

Step4 在 Euler 回路点序列中去除重复点，形成回路解。

适用范围：对称型 Δ TSP。

最坏情况： $\varepsilon = 3/2$ ；时间复杂度： $O(n^3)$ 。

2.5.6 r-opt 算法

该算法是一种局部改进搜索算法，由 Lin 等人（1965）提出。其思想是对给定的初始回路，通过每次交换 r 条边来改进当前解。对不同的 r ，优劣次序为：

$$2\text{-opt} < 3\text{-opt} < \dots < r\text{-opt}$$

但是，大量计算发现，3-opt 法比 2-opt 法好，而 4-opt 和 5-opt 却不比 3-opt 来的优越，况且 r 越大，计算时间越长。对于 3-opt 法，有一个经验公式告诉我们：求得最优解的概率为 $2^{-n/10}$ 。例如：对于 $n=50$ ，有 $P=2^{-50/10}=0.03$ ，只要随机选取 150 条初始路线，则求得最优解的概率可达 0.99。迄今为止，3-opt 法仍是一种相当有效的近似算法。

适用范围：对称型 Δ TSP。

最坏情况： $\varepsilon = 2(n \geq 8, r \leq n/4)$ ；时间复杂度： $O(n^r)$ 。

2.5.7 混合优化策略方法

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临近原则确定问题的整体解，再利用局部改进算法对其作进一步加工以得到问题的最终解。

这种算法往往能获得较好的解，但也很耗时，一般仅在对解有较高要求时采用。

2.5.8 概率算法 (probabilistic algorithm)

该算法能对任意给定的 $\varepsilon > 0$ ，在 $1 + \varepsilon$ 之内几乎处处解决 ΔTSP 。假定 G 位于单位正方形内，函数 $t(n)$ 映射到正有理数，满足：① $t \sim \log_2 \log_{2n}$ ；② 对所有 n ， n/t 是完全平方。则有：

Step1 以 $[t(n)/n]^{1/2}$ 为尺寸构成网络，将单位正方形分成 $n/t(n)$ 个子正方形，将 G 也分成至多 $n/t(n)$ 个子图；

Step2 用动态规划方法求解每个子图的最优回路；

Step3 将 $n/t(n)$ 个子图各自收缩为一点，其间距离定义为原子图的最优子回路间最短距离，并对新构成的图求最小生成树 T ；

Step4 将 $T \cup \{\text{各子图的最优子回路}\}$ 视为一可能有点、边重复的闭回路，根据三角不等式条件，归约重复点或边，得 TSP 回路。

适用范围：对称型 ΔTSP 。

最坏情况： $\varepsilon = 1 + (\text{任意给定正数})$ ；时间复杂度： $O(n \lg n)$ 。

2.5.9 模拟退火算法^[50] (simulated annealing, SA)

模拟退火算法是近年来特别引人注目的一种适应于解大型组合优化问题的技术，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与一般组合优化问题之间的相似性。SA 算法由某一较高初温开始，结合具有概

率突跳特性的 Metropolis 抽样策略在解空间中随机寻找目标函数的全局最优解，伴随温度参数的不断下降重复抽样过程，最终得到问题的全局最优解。

从算法的结构我们可以知道，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。

解决 TSP 问题的模拟退火算法的框架为：

给定起止“温度” T , T_0 和退火速度 α , 初始一条路径 C_0 ;

While ($T > T_0$) do

在 C_0 的邻域内产生另一条路径 C_1 ;

计算两条路径所引起的目标函数（能量）值得变化 ΔE ;

若 $\Delta E \leq 0$, 接受新值, 若 $\exp(-\Delta E/T) > \text{rand}(0,1)$ ($\text{rand}(0,1)$ 表示 0 到 1 之间的随机数),
也接受新值, 否则就拒绝;

确定新的参数值, 若扰动被接受, 则 $C_1 \rightarrow C_0$, 否则 C_0 不变化;

若接受新值, 降温 αT , 否则不降温;

END

模拟退火所得解的好坏与初始状态、温度函数等都有一定的联系, 降温较快的效果不一定很好, 效果好的, 其降温过程又极其缓慢。但由于该方法适用范围广, 并可人为控制迭代次数, 反复求解, 因此具有很强的实用性。

2.5.10 蚁群算法 (ant colony algorithm)

蚁群算法是一种新型的模拟进化算法, 由意大利学者 M·Dorigo、V·Maniezzo 和 A·Colorini 等人在 90 年代首先提出, 称之为蚁群系统 (ant colony system)。它是受到人们对自然界中真实的蚁群集体行为的研究成果的启发而提出的一种基于种群的模拟进化算法。生物学研究表明一群互相协作的蚂蚁能够找到食物源和巢之间的最短路径, 而单只蚂蚁则不能。蚂蚁间相互协作的方法是它们在运动过程中能在所经过的路径上留下一种称之为外激素 (pheromone) 的物质进行信息传递, 而且蚂蚁在运动过程中能够感

知这种物质，并以此指导自己的运动方向，因此由大量蚂蚁组成的蚁群集体行为便表现出一种信息正反馈现象：某一路径上走过的蚂蚁越多，则后来者选择该路径的概率就越大。

用蚁群算法求解 TSP 时，我们不难发现：蚁群算法的主要依据是信息正反馈原理和某种启发式算法的有机结合，这种算法在构造解的过程中，利用随机选择策略，这种选择策略使得进化速度较慢，正反馈原理旨在强化性能较好的解，却容易出现停滞现象。因此可以从选择策略方面进行修改，采用确定性选择和随机选择相结合的选择策略，并且在搜索过程中动态的调整作确定性选择的概率。当进化到一定代数后，进化方向基本确定，这时对路径上信息量作动态调整，缩小最好和最差路径上信息量的差距，并且适当加大随机选择的概率，以利于对解空间的更完全搜索，使算法性能明显得到改善。

2.5.11 遗传算法（Genetic algorithms）

遗传算法（简称 GA）是一种高度并行、随机和自适应的优化算法，其基本思想是基于 Darwin 的进化论和 Mendel 的遗传学说。该算法最早由美国密执安大学的教授于 1975 年创建，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。其所得结果的好坏，主要依赖于遗传代数和解组规模，在实际应用中只能根据具体要求，在合理的时间内对问题进行求解。

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确定适配配置函数；③算法参数的选取；④遗传算子的设计；⑤确定算法的终止条件。

遗传算法的早期应用主要围绕组合优化问题求解，如煤气管道的最优控制，TSP 问题等，近年来迅速扩展到机器学习、设计规划、神经网络优化、核反应堆控制、喷气发动机设计、通信网络设计、人工生命等领域，显示了遗传算法应用的巨大潜力。

近年来，遗传算法从理论到实际都已经取得了许多重要成果。由于它具有良好的全局搜索能力，是目前解决各种优化问题的最有效的方法，已经成为研究热点。遗传算法就其本质来说，主要是处理复杂问题的一种鲁棒性强的启发式随机搜索算法。因此遗传

算法在 TSP 问题求解方面的应用研究，对于构造合适的遗传算法框架、建立有效的遗传操作以及有效的解决 TSP 问题等有着多方面的重要意义。

2.5.12 神经网络算法

八十年代后期，美国、日本等国家出现了一股神经网络热潮，许多从事脑科学、心理学、计算机科学以及电子学等方面的专家都在积极合作，开展这一领域的研究。其早期思想源于四十年代，由于受到 Von Neumann 串行处理体系的限制，一直进展不大，直到 1982 年，美国生物物理学家 Hopfield 提出了一种离散神经网络（ANN）模型，才被认为是一个重大突破。而 Hopfield 于 1984 年又提出一种连续时间神经网络（HNN）模型，并由容易实现的电子线路构成。对于 HNN，给出适当的初始条件，在状态空间里反复使其更新状态，网络的能量随时间推移单调的减小，状态向着平衡态的方向更新。最后，网络的能量减至全局最小或局部最小，其状态稳定在某个平衡状态。利用 HNN 模型在状态空间里的这种能量最小（极小）化特性，可将它应用于 TSP 问题求解。

该方法的基本思想是通过对神经网络引入适当的能量函数，使之与 TSP 的目标函数相一致来确定神经元之间的联结权，随着网络状态的变化，其能量不断减少，最后达到平衡时，即收敛到一个局部最优解。

要解 n 城市的 TSP 问题，要把问题映射到一个神经网络上。可以使用 $n \times n$ 神经元矩阵。矩阵中的每个元的状态只能为 0 或 1，神经元的状态用 V_{si} 表示， $V_{si}=1$ 表示城市 x 在路径中第 i 个位置出现。一次有效路径使每行每列有且仅有一个元素为 1，其余为 0。为了最终解决 TSP 问题，必须构成这样的神经网络：在网络运行时，计算能量低，网络稳定后其输出状态表示城市被访问的次序。网络能量的极小点，对应于最佳（或较佳）路径的形成。其解决问题最关键的一步是构造能量函数。金海和等曾提出基于 HNN 的多城市旅行商问题的求解方法。它是把 HNN 学习算法作为基本算子，对城市群体按一定的规则进行有效的分割、计算和连接，来寻找巡回路径的最优解或满意解，取得了一定的成果。

2.6 本章小结

首先对 TSP 的问题进行了分析，介绍了其作为 NP 完整问题的求解难点。接着建立了数学模型并对 TSP 问题进行了分类。最后介绍了各种求解 TSP 问题的算法。其中重点介绍了模拟退火算，蚂蚁算法，遗传算法，神经网络算法这四种算法主要思想以及如何在 TSP 问题的应用。

第三章 一些典型算法在 TSP 问题中的应用

在上一章中详细介绍了求解 TSP 问题的各种算法，其中模拟退火算法、遗传算法、蚁群算法都是经典的算法，经常在实际研究中被采用。本章主要对这三种算法进行了实际的应用，并给出了结果。

3.1 模拟退火算法

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温度升高变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为 $e^{-\Delta E/(kT)}$ ，其中 E 为温度 T 时的内能， ΔE 为其改变量， k 为 Boltzmann 常数。用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f ，温度 T 演化成控制参数 t ，即得到解组合优化问题的模拟退火算法：由初始解 i 和控制参数初值 t 开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减 t 值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的初值 t 及其衰减因子 Δt 、每个 t 值时的迭代次数 L 和停止条件 S 。

3.1.1 模拟退火算法的模型

1、模拟退火算法的流程：

- (1) 初始化：初始温度 T （充分大），初始解状态 S （算法迭代的起点），每个 T 值的迭代次数 L

- (2) 对 $k=1, \dots, L$ 做第(3)至第6步:
- (3) 产生新解 S'
- (4) 计算增量 $\Delta t = C(S') - C(S)$, 其中 $C(S)$ 为评价函数
- (5) 若 $\Delta t < 0$ 则接受 S' 作为新的当前解, 否则以概率 $\exp(-\Delta t/T)$ 接受 S' 作为新的当前解。
- (6) 如果满足终止条件则输出当前解作为最优解, 结束程序。终止条件通常取为连续若干个新解都没有被接受时终止算法。
- (7) T 逐渐减少, 且 $T > 0$, 然后转第2步。

2、模拟退火算法新解的产生和接受可分为如下四个步骤:

第一步是由一个产生函数从当前解产生一个位于可行解空间的新解; 为便于后续的计算和接受, 减少算法耗时, 通常选择由当前新解经过简单地变换即可产生新解的方法, 如对构成新解的全部或部分元素进行置换、互换等, 注意到产生新解的变换方法决定了当前新解的邻域结构, 因而对冷却进度表的选取有一定的影响。

第二步是计算与新解所对应的目标函数差。因为目标函数差仅由变换部分产生, 所以目标函数差的计算最好按增量计算。事实表明, 对大多数应用而言, 这是计算目标函数差的最快方法。

第三步是判断新解是否被接受, 判断的依据是一个接受准则, 最常用的接受准则是 Metropolis 准则: 若 $\Delta t < 0$ 则接受 S 作为新的当前解 S' , 否则以概率 $\exp(-\Delta t/T)$ 接受 S' 作为新的当前解 S 。

第四步是当新解被确定接受时, 用新解代替当前解, 这只需将当前解中对应于产生新解时的变换部分予以实现, 同时修正目标函数值即可。此时, 当前解实现了一次迭代。可在此基础上开始下一轮试验。而当新解被判定为舍弃时, 则在原当前解的基础上继续下一轮试验。

模拟退火算法与初始值无关, 算法求得的解与初始解状态 S (是算法迭代的起点) 无关; 模拟退火算法具有渐近收敛性; 已在理论上被证明是一种以概率 1 收敛于全局最优解的全局优化算法; 模拟退火算法具有并行性。

3.1.2 参数设置

模拟退火算法参数难以控制，其主要问题有以下三点：

(1) 温度 T 的初始值设置问题。

温度 T 的初始值设置是影响模拟退火算法全局搜索性能的重要因素之一、初始温度高，则搜索到全局最优解的可能性大，但因此要花费大量的计算时间；反之，则可节约计算时间，但全局搜索性能可能受到影响。实际应用过程中，初始温度一般需要依据实验结果进行若干次调整。

(2) 退火速度问题。

模拟退火算法的全局搜索性能也与退火速度密切相关。一般来说，同一温度下的“充分”搜索（退火）是相当必要的，但这需要计算时间。实际应用中，要针对具体问题的性质和特征设置合理的退火平衡条件。

(3) 温度管理问题。

温度管理问题也是模拟退火算法难以处理的问题之一。实际应用中，由于必须考虑计算复杂度的切实可行性等问题，常采用如下所示的降温方式：

$$T(K+1)=K*T(K) \quad (3.1)$$

式中 k 为正的略小于 1 的常数， t 为降温的次数。

3.1.3 实验结果

在酷睿 TWO 双核 2.0 的 PC 机上，在 Visual C++ 编译环境下，编程实现了模拟退火算法程序，并利用它们求解了 48 个城市的 TSP 问题。

1、城市位置的标定

在本文中对城市位置的标定采用了坐标的形式，坐标文件的格式如下：

1 6734 1453

2 2233 10

3 5530 1424

.....
文件的每一行表示一个城市的信息，即，城市名称+空格+X坐标+空格+Y坐标。

2、参数设置

在上一节了解到模拟退火算主要涉及到了三个参数。对初始温度采用了一种估计的方法。如下式所示：

$$T_0 = K * \delta \quad (3.2)$$

其中 T_0 为估计的初始温度，K 为常数取值为 (10,20,100…)， δ 为最大的城市距离与最小的城市距离之差。温度下降方式： $T(k+1) = K * T(k)$ ， $K=0.95$ ；在退火速度方面对某一温度下的循环次数取为城市个数的三次方。

