西南交通大学

硕士学位论文

基于蚁群算法的铁路车辆路径问题研究

姓名：周宏敏

申请学位级别：硕士

专业：计算机应用技术

指导教师：楼新远

20080301

摘要

车辆路径问题是近几十年来运筹学、应用数学、网络分析、计算机应用及交通运输等学科研究的一个热点问题。开放式车辆路径问题是另一种类型的车辆路径问题，有着广泛的应用前景。它与基本的车辆路径问题的主要区别是不要求车辆完成运输任务后必须返回原出发点。蚁群算法是一种新的仿生类算法，属于随机搜索算法，多年来的研究已显示出蚁群算法在求解复杂优化问题方面的优越性。蚁群算法也存在一些缺陷，如收敛速度慢，容易陷入局部最优解等。

本文对带约束条件的开放式车辆路径问题从理论上进行研究，实现了铁路运输矢量图的分层显示，约束条件主要考虑了铁路运输网络的系统最优平衡，也就是全局最优，而并非仅仅是单个用户的路径寻优，使研究内容更具有实际应用价值。通过实验对比了蚁群算法和其它经典的人工智能算法后，发现无论是获得的解的质量还是迭代次数蚁群算法都明显优于其它人工智能算法，显示了其在解决复杂优化问题方面的竞争力。针对蚁群算法的缺点，很多学者提出了改进蚁群算法，本文分别编程实现了蚁群算法和典型的改进蚁群算法，通过实验观察到改进蚁群算法的性能明显提高，然后借鉴了某些改进蚁群算法的思想并与带约束条件的铁路VRP相结合对蚁群算法进行了新的改进，使其在求解问题时结果更好、速度更快。最后针对本文研究的实际问题，对蚁群算法中各参数的选取规则进行了归纳。

实验表明用改进蚁群算法求解带约束条件的铁路 VRP 是有效的，这一成功尝试再次表明蚁群算法在优化领域具有强大竞争力。本文在研究中考虑了能获得系统最优平衡的约束条件，使研究内容更贴合铁路运输的实际，不仅具有理论研究意义也富有实际应用价值。

关键词：车辆路径问题；蚁群算法；系统最优平衡；约束条件

Abstract

Vehicle routing problem is a hot question during near decades and is studied such as operations research, applied mathematics, network analysis, computer application and traffic transport, etc. . Open vehicle routing problem is a VRP of another kind and has extensive application prospects. Main difference between OVRP and VRP is that the former is not to require the original starting point must be returned after the vehicle have accomplished the task transported. ACO (ant colony optimization algorithm) is a kind of new bionic algorithm, belong to random search algorithms. The research for many years has already demonstrated ACO superiority in solving complicated optimization question. ACO has some defects too, such as disappearing slowly and easy to fall into local optimization etc. .

This thesis has carried on the theoretical research to OVRP taking restraint terms and has realized that the railway transportation vector graph is showed different layer. Restraint terms mainly consider the system optimum equilibrium of railway transport, that is to say the overall situation is optimum, not only seek excellently users' route, so consideration makes the research contents have actual application value even more. After this thesis has compared ACO with other classical artificial intelligent algorithm through the experiment, it is found that ACO quality solved and cost of time are all obviously superior to other artificial intelligent algorithm, which has shown ACO competitiveness in solving the complicated optimization problem. To the shortcoming of ACO, a lot of scholars have proposed a lot of improved ACO. This thesis programms ACO and classical improved ACO, through the experiment it is proved that the performance of improved ACO obviously improve. Deriving from the thought of some improved ACO and combining railway VRP taking restraint terms, this thesis carries on new improvement to ACO in order to make it better and faster. Finally, to the practical studied problem, this thesis summes up the choosing rule of ACO

parameter.

The experiment of this thesis indicates that railway VRP taking restraint terms is effectual to solve with ACO, which indicates again that ACO has strong competitiveness in optimization field. The thesis has considered the restraint terms that can obtain the system optimum equilibrium. It makes that research contents laminate reality of railway transportation and the contents not only have theoretical research meanings but also are rich in actual appliacation value.

key words:Vehicle Routing Problem; Ant Colony Optimization Algorithm; System Optimum Equilibrium; Restraint Terms

西南交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复印手段保存和汇编本学位论文。

本学位论文属于

1. 保密□，在　　年解密后适用本授权书；
2. 不保密□，使用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：周宏敏　　指导老师签名：楼梦麟

日期：2008.5.24　　日期：2008.5.24

西南交通大学学位论文创新性声明

本人郑重声明：所呈交的学位论文，是在导师指导下独立进行研究工作所得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中作了明确的说明。本人完全意识到本声明的法律结果由本人承担。

本学位论文的主要创新点如下：

1. 对铁路运输网络矢量图进行了分层，搜索速度明显提高；
2. 在吸收一些典型改进蚁群算法的思想后，针对研究的问题本文采取了以下改进措施：①确定性选择和随机性选择相结合的选择策略；②信息素全局更新规则；③对信息素浓度进行限制。实验表明这些改进是有效的；
3. 考虑了流量约束，目的就是要获得路网交通流的最优化分布状态，这样就能避免某路径同一时间被大量使用，造成拥挤直至道路堵塞，使研究的内容更加具有实际应用价值。

第1章 绪论

1.1 选题背景和研究意义

铁路是国民经济的命脉，它对社会经济发展和人民生活起着极为重要的作用。以前铁路线路简单，列车运行路线几乎不用选择，可以一目了然确定列车运行路线。但是，目前我国铁路线路四通八达，交错纵横，已经形成了复杂的运输网络。而且，随着国民经济的发展，现有的铁路线路资源已明显不足，发展趋势必定是还要不断修建新线路，运输网络会日益庞杂[1]。如何在复杂的铁路运输网络中，在约束条件的限制下，快速地确定列车运行的最优路径就比以前复杂得多了。

车辆路径问题广泛存在于生产和生活中，如物流运输中车辆线路的优化，旅游中景点游览线路的优化，连锁商店送货车线路优化等。由于车辆路径问题是运输组织优化的核心问题，对其进行广泛和深入地研究，完善其中的算法，既可以促进计算机学科的理论发展，又可以将这种发展应用于物流配送管理和交通运输管理中。对改进管理方法和提高运输效率具有非常重要的实际价值。运筹学[2]将车辆路径问题（Vehicle Routing Problem，VRP）分为开放式和闭合式两种。当车辆完成任务后必须返回原出发点，属于闭合式车辆路径问题；当不要求车辆完成任务后必须返回原出发点，属于开放式车辆路径问题（Open Vehicle Routing Problem，OVRP）。

OVRP 其主要应用领域是铁路、公路和航空运输中的运输路线优化，因为列车、公共汽车和飞机等的运行路线基本上都是属于开放式的，其运行路线和时刻表（运行图）的编制都可以抽象、归结为一些不同类型的 OVRP 来研究。但这些问题目前还没有很好地解决，如铁路列车运行图的优化编制算法问题，仍然是实现计算机编图的一个急需解决的问题[3]。显然从理论上对 OVRP 进行研究，将有助于这些实际问题的解决。本文研究的铁路车辆路径问题就属于开放式车辆路径问题范畴。

1.2 国内外研究现状

VRP最早由Dantzig和Ramser[4]于1959年提出，提出后很快引起运筹学、应用数学、组合数学、图论和网络分析、物流科技、计算机应用等学科专家与运输计划制定者和管理者的极大重视，成为相关领域的前沿和研究热点问题。各学科的专家对 VRP 问题进行了大量的理论研究和试验分析，取得了很大进展。

国外对 VRP 做了大量而深入的研究，Willard (1989)[5]首先将禁忌搜寻法应用于车辆路径问题上，设计重复的虚拟物流中心，将车辆路径问题转换成旅行商问题(TSP)，利用 2-OPT 或 3-OPT 算法求解车辆路线。Gendreau, Hertz and Laporte (1994)[6]使用插入法求解旅行商问题，再用贪婪法(Greedy Method)进行路线切割，从而产生初始解。Barbarosoglu & Ozgur (1999)[7]利用禁忌搜寻法为土耳其某物流公司构建一决定货车配送点顺序的方法 DETABA，以二种乱数选取节点的方法产生初始解，找到其中最佳的解作为初始解，再以插入法（Insertion Procedure）作为搜寻邻近解的移步算法，最后以 2-OPT 改善算法找到最优解的值。Su & Chen (1999)[8]成功地将自组织映射网络应用在车辆配送区域及路线规划问题的求解上，其算法的主要概念是利用类神经网络快速运算、自我组织与平行处理的特性，配合 M 个一维环状网络拓扑来表现车辆路线配送问题。OVRP 是 Schrage[9]在 1981 年提出的，学术界随后才对 OVRP 进行理论研究。Sariklis 和 Powell[10]于 2000 年 5 月发表了第一篇关于该问题的理论研究论文，作者对简单情形（只有车辆装载能力限制）的 OVRP 进行了研究，提出了用启发式方法构造的求解算法。Brandao[11]现在仍对该问题进行研究，他探索运用“禁忌搜索”方法来解决问题。

在我国随着国民经济的发展，车辆路径问题显得日益重要，我国理论界早已开始关注车辆路径问题的研究，并已取得初步成果。启发式算法和一些混合算法被学者们广泛的利用。国内研究的相关领域除了 VRP 以外，还包

括中国邮递员问题(Chinese Postman Problem，简称 CPP)、有向中国邮递员问题 (Directed Chinese Postman Problem，简称 DCPP)等。西南交通大学的李军教授和郭耀煌教授[12]-[14]对车辆优化调度的基础理论及各类问题进行了系统的研究。李大为等[15]以 TSP 的最近距离启发式为基础，通过设置评价函数来处理时间窗约束，求解了简单的 VRP。另外在利用人工智能算法(如：遗传算法、神经网络方法、模拟退火等)对简单 TSP 的求解取得了一定成果。蔡延光等[16]应用模拟退火法针对满载问题进行了求解。张涛等[10]通过遗传算法保证搜索的全局性，用 3-OPT 算法来加强局部搜索能力，得到针对 VRP 的混合算法。这类算法目前已可求解较大规模的问题。肖朋等[17]通过构造 VRP 的染色体表达，采用基因换位算子进行染色体重组，实现了新颖的单亲遗传算法。此外，霍雪丽等[18]基于近年来出现的新型智能优化思想：人工蚂蚁系统，给出了一种可快速求解 VRP 问题的蚂蚁搜索算法。通过定义基本的人工蚂蚁转移概率，并结合局部搜索策略，用迭代次数控制算法的运行时间，从而使该算法具有实用意义和可操作性。魏俊华等[19]提出了一种分段编码方法，编码中的各段代表相应车辆路径的需求城市集合，以非完全连通网络为研究对象，基于分段遗传编码，构造了车辆路径问题的遗传算法。尹晓峰等[20]针对蚁群算法过早收敛的问题，引入节省量和车辆载重利用率两种启发式信息对蚁群算法进行改进，并加入 2-OPT 方法对问题求解进行局部优化，计算机仿真结果表明，这种混合蚁群算法对求解车辆路径问题有较好的改进效果。虽然我国学者早已开始关注 VRP，但由于我国用人工智能算法解决 VRP 是 20 世纪 90 年代以后才逐渐兴起的，比国外相对落后，这方面的研究还有待于进一步提高。

1.3 本文的主要内容

论文首先阐述了课题的研究背景和研究意义，给出了车辆路径问题的研究现状；然后对车辆路径问题进行分析，介绍了车辆路径问题的基本模型及

目前解决该问题的常用方法，实验对比了蚁群算法与常用人工智能算法的性能后，选择蚁群算法解决问题。介绍了蚁群算法的原理和模型，编程实验了基本蚁群算法和几种有代表性的改进蚁群算法的性能。借鉴以往某些改进算法的思想并结合研究的实际问题对蚁群算法进行了新的改进。在解决研究的具体问题时，先抽象出待解决问题的模型，对比了邻接表和邻接矩阵的优缺点后选择邻接表作为无向图的存储结构；介绍并实现了铁路运输矢量图的分层显示；阐述了蚁群算法的设计过程，包括线路构造、信息素的更新规则等，并在算法实现过程中考虑了能够获得系统最优平衡的约束条件；对算法中的参数选取进行归纳。最后总结了本文的主要工作，以及有待改进的地方，对进一步的研究方向进行展望。

第2章 VRP的提出及各类VRP的算法

2.1 VRP的提出

VRP的一般描述是：对一系列给定的客户（送货点或取货点），确定适当的配送车辆行驶路线，使其从配送中心出发，有序地通过它们，最后返回配送中心，并在满足一定的约束条件下（如车辆载重量、客户需求量、时间窗限制等），使总运输成本达到最小（如使用车辆数最少、车辆行驶路程或时间最短等）。其中，一般把最小的车辆数作为第一目标，而最低的行驶成本作为第二目标。参见图2-1：

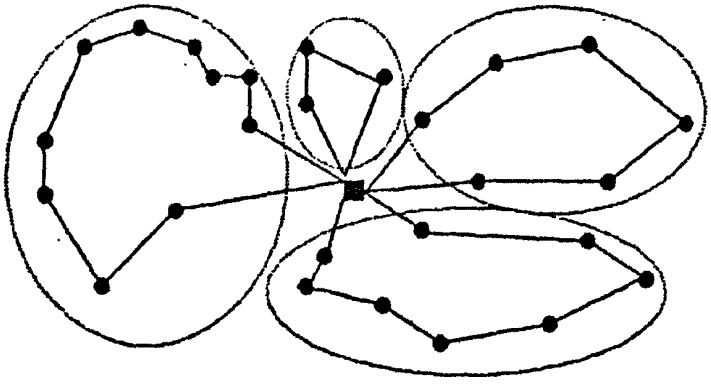

图2-1 车辆路径问题示意图

2.2 VRP的分类

现在的车辆路径问题在基本的VRP的基础上，在学术研究和实际应用上产生了许多不同的延伸和变化型态，根据研究的重点不同，VRP可分为以下几类[21]：

①按已知信息的特征可分为确定性VRP和不确定性VRP，其中不确定性VRP可进一步分为随机车辆路径问题(SVRP)和模糊车辆路径问题(FVRP)；

②按约束条件可分为带容量限制的车辆路径问题（CVRP）、带距离约束的车辆路径问题（DVRP）以及带有时间窗的车辆路径问题（VRPTW）等。

③按车辆载货状况分，有满载车辆路径问题和非满载车辆路径问题；

④按配送中心的多少来分，有单车场车辆路径问题（SVRP），即一般车辆路径问题（VRP）和多车场车辆路径问题（MVRP），其中 MVRP 又可以根据是否每辆车都有固定的终点车场分为终点车场固定的车辆路径问题和终点车场不固定的车辆路径问题；

⑤按车辆类型分，有单车型车辆路径问题和多车型车辆路径问题；

⑥按优化目标数来分，有单目标车辆路径问题和多目标车辆路径问题；

⑦按任务特征分，有纯装车辆路径问题、纯卸车辆路径问题及装卸混合车辆路径问题。

2.3 求解 VRP 的算法

目前已提出许多用于求解 VRP 的算法，针对各种不同类型的 VRP，本文对人们提出的一些算法进行了归纳，大致将它们分为两类：精确算法和人工智能算法。

2.3.1 精确算法

精确算法可以分为三个大类：直接树搜索算法、动态规划方法和整数线性规划。

（1）K 度中心树算法

该方法是 Christofides 等人[22]提出的。对固定 m 的 m-TSP 进行 k 度中心树松弛处理 。所以该方法需要知道所需车辆数的下界。其模型是用边的角度建立的，出发点用一条边表示，其它点用两条边表示。通过拉格郎日松弛法，将其中的一个约束条件消去，并进一步将原来的最小化问题转化为 3 个易求解的子最小化问题，然后进行求解 [23]。

（2）分支定界算法

该方法是 Laporte 等人[24]提出的。它利用了 VRP 和其放松形式 m-TSP 间的关系。根据 Lenstra 等人[25]所给出的 m 的上界 mv，m-TSP 可转化为 1-TSP。

关键步骤是，引入 mv-1 个伪出发点，n'=n+mv-1,v'=(1,……, n'), A'=A∪{(i, j)：i,j∈v',i≠j 或 j∈v'\v}。接下来用分支定界法求解。

（3）动态规划法

动态规划法是由 Eilon 等人[26]首先提出的。它针对的也是固定车辆数的 VRP，通过递归方法求解。为减少问题的计算规模，引入可行性规则或松弛过程减少状态的数量。其后，Christofides[27]提出了状态空间松弛，极大地减少状态数量。该方法要求：转换函数易于求解，映射出来的范围小，可求得很好的下界。目前可求解有 50 个客户的 VRP[28]。

（4）集分割和列生成

这种方法直接考虑可行解集合，在此基础上进行优化，因此建立的 VRP 模型最简单。但缺陷在于如果问题所受的约束不严格，则所需计算的状态空间非常大。另外，要确定每个可行解的最小成本很困难。对于其中规模相对较小的、约束严格的问题，可通过线性松弛、引入割平面进行求解。于是 Rao[29]等人引入了列生成方法进行求解，在此方法中，原问题被转化为简化问题，考虑的范围是所有可能的可行解的子集。在此基础上重复求解。通过引入优化对偶变量向量，对该简化问题松弛，通过计算列的最小边际成本，确定最优解。其算法本质上是最短路径算法，同时结合了分支定界算法。Desrocher[30]用它求解有 100 个客户的带时间窗的 VRP。

（5）三下标车辆流方程

Fisher 等人针对带能力约束、时间窗口以及无停留时间的 VRP 问题，提出了三下标车辆流方程。在该方程中，其中两个下标表示弧或边，另外一个下标表示特定车辆的序号。基于 Benders[31]的分解技术，他们提出了一种启发式算法，保证在有限的步骤内找到优化解。Martello 和 Desrochers 分别提出了相应的改进算法[32]。

（6）二下标车辆流方程

对于对称 CVRP 和 DVRP，可通过去掉表示车辆序号的下标，引入所需车辆数的下界，得到一个更为紧凑的方程。它所对应的算法结合了爬山法的思想，其算法核心仍然是线性规划，若得到的解是分数解，则用分枝定界方法求其整数解。该方法是由 Laporte 等人提出的，已用它来求解了规模为 60 的 VRP[33]。

总的来说，精确算法基于严格的数学手段，在可以求解的情况下，其解通常要优于人工智能算法，这就是它们为什么在商业软件中仍被广泛使用的原因。但由于引入严格的数学方法，因而无法避开指数爆炸问题，使该类算法只能有效求解中小规模的确定性 VRP。具体到每个算法，它们都有其适用的范围和特点。给定下界和相关的分枝定界算法是从所要访问的点的角度出发建立的，因此不仅适用于对称 VRP，还适用于非对称的 VRP；三下标车辆流方程在模型中有效引入了代表时间窗口的变量，从而可适用于通用任务分配问题（GAP）和带时间窗的 TSP（TSPTW）[34]；二下标车辆流方程是由 TSP 的 SYS 方程扩展而来。由于去掉了代表车辆序号的下标，形式上更为紧凑，具有更少的约束条件，所以仅适用于对称的 CVRP 和 DVRP，而且特别适用于这两种 VRP 中约束条件比较宽松的问题[35]。而 k 度中心树、动态规划和集分割等方法在对应的模型中，以不同的方式给出能力约束、子回路约束等约束条件，适用于约束条件严格的问题。许多专家认为，在精确性算法领域，几乎已无进行重大改进的余地，历史已经到了翻开新的一页的时候了，这就是下面的人工智能算法

2.3.2 人工智能算法

在求解中小规模 VRP 时，人工智能算法与精确算法相比，在精度上不占优势。但在求解大规模 VRP 时，人工智能算法总可以在有限时间里，找到满意的次优解/可行解，这是精确算法难以做到的。因此，在实际应用中，人工智能算法要更广泛。常见的有特色的人工智能算法有以下几种：