3、实验结果

首先要读取城市坐标文件，这里选取了 48 个城市的坐标文件。其次就要进行最优路线计算，计算的过程如图 3.1 所示：



图 3.1 计算过程

最优路线为：5→25→21→47→17→27→19→37→6→28→36→7→18→44→31→46→33→15→12→11→40→9→38→8→1→16→22→3→34→41→29→2→42→26→4→35→45→10→24→32→39→13→23→14→48。总路程 37269。

最后用 MATLAB 画出路径优化过程图和求解路径图，如图 3.2 所示：（大的红*表示路径开始城市，途经城市依次用蓝色方块和红色*标示）

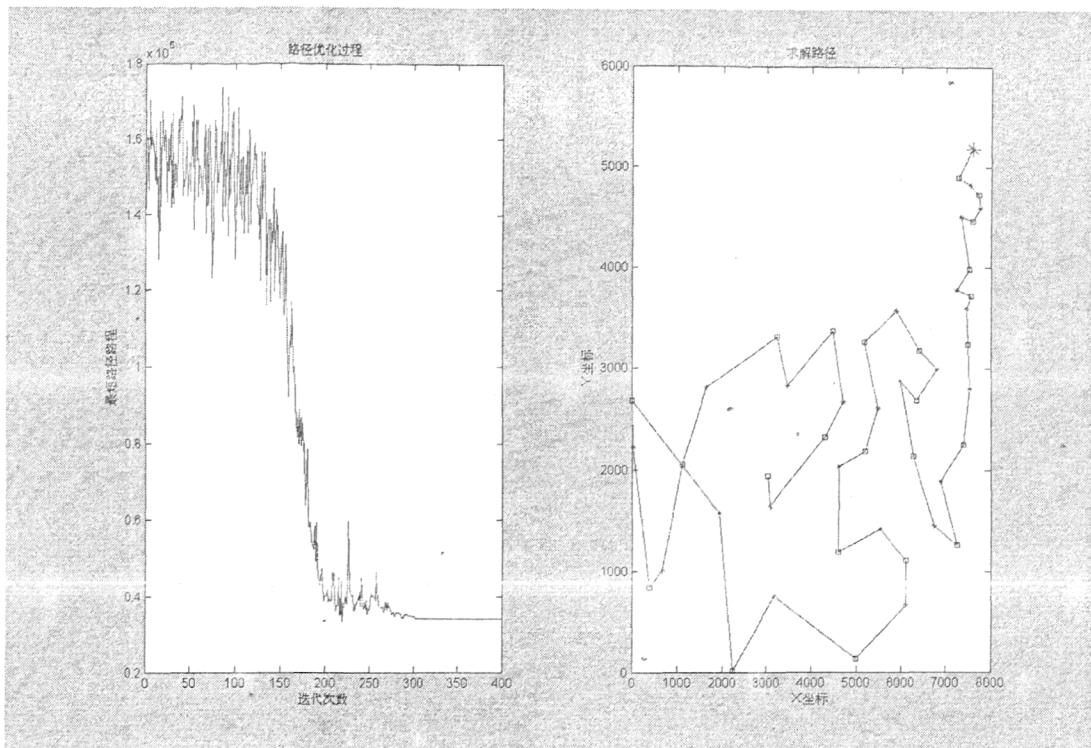


图 3.2 优化过程和最优路线

3.2 蚁群算法

20 世纪 50 年代中期创立了仿生学，人们从生物进化的机理中受到启发，提出了许多用以解决复杂优化问题的新方法，如遗传算法、进化规划、进化策略等。蚁群算法最早由意大利学者 M. Dorigo 于 1992 年，在他的博士论文中首次系统提出，并用该方法求解旅行商问题 (TSP)、指派问题 (assignment problem)，取得了一系列较好的实验结果。受其影响，蚁群系统模型逐渐引起了其他研究者的注意，并用该算法来解决一些实际问题，证明蚁群算法在求解复杂优化问题（特别是离散优化问题）方面具有一些优越性。

3.2.1 蚁群算法的基本原理

人工蚁群算法是受到人们对自然界中真实的蚁群集体行为的研究成果的启发而提出的一种基于种群的模拟进化算法。M.Dorigo 等人首次提出该方法时充分利用了蚁群搜索食物的过程与著名的旅行商问题(TSP)之间的相似性，通过人工模拟蚂蚁搜索食物的过程(即：通过个体之间的信息交流与相互协作最终找到从蚁穴到食物源的最短路径)来求解 TSP，为了区别于真实蚂蚁群体系统，我们称这种算法为“人工蚁群算法”。像蚂蚁这类群居昆虫，虽然单个蚂蚁的行为极其简单，但由这样的单个简单的个体所组成的蚁群群体却表现出极其复杂的行为，能够完成复杂的任务，不仅如此，蚂蚁还能够适应环境的变化，如：在蚁群运动路线上突然出现障碍物时，蚂蚁能够很快地重新找到最优路径。蚁群是如何完成这些复杂的任务的呢？人们经过大量研究发现，蚂蚁个体之间是通过一种称之为外激素(pheromone)的物质进行信息传递。从而能相互协作，完成复杂的任务。蚁群之所以表现出复杂有序的行为，个体之间的信息交流与相互协作起着重要的作用。蚂蚁在运动过程中，能够在它所经过的路径上留下该种物质，而且蚂蚁在运动过程中能够感知这种物质的存在及其强度，并以此指导自己的运动方向，蚂蚁倾向于朝着该物质强度高的方向移动。因此，由大量蚂蚁组成的蚁群的集体行为便表现出一种信息正反馈现象：某一路径上走过的蚂蚁越多，则后来者选择该路径的概率就越大。蚂蚁个体之间就是通过这种信息的交流达到搜索食物的目的。

3.2.2 基本蚁群系统模型

为了便于理解，我们以求解平面上 n (表示城市序号) 个城市的 TSP 问题 ($n = 0, 1, \dots, n - 1$) 为例说明蚁群系统模型。对于其它问题，可以对此模型稍做修改便可应用。为模拟实际蚂蚁行为，首先引进如下记号：设 m 是蚁群中蚂蚁的数量， d_{ij} ($i, j = 1, 2, \dots, n$) 表示城市 i 和城市 j 之间的距离， $b_i(t)$ 表示 t 时刻位于城市 i 的蚂蚁的个数。

$$m = \sum_{i=1}^n b_i(t) \quad (3.3)$$

$\tau_{ij}(t)$ 表示 t 时刻在 i, j 连线上残留的信息量。初始时刻，各条路径上信息量相等，设 $\tau_{ij}(0)=C$ (C 为常数)。蚂蚁 k ($k=1, 2, \dots, m$) 在运动过程中，根据各条路径上的信息量决定转移方向， $p_{ij}^k(t)$ 表示在 t 时刻蚂蚁 k 由位置 i 转移到位置 j 的概率

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha(t) \eta_j^\beta(t)}{\sum_{\alpha \in allowed_k} \tau_{ij}^\alpha \eta_j^\beta(t)}, & j \in allowed_k \\ 0 & otherwise \end{cases} \quad (3.4)$$

其中， $allowed_k = \{0, 1, \dots, n-1\} - tabu_k$ 表示蚂蚁 k 下一步允许选择的城市。与实际蚁群不同，人工蚁群系统具有记忆功能， $tabu_k$ ($k=1, 2, \dots, m$) 用以记录蚂蚁 k 当前所走过的城市，集合 $tabu_k$ 随着进化过程作动态调整。随着时间的推移，以前留下的信息逐渐消逝，用参数 $1 - \rho$ 表示信息消逝程度，经过 n 个时刻，蚂蚁完成一次循环，各路径上信息量要根据下式作调整：

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta \tau_{ij} \quad (3.5)$$

$$\Delta \tau_{ij} = \sum_{k=1}^n \Delta \tau_{ij}^k \quad (3.6)$$

$\Delta \tau_{ij}^k$ 表示第 k 只蚂蚁在本次循环中留在路径 ij 上的信息量， $\Delta \tau_{ij}$ 表示本次循环中路径 ij 上的信息量的增量。

$$\Delta \tau_{ij}^k = \begin{cases} \frac{Q}{L}, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过 } ij \\ 0 & otherwise \end{cases} \quad (3.7)$$

其中 Q 是常数， L_k 表示第 k 只蚂蚁在本次循环中所走路径的长度。在初始时刻， $\tau_{ij}(0)=C(const)$ ($i, j = 0, 1, \dots, n-1$)。 α, β 分别表示蚂蚁在运动过程中所积累的信息及启发式因子在蚂蚁选择路径中的所起的不同作用。 η_{ij} 表示由城市 i 转移到城市 j 的期望程

度，可根据某种启发式算法具体确定。根据具体算法的不同， $\tau_{ij}(t)$ ， $\Delta\tau_{ij}(t)$ 及 $p_{ij}^k(t)$ 的表达式可以不同，要根据具体问题而定。M.Dorigo 曾给出三种不同模型，分别称之为 ant cycle system、ant quantity system、ant density system。它们差别在于表达式的不同。在 ant quantity system 模型中：

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{d_{ij}}, & \text{若第 } k \text{ 只蚂蚁在时刻 } t \text{ 和 } t+1 \text{ 之间经过 } ij \\ 0 & \text{otherwise} \end{cases} \quad (3.8)$$

在 ant density system 模型中：

$$\Delta\tau_{ij}^k = \begin{cases} Q, & \text{若第 } k \text{ 只蚂蚁在时刻 } t \text{ 和 } t+1 \text{ 之间经过 } ij \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

它们的区别在于：后两种模型中，利用的是局部信息，而前者利用的是整体信息，在求解 TSP 问题时，性能较好。因而通常采用它作为基本模型。参数 Q ， C ， α ， β ， ρ 可以用实验方法确定其最优组合。

3.2.3 应用蚁群算法要注意的问题

应用蚂蚁的行为特性求解 TSP 问题时，每只人工蚂蚁的行为还必须符合下列规律：

①是根据路径上的信息素浓度，以某种概率来选取下一步的路径；②是不再选取自己本次循环已经走过的路径作为下一步的路径（可以通过一个数据结构来控制这一点）；③是当完成一次循环后，根据整个路径长度来释放相应浓度的信息素，并更新走过路径上的信息素浓度。

引入变量 $\tau_{ij}(t), \Delta\tau_{ij}, \Delta\tau_{ij}^k, p_{ij}^k$ ，其中 $1 \leq i, j \leq n, 1 \leq k \leq m$ ， n 表示 TSP 问题的规模，即城市数，而 m 则表示人工蚂蚁的个数。下面解释上述变量的含义及其关系。

$\tau_{ij}(t)$ 表示在 t 时刻边 (i, j) 上的信息素浓度。当蚂蚁完成一次循环后，相应边上的信息素浓度为 $\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$ ，即等于上一时间段残留下的信息素浓度加上当前时

间段新增加的信息素浓度。其中 ρ 为一个取值范围在 0 到 1 之间的常数，显然， $1 - \rho$ 表示在时间段 t 到 $t+1$ 之间信息素浓度的挥发强度因子。

$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k$ 。其中 $\Delta\tau_{ij}^k$ 是第 k 只蚂蚁在时间 t 到 $t+1$ 之间，在边 (i, j) 上增加的信息素浓度。M.Dorigo 针对 $\Delta\tau_{ij}^k$ 取值的不同给出了蚂蚁算法的三种模型^[7]，分别称之为 ant-cycle system、ant-quantity system、ant-density system。

对于 ant-cycle system， $\Delta\tau_{ij}^k$ 定义为：

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{如果第 } k \text{ 只蚂蚁在时间 } (t, t+1) \text{ 之间经过边 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (3.10)$$

对于 ant-quantity system， $\Delta\tau_{ij}^k$ 定义为：

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{C_{ij}}, & \text{如果第 } k \text{ 只蚂蚁在时间 } (t, t+1) \text{ 之间经过边 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (3.11)$$

对于 ant-density system， $\Delta\tau_{ij}^k$ 定义为：

$$\Delta\tau_{ij}^k = \begin{cases} Q, & \text{如果第 } k \text{ 只蚂蚁在时间 } (t, t+1) \text{ 之间经过边 } (i, j) \\ 0, & \text{否则} \end{cases} \quad (3.12)$$

在上述公式 (3.10)、(3.11) 和 (3.12) 中 Q 是一个常量，用来表示蚂蚁完成一次完整的路径搜索后，所释放的信息素总量。 L_k 表示第 k 只蚂蚁的路径总费用，它等于第 k 只蚂蚁经过的各段路径所需费用的总和。显然，蚂蚁不会在其没有经历过的路径上释放信息素。值得注意的是，后两种模型中（如公式 (3.11) 和 (3.12) 所示）利用的是路径的局部信息，而 (3.10) 利用的是路径的整体信息，在求解 TSP 问题时，第一个模型性能较好。

定义第 k 只蚂蚁当前时刻在顶点 V_i ，下一步选择顶点 V_j ，即选择路径 (i, j) 的概率为：

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \times [\eta_{ij}(t)]^\beta}{\sum_{l \in allowed_k} [\tau_{il}(t)]^\alpha \times [\eta_{il}(t)]^\beta}, & if \quad j \in allowed_k \\ 0, & otherwise \end{cases} \quad (3.13)$$

其中 $\eta_{ij} = \frac{1}{C_{ij}}$, C_{ij} 为路径 (i, j) 所需费用。 α, β 为两个参数, 分别用来控制信息素浓度和