（1）Clarke-Wright 节约算法

Clarke-Wright 节约算法（Clarke and Wright Savings Algorithm）是求解 VRP 的最著名的算法[36]，用来解决车辆数不固定的 VRP。该算法是一种基于节约准则的逐步构解算法。其基本思想是：当将两条线路（0,……,i,0）和（0,j,……,0）合并成一条线路时，则所有带来的路程长度节约值为 $s_{ij}=c_{i0}+c_{0j}-c_{ij}$；$s_{ij}$ 越大，说明把 i 和 j 连接在一起（即合并相应的两条线路）会使总费用减少越多，若根据 s_{ij} 的大小来构造线路，就有可能得到总费用较小的解。

（2）扫描算法

扫描算法（Sweep Algorithm）是由 Gillett 和 Miller[37]于 1974 年提出的。算法分两个阶段来构造一组解，首先以车场为旋转中心转动一条射线将客户点按区域进行划分成组；然后对每一区域内的客户点求解一个 TSP，以确定一条车辆行驶路线。参见图 2-2：

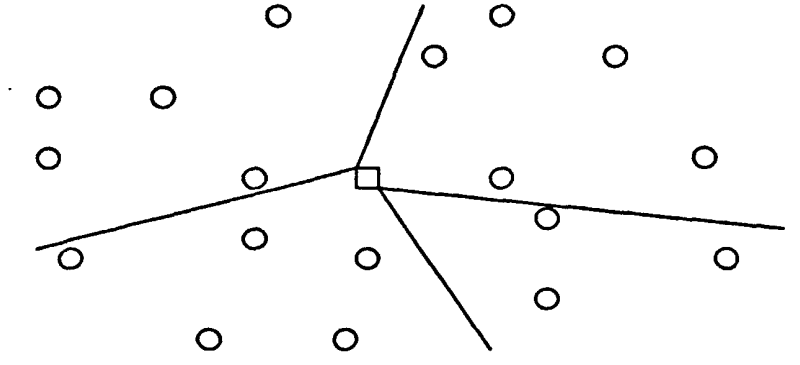

图 2-2 由扫描法划分的区域

算法的一个简单实现方式是，以车场为轴心建立极坐标，随机选取一个种子客户，令其在极坐标上的夹角度数为 0，以车场至种子客户的连线为射线开始扫描（顺时针或逆时针），并将各客户点按其夹角度数的升序排列。然后按以下步骤进行：

步骤1（线路初始化）：选择一辆尚未被使用的车辆k。

步骤2（线路构造）：从尚未指派给任何车辆的、夹角度数最小的客户点开始，只要不超过车辆的装载能力，就按顺序将客户指派给车辆k。如果还有未被指派的客户，返回步骤1。

步骤3（线路优化）：通过求解相应的TSP（精确地或近似地），分别对每条线路进行优化。

（3）Chrisofides-Mingozzi-Toth两阶段算法

它主要面向CVRP和DVRP。该算法的求解过程分为两个阶段：第一阶段按最小路径的原则形成初始解，然后用k-OPT算法对所得的各子路径分别进行优化；第二阶段是在各子路径间进行点的交换，以减少总行程，然后再用k-OPT算法对点交换后的子路进行优化。该算法的优点是，在计算过程中，考虑了所需要访问的点数量增加的情况[38]。

（4）禁忌搜索（Tabu Search）

Gendreau等人[39]最先将该方法应用于VRP。先构造一系列的解，然后对所得解不断地进行改进。该算法所得到的解不一定是可行解，它们对可行性的偏离程度是通过目标函数里的罚函数来体现的。该算法求解过程中的邻域，是通过GENI过程得到的。它是针对VRP的比较好的人工智能算法，可以成功地应用于许多经典的VRP。其后E.Tailard等人通过按角度和路径重心对原问题的空间进行分割，再用禁忌搜索结合模拟退火对子问题求解，实现了对问题求解的并行化。

（5）模拟退火算法

模拟退火算法是Kirkpatrick等人于1982年将固体退火思想引入组合优化领域，并提出的一种适合求解大规模组合优化问题，特别是NP-完全问题的算法。它源于对固体退火过程的模拟，采用Metropoplis接受准则，并用一组称之为冷却表的参数控制算法过程，使算法在多项式里给出一个近似

最优解。该算法是一种随机的启发式搜索方法，是对局部搜索方法的改进，即为避免搜索过程陷入局部最优，允许在一定条件下接受恶化解。

（6）遗传算法

遗传算法是一种仿生物进化过程的全局随机搜索方法，由美国 Michigan 大学的 John Holland 教授于 1975 年提出。表 2-1 中列出了在算法设计中要用到的一些生物遗传学的基本概念及其与优化问题的对应关系。

表 2-1 生物遗传概念与优化问题的对应关系

生物遗传概念	在优化问题中表示的概念
个体（individual）	一个可行解
染色体（chromosome）	解的编码（字符串、向量等），即解的表示形式
基因（gene）	解中每一分量的特征（如各分量的值）
个体适应度（fitness）	解的目标函数值或所对应的适应函数值
种群（population）	多个可行解组成的一个集合，可行解的个数称为种群的规模
生物进化过程	求解的迭代过程
适者生存	目标函数值越好的解，被选择作为下一迭代过程的当前解的可能性越大

它的基本思想可用上述术语描述为：从优化问题的一个种群（一组可行解）开始，按照适者生存和优胜劣汰的原理，逐代演化产生出越来越好的一个种群（一组可行解）。在每一代，根据个体（可行解）的适应度（目标函数值）的优劣挑选一部分优良个体复制到下一代，并对其进行交叉和变异操作，产生出代表新的解集合的种群。这个过程将导致种群像自然进化一样的子代种群比父代更加适应于环境（即新可行解比旧可行解更接近问题的最优解），整个进化过程中的最优个体就作为问题的最终解。J.Lawrence 最先将该方法用于 VRP 的研究，并可有效求解带时间窗的 VRP。鉴于传统的遗传

算法是个大范围、粗粒度的寻优算法，因此近年来许多学者对遗传算法进行了改进，该算法目前已相对比较成熟。

下面把求解 VRP 的各种算法及其适用范围列在表 2-2 中：

表 2-2 求解 VRP 的各种算法及其适用范围

算法			适用范围
精确性算法	精确性算法基于严格的数学手段，在可以求解的情况下，其解通常要优于人工智能算法。但由于严格的数学方法，因而无法避开指数爆炸问题，从而使该类算法只能有效求解中小规模的 VRP	给定下界和相关的分支定界算法	给定下界和相关的分支定界算法是从所要访问的点的角度出发建立的，因此不仅适用于对称 VRP，还适用于非对称的 VRP。
		三下标车辆流方程	三下标车辆流方程在模型中有效引入了代表时间窗的变量，从而可适用于通用任务分配问题（GAP）和带时间窗的 TSP（TSPTW）。
		二下标车辆流方程	二下标车辆流方程是由 TSP 的 SYM 方程扩展而来。由于去掉了代表车辆序号的下标，形式上更为紧凑，具有更少的约束条件，所以仅适用于对称的 CVRP 和 DVRP，而且特别适用于这两种 VRP 中约束条件比较宽松的问题。
		k度中心树和相关算法	这三种算法在对应的模型中，以不同的方式给出能力约束、
		动态规划法	

<table>
<tr><th colspan="3">算法</th><th>适用范围</th></tr>
<tr><td></td><td></td><td>集分割和列生成</td><td>子回路约束等约束条件，适用于约束条件严格的问题。</td></tr>
<tr><td rowspan="6">人工智能算法</td><td rowspan="6">在求解中小规模VRP时，人工智能算法与精确算法相比，在精度上不占优势。但在求解大规模VRP时，人工智能算法总可以在有限时间里，找到满意的次优解/可行解，这是精确算法难以做到的。因此，在实际应用中，人工智能算法要更广泛。</td><td>Clark-Wright 节约算法</td><td>此算法归并子路径的基本前提是要满足车辆的能力约束，因此该方法也适用于CVRP</td></tr>
<tr><td>Sweep 扫描算法</td><td>Sweep 通过角度的大小对客户/节点加以聚类，本质上是将距离近的客户归并到一个子路径中，同时聚类要满足的可行性条件是车辆的能力约束，因此该方法也适用于CVRP。</td></tr>
<tr><td>Chrisofides-Mingozzi-Toth 两阶段算法</td><td rowspan="4">这几种算法要满足的可行性条件可以是能力约束，也可以是时间约束，因此这些方法适用于CVRP和DVRP。</td></tr>
<tr><td>禁忌搜索</td></tr>
<tr><td>模拟退火算法</td></tr>
<tr><td>遗传算法</td></tr>
</table>

人工智能算法中还有一种非常有效的算法，就是蚁群算法，本文将选择蚁群算法求解问题，为何选择该算法以及该算法与其他人工智能算法的比较等内容将在第三章进行介绍。

2.4 开放式车辆路径问题

开放式车辆路径问题（Open Vehicle Routing Problem，OVRP）是另一种

类型的车辆路径问题。它与基本车辆路径问题（VRP）的主要区别是不要求车辆完成取送任务后必须返回原出发点，因此，车辆的行驶路线是开放式的。参见图 2-3：

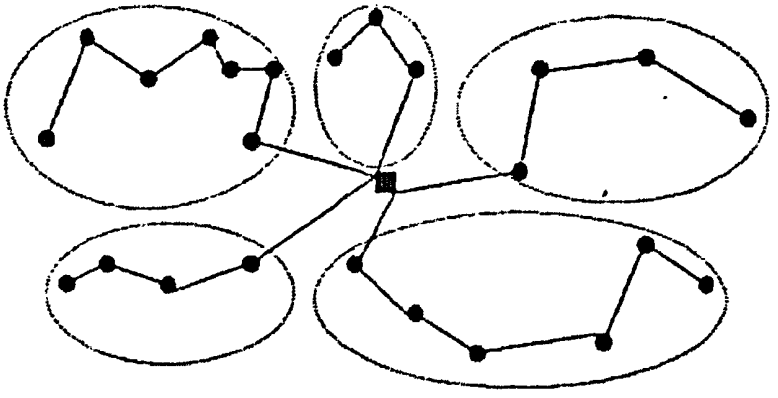

图 2-3 开放式车辆路径问题示意图

开放式车辆路径问题的一般描述是：有一个车场、一组有确定需求量的客户（送货点或取货点）、以及有确定装载能力和运营成本的一组车辆，并已知车场与各客户间、客户与客户间的车辆行驶费用，问题是确定一组车辆行驶路线，使得如下两个目标达到最小：