路径长度的相对重要程度。 $allowed_k$ 是第 k 只蚂蚁在满足人工蚂蚁行为规律 3 个条件的前提下, 下一步可以选择的路径集合。

根据上面的分析, 再引入一个控制“蚂蚁不再选取自己本次循环已经走过的路径作为下一步路径”这一行为特性的数据结构 $visited-city_k$, 表示第 k 只蚂蚁访问过的城市集合, 其中 $1 \leq k \leq m$ 。在初始化的时候, m 只人工蚂蚁被放置在不同的城市上, 赋予每条边的信息素浓度初值为一常数 $\tau_{ij}(0) = C$ 。每只蚂蚁的 $visited-city_k$ 的第一个元素赋值为它所在的城市。当蚂蚁完成一次完整的寻径过程后(即从某个城市出发走完所有城市一次并回到出发点), 计算 $\Delta\tau_{ij}^k$, 并更新每条边上的信息素浓度, 然后开始新一轮循环。直到迭代次数达到事先定义好的最大循环次数 NC 或所有的蚂蚁都选择了同一路经时, 程序终止。

3.2.4 算法的实现

算法描述如下:

① 初始化: $t = 0, nc = 0, \tau_{ij}(t) = C, \Delta\tau_{ij} = 0$, 并置 m 只蚂蚁于 n 个城市上。

其中 nc 为迭代循环的次数, t 是一个“伪”时间控制变量。

② 将各蚂蚁的初始出发点置于当前解集 $visited-city_k$ 中;

对每只蚂蚁 $k (k = 1, 2, \dots, m)$ 按概率 p_{ij}^k 移至下一顶点 j ;

并将 j 置于当前解集 $visited-city_k$ 中。

- ③ 计算各蚂蚁的目标函数值 L_k ($k = 1, 2, \dots, m$)；记录当前的最好解。
- ④ 按信息素浓度调整规则更新每条边的信息素浓度。
- ⑤ 对各边 (i, j) ，置 $\Delta\tau_{ij} = 0, nc = nc + 1$ 。
- ⑥ 若 $nc < NC$ 且无退化行为（即找到的都是相同解），则转步骤②；否则结束并输出当前最好解。

3.2.5 实验结果

在酷睿 TWO 双核 2.0 的 PC 机上，在 MATLAB 编译环境下，编程实现了蚁群算法程序，并利用它们求解了 30 个城市的 TSP 问题。结果如图 3.3 所示：

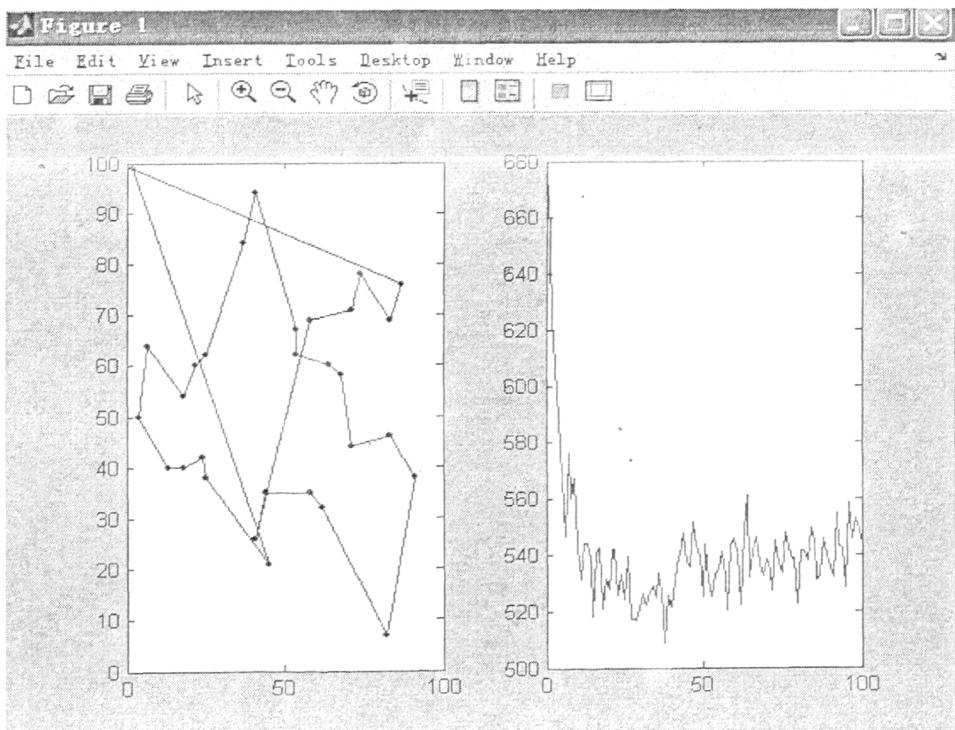


图 3.3 蚁群算法优化过程和最优路线

最佳路径：10→21→20→19→11→7→8→14→15→24→25→26→29→28→27→16→17→22→23→30→12→13→4→5→6→1→2→9→3→18。总路程 429.6818

3.3 遗传算法

遗传算法 (GA) 是一种通过模拟自然进化过程搜索最优解的方法。自从生物的进化理论得到人们的接受之后，生物学家就对进化机制产生了极大的兴趣。化石纪录表明我们所观察到的复杂结构的生命是在相对短的时间内进化而来的，这一点让包括生物学家在内的许多人感到惊奇。虽然目前关于推动这个进化的机制还没有完全弄清楚，但它们的某些特征已为人们所熟知。进化是发生在作为生物体结构编码的染色体上，通过对染色体的译码部分地生成生物体。人们现在还不完全清楚染色体的编码和译码过程的细节，但下面几个关于进化理论的一般特性已广为人们所接受：进化过程是发生在染色体上，而不是发生在它们所编码的生物体上。自然选择把染色体以及由它们所译成的结构的表现联系在一起，那些适应性好的个体的染色体经常比差的染色体有更多的繁殖机会。繁殖过程是发生在进化发生的那一刻。变异可以使生物体子代的染色体不同于它们父代的染色体。通过结合两个父代的染色体中的物质，重组过程可以在子代中产生有很大差异的染色体。

生物进化没有记忆。有关产生个体的信息包含在全体的染色体的集合以及染色体编码的结构之中，这些个体会很好地适应它们的环境。遗传算法利用简单的编码技术和繁殖机制来表现复杂的现象，从而解决非常困难的问题。特别是由于它不受搜索空间的限制假设的约束，不必要求诸如连续性、导数存在和单峰等假设，以及其固有的并行性，遗传算法已经在最优化、机器学习和并行处理等领域得到了广泛应用。

3.3.1 GA 基本概念和基本操作

根据遗传理论，遗传算法的基本操作对象被称为染色体或个体(chromosome)。每个染色体是一个知识结构，代表求解问题的一个可行解。染色体通常用字符串或位串来表示，一定长度的位串被称为染色体的基因(Gene)。

群体或种群(Population)是由一组染色体所构成，它描述 GA 搜索的遗传空间。在搜

索过程中，用适应度函数(Fitness Function)来评价每个染色体的优劣，其值越大(适应度值大)，相应的染色体所代表的解就越优。适应度函数的选择是能有效地指导搜索空间沿着面向优化参数组合方向逐渐逼近最佳参数优化组合。

利用遗传算法求解问题的第一步就是要对问题的解进行某种形式的(Encoding)，将解的表示转换成遗传空间解的表示，即用字符串构造染色体，其相反的操作为解码(Decoding)。编码形式的选择也是影响遗传算法效率的重要因素之一。对问题的解施行编码以及初始化种群之后，对种群的遗传操作主要有以下三种：选择、交换和变异。

选择(Selection)操作就是根据各染色体的适应度，在群体中按一定的概率选择可作为父本的染色体，选择的依据是适应度大的染色体被选中的概率也大。

交换(Crossover)操作是按一定的概率随机地交换一对父本染色体的基因以形成新的子染色体。

变异(Mutation)操作是按一定的概率随机地改变一个父染色体的基因值以形成新的子染色体。

3.3.2 遗传算法描述

关于基本的遗传算法有许多种描述形式，它们之间有较小的差异。我们这里给出其中的一种：

- (1) 随机产生一个初始群体；
- (2) 对群体进行迭代，指到满足停止准则；
 - (i) 计算每个个体的适应值；
 - (ii) 应用选择、杂交和变异算子产生下一代群体。

(3) 把在任一代中出现的最好个体作为遗传算法的执行结果。这个结果可以表示问题的一个解(近似解)。值得注意的是，遗传算法按不依赖于问题本身的方式作用在特征串群体上。遗传算法搜索可能的特征串空间找到高适应值串，为了指导这个搜索，算法仅用到与在搜索空间中检查过的点相联系的适应值。不管求解问题的本身多么复杂，遗传算法通过执行同样的、惊人简单的复制、杂交和偶尔的变异操作来完成它的搜索。

在实际应用中，遗传算法能够快速有效地搜索复杂、高度非线性和多维的空间。出人意料的是遗传算法并不知道问题本身的信息，也不了解适应值的度量。使用遗传算法时，一般可以利用特定领域的知识来选择表示方案和适应值度量，并且在选择群体规模、演化代数、控制执行各种遗传算子的参数、停止准对和指定算法结果的方法上也可以采取附加的判断，所有这些选择都可能影响到遗传算法在求解问题中的执行效果，甚至关系到它能否起作用。但总的来说，遗传算法仍是按不依赖于问题本身的方式快速搜索未知的空间以找到高适应度值点。

3.3.5 求解 TSP 的遗传操作方法

应用遗传算法求解 TSP，表示方法非常简单，即人们普遍把染色体表示成所有城市的一个排列。假设有 n 个城市，一条可能的路径可以编码为长度为 n 的整数向量 (i_1, i_2, \dots, i_n) ，其中 i_k 表示第 i_k 个城市。这个向量是 1 到 n 个排列，换句话说，从 1 到 n 的每个整数在这个向量中正好出现一次。

种群的初始化

初始化种群首先要解决的问题是确定种群的规模，即种群中个体的数量。然而这个规模的选取现在还没有一个统一的标准，也就是说，种群中应包含多少个体才能使算法效果最佳，现在还缺乏理论的指导。有人采用一个确定的常数，比如 100，也有人主张这个数里应与问题的规模 n 有关，如就选为 n 或 n 的倍数。还有人使用动态的方法，在算法执行过程中，根据具体情况和需要确定每一代群体的规模。

求解 TSP 的遗传算法初始化群体的常用方法一般有两种：随机产生初始个体和利用局部搜索算法。随机选取初始个体是被普遍采用的初始化方法，它具有快速和能够保证群体多样性等特点。但大量试验结果表明：随机产生的个体适应度值太低，它们需要更多迭代次数的遗传操作来得到优化。而遗传算法的主要遗传操作——交换会使得群体中个体越来越趋于相同，致使种群在一定的迭代次数后很难得到进化或停滞不前，而此时的解离最优解相去甚远。这使得它的两大优点丧失殆尽。利用局部搜索算法初始化种群

可以解决这一问题，但局部搜索算法将增加算法的时间开销。特别是种群规模较大时，单初始化种群就将占用整个算法时间的很大比重，并使得算法无法在合理的时间内完成。采用快速的近似算法初始化种群将是一种折衷方案。

选择操作

选择操作的一种最常用技术叫做赌盘选择，它的基本步骤为：

- (1) 将群体中所有个体的适应度值相加求总和；
- (2) 产生一个在 0 与总和之间的随机数 m；
- (3) 从群体中编号为 I 的个体开始，将其适应度值与后继个体的适应度值相加，直到累加和等于或大于 m：其中那个最后加进去的个体就是所要选择的个体。

赌盘选择的结果是返回一个随机选择的个体。尽管选择过程是流动的，但每个个体被选择的机会却直接与其适应度值成比例。那些没被选中的个体则被从群体中淘汰出去。当然，由于选择的随机性，群体中适应度值最差的个体有时也可能被选中，这会影响到遗传算法的执行效果，但随着进化过程的进行，这种偶然性的影响将会是微不足道的。现在被人们采用较多的选择操作方法还有线性选择和非线性选择。线性选择是根据群体 P 中各染色体的适应度，按照一定的概率选出一定数量的染色体，并将其拷贝到基因池 T 中。则基因将由 P 中染色体 S_i 的 n_i 个拷贝构成，其中 $n_i = F_i m / \sum_{j=1}^m F_j$ 。

若 n_i 不是整数则进行舍入运算使 $\sum_{i=1}^m n_i = m$ 。

非线性选择是在产生基因池时引入某些非线形函数，如使用 S 型函数

$$S(F) = \begin{cases} 0, & \text{当 } F_i \leq F_{\min} \text{ 时} \\ 2(F_i - F_{\min})^2 / (F_{\max} - F_{\min}), & \text{当 } F_{\min} \leq F_i \leq (F_{\max} + F_{\min})/2 \text{ 时} \\ 1 - 2(F_i - F_{\min})^2 / (F_{\max} - F_{\min}), & \text{当 } (F_{\min} + F_{\max})/2 \leq F_i \leq F_{\max} \text{ 时} \\ 1, & \text{当 } F_i \geq F_{\max} \text{ 时} \end{cases} \quad (3.14)$$

交换操作

应用遗传算法求解 TSP，通常把染色体表示成所有城市的一个排列。在这种表示方法下，传统的杂交算子产生的向量很可能不是从 1 到下一个排列，即产生了无意义的路径。为了排除这一情况，必须定义保持编码有效性的杂交算子。关于杂交过程，这里需要的是交换一个向量上的元，而不是替换它们。下面是三种求解 TSP 常用的杂交算子，它们都属于多点交换的范畴。

(1) 基于次序的杂交

这种杂交算子首先在两个父代向量上选取一组元位置，然后把一个父代在这组位置上的元的次序强加到另一个父代对应的元上。举例如下：

父代 1: 1 2 3 4 5 6 7 8 9 10

父代 2: 5 9 2 4 1 10 7 3 8 6

所选位置: xx xx

子代 1: 1 9 3 4 5 2 6 8 7 10

子代 2: 2 9 3 4 6 1 10 7 5 8

(2) 基于位置的杂交

这种杂交算子首先在两个父代向量上随机选取一组位置，然后把一个父代向量上被选元的位置强加到另一个父代向量对应的元上。如下：

父代 1: 1 2 3 4 5 6 7 8 9 10

父代 2: 5 9 2 4 6 1 10 7 3 8

所选位置 xx xx

子代 1: 1 9 2 3 6 4 5 7 8 10

子代 2: 9 2 3 4 5 6 1 8 10 7

(3) 部分影射杂交

在两个父代向量上随机选取一段，用两个父代向量在所选段内元的对应对来定义一系列交换，这些交换可以在每个父代向量上分别执行。假设下面两个向量表示两个可能的路径：

2 6 4 3 | 8 1 5 | 10 7 9

8 5 1 10 | 7 6 2 | 4 3 9

其中的对应对是 8:7, 1:6 和 5:2，部分影射杂交生成如下两个有效路径：

5 1 4 3 | 7 6 2 | 10 8 9

7 2 6 10 | 8 1 5 | 4 3 9

纵观以上三种杂交算子，无论是杂交位置还是基因段的选取都是随机的，或者说缺乏有效的指导，所产生的子代被恶化的可能性理论上远高于被优化的可能性。这势必影响交换操作的效率。该算法在部分影射杂交的基础上进行了改进，在两个父代染色体上

选择基因段时，让它们以某个相同的点为中心，例如

父代 1: 4 3 8 1 5 10 | 7 9 2 | 6

父代 2: 1 10 7 6 2 4 | 3 9 8 | 5

由于(792)和(398)是两个不同的基因段，必有优劣之分。在执行部分影射杂交后，产生后代至少有一个被优化的概率将有所提高，减少了交换操作的盲目性，提高了算法的执行效率。

3.3.6 实验结果

在酷睿 TWO 双核 2.0 的 PC 机上，在 Visual C++ 编译环境下，编程实现了模拟退火算法程序，并利用它们求解了 48 个城市的 TSP 问题。

1、城市位置的标定

在本文中对城市位置的标定采用了坐标的形式，坐标文件的格式如下

1 6734 1453

2 2233 10

3 5530 1424

.....