①服务所有客户所需要的车辆数；

②总的车辆行驶费用；

并满足三个主要的约束条件：

①每条路线开始于车场，终止于某一个客户点（或反过来）；

②每个客户点由一辆车访问（服务）一次且仅一次，且其需求量全部得到满足；

③每一条路线上各客户点的需求量之和不超过其车辆装载能力。

通常都认为，多用一辆车所带来的固定费用的增加，总是超过其因总行驶距离缩短所带来的节省，因此，一般把最少的车辆数作为第一目标，而最

小的行驶费用作为第二目标。

在实际应用中碰到的OVRP是多种多样的，可比照VRP的分类方法对其进行分类。从基本的VRP中去掉车辆必须返回原出发点的限制并没有使问题变得简单和容易。在OVRP中，每一条路线都是哈密顿路径(Hamiltonian Path)，而在VRP中是哈密顿圈（Hamiltonian Cycle）。虽然目前已有许多求解VRP的好算法，但都不能直接用于求解OVRP，且一般也不主张把OVRP转化为VRP作为其求解过程中的一部分，这是因为一个好的OVRP的解往往与相应的VRP的解相距甚远。正如Syslo[41]等所述，“从网络中的一个最小的哈密顿圈中去掉其最大的边，并不能得到最小的哈密顿路径”。本文用人工智能算法中的蚁群算法对OVRP进行求解。

第3章 基本蚁群算法及其实现

3.1 蚁群算法的提出

优化理论和方法在近几十年里得到了长足发展，伴随着计算机技术的快速更新，其应用范围日益延伸，长期以来人们一直对此进行不懈的研究，提出很多卓有成效的求解方法，各种优化方法层出不穷。自20世纪50年代中期创立了仿生学以来，人们从生物进化机理中受到启发，提出许多解决复杂优化问题的新方法，如遗传算法、进化规划等。其中蚁群算法是由意大利学者M.Dorigo[42]-[44]等人提出的新型模拟进化算法。该算法充分利用了蚁群搜索食物的过程与旅行商问题（TSP）之间的相似性，通过人工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问题。为了区别于真实的蚂蚁群体，我们称这种算法为“人工蚁群系统算法”，简称“蚁群算法”。

3.2 蚁群算法的基本原理

人们经过大量的研究发现，自然界的蚂蚁在运动路径上释放出的特殊激素——信息素，在蚂蚁个体之间的信息交流和相互协作起着重要作用，其浓度与蚂蚁走过的路径长短有关，路径越短其浓度越高。当别的蚂蚁走到前面蚂蚁走过的路径时可以感知环境中这种信息素的存在及强度，并倾向于朝着信息素浓度高的方向移动，即选择信息素浓度高的路径的概率会相对较大。于是，由大量蚂蚁完成的高度自组织行为便形成一个信息正反馈机制：某一路径上走过的蚂蚁越多，该路径上的信息素浓度越大，则后来者选择该路径的概率越大；蚂蚁少的路径信息素浓度越低，选择概率就小，并且随着时间的推移逐渐挥发直至消失，这样的路径最后被淘汰，最终蚂蚁集体获得最优路径[44]。

蚁群算法是人们受到自然界中蚂蚁集体行为的启发而提出的一种全新模拟进化算法，属于随机搜索算法。蚁群算法采用人工蚂蚁模拟自然界蚂蚁

的寻找方式，每个人工蚂蚁的行为要符合下列规律：

①根据路径上的信息素浓度，以相应的概率来选取下一步路径；

②不再选取自己本次循环已经走过的路径为下一步路径，用一个数据结构（tabu list）来控制这一点；

③当完成了一次循环后，根据整个路径长度来释放相应浓度的信息素，并更新走过的路径上的信息素浓度。

3.3 蚁群算法的应用

蚁群算法最初用于解决旅行商问题。自从在著名的 TSP[47][48]和工件排序问题上取得成效以来，已经陆续渗透到其他领域中，其中最成功的是在组合优化问题中的应用，如图着色问题、大规模集成电路设计、通讯网络中的路由问题以及负载平衡问题、车辆调度问题等等。可以将这些应用分为两类：一类应用于静态组合优化问题，其典型代表有 TSP、二次分配问题（quadratic assisgnment problem,QAP）、车辆调度问题、车辆路由问题等[45][46]；另一类应用于动态组合优化问题，蚁群算法在动态组合优化问题研究中的应用主要集中在通讯网络方面。这主要是因为网络优化问题有一些特征，如内部信息和分布计算，随机动态，以及异步的网络状态更新等，与蚁群优化算法的特征匹配得很好[47]。蚁群算法的一些经典应用如表 3-1 所示。

表 3-1 蚁群算法应用表

问题名称	作者	算法名称	年代
旅行商问题（TSP）	Dorigo, Maniezzo and Colotni	AS	1991
	Gambardella and Dorigo	Ant-Q	1995
	Dorigo and Gambardella	ACS and ACS3-opt	1996
	Bullnheimer,HartlandStrauss	ASrank	1997
	Stutzle and Hoos	MMAS	2007
二次分配问题（QAP）	Maniezzo,Colomi and Dorigo	AS-QAP	1994

问题名称	作者	算法名称	年代
	Gambardella, Tailland and Dorigo	HAS-QAP	1997
	Stutzle and Hoos	MMAS-QAP	1997
	Maniezzo	ANTS-QAP	2000
	Maniezzo and Colorni	AS-QAP	2001
调度问题（JSP）	Colorni, Dorigo and Maniezzo	AS-JSP	1994
	Stulzle	AS-FSP	1999
	Bauer et al.	ACS-SMTTP	2001
	Den Besten, Stutzle and Dorigo	ACS-SMTWTP	2005
车辆路径问题（VRP）	Bullnheimer, Hartle and Strauss	AS-VRP	1997
	Gambardella, Taillard and Agtazzi	HAS-VRP	2003
带时间窗的VRP	Leguizamon	HAS	2006
有向连接网络路由	Schoonderwoerd et al.	ABC	1996
	White, Pagurek and Oppacher	ASGA	1998
	Di Caro and Dorigo	AntNet-FS	2002
	Bonabeau et al.	ABC-smart ants	2005
无向连接网络路由	Di Caro and Dorigo	AntNet and AntNet-TA	1997
	Subraunartian, Druschel and Chen	Regular ants	2002
	Heusse et al.	CAF	2005
	Van der Put and Rothkrantz	ABC-backward	2006
有序排列问题(SOP)	Gambardella and Dorigo	HAS-SOP	1997
图着色问题（GCP）	Costa and Hertz	ANTCOL	2004
最短公共父序列问题（SCS）	Michel and Middendarf	AS-SCS	1998
频率分配问题（FAP）	Maniezzo and Carbonaro	ANTS-FAP	2003

问题名称	作者	算法名称	年代
一般分配问题（GAP）	Ramalhinho Lourenco and Serra	MMAS-GAP	1998
多重背包问题（MKP）	Leguizamon and Michalewica	AS-MKP	1999
网络路由	Navarro Varela and Sinclair	ACO-VWP	2006
多余分配问题（RAP）	Liang and Smith	ACO-RAP	2005

3.4 蚁群算法的特点

3.4.1 蚁群算法的优点

（1）蚁群算法是一种结合了分布式计算、正反馈机制和贪婪式搜索的算法，具有很强的搜索较优解的能力。正反馈能够快速地发现较优解，分布式计算避免了早熟收敛，而贪婪式搜索有助于在搜索过程的早期找出可接受的解决方案，缩短了搜索时间。

（2）蚁群算法具有很强的并行性。

（3）可以不通过个体之间直接通信而是通过信息素进行合作，这样的系统具有更好的可扩充性。由于随着系统中个体增加而增加的系统通信开销在这里将非常小。

3.4.2 蚁群算法的缺点

（1）蚁群算法一般需要较长的搜索时间。蚁群中个体的运动是随机的，虽然通过信息交换能够向着最优路径进化。但是当问题规模较大时，蚁群算法这种大空间的多点全局搜索，不可避免地增加了搜索所需的时间。

（2）该算法容易出现停滞现象，即搜索进行到一定程度后，所有个体所发现的解完全一致，不能对解空间进行进一步搜索，不利于发现更好的解。

（3）蚁群算法是一种概率算法，从数学上对它的正确性与可靠性的证明还是比较困难的，所做的工作也比较少[48]。而且蚁群优化算法在解决问题的时候，算法系统的高层次的行为是需要通过低层次的蚂蚁之间的简单行为交互涌现产生的。单个蚂蚁控制的简单并不意味着整个系统设计的简单，设计

者必须能够将高层次的复杂行为（也就是系统所要执行的功能，例如旅行商问题、车辆路径问题、图着色问题）映射到低层次的蚂蚁的简单行为（例如信息素的释放）上面，而这二者之间是存在较大差别的。并且在系统设计时也要保证多个个体简单行为的交互能够涌现出我们所希望看到的高层次的复杂行为。这可以说是蚁群算法乃至群集智能中一个极为困难的问题。