文件的每一行表示一个城市的信息，即，城市名称+空格+X 坐标+空格+Y 坐标。

2、实验结果

首先要读取城市坐标文件，这里选取了 48 个城市的坐标文件。其次就要进行最优路线计算，最后用 MATLAB 对最佳路线和迭代次数进行了画图，如图 3.4 所示：（大的红*表示路径开始城市，途经城市依次用蓝色方块和红色*标示）

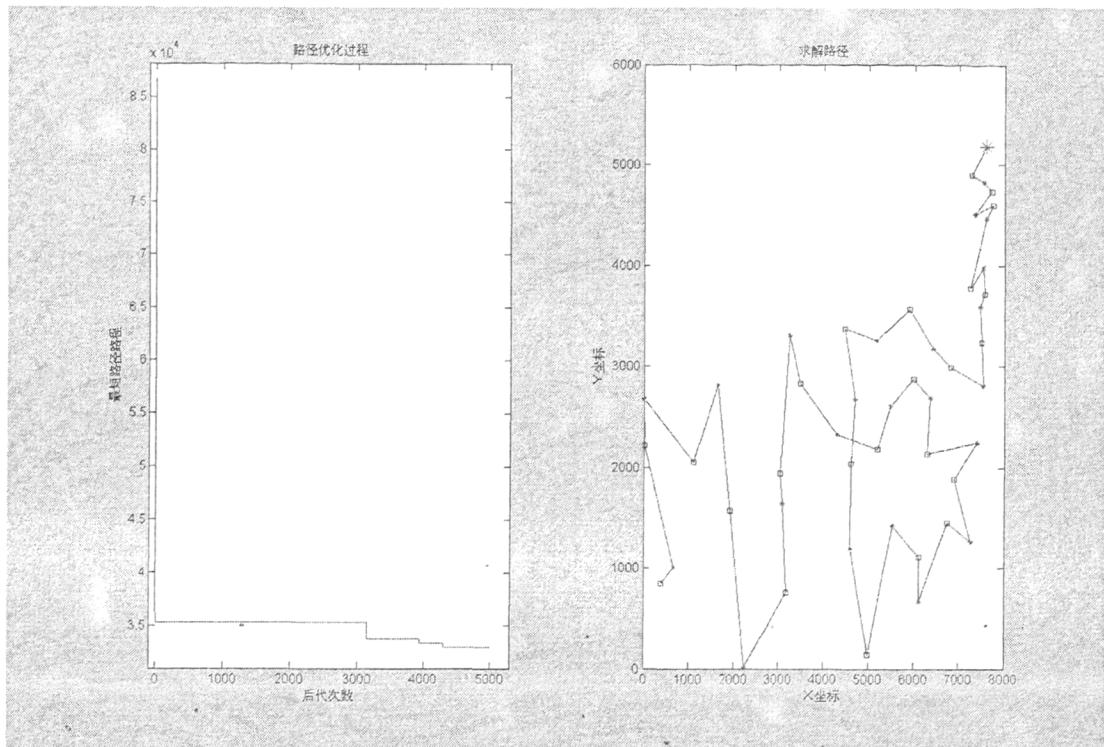


图 3.4 优化过程和最优路线

最优路线为：12→15→40→9→1→8→38→31→44→30→43→17→27→19→37→6→28→7→18→36→46→33→20→47→21→32→39→48→29→2→26→4→35→10→24→42→5→14→13→25→34→41→16→22→3→23→11。总路程 35493。

3.4 本章小结

选择了遗传算法、模拟退火算法和蚁群算法这三个比较有代表性的算法进行了编程实现。针对每个算法分别介绍了主要算法思想和实现步骤，最后给出了其求解 TSP 问题的结果。

通过研究这三种算法的求解结果发现：模拟退火算法的优化过程较长；蚁群算法同样是搜索时间比较长，也容易陷于局部最优解，使搜索停滞。遗传算法实际应用时易出现早熟收敛和收敛性差等缺点。这三种算法都在求解速度上遇到了很大的困难。

第四章 人工神经网络求解 TSP 问题

人工神经网络技术作为人工智能技术新手段，是目前国际上迅速发展的前沿研究方向之一。人工神经网络是用大量简单的基本单元——神经元广泛地互相连接而成的复杂的网络系统，反映了人脑功能的许多基本特征，是对人脑神经系统的某种简化、抽象和模拟。人工神经网络在限制性条件满足与最优化、数据压缩、预测与危险判断、控制、内容可寻址存储、多传感器数据融合、模式识别以及诊断等许多领域都有着广泛的应用前景，并且已取得了很多丰硕的成果。其中，Hopfield 神经网络被广泛应用在 TSP 求解问题中。对于 TSP 这样一个组合优化问题，Hopfield 神经网络可以把目标函数转化为网络的能量函数，把问题的变量对应到网络的状态。这样，当网络的能量函数收敛于极小值时，问题的最优解也随之求出。由于神经网络是并行计算的，其计算量不随维数的增加而发生指数性“爆炸”，因而对于优化问题的高速计算特别有效。本章主要介绍了如何应用 Hopfield 神经网络对 TSP 问题进行求解。

4.1 人工神经网络简介

人工神经网络（简称神经网络）是神经系统的模拟，包括了大脑神经系统的许多特征。人脑是一个高度复杂的、非线性的和并行的计算机器（信息处理系统）。人脑能够组织它的组成成分，即神经元，以比今天已有的最快的计算机还要快许多倍的速度进行特定的计算（如模式识别、感知和运动神经控制）。由于目前人类对大脑神经系统及其智能机理研究水平以及相关科学技术水平有限，神经网络对大脑神经系统进行了合理的简化和抽象。神经网络是一个由大量结构简单的处理单元，即神经元，通过广泛的连接而形成的神经系统。通过大量神经元的并行计算和分布存储，神经网络具有很强的计算能力和存储能力，因此神经网络是一种基于大规模并行的分布式处理系统。神经网络在两方面与大脑相似：

1、神经网络获取的知识是从外界环境中学习得来的。

2、互连神经元的连接强度，即突触权值，用于存储获取的知识。

设计一个神经网络模型需要考虑如下三个方面的因素：

◆ 网络拓扑结构。前向式神经网络(feed forward)还是反馈式神经网络(feed back)，网络的层数，神经元的数目。

◆ 神经元的类型。神经元输入输出函数的类型，是可以同模拟电路实现的简单神经元还是必须用数值电路实现的复杂神经元，是连续型神经元还是离散型神经元。

◆ 学习训练机制。是有监督的学习还是无监督的自组织学习方式，所用的学习规则和训练方法。

这些因素也是用来划分神经网络类型的标准。例如，按照网络拓扑结构，可以将神经网络分为前向式神经网络和反馈式神经网络；按照神经元类型，可以分为连续型神经网络和离散型神经网络；按照学习机制，则可分为有监督的学习和无监督的学习。

4. 2 Hopfield 神经网络

Hopfield 网络是一种互连型网络的一种，它引入类似于 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）相对应，并将其转换为神经网动力学系统的演化问题。其演变过程是一个非线性动力学系统，可以用一组非线性差分议程描述（离散型）或微分方程（连续型）来描述。系统的稳定性可用所谓的“能量函数”进行分析。在满足条件的情况下，某种“能量函数”的能量在网络运行过程中不断地减少，最后趋于稳定的平衡状态。对于一个非线性动力学系统，系统的状态从某一初值出发经过演变后可能有如下几种结果：

渐进稳定点（吸引子）、极限环、混沌、状态发散。

因为人工神经网络的变换函数是一个有界函数，故系统的状态不会发生发散现象。目前，人工神经网络经常利用渐进稳定点来解决某些问题。如果把系统的稳定点视为一

个记忆的话，那么从初态朝这个稳定点的演变过程就是一个寻找记忆的过程。如果把系统的稳定点视为一个能量函数的极小点，而把能量函数视为一个优化问题的目标函数，那么从初态朝这个稳定点的演变过程就是一个求解该优化问题的过程。因此，Hopfield 神经网络的演变过程是一个计算联想记忆或求解优化问题的过程。实际上，它的解决并不需要真的去计算，而是通过构成反馈神经网络，适当地设计其连接权和输入就可以达到这个目的。

4.2.1 离散型 Hopfield 神经网络

在离散型 Hopfield 神经网络中，每个神经元节点的输出只可以有两值状态，-1 或 1（0 或 1），神经网络的激活函数一般采取阈值函数。

设

$$y(t) = (y_1(t), y_2(t), \dots, y_n(t))^T, \quad t=1, 2, \dots, \quad (4.1)$$

表示一个含有 n 个神经元的离散型 Hopfield 神经网络在 t 时刻的状态，其中， $y_i(t)$ 分别对应于第 i 个神经元的状态。则离散型 Hopfield 神经网络可以用下面的公式表示：

$$y(t+1) = \varphi(W \cdot y(t) - T). \quad (4.2)$$

这里， $W = (w_{ij})$ 是 $n \times n$ 的连接权值矩阵， $T = (T_1, T_2, \dots, T_n)$ 是阈值向量。激活函数 φ 一般取符号函数：

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0 \end{cases}. \quad (4.3)$$

或者

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}. \quad (4.4)$$

根据神经元的状态更新方式，离散型 Hopfield 神经网络又可分为两种操作模式：同步（并行）模式和异步（串行）模式。在同步模式中，每步都更新所有的神经元，而在异步模式中，每步只更新一个神经元。相应的算法描述如下：

同步模式算法：

- 1、给定一个初始的网络状态 $y(0)$, 令 $t=0$;
- 2、给定一个终止规则 C ;
- 3、计算 $y(t+1) = \varphi(W \cdot y(t) - T)$;
- 4、如果 $y(t+1)$ 满足终止规则 C 则停止, 否则令 $t=t+1$, 转步骤 3。

异步模式算法:

- 1、给定一个初始的网络状态 $y(0)$, 令 $t=0$;
- 2、给定一个终止规则 C ;
- 3、按照 $(1, 2, \dots, n)$ 的某个排列顺序或随机选择一个 $i(t) \in (1, 2, \dots, n)$;
- 4、计算 $y_{i(t)}(t+1) = \varphi(W_{i(t)}y(t) - T_{i(t)})$ (这里 $W_{i(t)}$ 是权值矩阵 W 的第 $i(t)$ 行构成的向量);
- 5、令 $y_j(t+1) = y_j(t), \quad j=1, 2, \dots, n, \quad j \neq i(t)$;
- 6、如果 $y(t+1)$ 满足终止规则 C 则停止, 否则令 $t=t+1$, 转步骤 3。

离散型 Hopfield 神经网络的一个重要应用是用作联想存储器, 它也可以用来求解优化问题。

4.2.2 连续型 Hopfield 神经网络

连续型 Hopfield 神经网络与离散型 Hopfield 神经网络具有相同的拓扑结构, 它们的不同之处就在于采用了 Sigmoid 函数代替阈值函数作为神经元的激活函数。相应的, 神经元也就不再只有两种状态了, 而是在一个范围内连续变化。

连续型 Hopfield 神经网络可以由一个放大器电路实现。其中, 运算放大器模拟神经元的激活函数, 电压 u_i 为激活函数的输入, 也称为神经元 i 的内部状态, 各并联的电阻 R_{ij} 值决定各神经元之间的连接强度; 电容 C_i 和电阻 r_i 模拟生物神经元的输出时间常数, 电流 I_i 模拟阈值。整个网络是由 n 个同样的模型并联组成, 每个模型都具有相同的输入矢量 $V = (v_1, v_2, \dots, v_n)$, a_i 为神经元 i 的输出。根据克西霍夫电路定律, 它的动力学方程描述为:

$$C_i \frac{du_i}{dt} = -\frac{u_i}{R_i} + \sum_{j=1}^n w_{ij} v_j + I_i \quad (4.5)$$

$$v_i = \varphi_i(u_i) = \frac{1}{2} \left(1 + \tanh \left(\frac{u_i}{T} \right) \right) \quad (4.6)$$