3.5 蚁群算法模型

3.5.1 TSP 问题

为了说明蚁群算法模型，首先引入旅行商问题。

旅行商问题就是指给定 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。

3.5.2 蚁群算法模型

为模拟实际蚂蚁的行为，首先引入如下记号：

m—蚁群中蚂蚁数量；

$b_i(t)$—t 时刻位于城市 i 的蚂蚁的个数，$m=\sum_{i=1}^{n} b_i(t)$；

d_{ij}—两城市 i 和 j 之间的距离；

η_{ij}—边(i,j)的能见度，反映由城市 i 转移到城市 j 的启发程度，这个量在蚂蚁系统的运行中不改变；

Δt_{ij}—蚂蚁 k 在边(i,j)上留下的单位长度轨迹信息素量；

p_{ij}^{k}—蚂蚁 k 的转移概率，j 是下一步要访问的城市。

每只蚂蚁都是具有如下特征的简单主体：

①在从城市 i 到城市 j 的运动过程中或是在完成一次循环后，蚂蚁在边(i,j)

上释放一种物质，称为信息素轨迹；

②蚂蚁概率地选择下一个将要访问的城市，这个概率是两城市间距离和连接两城市的路径上存有轨迹量的函数；

③为了满足问题的约束条件，在完成一次循环以前，不允许蚂蚁选择已经访问过的城市。以 TSP 为例，基本蚁群算法求解框架如图 3-1 所示：

```
Procedure 蚁群算法
初始化
for t=1 to 循环次数 do
Begin
  for k=1 to m do
  Begin
    Repeat
      根据概率 p_ij^k(t) 选择下次访问的节点 j
    Until
      蚂蚁 k 完成一条路径
  计算蚂蚁 k 生成的路径长度 L_k
  保存目前为止找到的最好路径
  End
  修改所有路径的轨迹强度 c_ij
End
```

图 3-1 基本蚁群算法框架

基本蚁群算法的流程如图 3-2 所示：

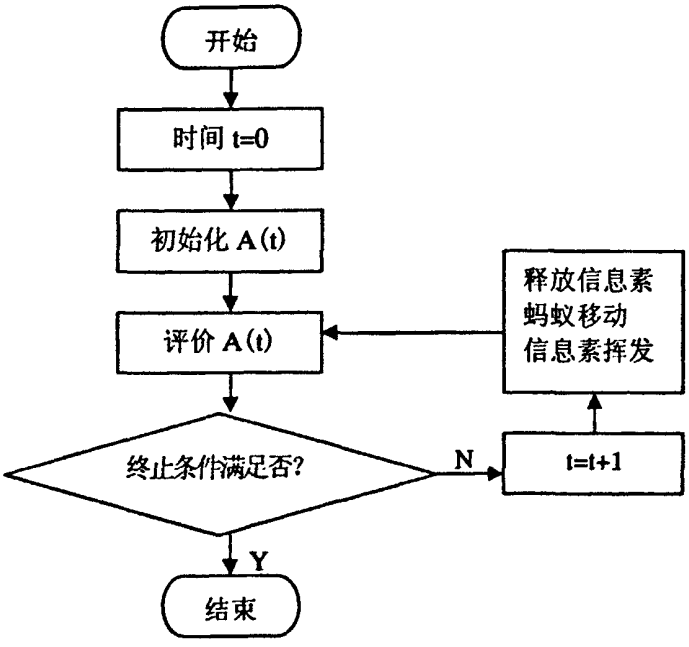

图 3-2 基本蚁群算法流程图

初始时刻，各条路径上的信息素量相等，设 $t_{ij}(0)=C$(C 为常数)。蚂蚁 k(k=1,2,…，m)在运动过程中根据各条路径上的信息素量决定转移方向。蚂蚁系统所使用的状态转移规则被称为随机比例规则，它给出了位于城市 i 的蚂蚁 k 转移到城市 j 的概率。在 t 时刻，蚂蚁 k 在城市 i 选择城市 j 的转移概率 $p_{ij}^{k}(t)$ 见式（3-1）

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{\mu\in allowed_k}\tau_{i\mu}^{\alpha}(t)\eta_{i\mu}^{\beta}(t)}, & j\in allowed_k\\ 0, & otherwise\end{cases} \tag{3-1}$$

式中 $allowed_k$—表示蚂蚁 k 下一步允许选择的城市；

α 和 β —为两个参数，分别反映了蚂蚁在运动过程中所积累的信息和启发信息在蚂蚁选择路径中的相对重要性。

由公式（3-1）可知，转移概率 $p_{ij}^{k}(t)$ 与 $\tau_{ij}^{\alpha}(t)\cdot\eta_{ij}^{\beta}(t)$ 成正比。与真实蚁群不同，人工蚁群系统具有记忆功能。为了满足蚂蚁必须经过所有 n 个不同城

市这个约束条件，为每只蚂蚁都设计了一个数据结构，称为禁忌表(tabu list)。禁忌表中记录了在 t 时刻蚂蚁已经走过的城市，不允许该蚂蚁在本次循环中再次经过这些城市。当本次循环结束后，禁忌表被用来计录该蚂蚁当前所建立的解决方案(即蚂蚁所经过的路径长度)。之后，禁忌表被清空，该蚂蚁又可以自由地进行选择。

经过 n 个时刻，蚂蚁完成一次循环，各路径上信息素量根据公式（3-2）调整

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta \tau_{ij}(t,t+n) \qquad (3\text{-}2)$$

公式（3-2）中的$\Delta \tau_{ij}(t,t+1)$由公式（3-3）求解

$$\Delta \tau_{ij}(t,t+n) = \sum_{k=1}^{m} \Delta \tau_{ij}^{k}(t,t+n) \qquad (3\text{-}3)$$

式中 $\Delta \tau_{ij}^{k}(t,t+n)$—第 k 只蚂蚁在时刻(t,t+n)留在路径(i,j)上的信息素量，其值视蚂蚁表现的优劣程度而定，路径越短，信息素释放的就越多；

$\Delta \tau_{ij}(t,t+n)$—表示本次循环中路径(i,j)的信息素量的增量；

$(1-\rho)$—信息素轨迹的衰减系数，通常设置系数 $\rho < 1$ 来避免路径上轨迹量的无限增加。

根据 τ_{ij} 和 $\Delta\tau_{ij}^{k}$ 表达形式的不同，M.Dorigo 曾给出三种不同模型，分别称为蚁周系统(Ant-Cycle Ssystem)、蚁量系统(Ant-Quantity System)、蚁密系统(Ant-Density System)。在蚁周系统模型中，$\Delta\tau_{ij}^{k}(t,t+n)$表示更新蚂蚁 k 所走过的路径，(t,t+n)表示蚂蚁经过 n 步完成一次循环，找到一条路径，具体更新值由公式（3-4）给出

$$\Delta\tau_{ij}^{k}(t,\ t+n) = \begin{cases} \dfrac{Q}{L_k}, & \text{如果蚂蚁k在本次循环中经过}(i,\ j) \\ 0, & \text{否则} \end{cases} \qquad (3\text{-}4)$$

式中 L_k—第 k 只蚂蚁在本次循环中所走的路径长度。

在一系列标准测试问题上运行的实验表明，Ant-Cycle 算法的性能优于其他两种算法。因此，对蚁群算法的研究正朝着更好地了解蚁周系统模型的方向发展。现在蚁周系统模型通常被称为蚁群算法，而另外两种算法模型被放弃了。

3.6 蚁群算法与其它人工智能算法的比较

目前提出的人工智能算法已有很多，比较有代表性的有模拟退火算法、遗传算法、蚁群算法等。TSP 与 VRP 本质上具有相似性，但 TSP 形式简单、易于描述，属于典型的 NP 问题；更为重要的一点是由于对 TSP 问题的研究已经非常深入，形成了标准的测试数据（TSPLIB），并且在 TSPLIB 中有最优结果可供参考。所以本文在算法的选择与改进方面都是以 TSP 为例进行编程实现，用 TSPLIB 中的标准测试数据来检验各算法的有效性。

（1）模拟退火算法求解框架如图 3-3：

```
Begin
    任选一个初始解 x0，确定初始温度 t0和每一个 t 值下进行迭代的次
数 L;
    xi=x0;  (置初始解为当前解)
    k=0;  (温度变化计数器置 0)
    Repeat
      l=0;  (迭代次数计数器置 0)
      Repeat
        从邻域 N (xi) 中随机选一 xj，计算 Δf=f(xj)-f(xi);
        if (Δf<0)
            xi = xj;
        else if exp(-Δf/tk)>random[0,1]
                xi = xj; l=l+1;
        until l=L;
        k=k+1; tk =t(k);
     until 满足终止条件;
END;
```

图 3-3 模拟退火算法框架

模拟退火算法常用的终止条件有：①给定一个确定的总迭代次数；②设定终止温度；③给定当前的最好解保持不变的最大连续迭代次数。本文程序中判断是否结束模拟退火算法，使用设定终止温度（即零度法）的判断方式，T（k）<= 0.01，结束计算。

（2）遗传算法的流程如图 3-4 所示：

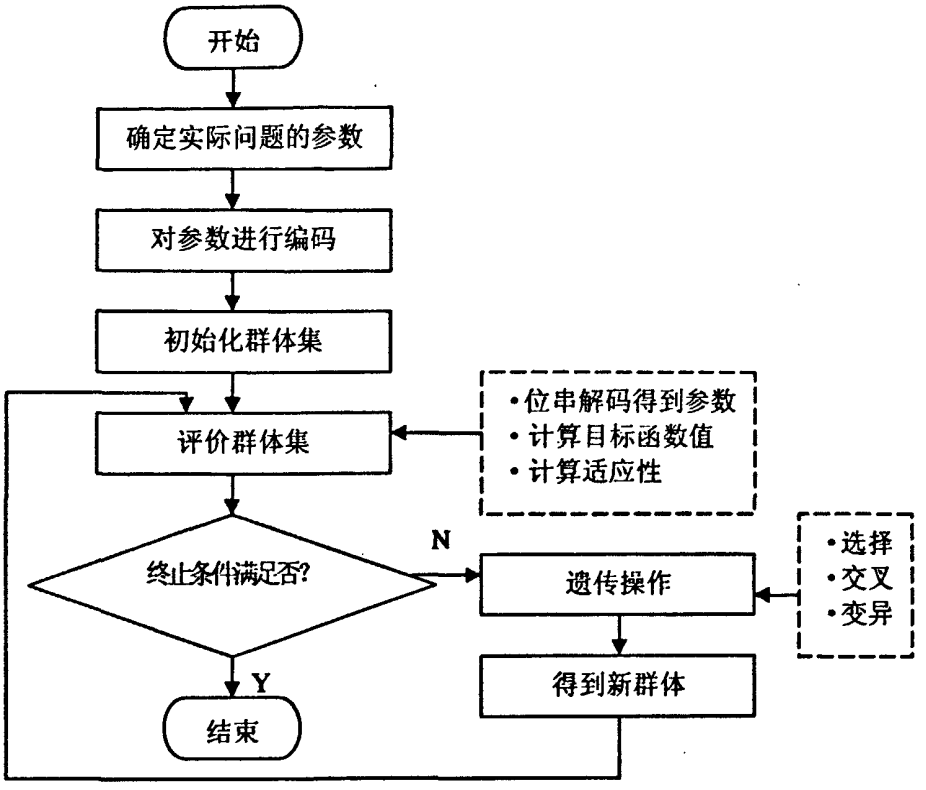

图 3-4 遗传算法流程图

下面对模拟退火算法、遗传算法与蚁群算法进行比较。考虑两组实验，数据均来自 TSPLIB，第 1 组是包含 30 个城市的 Oliver30；第 2 组是包含 50 个城市的 Eil50。因为这两组问题在 TSPLIB 中是比较有代表性的，因此用这两组 TSP 测试问题进行实验是很必要的。在 VC++6.0 环境下实现用三种算法分别解决两组问题，表 3-2 给出了三种算法在解决第 1 组问题时的结果，表 3-3 给出了三种算法解决第 2 组问题的结果，每轮实验进行 10 次取平均值，其中的迭代次数是指获得最优解所需的代数。

表 3-2 三种算法解决 Oliver30 结果（TSPLIB 中标注的最优解为 420.371）

实验轮数	模拟退火算法	遗传算法	蚁群算法
1	426	421	421
迭代次数	[24523]	[3500]	[1485]
2	424	420	420
迭代次数	[24617]	[3426]	[1494]
3	426	424	420
迭代次数	[23996]	[3119]	[1486]
4	422	422	421
迭代次数	[22045]	[3208]	[1493]
5	423	421	421
迭代次数	[24652]	[3186]	[1522]

表 3-3 三种算法解决 Eil50 结果（TSPLIB 中标注的最优解为 428.000）

实验轮数	模拟退火算法	遗传算法	蚁群算法
1	443	428	429
迭代次数	[68417]	[25000]	[2412]
2	439	436	428
迭代次数	[66108]	[24972]	[2394]
3	432	440	430
迭代次数	[67107]	[25002]	[2402]
4	439	437	429
迭代次数	[68100]	[23406]	[2459]
5	440	430	428
迭代次数	[67996]	[23529]	[2481]

实验结果表明，蚁群算法所找出的解的质量最高，遗传算法次之，而模

拟退火算法最低。收敛速度是评价一种算法性能好坏的重要方面，从表 3-2 和表 3-3 可以看出，在同规模的测试问题上三种算法收敛到最优解的迭代次数相差很大。在 Oliver30 中，蚁群算法收敛到最优解在第 1485 代，遗传算法是在 3119 代，而模拟退火算法是在 22045 代。随着城市规模的增大，差距也随之增大。在 Eil50 中，蚁群算法收敛到最优解是在第 2402 代，遗传算法是在 23406 代，而模拟退火算法的收敛速度最慢，在 66108 代才收敛到最优解。以上结果表明，蚁群算法的收敛速度最快。

模拟退火算法需要一个相当长的优化过程，这是该算法最大的缺点。遗传算法容易早熟并且收敛慢。蚁群算法作为一种模拟进化算法，具有正反馈、分布式计算以及贪婪启发式搜索等主要特点。正反馈能够快速地发现较好解；分布式计算使得算法具有很强的并行性，避免了早熟收敛；而贪婪启发式搜索有助于搜索过程的早期就找出可接受的解，缩短了搜索时间。实验表明，和其它算法相比，蚁群算法更具优势。所以在比较了几个典型的人工智能算法后决定采用蚁群算法解决本文研究的问题。然而任何一个算法都不是完美的，蚁群算法存在搜索速度较慢、容易陷入局部优化等问题，为克服这些缺点，本文对基本蚁群算法进行了改进，使其在解决本文研究的实际问题时更加有效。

第4章 蚁群算法的改进及实现

4.1 蚁群算法的改进

一系列研究结果发现，用基本蚁群算法求解问题时容易出现两个问题[49]，一是搜索容易陷入局部最优解，即搜索进行到一定程度后，所有的个体发现的解基本一致，出现停滞现象，不能再对解空间进一步搜索，导致可能无法找到全局最优解；二是收敛到全局最优解的时间较长，求解的结果反复在局部最优解和全局最优解之间震荡。在应用过程中，为了提高算法的性能，人们提出了种种改进措施，比较有代表性的改进算法有带精英策略的蚁群算法、基于优化排序的蚁群算法、Ant-Q System、最大-最小蚁群算法、最优-最差蚁群算法、混合蚁群算法。本章分别把几种典型改进算法编程加以实现，并以 TSP 的标准测试数据对各算法进行测试对比。吸取某些改进算法的思想，结合本文研究的具体问题，寻找有效的解决方案。

4.1.1 带精英策略的蚁群算法

带精英策略的蚁群算法是最早的改进蚁群算法。之所以用精英策略这个词，是因为在某些方面它类似于遗传算法中所使用的精英策略。总的来说，在遗传算法中，如果应用选择、重组和突变这些遗传算子，一代中的最适应个体（一代循环中的最优解）有可能不会被保留在下一代中。在这种情况下，那个最适应个体的遗传信息将会丢失。因此，在遗传算法中，精英策略的思想就是为了保留一代中的最适应个体。类似地，在带精英策略的蚁群算法中，为了使目前为止所找出的最优解在下一循环中对蚂蚁更有吸引力，在每次循环之后给予最优解以额外的信息素量。这样的解被称为全局最优解（Global-best Solution），找出这个解的蚂蚁被称为精英蚂蚁（Elitist Ants）。信息素量根据公式（4-1）进行更新

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \qquad (4\text{-}1)$$

式中 $\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$，$\Delta\tau_{ij}^{k}$和$\Delta\tau_{ij}^{*}$按公式（4-2）、（4-3）求解

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{如果蚂蚁}k\text{在本次循环中经过路径（i，j）}\\ 0, & \text{否则}\end{cases} \tag{4-2}$$

$$\Delta\tau^{*}=\begin{cases}\sigma*\dfrac{Q}{L^{*}}, & \text{如果边（i，j）是所找出的最优解的一部分}\\ 0, & \text{否则}\end{cases} \tag{4-3}$$

式中 $\Delta\tau_{ij}^{*}$—精英蚂蚁引起的路径（i,j）上的信息素量的增加；

σ—精英蚂蚁的个数；

L^{*}—所找出的最优解的路径长度；

研究结果表明，使用精英策略可以使蚁群系统找出更优的解，并且在运行的更早阶段就能找出这些解[50]。但是，如果所使用的精英蚂蚁过多，搜索会很快地集中在极优值周围，从而导致搜索早熟收敛。上面已经提到，早熟收敛就是指所有蚂蚁都沿着同一条路径移动，重复地建立相同的解决方案，从而不能找出更好的解来。因此，需要适当地选择精英蚂蚁的数量。

4.1.2 基于优化排序的蚁群算法

基本蚁群算法和带精英策略的蚁群算法都有一个缺点，若在进化过程中，解的总质量提高了，解元素之间的差异减小了，导致选择概率的差异也随之减小，使得搜索过程不会集中在到目前为止所找出的最优解附近，从而阻止了对更优解的进一步搜索。当路径长度变得非常接近，特别是当很多蚂蚁沿着局部极优的路径行进时，则对短路径的增强作用被削弱了。

在遗传算法中，为了解决这种维持选择压力的问题，一个可行的选择机制是排序，根据适应度对种群进行分类，被选择的概率取决于个体的排序。适应度越高表明该个体越优，个体在种群中的排名越靠前，则被选择的概率就越高。

将遗传算法中排序的概念扩展应用到蚁群算法中，称之为基于优化排序的蚁群算法。具体实施如下：当每只蚂蚁都生成一条路径后，蚂蚁按路径长度（$L_1 \leqslant L_2 \leqslant \cdots \leqslant L_n$），蚂蚁对信息素轨迹量更新的贡献根据该蚂蚁的排名$\mu$的位次进行加权。此外，只考虑$\omega$只最好的蚂蚁。因此，要以有效避免上述的某些局部极优路径被很多蚂蚁过分重视的情况发生。因此σ为到目前为止所找出的最优路径上轨迹贡献的权值，所以这不会被其他任何权值所超过。在排列中，前σ-1只蚂蚁中的一只所经过的边获得一定量的信息素，其量正比于该蚂蚁的排名位次[51]。此外，到目前为止找出最优路径的蚂蚁所经过的边也将获得额外的激素量（这相当于带精英策略的蚂蚁算法中精英蚂蚁的激素更新）。在这样一个带精英和排序混合策略的设置中，轨迹量根据公式（4-4）进行更新

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \qquad (4\text{-}4)$$

式中 $\Delta\tau_{ij} = \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu}$，表示$\sigma$-1只蚂蚁在城市（i,j）之间根据排名对信息素轨迹量的更新。$\Delta\tau_{ij}^{\mu}$和$\Delta\tau_{ij}^{*}$按公式（4-5）和（4-6）计算

$$\Delta\tau^{\mu} = \begin{cases} (\sigma-\mu)\dfrac{Q}{L^{\mu}}, & \text{如果第}\mu\text{只最好蚂蚁经过路径（i, j）} \\ 0, & \text{否则} \end{cases} \qquad (4\text{-}5)$$

$$\Delta\tau_{ij}^{*} = \begin{cases} \sigma * \dfrac{Q}{L^{*}}, & \text{如果边(i, j)是所找出的最优解的一部分} \\ 0, & \text{否则} \end{cases} \qquad (4\text{-}6)$$