其中, φ_i 为激活函数, T 是控制 Sigmoid 函数陡度的参数。 w_{ij} 为神经元 j 到神经元 i 的连接权值, 而 R_i 由下式确定:

$$\frac{1}{R_i} = \frac{1}{r_i} + \sum_{j=1}^n \frac{1}{R_j} \quad (4.7)$$

将方程 (2.10) 两边同时除以 C_i , 得到如下的动力学方程

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + \sum_{j=1}^n w_{ij} v_j + I_i \quad (4.8)$$

$$u_i = \varphi_i^{-1}(v_i), \quad \tau_i = R_i C_i \quad (4.9)$$

通常我们可以通过构造一个 Liapunov 函数来分析连续型 Hopfield 神经网络的收敛性和稳定性。Hopfield 神经网络能量函数定义为:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} v_i v_j - \sum_i I_i v_i + \sum_i \frac{1}{\tau_i} \int_0^v \varphi_i^{-1}(v) dv \quad (4.10)$$

当权值矩阵 W 对称时 (Vidyasagar 证明了在某种情况下, 即使 W 非对称, Hopfield 神经网络也是收敛的), 将能量函数对时间求导可得:

$$\begin{aligned} \frac{dE}{dt} &= -\sum_i \frac{dv_i}{dt} \left(\sum_j w_{ij} v_j + I_i - \frac{u_i}{\tau_i} \right) \\ &= -\sum_i \left(\frac{dv_i}{dt} \right) \left(\frac{du_i}{dt} \right) = -\sum_i \left(\frac{d\varphi_i^{-1}(v_i)}{dt} \right) \left(\frac{dv_i}{dt} \right)^2 \end{aligned} \quad (4.11)$$

因为 $\varphi_i^{-1}(v_i)$ 是单调递增函数, 所以有:

$$\frac{dE}{dt} \leq 0 \quad (4.12)$$

且

$$\frac{dE}{dt} = 0 \Leftrightarrow \frac{dv_i}{dt} = 0, \text{ 对 } \forall i \quad (4.13)$$

因为能量函数有界，所以(4.12)、(4.13)式表明，(4.8)式描述的 Hopfield 神经网络中，系统状态总是朝着能量函数减小的方向运动，最终收敛到能量函数 E 的局部极小点上。

能量函数不仅能用来分析神经网络，而且也可以用来构造神经网络。在求解一个优化问题，先选择一个合适的表示方法，将神经网络的输出和问题的解对应起来。然后构造能量函数，使其最小值对应于原问题的最优解。最后由能量函数推导出神经网络的动力学方程及各个参数，构造出硬件电路进行求解或用计算机软件进行仿真模拟。

4.2.3 采用连续 Hopfield 网络求解 TSP 问题

本文的做法是将 TSP 的合法解映射为一个置换矩阵，并给出相应的能量函数，同时将满足置换矩阵要求的能量函数最小值与 TSP 问题的最优解相对应。具体步骤如下：

1) 将神经元网络的输出状态与问题的解对应起来

使用一个 $n \times n$ 神经网络，用神经元的状态来表示某一城市在某一条有效路径中的位置，例如神经元 x_i 的状态用 v_{xi} 表示，其中 $x \in \{1, 2, \dots, n\}$ 表示第 x 个城市 C_x ，而 $i \in \{1, 2, \dots, n\}$ 表示城市 C_x 在路径中的第 i 个位置出现； $v_{xi}=0$ 表示 C_x 在路径中第 i 个位置不出现，此时第 i 个位置上是其它城市。

由此可见， $n \times n$ 矩阵 V 可以表示 n 个城市 TSP 问题的一次有效的路径，即 V 矩阵可以唯一的确定对于所有城市的访问次序。

2) 构造神经网络的能量函数

构造神经网络的能量函数为：

$$E = E_1 + E_2 + E_3 + E_4 \quad (4.14)$$

其中 $E_1 = \frac{A}{2} \sum_x \sum_i \sum_{j \neq i} v_{xi} v_{xj}$ ， $A > 0$ 为常数，此项保证矩阵 V 的每一行不多于一个 1 时（即每个城市只去一次）， $E_{1\min} = 0$ 最小。

$E_2 = \frac{B}{2} \sum_i \sum_x \sum_{y \neq x} v_{xi} v_{yi}$, $B > 0$ 为常数, 此项保证矩阵 V 的每一行不多于一个 1 时 (即每次访问而且只访问一个城市), $E_{2\min} = 0$ 最小。

$E_3 = \frac{C}{2} \left(\sum_x \sum_i v_{xi} - n \right)^2$, $C > 0$ 为常数, 此项保证矩阵 V 中的 1 的个数恰好为 n 时,

$E_{3\min} = 0$ 最小。

$E_4 = \frac{D}{2} \sum_x \sum_{y \neq x} \sum_i d_{xy} v_{xi} (v_{y,i+1} + v_{y,i-1})$, $D > 0$ 为常数, 此项定义中包含有效路径的长度信息。

3) 由能量函数逆推神经网络的结构

将式 (4.10) 与标准能量函数相比较, 可得神经元 x_i 与 y_j 之间的连接权值为

$$T_{xi,yj} = -A \delta_{ij} (1 - \delta_{ij}) - B \delta_{ij} (1 - \delta_{ij}) - C - D d_{ij} (\delta_{j,j+1} + \delta_{j,j-1}) (1 - \delta_{ij}) . \quad (4.15)$$

其中 δ_{ij}, δ_{yy} 定义为 $\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & \text{其他} \end{cases}$,

偏置输入为:

$$I_n = Cn \quad (4.16)$$

4) 建立网络, 令其运行, 当网络稳定时, 得到问题的最优解。

将网络结构 (4.15) 式代入网络的运行方程中可得:

$$C_n \frac{dU_{xi}}{dt} = -\frac{U_{xi}}{R_{xi}} - A \sum_{j \neq i} v_{xj} - B \sum_{y \neq x} v_{yi} - C \left(\sum_x \sum_i v_{xi} - n \right) - D \sum_{y \neq x} d_{iy} (v_{y,i+1} + v_{y,i-1})$$

$$v_{xi} = g(u_{xi}) \quad (4.17)$$

选择适当的参数值 (A, B, C, D) 和初值 U_0 , 按上式迭代直到收敛。TSP 中 v_{xi} 值要求为 0 或 1, 但用连续型网络时 v_{xi} 在 $[0,1]$ 区间变化, 因此实际计算时, v_{xi} 应在 $[0,1]$ 区间变化, 在连续演变过程中少数神经元的输出值逐渐增大, 其他则逐渐减少, 最后收敛到的状态符合要求即可。

4.2.4 算法的实现

为了模拟计算，首先将状态方程（4.17）离散化为差分方程：

$$u_x(n+1) = u_x(n) + h \left(-A \left(\sum_{j=1}^n V_{xj} - 1 \right) - B \left(\sum_{i=1}^N v_{xi} - 1 \right) - D \sum_{y=1}^N d_{xy} V_{yx,i+1} \right) \quad (4.18)$$

$$V_{xi} = \frac{1}{2} \left(1 + \tanh \left(\frac{u_x}{u_0} \right) \right) \quad x, i = 1, 2, \dots, N \quad (4.19)$$

初始化 U: U 在 0 附近取值。

虽然式子（4.17）是单调递增函数，已证明随着时间的推移，有 $\frac{dE}{dt} \leq 0$, $\frac{dE}{dt} = 0$ 的关系。但是，考虑到上述能量曲线和状态矩阵的变化，不能用后者判别是否到达极小值，而是检查是否得到合法回路。其次，记录合法回路的长度，要求第 $r+1$ 次路径的长度必须小于或等于第 r 次路径的长度，因为局部极小很多，除非得到全局最优解，连续两次合法路径的长度相等的概率几乎等于零，所以利用以上要求不难获得全局最优解。第 r 次计算路径长度的主要步骤为：

步骤 0：若 $r=1$ ，设初始化矩阵为 $V(N,N)$ ，其中 N 为景点个数。令迭代次数 $count=0$ ，转入步骤 1。若 $r>1$ ，对 $V(N,N)$ 任一行进行一个微小的扰动，令迭代次数 $count=0$ ，转入步骤 1。

步骤 1：对 $x, i = 1, 2, \dots, N$ 的式（4.18）和（4.19）进行 1 次迭代运算，记录迭代次数 $count=count+1$ 。

步骤 2：检查超时。如果 $count$ 达到迭代次数的上限时，终止第 r 次循环，计算路径长度，令 $r=r+1$ 转入步骤 0，否则，转入步骤 3。

步骤 3：判别合法路径。

(1) 当 V 中的每个元素是 0 (即 $V \leq 0.01$) 或 1 (即 $V > 0.99$) 时，继续执行 (2)；否则，转入步骤 1。

(2) 当每行每列恰有一个 1 时，继续执行步骤 4。否则，转入到步骤 1。

步骤 4：计算路径长度，从而求得状态矩阵，计算对应的回路长度。

步骤 5：求最短的回路长度。若 $r=1$ ，记录求得的回路长度 L 。若 $r>1$ ，第 r 次求得的回路长度 L 不大于第 $r-1$ 次求得的回路长度时，记录求得的回路长度 L ，结束第 r 次运算，令 $r=r+1$ 转入步骤 0。否则，转入步骤 1。

4.3 对现有算法的改进

应用 Hopfield 神经网络对 TSP 问题进行求解，在速度上有很大的优势。但是该方法借助于 Lyapunov 方法的思想构造能量函数，但并不知道网络平衡点位置，不能保证 $dE/dt=0$ 的点就是问题真正的极小点，故其稳定性分析不严格。利用该方法求解时，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。因此，很有必要对上述算法进行一定的改进。

4.3.1 对参数的分析和改进^[52]

我们使用了 $n \times n$ 个神经元构成的连续型 Hopfield 网络求解 TSP。网络能量函数为：

$$E = \frac{A}{2} \sum_x \sum_i \sum_{j \neq i} v_{xi} v_{xj} + \frac{B}{2} \sum_i \sum_x \sum_{x \neq y} v_{xi} v_{xy} \\ + \frac{C}{2} \left(\sum_x \sum_i v_{xi} - n \right)^2 + \frac{D}{2} \sum_x \sum_{x \neq y} \sum_i d_{xy} v_{xi} (v_{y(i+1)} + v_{y(i-1)}) \quad (4.20)$$

其中 $A>0$, $B>0$, $C>0$, $D>0$ 为常数。为以后讨论方便，我们定义 (4.20) 式中前三项为惩罚项，用 E_p 表示，最后一项为目标项。

在上一节提出 Hopfield 模型时，我们对 Hopfield 网络模型的参数做出了以下假定： $A=500$, $B=500$, $C=200$, $D=500$ 。我们就以此为基础来研究参数 A 、 B 、 C 、 D 、 Δt 和城市分布对网络性能的影响。

在进行研究之前，先假定一种城市拓扑。考虑到把城市间路线长度作为研究对象比城市坐标更具有广泛性（城市间的实际路线长度并非就是城市坐标间的几何距离），为

此选城市间的距离矩阵作为研究对象。

选用一个城市距离矩阵如表 4.1 所示。按照这种城市间路线长度矩阵，我们的研究结果如下。

表 4.1 城市间路程长度矩阵

	1	2	3	4	5	6	7	8	9	10
1	0.000	0.184	0.163	0.872	0.523	0.883	0.809	0.979	0.136	0.919
2	0.184	0.000	0.086	0.329	0.009	0.021	0.819	0.654	0.021	0.428
3	0.163	0.086	0.000	0.939	0.087	0.940	0.288	0.059	0.158	0.847
4	0.872	0.329	0.939	0.000	0.140	0.697	0.695	0.441	0.705	0.757
5	0.523	0.009	0.087	0.140	0.000	0.063	0.090	0.706	0.355	0.733
6	0.883	0.021	0.940	0.697	0.063	0.000	0.959	0.923	0.586	0.641
7	0.809	0.819	0.288	0.695	0.090	0.959	0.000	0.057	0.219	0.317
8	0.979	0.654	0.059	0.441	0.706	0.923	0.057	0.000	0.306	0.561
9	0.136	0.021	0.158	0.705	0.355	0.586	0.219	0.036	0.000	0.746
10	0.919	0.428	0.847	0.757	0.733	0.641	0.317	0.561	0.746	0.000

1、参数 A

把参数 A 的值从 100 递增到 1000，每次变化 A 递增 100。对于每个不同的 A 值，对网络进行同一种初始化，得到的结果如表 4.2 所示。

其它参数的设置为：B=500、C=200、D=500、DT=0.000001、TIMES=3000，其中 TIMES 为最多网络迭代次数，超过这个次数，网络迭代终止，下同。

从表 4.2 可以看出，网络能量与参数 A 之间并没有很明确的对应关系。A 较小时网

表 4.2 参数 A 对网络能量的影响

A	100	200	300	400	500	600	700	800	900	1000
E_m	845	600	984	714	1040	980	1191	1157	997	1073
E_p	703	305	615	400	400	554	462	874	875	400

络最小能量可能较大，如 A=100 时，最小能量 $E_m = 845$ ，相应惩罚函数项 E_p 为 703；A

较大时网络最小能量也可能较小，如 $A=900$ 时， $E_m = 997$ ，相应惩罚函数项 E_p 为 875。这是由于网络惩罚项对网络能量的约束力还不是很强，该收敛到 1 的神经元的输出值离 1 的差距还很大，因此网络收到的惩罚很严重。但从表 4.2 仍可以看出，随着 A 值得增大，网络最小能量和相应的惩罚项都有变大的趋势，但这并不绝对。可以肯定的是，在网络没有满足预订约束的情况下， A 的值越大，网络收到的惩罚就越大，表现在能量上就是它的惩罚项越大。

对于某个给定的 A 值，如 $A=400$ ，把网络随机初始化十次，其最小能量和相应惩罚项如表 4.3 所示。

从表 4.3 可以看出，即使网络的参数保持不变，网络的最小能量和惩罚项的分布很

表 4.3 $A=400$ 时的网络能量和相应的惩罚项

运行次数	1	2	3	4	5	6	7	8	9	10	平均值
E_m	714	1298	1388	1395	974	1045	1095	1076	686	647	1032
E_p	400	1188	874	400	384	800	812	800	400	400	646

没有规则。说明在网络参数不变的情况下，网络的输入初始化设置对网络的性能影响较大。

2、参数 B

同对参数 A 的研究一样，对于同一种网络初始化，参数 B 得到的结果与参数 A 相似。网络能量与参数 B 之间的关系，如同它与参数 A 的关系一样，并不明显。参数 B 对网络的影响与参数 A 对网络的影响完全类似，对它的分析可以参照前面对参数 A 的分析结论。

3、参数 C

同对前面参数的研究一样，对于同一种网络初始化，得到的结果如表 4.4 所示：

表 4.4 参数 C 对网络能量的影响

C	50	100	150	200	250	300	350	400	450	500
E_m	373	667	913	1040	952	923	993	1368	1542	1563
E_p	225	274	420	400	126	538	644	1188	1222	1296

其它参数的设置为: A=500、B=500、D=500、DT=0.000001、TIMES=3000。

从表 4.4 可以看出, 网络能量随着参数 C 的递增而呈增加趋势, 同样, 这不是绝对的。可以看到, C 较小时网络能量较小, C 较大时网络能量较大。这是由于网络惩罚项 C 对网络能力的约束力相对于约束项 A 和 B 来说较强, 它是网络全局的惩罚项, 且 C 的值越大, 惩罚越大。如果网络最终运行得到的结果距离理想结果的差距很大时, 惩罚部分的值会很大, 如 C=400 时, $E_m=1368$ 而惩罚项为 1188, 目标项仅为 $E_m-E_p=180$, 可以想象, 这次网络运行的结果是错误的。在这十次网络的运行中, 仅在 C=250 时的结果是正确的。此时, $E_m=952$ 而惩罚项为 126, 它能够较好的满足网络的约束, 因而得到了有效的解, 结果是: 3-8-7-10-2-6-5-4-1-9-3, 其路径总长度是 2.251, 这个解的质量是比较好的。

4、参数 D

改变参数 D 值, 对于同一种网络初始化, 得到的结果如表 4.5 所示。其它参数为: A=500、B=500、C=500、DT=0.000001、TIMES=3000。

表 4.5 参数 D 对网络能量的影响

D	100	200	300	400	500	600	700	800	900	1000
E_m	541	407	750	650	1040	789	868	1125	1243	1190
E_p	71	98	229	356	400	400	400	876	875	875

从表 4.5 可以看出, 网络能量随着参数 D 的递增而呈增加的趋势。

值得注意的是, 随着 D 的增加, 网络能量的惩罚部分明显的增加。说明 D 值越大, 加在网络上的惩罚就越大。由此, 对 D 项在网络收敛方面的影响作了一些研究, 假定参数设置同上。

基于随机选择的城市间的路径长度, 把 D 从 100 变化到 1000, 每次进行 100 次网

络运行，而每次网络运行之前进行输入随机初始化，结果如表 4.6 所示。

表 4.6 参数 D 对网络收敛性的影响

D	有效收敛率	最优路径 C_{exp}	平均路径长度
100	100%	3.34	5.08
200	89%	3.80	4.84
300	46%	3.61	4.43
400	10%	3.44	3.98
500	2%	3.70	3.95
600	1%	3.39	3.39
700	0	/	/
800	0	/	/
900	0	/	/
1000	0	/	/

从上表可以看出，参数 D 的值越小，网络的有效收敛率就越高。而 D 最小时网络的最优路径解也是最好的，但是平均的路径长度则不是很好，相对而言，在上例中它是最差的，D=100 时平均路径长度为 5.08。这说明参数 D 较小时网络的运行结果有一定的不稳定性。

5、参数 Δt

把参数 Δt 从 10^{-6} 变化到 10^{-4} ，前面每次递增 10^{-6} ，后面相应的加大。对于同一种网络初始化，得到的结果如表 4.7 所示：

表 4.7 Δt 对网络能量的影响

Δt	1	2	3	4	5	10	15	20	21	25
E_m	1498	740	1846	1053	1239	1292	989	1089	10^4	10^4
E_p	508	229	1375	400	400	444	237	283	10^4	10^4

注：表中， Δt 的单位为 10^{-6} 。其它参数的设置为：A=500、B=500、C=200、D=500、

TIMES=3000。

从表 4.7 中可以看出，时间间隔 Δt 在 10^{-6} 和 2×10^{-5} 时，网络就要出问题了：因为网络的反馈过大使得神经元的输出很快全变为 1 或者全变为 0，反映在能量函数上就是网络最小能量和能量的惩罚部分相等，而目标函数部分则为 0，由此网络运行完全失去意义。因此，兼顾网络运算时间，在参数即定为 A=500、B=500、C=200、D=500 的情况下，我们选择时间间隔 Δt 的范围是在 10^{-6} 和 2×10^{-5} 之间，此时网络运行正常。

6、结论

我们在 Hopfield 对网络模型的参数作出假定（A=500、B=500、C=200、D=500）的基础上，通过以上研究，对参数有以下结论：

(1) 对于网络的行约束和列约束，其参数 A 和 B 应该定义在 400~700 的范围内。太大或太小都会引起网络工作的严重不正常，表现在能量函数上就是惩罚项会变得很大，目标项变得太小，网络无法收敛到预料的神经元解。

(2) 对于网络的全局约束项，其参数 C 应该在 150~250 之间。太大或太小也会比较严重的影响网络的收敛性能，导致与 A 和 B 相类似的结果。

(3) 网络目标项参数 D 对于网络的性能有较明显的影响。D 较小时，网络的收敛率有大大的提高。

(4) 选择时间间隔 Δt 的范围是在 10^{-6} 和 2×10^{-5} 之间，此时网络运行正常。

从以上的分析表明：网络参数值得确定至关重要，对神经网络的收敛性能有直接的影响。本文进一步完善了网络参数的取值，为今后的研究及实际工作提供了有益帮组，为真正达到用神经网络解决优化问题的目的提供了捷径。

4.3.2 对能量函数的改进

在上两节中我们采用的能量函数如下式所示：

$$E = \frac{A}{2} \sum_{x=1}^n \sum_{i=1}^n \sum_{j=1, j \neq i}^n V_{xi} V_{xj} + \frac{B}{2} \sum_{x=1}^n \sum_{y=1, y \neq x}^n \sum_{i=1}^n V_{xi} V_{yi}$$

$$+ \frac{C}{2} \left(\sum_{x=1}^n \sum_{i=1}^n V_{xi} - n \right)^2 + \frac{D}{2} \sum_{x=1}^n \sum_{y=1}^n \sum_{i=1}^n d_{xy} V_{xi} (V_{y,i+1} + V_{y,i-1}) \quad (4.21)$$

其中第三项仅在网络输出全为 0 时方起约束作用，否则前两项已经保证了第三项成立。如前两项做修改，那第三项完全可省去，于是，TSP 的能量函数简化为：

$$\frac{A}{2} \sum_{x=1}^n \left(\sum_{i=1}^n V_{xi} - 1 \right)^2 + \frac{B}{2} \sum_{i=1}^n \left(\sum_{x=1}^n V_{xi} - 1 \right)^2 + \frac{D}{2} \sum_{x=1}^n \sum_{y=1}^n \sum_{i=1}^n V_{xi} d_{xy} V_{xi} V_{y,i+1} \quad (4.22)$$

由行列的俄对称性取 $B=A$ ， D 项由式 (4.21) 的 $(V_{y,i+1} + V_{y,i-1})$ 两项简化为 $V_{y,i+1}$ 一项，项目方程式仍满足优化目标约束之要求。神经元 (x, i) 的动态方程有关系：

$$\frac{d u_{xi}}{dt} = -\frac{\partial E}{\partial V_{xi}} (x, i = 1, 2, \dots, n) \quad (4.23)$$

导出：

$$\frac{d u_{xi}}{dt} = -A \left(\sum_{j=1}^n V_{xj} - 1 \right) - A \left(\sum_{y=1}^n V_{yi} - 1 \right) - D \sum_{y=1}^n d_{xy} V_{yi} V_{y,i+1} \quad (4.24)$$

神经元特性曲线可选 Sigmoid 函数，可选分段线性函数，网络的权重 T_{xi} ， y_i 和外部输入偏置电流 I_{xi} 可表示为：

$$T_{xi,yi} = A \delta_{xy} - A - \delta_{iy} - D_{xy} - \delta_{y,i+1} \quad (4.25)$$

$$I_{xi} = 2A \quad (4.26)$$

理论上证明了该算法的正确性，模拟结果也表明此算法的有效性。下面的程序描述了 Hopfield 神经网络求解 TSP 的改进算法，其中 A 、 D 为实验值。

算法步骤如下：

- ① 设 $t=0$, $A=1$, $D=1$ 。② 读入 n 个城市之间距离 d_{xy} ($x, y = 1, 2, \dots, n$)。③ 计算权重 $T_{xi,yi}$ 和输入偏置 I_{xi} (见式 (4.25) 和 (4.26))。④ $U_{xi}(t)$ 初始值在 0 附近产生 ($x, i = 1, 2, \dots, n$)。⑤ 计算 $V_{xi}(t)$ ($x, i = 1, 2, \dots, n$)， $V_{xi}(t) = 1/2(1 + \tanh U_{xi}(t)/U_0)$ ，取 $U_0 = 0.02$ 。⑥ 利用动态方程计算 $\Delta U_{xi}(t)$: ($x, i = 1, 2, \dots, n$)

$$\Delta U_{xi}(t) = \sum_{y=1}^n \sum_{j=1}^n T_{xi,yj} V_{yj} + I_{xi} \quad (4.27)$$

- ⑦ 根据一阶欧拉算法计算 $U_{xi}(t+1): U_{xi}(t+1) = U_{xi}(t) + \Delta U_{xi}(t)$ $\Delta t, x, i = 1, 2, \dots, n$ 。取 $\Delta t = 0.5$ 。⑧ 如果系统达到平衡状态，则终止，否则返回⑤。

4.3.3 对上述算法的进一步改进

在上一节我们求解 TSP 的能量函数进行了改进，这个算法具有收敛速度快，无避免无效解，对网络参数不敏感等优点。通过对问题及算法进行分析，我们可以发现，这种表示方法仍然有个缺陷，那就是解的等价类的存在问题，就是说，每个解都存在 $N-1$ 个解与之完全等价。例如，路线 DACBD 和路线 ACBDA, CBDAC, BDACB 是完全等价的。这种等价解在计算过程中不必要的重复出现，降低了计算效率。我们可采取固定起点的方法来消除这种现象，比如固定起点为 A，用 $(N-1) \times (N-1)$ 阶矩阵来表示有效解。如访问路线 $x_1 x_2 x_3 x_4 x_1$ 可表示为图 4.3 的形式：

$$\begin{matrix} & 1 & 2 & 3 \\ x_2 & 0 & 1 & 0 \\ x_3 & 1 & 0 & 0 \\ x_4 & 0 & 0 & 1 \end{matrix}$$

图 4.3：有效解的矩阵形式

神经元 (x, i) 的输出仍用 V_{xi} 表示，此时神经元共有 $(N-1)^2$ 个， $V_{xi}=1$ 表示从 A 市出发先到达市 x，相应的能量函数可在原有基础上增加 A 市与其他各城市距离信息，则变为：

$$E = \frac{A}{2} \sum_{x=1}^{N-1} \left(\sum_{i=1}^{N-1} V_{xi} - 1 \right)^2 + \frac{A}{2} \sum_{i=1}^{N-1} \left(\sum_{x=1}^{N-1} V_{xi} - 1 \right)^2 + \frac{D}{2} \left[\sum_{x=1}^{N-1} \sum_{y=1}^{N-1} \sum_{i=1}^{N-2} V_{xi} d_{xy} V_{y,(i+1)} \right. \\ \left. + \sum_{y=1}^{N-1} d_{0y} V_{y1} + \sum_{y=1}^{N-1} d_{y0} V_{y(N-1)} \right] \quad (4.28)$$

神经元的动态方程有关系： $\frac{d u_{xi}}{dt} = -\frac{\partial E}{\partial V_{xi}}$ ($x, i = 1, 2, \dots, N-1$) 并导出：

$$\frac{d u_{xi}}{dt} = -A \left[\sum_{j=1}^{N-1} V_{xj} - 1 \right] - A \left[\sum_{y=1}^{N-1} V_{yi} - 1 \right] - B \sum_{y=1}^{N-1} d_{xy} V_{y,(i+1)} \quad (4.29)$$

其中 $1 < i < N - 1$

$$\frac{d u_{x1}}{dt} = -A \left[\sum_{j=1}^{N-1} V_{xj} - 1 \right] - A \left[\sum_{y=1}^{N-1} V_{yi} - 1 \right] - \frac{B}{2} \left[\sum_{y=1}^{N-1} d_{xy} V_{y2} + d_{x0} \right] \quad (4.30)$$

$$\frac{d u_{x(N-1)}}{dt} = -A \left[\sum_{j=1}^{N-1} V_{xj} - 1 \right] - A \left[\sum_{y=1}^{N-1} V_{yi} - 1 \right] - \frac{B}{2} \left[\sum_{y=1}^{N-1} d_{xy} V_{y(N-2)} + d_{x0} \right] \quad (4.31)$$

上述算法的能量函数及导出的神经元动态方程与前面的算法本质上是相同的，从而具有前面算法的优越性，但神经元由原来的 N^2 个减为 $(N-1)^2$ 个，在求解过程中减少了不必要的等价解，从而提高计算效率。

4.4 实验结果

为了测试算法的有效性，分别选取 10 城市、20 个城市和 40 个城市，采用不同的迭代次数对算法进行模拟。城市坐标为：

10 城市：6734 1453; 2233 10; 5530 1424; 401 841; 3082 1644; 7608 4458; 7573 3716;
7265 1268; 6898 1885; 1112 2049,

20 城市：37 52; 49 49; 52 64; 20 26; 40 30; 21 47; 17 63; 1 62; 52 33; 51 21;
42 41; 31 32; 5 25; 12 42; 36 16; 52 41; 27 23; 17 33; 13 13; 57 58

30 城市：24 53; 33 103; 55 24; 40 84; 82 44; 76 58; 73 16; 65 68; 98 18; 11 20;
54 6; 89 73; 47 26; 46 35; 47 83; 61 66; 11 84; 62 90; 32 23; 59 361; 48 39; 61 10;
59 22; 13 28; 43 22; 67 16; 75 48; 71 31; 37 76; 19 56