式中 μ—最好蚂蚁排列的顺序号；

$\Delta\tau_{ij}^{\mu}$—由第μ只最好蚂蚁引起的路径（i,j）上的信息素量的增加；

L_{μ}—第μ只最优蚂蚁的路径长度；

$\Delta\tau_{ij}^{*}$—由精英蚂蚁引起的路径（i,j）上的信息素量的增加；

σ—精英蚂蚁的数量；

L^*—所找出的最优解的路径长度。

4.1.3 Ant-Q System

蚂蚁圈算法与基本蚁群算法主要有三点不同[52]：蚂蚁的转移规则不同；新增了对各条路径信息素量调整的局部更新规则；全局更新规则不同。下面将作具体说明。

（1）转移概率

在 Ant-Q System，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调整确定性选择的概率。当进化到一定次数后，进化方向已经基本确定，这时对确定性选择概率 q_0 作动态调整，适当加大随机选择的概率，以利于解空间的更完全搜索。位于第 i 个结点的蚂蚁 k 按公式（4-7）的选择策略移动到结点 j：

$$j=\begin{cases}\arg\max_{\mu\in allowed_k}\left\{\left[\tau_{i\mu}(t)^{\alpha}\right]\left[\eta_{i\mu}(t)^{\beta}\right]\right\}, & q\le q_0\\ h, & q>q_0\end{cases} \qquad (4\text{-}7)$$

式中 $\arg\max_{\mu\in allowed_k}\left\{\left[\tau_{i\mu}(t)^{\alpha}\right]\left[\eta_{i\mu}(t)^{\beta}\right]\right\}$—在与 i 相关联的边集中，$\left\{\left[\tau_{i\mu}(t)^{\alpha}\right]\left[\eta_{i\mu}(t)^{\beta}\right]\right\}$最大的边另一结点；

q_0—确定性选择概率；

q—给定参数，$0<q<1$，q 是（0，1）内服从均匀分布的随机变量；

h 依照公式（4-8）在 U_i=allowed$_k$-$\arg\max_{\mu\in allowed_k}\left\{\left[\tau_{i\mu}(t)^{\alpha}\right]\left[\eta_{i\mu}(t)^{\beta}\right]\right\}$ 内随机取值：

$$p_{ih}^{k}(t)=\frac{\left[\tau_{ih}(t)^{\alpha}\right]\left[\eta_{ih}(t)^{\beta}\right]}{\sum_{\mu\in U}\left[\tau_{i\mu}(t)^{\alpha}\right]\left[\eta_{i\mu}(t)^{\beta}\right]},\ h\in U_i \qquad (4\text{-}8)$$

（2）信息素局部更新规则

边(i,j)按公式（4-9）进行更新：

$$\tau_{ij}(t)=(1-\delta)\tau_{ij}(t)+\delta\left[\Delta\tau_{ij}(t)+\gamma\cdot\max_{\mu\in allowed_k}\tau_{i\mu}(t)\right] \quad (4\text{-}9)$$

局部更新是为了避免所有蚂蚁都选择同一条路径。当蚂蚁选择完一条路径后，该路径在此迭代周期内的剩余步数中再被使用的可能性就会降低，这个过程称为信息素挥发，γ称为挥发因子，δ称为学习率，通过引入挥发因子，可以做到对过去信息的慢慢遗忘，因而能够强化后来学习得到的知识，这样可以使较少的路径得到更多的访问机会，搜索的范围会更加广，增加蚂蚁选择其它边的概率，防止算法收敛到局部最优解，有利于发现更好解，不致过早出现停滞现象。

（3）信息素全局更新规则

在 Ant-Q System 的迭代过程中，全局更新原则只对获得最短路径的蚂蚁实施。这样可以使较好的解得到更多的加强，能使算法较快的收敛到最优解。当所有蚂蚁均完成一次循环时，信息素按公式（4-10）更新

$$\tau_{ij}(t+n)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t,t+n) \quad (4\text{-}10)$$

$\Delta\tau_{ij}(t,t+n)$按公式（4-11）计算

$$\Delta\tau_{ij}(t,t+n)=\begin{cases}\dfrac{Q}{L_{best}}, & \text{如果最优路径上的蚂蚁在其路径中使用(i, j)}\\ 0, & \text{否则}\end{cases} \quad (4\text{-}11)$$

4.1.4 最大-最小蚁群算法（Max-Min Ant System）

MMAS 是到目前为止解决 TSP、QAP 等问题最好的改进蚁群算法[53]。和其他寻优算法比较起来，它仍然属于较好的解决方案之一。MMAS 对基本蚁群算法进行了几点改进。考虑本论文研究的内容，拟采用该改进算法的某些思想求解问题。下面对该改进算法进行具体介绍

（1）信息素更新

一次循环中只有最短路径的蚂蚁才进行信息素修改增加，这与 Ant-Q System 调整方法相似；所不同的是，MMAS 没有使用信息素局部更新规则，

仅让每一代中最好个体所走路径上的信息量作调整，以加快收敛速度[54]-[56]。但是这样容易出现停滞现象，于是对信息素的浓度加以限制。

（2）信息素限定

为了避免算法过早收敛到非全局最优解，应当有效地控制人工蚂蚁选择下一个解的概率，而该概率直接取决于信息素浓度和启发式信息。由于启发式因子在算法运行开始时已经设定并且固定不变，因此可以通过限定信息素的范围，弱化信息素对路径选取的影响，一方面使较优路径上的边得到更多的使用，另一方面又要探索未被充分利用的边。

将各路径的信息素浓度限制在$[t_{min},t_{max}]$之间，超出这个范围的值被强制设为 t_{min} 或者 t_{max}，避免了信息素的无限制累加和可能出现的信息素为零的现象。t_{max} 的设置是一个渐进的过程，在算法启动时将所有支路上的信息素浓度初始化为最大值，每次迭代后，按挥发系数 ρ 降低痕迹浓度，只有最佳路径上的支路才允许增加其浓度，并保持在高水平上。t_{min} 的值可以根据经验选取。MMAS 可以有效地避免某条路径上的信息量远大于其余路径，不会使所有的蚂蚁都集中到同一条路径上，从而有进一步寻优的可能。

（3）信息素初始化

为了更加充分地进行寻优，各路径信息素初始化为一个很大的值（通常大于 t_{max}），由于信息素范围的限制，在第一次迭代后设为最大值 t_{max}，这样可以在一开始迭代时增大搜索的范围，因为由于信息素的挥发（挥发因子为 ρ），第一次迭代后，解元素的信息素轨迹的相对差别最大可达 ρ，第二次可达 ρ^2，依次类推。若将信息素初始化为下限值 t_{min}，这种差别就会更大，是将信息素初始化为 t_{max} 时的 6 倍[57]。将信息素初始化为 $t(1)=t_{max}$，是让可行解的各元素间信息素浓度差别不会过大，这样可以使算法遍历搜索空间，不致早熟。实验表明，将初始值设为 $t(1)=t_{max}$ 可以改善最大-最小蚁群系统的性能。从实验结果看，MMAS 在防止算法过早停滞及有效性方面比

基本蚁群算法有较大改进，明显优于基本蚁群算法。

4.1.5 最优最差蚁群算法（Best-Worst Ant System，BWAS）

该改进算法在基本蚁群算法的基础上进一步增强了搜索过程的指导性，使得蚂蚁的搜索更集中于到当前循环为止所找出的最好路径的邻域内[58]。蚁群算法的任务就是引导问题的解向着全局最优的方向不断进化。这种引导机制建立的基础是，一个解决方案越好，越可能在它的附近找出更优的解。因此，将搜索集中于所找出的最优解附近是合理的[57]-[59]。该算法的思想就是对最优解进行更多限度的增强，而对最差解进行削弱，使得属于最优路径的边与属于最差路径的边之间的信息素量差异进一步增大，从而使蚁群的搜索行为更集中于最优解的附近。该算法主要修改了蚁群算法中全局更新公式。当所有蚂蚁完成一次循环后，增加对最差蚂蚁所经过的路径信息素的更新。若（r,s）为最差蚂蚁路径中的一条边，且不是最优蚂蚁路径中的边，则该边上的信息素量按公式（4-13）调整

$$\tau(r,s)=(1-\rho)\cdot\tau(r,s)-\varepsilon\cdot\frac{L_{worst}}{L_{best}} \qquad (4\text{-}13)$$

式中，ε—该算法中引入的一个参数；

L_{worst} — 当前循环中最差蚂蚁的路径长度；

L_{best} — 当前循环中最优蚂蚁的路径长度；

$\tau(r,s)$— 城市 r 和城市 s 之间的信息素轨迹量。

4.1.6 混合蚁群算法

作为一种全局搜索算法，蚁群算法能够有效地避免局部极优。但同时，对大空间的多点全局搜索，却不可避免地增加搜索需要的时间。因此可以将蚁群算法与其它搜索算法结合起来，提高蚁群算法的性能。例如与禁忌搜索算法相结合，利用禁忌搜索算法的优点来提高蚁群算法的性能[59]。简单的禁忌搜索算法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，并利用藐视准则来赦免一些优良状态，其中邻域结构、候选解集、禁忌对象、禁忌长度、藐视准则、终止准则等是影响禁忌搜索算法性能的关键。禁忌搜索算法流程见图 4-1。