坐标格式：A X Y；其中 A 表示城市名称；X 表示城市横坐标；Y 表示城市纵坐标。

结果如图 4.1、图 4.2 和图 4.2 所示：

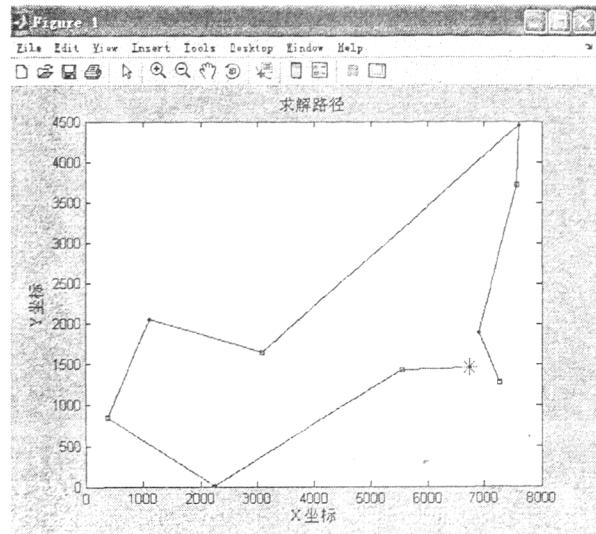


图 4.1 10 城市最佳路线图

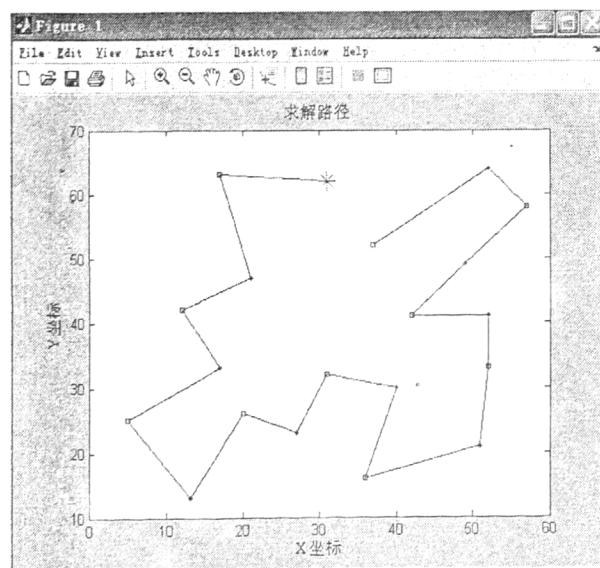


图 4.2 20 城市最佳路线图

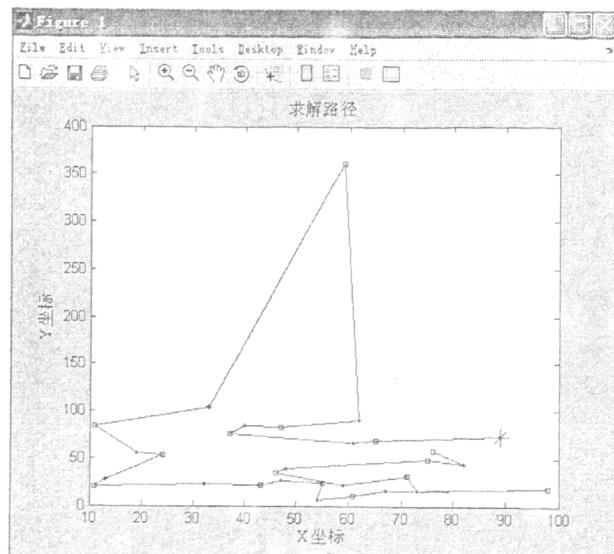


图 4.3 40 城市最优路线

10 城市最佳路径: $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 10 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 9 \rightarrow 8$; 总路径 19520

20 城市最佳路径: $8 \rightarrow 7 \rightarrow 6 \rightarrow 14 \rightarrow 18 \rightarrow 13 \rightarrow 19 \rightarrow 4 \rightarrow 17 \rightarrow 8 \rightarrow 12 \rightarrow 5 \rightarrow 15 \rightarrow 10 \rightarrow 9 \rightarrow 16 \rightarrow 11 \rightarrow 2 \rightarrow 20 \rightarrow 3 \rightarrow 1$; 总路径 243.1792.

30 城市最佳路径: $12 \rightarrow 8 \rightarrow 16 \rightarrow 29 \rightarrow 4 \rightarrow 15 \rightarrow 18 \rightarrow 20 \rightarrow 2 \rightarrow 17 \rightarrow 30 \rightarrow 1 \rightarrow 24 \rightarrow 10 \rightarrow 19 \rightarrow 25 \rightarrow 13 \rightarrow 3 \rightarrow 11 \rightarrow 22 \rightarrow 26 \rightarrow 9 \rightarrow 7 \rightarrow 28 \rightarrow 23 \rightarrow 14 \rightarrow 21 \rightarrow 27 \rightarrow 5 \rightarrow 6$; 总路径 978.533

此外, 运用该方法我们还获得了中国 31 个省会城市的长为 15521km 的最优巡回路径为: 北京→石家庄→太原→呼和浩特→银川→西安→兰州→西宁→乌鲁木齐→拉萨→成都→昆明→贵阳→南宁→海口→广州→长沙→武汉→南昌→福州→台北→上海→杭州→南京→合肥→郑州→济南→天津→沈阳→长春→哈尔滨→北京。

4.5 本章小结

本章提出了一种基于 Hopfield 神经网络的求解算法。在介绍了 Hopfield 神经网络模型之后, 详细介绍了如何实现其对 TSP 问题的求解。在分析该方法的缺陷之后, 分别就参数设置和能量函数对现有算法进行了改进, 并针对有重复解的现象, 对算法提出了进一步的改进。最后给出了改进算法求解的结果。

实验结果表明算法改进后在求解速度和算法收敛性上都有了很大的提高。

第五章 应用 Hopfield 神经网络解决西安市旅游问题

西安作为中国著名的历史文化名城，同北京、南京、洛阳、开封、杭州共为我国六大古都。先后有周、秦、汉、唐等 13 余个王朝在此建都，是六大古都中最古老、建都时间最长的一个。文化遗存丰富、文物古迹荟萃，帝都风韵犹存，拥有十分丰富的旅游资源。据不完全统计，西安市及其周边地区划分的十大旅游区共有景点 177 个；其中仅西安市就有大小景点 60 多个。面对如此众多的旅游景点，作为一名有客观条件约束的游客，如果不能够合理的安排自己的行程、正确的选择合适的旅游路线，无疑会给游客自身带来不必要的浪费。

对于旅行商问题来说，它的限制条件只是要求旅行者遍历所有城市当且仅当一次，而使得路径最小，除此之外并没有其余额外的附加条件。而西安旅游问题则稍有不同，该问题不仅和旅行商问题有着同样的要求，而且还有许多实际的约束，比如：景区风景资源能否满足游客的需求；景区气候条件如何；游客在本次旅游过程中所能付出的精力、物力、财力；景点的门票以及交通费用等。所以，对于西安市旅游路线的选择问题，除了按照旅行商问题给出的方法予以解决外，还应当给予适当的补充。

下面先看一些概念。

定义 5.1 游客在旅游过程中所获得的身心上满足程度称为旅游收益。它主要由景区风景资源、景区气候条件、配套服务设施等确定。

定义 5.2 游客在整个旅游过程中所付出的精力、物力、财力称为旅游付出。它主要由景点门票、交通消费（包括交通时间、舒适方便程度、路线、费用等）、食宿购物消费等服务设施确定。

定义 5.3 旅游过程中旅游收益与旅游付出的比值称为旅游效用 G。

定义 5.4 每单位旅游距离上的旅游效用称为旅游有效距离 D（简称有效距离），即旅游有效距离 $D = \text{旅游效用 } G / \text{旅游距离 } d$ 。它在普通交通距离的基础上考虑了景点风景状况、行车时间、行车费用、行车舒适方便程度等影响因素，既反映了游客对景点的认可程度，又反映风景区路面交通状况的好坏。

定义 5.5 由于不同的游客旅游要求不同，因此根据其要求对各指标赋以不同的权重，由此得出的景点称为最佳景点。此类景点未必与生活中公认的“好”景点完全一致，但它对这类游客确是最合适的景点，因此称之为这类游客的最佳景点。

在给出以上概念的基础上，我们把以上概念加到求解 TSP 问题的模型中，便可以得到旅游景点的选择方法。

5.1 旅游景点向量矩阵

本文考虑的旅游路线问题是从一个旅游景点出发，对每个要游览的景点都游览一遍，且每个景点只经过一次。基于这种思想，构建模型如下：

用网络框架表示旅游路线问题，每个神经元对应于一个按次序访问的景点。由于各神经元的输出在 0 到 1 之间，两个极端取值就对应着是访问还是不访问。例如 4 个景点，如果某个景点被第 3 个访问，景点可以表示成向量 $(0,0,1,0)$ 。对 4 个景点的地图，可以创建一个包括 $4 \times 4 = 16$ 个神经元的 Hopfield 网络。这意味着，一条路线可以表示成一个 $n \times n$ 的神经元输出 V_s 的状态矩阵 V_s （列表示路线中的不同位置，行表示景点。约定元素 $V_{s,i}$ 的第 1 个下标表示行，第 2 个下标表示列）。

$$V_s = \begin{bmatrix} V_{11} & V_{12} & V_{13} & V_{14} \\ V_{21} & V_{22} & V_{23} & V_{24} \\ V_{31} & V_{32} & V_{33} & V_{34} \\ V_{41} & V_{42} & V_{43} & V_{44} \end{bmatrix} \quad (5.1)$$

假设 4 个景点 $P = (p_1, p_2, p_3, p_4)$ 应该按下面顺序访问： $p_3 \Rightarrow p_1 \Rightarrow p_4 \Rightarrow p_2$ ，则状态矩阵为 V_p ：

$$V_p = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.2)$$

5.2 景点的选取

不同的游客对西安的景点有着不同的爱好，而西安市周边也有许多不同的景点，按照现有常识，我们知道西安市的旅游景点（如图 5.1）大体可分为 5 条路线—市内、东线、西线、北线和南线。考虑到各种因素，选取的景点包括了西安具有代表性的著名景点，有名胜古迹、寺院、陵墓、博物馆、山水等。



图 5.1：西安市旅游景点图

5.2.1 游客对景点的评价^[53]

在西安市旅游问题中加入游客对景点的评价，就是为了使选出来的景点更适合游客自身的需求，使选景、选线更符合实际要求。作为其中的一个特例。即游客没有特殊要求时，最佳景点应以“景点风景资源优、服务设施好、同时游玩付出代价适中”的原则来评价。则评价出的最佳景点必与公认的好景点一致。例如可以分配权重为(0.5,0.35,0.15)。出于国情考虑，大多数游客是属于中低收入者，在游玩过程中优先考虑

的是风景及价格。因此，此类游客的旅游需求为“景点风景区资源较优且无特殊风景要求，服务设施尚可，但游玩代价最小”。

假设某位游客在选定景点范围以后，对该范围内的景点依次进行如定义 5.1—5.5 给出的评价（均小于 1），把游客对同一景点的所有评价取平均值就可以得到该游客对景点 i 的综合评价 λ_i （小于 1）。

5.2.2 景点坐标的确立^[54]

假设该游客选取的景点范围中包括如下景点：法门寺、乾陵、楼观台、杨贵妃墓、翠华山、陕西省历史博物馆、大雁塔、秦始皇陵、秦俑博物馆、黄帝陵、华清池、司马迁祠、钟楼、青龙寺和华山。并且依据自身的旅游要求以及各景点的知名度对各个景点依次做出的综合评价为：

$$\lambda_i = [0.45 \ 0.63 \ 0.32 \ 0.85 \ 0.86 \ 0.62 \ 0.89 \ 0.91 \ 0.83 \ 0.43 \ 0.88 \\ 0.52 \ 0.82 \ 0.43 \ 0.95] \quad (5.3)$$

式中值越大表示景点的知名度越高，同时也表示是西安市最有代表性的旅游景点，是一般来西安旅游的游客必会游览的地方。

首先，利用西安旅游地图，用画图板进行编辑，将地图的左下角设置为坐标原点(0, 0)。给出各景点的初始坐标：法门寺(51, 217)；乾陵(83, 251)；楼观台(69, 83)；杨贵妃墓(110, 147)；翠华山(367, 14)；陕西省历史博物馆(275, 130)；大雁塔(286, 15)；秦始皇陵(326, 191)；秦俑博物馆(386, 176)；黄帝陵(382, 316)；华清池(356, 227)；司马迁祠(435, 297)；钟楼(263, 177)；青龙寺(320, 132)；华山(416, 227)。

其次，由于画图板的坐标原点在左上角，所以必须对所有景点作如下坐标变换：

$$\begin{cases} x = |x - x_0| \\ y = |y - y_0| \end{cases} \quad (5.4)$$

式中: (x_0, y_0) 为在左下角选取的一个作为新坐标原点的旧坐标。这种坐标变换保证了各景点方位及相对位置不变, 仍然符合地图规则(上-北, 下-南, 左-西, 右-东)。所选景点的坐标经过变换形成的新矩阵为:

$$P_{Coord_0} = \begin{bmatrix} 51 & 83 & 69 & 110 & 367 & 275 & 286 & 326 & 386 & 382 \\ 217 & 251 & 83 & 147 & 14 & 130 & 115 & 191 & 176 & 316 \\ 356 & 435 & 263 & 320 & 416 \\ 227 & 297 & 177 & 132 & 227 \end{bmatrix} \quad (5.5)$$

对于各个景点坐标, 为了方便显示, 将每个坐标值归一化在(0, 1)之间。将景点坐标归一化后的矩阵为:

$$P_{Coord_1} = \begin{bmatrix} 0.051 & 0.083 & 0.069 & 0.110 & 0.367 & 0.275 & 0.286 & 0.236 & 0.386 & 0.382 \\ 0.217 & 0.251 & 0.083 & 0.147 & 0.014 & 0.130 & 0.115 & 0.191 & 0.176 & 0.316 \\ 0.356 & 0.435 & 0.263 & 0.320 & 0.416 \\ 0.227 & 0.297 & 0.177 & 0.132 & 0.227 \end{bmatrix} \quad (5.6)$$

将综合评价 λ_i 与 P_{Coord_1} 综合即可得到初始化矩阵 V:

$$V = \begin{bmatrix} 0.2505 & 0.3565 & 0.1945 & 0.48 & 0.6135 & 0.4475 & 0.588 & 0.618 & 0.608 \\ 0.3335 & 0.3505 & 0.2015 & 0.4985 & 0.437 & 0.375 & 0.5025 & 0.5505 & 0.503 \\ 0.406 & 0.618 & 0.4775 & 0.5415 & 0.375 & 0.683 \\ 0.373 & 0.5535 & 0.4085 & 0.4985 & 0.281 & 0.5885 \end{bmatrix} \quad (5.7)$$

5.3 实验结果

分别选取 10 个景点和 15 个景点对该算法进行模拟, 求出最优旅游路线(即最短路线)。在模拟的过程中我们发现: 由于初始值的随机性, 达到最优结果的迭代次数也不一样。对于 10 个景点, 迭代次数在 200~450 之间, 大部分集中在 250~350 之间。对于 15 个景点, 迭代次数在 200~550 之间, 大部分集中在 250~350 之间。说明该算法对景点数在 20 以下时不太敏感, 所以对参数的改动不会对算法有太大影响。但是, 算

法对神经元初值比较敏感，不同的初值达到最优解的迭代次数不同，所以在本算法中，选择合适的初始值十分必要。本算法的初始值产生方法是，首先将综合评价 λ 与景点的编辑坐标归一化的结果综合得到的矩阵作为初始矩阵，然后加一个偏置值 0.05。在神经网络的迭代中，会随机的对状态矩阵的任一行进行一微小的扰动（类似于十进制的遗传算法的变异），以加速迭代。

第六章 总结与展望

6.1 总结

本文在研究了大量参考文献的基础上，对 TSP 问题进行了深入的分析，并研究了大量求解 TSP 的算法。选择了遗传算法、模拟退火算法和蚁群算法这三个比较有代表性的算法进行了编程实现，在比较了这三种算法的基础上提出了一种基于 Hopfield 神经网络的 TSP 问题的求解方法。并对现有的算法进行了改进得到了比较满意的结果。最后应用改进后的算法对西安旅游问题进行了求解并得到了最优的旅游路线。具体的工作如下：

- (1) 研究了大量求解 TSP 的算法，并进行了整理和分类，给出了算法的主要思想。
- (2) 建立了 TSP 问题的数学模型，并对问题的复杂度进行了分析。
- (3) 给出了遗传算法、模拟退火算法和蚁群算法的具体算法流程，并进行了编程实现。应用这三种算法分别解决了 48 城市，51 城市，41 城市的 TSP 最优路线求解问题，并对结果进行了分析。
- (4) 分析 Hopfield 神经网络模型，并在此基础上应用连续型 Hopfield 神经网络对 TSP 问题进行求解。经过研究发现该方法在求解 TSP 问题上有很大的不稳定性，能否得到最后结果很大程度上取决于初始参数的设置。在认识到这一点后对，对每个参数对结果的影响进行了分析，最后给出了参数的合理设置方法。本文还对能量函数进行了改进，使得问题的求解更加快速准确。最后对‘出现重复解的问题’进行了解决采用了一种从固定起点出发的办法。最后给出了新算法的具体的求解步骤，并进行了编程实现。
- (5) 把西安旅游路线选择问题转换成一个 TSP 问题，并应用本文提出的新算法进行了实现。结合西安旅游地图，根据具体的旅游问题给出了网络的能量函数，进而构建了一个 Hopfield 神经网络。选取了西安的著名旅游景点，对景点进行了变换和归一化，对算法编程进行实验。实验结果表明，该方法对 10 个景点和 15 个景点的迭代次数大都集中在 250~350 之间，说明该方法对于处理旅游路线的选择问题是行之有效的。

6.2 展望

人工神经网络技术作为人工智能技术新手段，是目前国际上迅速发展的前沿研究方向之一。本文应用了 Hopfield 进行了 TSP 问题的求解只是一个简单的应用。人工神经网络在限制性条件满足与最优化、数据压缩、预测与危险判断、控制、内容可寻址存储、多传感器数据融合、模式识别以及诊断等许多领域都有着广泛的应用前景，这也是作者今后的主要研究方向。

另外，本文求解西安旅游问题的实例可以进一步做成一套旅游指南系统。游客可以在该系统上任意选择想去的旅游景点，然后系统给出最佳的旅游路线。再结合旅游景点设计一些图片和视频介绍，会使该系统更加有吸引力。这样一套系统因该很有市场前景，作者也会继续做这方面的研究工作。

参考文献

- [1] 鄢克雨. Hopfield 神经网络的稳定性分析[D]. 电子科技大学. 2004
- [2] 李玮. 关于旅行商问题的改进遗传算法[D]. 重庆大学. 2004
- [3] 梁艳春, 冯大鹏等. 遗传算法求解旅行商问题的基因片段保序. 系统工程理论与实践 [J]. 2000(4):7-8
- [4] 严晨, 王直杰. 以 TSP 为代表的组合优化问题研究现状与展望. 计算机仿真[J]. 2007, 24(6):171-173
- [5] 陈志平, 徐宗本. 计算机数学—计算复杂性理论与 NPC、NP 难问题的求解[M]. 北京: 科学出版社, 2001.
- [6] Reeves CR(Ed).Modern Heuristic Techniques for Combinatorial Problems.[M]. Oxford: Blackwell Scientific Publications,1993.
- [7] 邢文训, 谢金星. 现代优化计算方法[M]. 北京: 清华大学出版社, 1999.
- [8] S Lin, B W Kernighan..An effective heuristic algorithm for the traveling salesman problem[J].Operations Research. 1973(21):498-516
- [9] H.J.hopfield&D.Tank."Neural"computation of decisions in optimization problems[J].Biolog Cybernet. 1985(52):141-152
- [10] G V Wilson & G S Pawley.On the stability of the traveling salesman problem algorithm of Hopfield and Tank[J].Biolog.Cybernet.1988(58):63-70
- [11] Sreeram V B Aiyer,Mahesan Niranjan and Frank Fallside.A theoretical investigation into the performance of the Hopfield model[J],IEEE Transactions on Neural Networks,June,1990,1(2):204-215.
- [12] S Abe. Theories on the Hopfield neural networks [J]. Proc. I J C N N .,1989,I:557-564.
- [13] G C Fox.,Physical computation .Concurrency[J].Practice and Experience,1991 , 3(6):627-653.
- [14] F Glover. Future paths for integer programming and links to artificial intelligence[J], Computers and Operations Research,13:533-549,1986.

- [15] S Kirkpatrick,Jr C D Gelatt,M P Vecchi,Optimization by simulated annealing[J].Science,220:671-680,May,1983.
- [16] Garrison W Greenwood, Ajay Gupta, Scheduling task in multiprocessor system using evolutionary strategies[J].The International Joint Conference on Neural Networks, Nagoya, Japan,216-2242,1993.
- [17] N Metropolis, et al. Equation of state calculations by fast computing machines[J].Journal of Chemical Physics,1953,21:1087-1092.
- [18] 郭茂祖,洪家荣. 基于模拟退火算法旅行商问题的并行实现. 哈尔滨理工大学学报 [J], 1997,2(5):84-85.
- [19] J H Holland., Adaption in Natural and Artificial System[M], The University of Michigan Press,1975:150-195.
- [20] 潘立登, 黄晓峰. 用改进的遗传算法求解中国旅行商问题. 北京化工大学学报 [J], 1997,24(1):31-34.
- [21] 罗映红. 遗传算法求解组合优化问题研究. 甘肃科学学报[J], 1997,9(2):101-103.
- [22] 林焰, 郝聚民, 纪卓尚, 戴寅生. 隔离小生境遗传算法研究. 系统工程学报 [J], 2000,15(1):64-65.
- [23] 覃俊, 蓝雯飞, 兰华荣. 用遗传算法求解旅行商问题. 中南民族学院学报(自然科学版) [J], 2000,19(1):41-42.
- [24] 胡小兵, 吴树范, 江驹. TSP 的一种改进遗传算法. 计算技术与自动化 [J], 2000,19(4):121-123.
- [25] 谢秉磊, 李军, 刘建新. 有时间约束旅行商问题的启发式遗传算法. 西南交通大学学报[J], 2001,36(2):34-35.
- [26] 周江鸣. 用玻尔兹曼网络解旅行商问题. 计算机工程与应用[J], 1997,10:60-61.
- [27] 党建武, 靳蕃. 神经网络求解 MTSP 的应用研究. 铁道学报[J], 1997,19(5):53-54.
- [28] 姜国均. Hopfield 网络解 TSP 的改进算法. 浙江大学学报(理学版) [J], 2001,28(2):89-90.
- [29] K.Aihara,T.Takabe and M.Toyoda, Chaotic-neural-networks[J], Physics Letters A, 1990, 144(6,7):333-340.

- [30] Luonan Chen and Kazuyuki Aihara, Chaotic Simulated Annealing by a Neural Network Model with Transient Chaos[J].Neural Networks,1995,8(6):915-930.
- [31] 王凌, 郑大钟. 一种基于退火策略的混沌神经网络优化算法[J]. 控制理论与应用. 2000,17(1):47-49.
- [32] Isai Tokuda,Kazuyuki Aihara and Tomomasa Nagashima,A daptive annealing forchaotic optimization[J].Physical ReviewE.,1998,58(4):5157-5160.
- [33] E Bonabeau, M Dorigo,G Theraulaz,Inspiration for optimization from social insect behavior[J].Nature,2000,406(6):39-42.
- [34] A Colorni, M Dorigo, Heuristics from nature for hard combinatorial optimization problems[J].International Trans Operational Research,1996,3(1):1-21.
- [35] 董梅, 谢楠琳. 基于蚂蚁行为的景点群最优路线选择[J]. 计算机工程, 2003,29(19):24-27.
- [36] 吴斌, 史忠植. 一种基于蚁群算法的 TSP 问题分段求解算法[J]. 计算机学报, 2001,24(12):54-55.
- [37] 洪炳熔, 金飞虎, 高庆吉. 基于蚁群算法的多层前馈神经网络[J]. 哈尔滨工业大学学报, 2003,35(7):12-13.
- [38] 曹浪财, 罗键, 李天成. 智能蚂蚁算法—蚁群算法的改进. 计算机应用研究 [J]. 2003,10:62-63.
- [39] 胡小兵. PASCP 在大规模 TSP 中的应用[J]. 计算机仿真, 2004,21(7):50-51.
- [40] Kalin Penev,Guy Littlefair,Free Search — a comparative analysis[J].Information Sciences,2005.173-193.
- [41] 刘岩, 董占球, 韩承德. 按自然法则计算的应用—TSP 的处理[J].Computer Research & Development, Feb.1997,34(2):112-114.
- [42] DANG Jian-wu Chen Yi,Study on a Polynomial Time Evolution Algorithm for the Traveling Salesman Problem 兰州铁道学院学报（自然科学版）[J]. 2001,20(1):43-45.
- [43] 谢开贵, 曾晓辉, 李春燕等. 免疫算法与其他随机优化算法的比较分析. 重庆大学学报[J]. 2003,26(11):37-38.
- [44] M Hasegawa,T Ikeguchi and K Aihara,Combination of Chaotic Neurodynamics with the

2-opt Algorithm to Solve Traveling Salesman Problems[J].Physical Review Letters,Sep,1997,79(12):2344-2347

- [45] Lovorka Pantic, Joaquin J Torres, Hilbert J Kappen and Stan C A M Gielen. Associative memory with dynamic synapses[J]. Neural Computation,2002
- [46] 马良.旅行推销员问题的算法综述.数学的实践与认识[J].2000,30(2):156-165
- [47] 穆艳玲.遗传算法在 TSP 问题中的应用[D].天津师范大学.2004
- [48] 刘海.旅行商问题优化算法设计及理论研究初步[D].华南理工大学.2002
- [49] 杨忠等.求解中国旅行商问题的新结果.数据采集与处理[J].1993,8(3):177-178.
- [50] 田贵超,黎明,韦雪洁.旅行商问题的几种求解方法.计算机仿真[J].2006,23(8):154-159.
- [51] 马向玲,田宝国.Hopfield 网络应用实例分析.计算机仿真[J].2003,20(8):66-68.
- [52] 张玉艳,陈萍.Hopfield 神经网络求解 TSP 中的参数分析.微电子学与计算机[J].2003,5:8-10.
- [53] 王宇.风景区旅游景点的综合评判及最优旅游路线的选择.宁德师专学报[J].2003,15(2):149-152.
- [54] 栗雪娟,崔尚身,张柯.最佳旅游路线选择的神经网络方法.交通与计算机[J].2006,24(132):105-107.

致谢

在论文完成之际，首先感谢我的导师白艳萍教授对我的悉心指导。白老师学识渊博、治学严谨、平易近人，具有高度的敬业精神，她以其宽广的视野、敏锐的洞察力、严谨的治学态度在学术上给了我悉心的指导，使我在学术上不断进步。导师勤勉的敬业精神和一丝不苟的工作态度深深感动我，这将成为我一生工作和学习的好榜样。

在两年多的课题研究过程中，白老师从论文选题、研究内容的确定及研究方法的实施都给予了艰辛与汗水，在此对白老师表示深深的感谢。在白老师身上，我不仅获得了无尽的知识，更学到了做人和治学的道理。

感谢胡红萍老师对我课题耐心地指导。

同时，也衷心感谢在我完成课题的过程中，给予我鼓励、帮助和支持的同窗好友杨晓燕、康爱花同学。

最后感谢我的父母对我的理解、关怀和支持，正是他们的爱给予了我无穷的信心和勇气去完成学业。

谨以此文，献给我的母校。

感谢评审专家对论文的不足之处提出宝贵的意见。

攻读硕士学位期间发表的论文及所取得的研究成果

- [1]、兰兆青, 白艳萍. 遗传算法在排课问题中的应用. [J]太原师范学院学报（已录用）
- [2]、兰兆青, 白艳萍, 李飞. 遗传算法解 TSP 问题的程序设计. [J]太原师范学院学报（已录用）