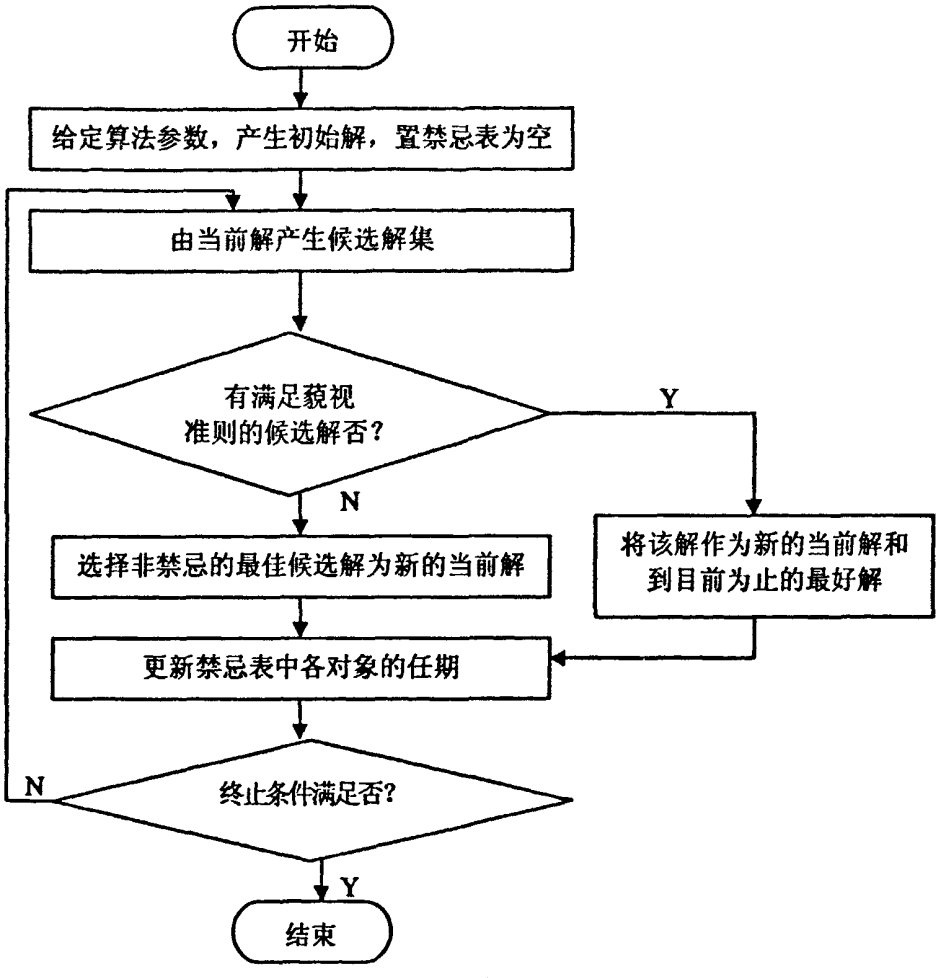

图 4-1 禁忌搜索算法流程图

4.2 编程实现典型改进蚁群算法及结果比较

4.2.1 三种模型的实验结果

在 VC++环境下编程实现几种有代表性的改进蚁群算法。实验选取的测试数据为 TSPLIB 中 20 个城市的 TSP。所有的实验都进行 2000 次循环，蚂

蚁的数量与城市相同 M=20。参数值分别设为 σ =1，β =2，ρ =0.5，Q=100[60]。每轮实验进行 10 次取平均值。

表 4-1 为蚁周，蚁密和蚁量模型的 10 次实验结果。结果表明，蚁周模型明显优于蚁密和蚁量模型。

表 4-1 蚁周系统，蚁密系统和蚁量系统的实验结果

实验轮数	蚁密模型	蚁量模型	蚁周模型
1	27.3098	27.3756	25.6897
2	29.4701	32.5986	27.5642
3	30.9256	27.3091	24.8478
4	33.0274	27.7834	26.4789
5	27.3091	27.0119	28.1247
6	32.0363	30.7230	25.3646
7	27.0589	27.0620	27.0150
8	27.3091	27.0588	25.3263
9	27.3091	30.4478	26.2456
10	26.5369	29.7784	26.4879
平均长度	28.8292	28.7149	26.3165
最优长度	26.5369	27.0119	24.8478

三类模型性能差异产生的原因是指导搜索过程的反馈信息的类型不同。蚁周系统模型使用全局信息[61]，即蚂蚁释放的信息素量正比于所生成解的优劣度。蚂蚁生成的路程越短，它在这条路径上贡献的信息素量就越多。而蚁密和蚁量模型使用的是局部信息，它们的搜索过程不受解优劣度的影响，换句话说，它们中的启发信息 t_{ij} 只是 η_{ij} 的增强，而在蚁周模型中 t_{ij} 代表了一个与 η_{ij} 有关的信息的不同类型[62]。

4.2.2 基本蚁群算法与 Ant-Q System、最大-最小蚁群算法、最优-最差蚁群算法的实验结果

实验选取TSPLIB中20个城市的TSP。基本蚁群算法的参数与前面用到的蚁周系统模型的参数一致；Ant-Q System中的参数设为m=20，α=1，β=2，$\delta=\rho$=0.1，γ=0.5，q_0=0.9，t0=$(n \cdot L_{nn})^{-1}$（其中n是城市数量，L_{nn}是由最近的邻域启发产生的一个路径长度）；最大-最小蚁群算法的参数为m=20，α=1，β=2，ρ=0.9，t_{min}=0.01，t_{max}=10；最优-最差蚁群算法的参数e分别设为{0,0.2,0.5,1,5,20,50,80,90,100}。结果见表4-2：

表4-2 基本蚁群算法与改进蚁群算法实验结果的比较

实验轮数	基本蚁群算法	Ant-Q System	最大-最小蚁群系统	最优-最差蚁群系统（每轮实验取不同的e值）
1	26.0046	25.0089	24.5022	24.7538
2	27.2546	26.4424	24.0009	24.7126
3	27.5642	26.4424	24.0009	24.5262
4	24.8476	25.1247	24.5927	24.8267
5	26.4789	24.7862	24.5374	24.5262
6	26.4789	25.2234	24.6485	24.9137
7	25.3746	25.7128	24.5927	24.0013
8	27.0152	24.9998	24.6942	24.6898
9	28.1247	25.3623	24.2110	24.1239
10	25.6004	26.3297	24.5638	24.7946
平均长度	26.4744	25.5433	24.4344	24.5869
最优长度	24.8476	24.7862	24.0009	24.0013

实验结果表明，最大-最小蚁群算法能够获得20个城市TSP的最优解，其所获得的平均解的质量甚至比基本蚁群算法的最优解还要好。考虑本文研究的实际问题，本文在线路构造时拟采用Ant-Q System算法中的某些思想，将确定性选择和随机选择相结合，在信息素更新时拟采用最大-最小蚁群算法

的改进思想。

第5章 改进蚁群算法求解带约束条件的铁路VRP

5.1 VRP问题模型

VRP是一个典型的组合优化问题，其模型是对车辆路径问题进行理论研究的基础。现实问题中的运输规划往往更加复杂，建模时要考虑到不同的约束条件，如车辆的工作时间、客户定义的时间窗、多运输中心的交互作用等约束条件，因此车辆路径问题的表现形式通常是多种多样的。但是，万变不离其宗，由实际运输问题中的约束条件衍生的各种变形问题仍以一般 VRP 为原型。本文研究的OVRP的数学模型可简化描述为：

$$\min f(x)$$

$$g(x)\geqslant 0,\quad x\in D$$

其中，f(x)为目标函数，g(x)为约束函数，x为决策变量，D表示有限个点组成的集合。

5.2 图论基础及图的存储结构

铁路运输网络可描述为带权的无向图G=(V,A)，其中V为节点集，A为边集。对比邻接矩阵和邻接表后，采用邻接表这一数据结构来存储铁路运输网络图。对于边很多的图，如稠密图，适于用邻接矩阵存储，因为相对于邻接表来说，邻接矩阵占用空间少；而对于顶点多而边少的图，若用邻接矩阵存储，那么对应的邻接矩阵将是一个稀疏矩阵，存储利用率很低。铁路运输网络总体来说是稀疏网络，并且包含成百上千的站点，为避免造成存储空间的严重浪费和数据冗余，采用邻接表存储来提高存储空间利用率。邻接表数据结构已被证明是稀疏网络表达中最有效的数据结构，在交通网络最优路径的求解算法中得到了广泛应用。图5-1是模拟局部铁路运输网络的无向图，图中的顶点代表铁路站点，括号中的数字是站点的坐标。图5-2是图5-1的邻接表存储结构。

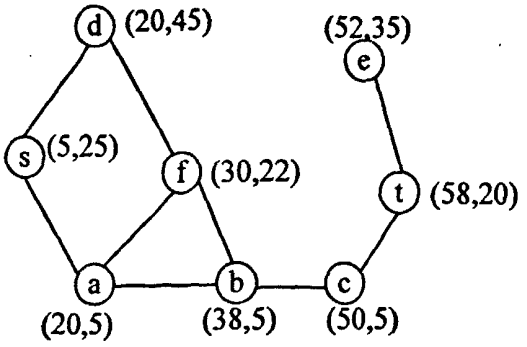

图 5-1 局部铁路运输网络模拟图

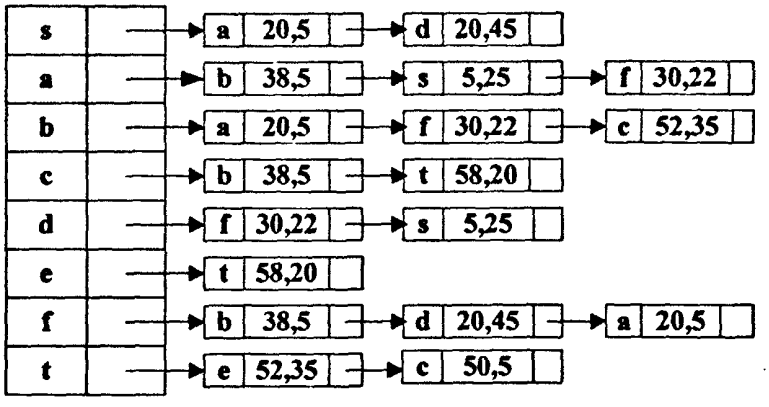

图 5-2 邻接表存储结构

5.3 铁路运输网络图的分层显示

确定了图的存储结构，接下来要实现的就是路网矢量图的分层显示。将省会和枢纽站放在第一层，将二等站和三等站放在第二层。之所以给路网分层，是因为目前我国铁路线路错综复杂，将整个路网都放在一个图层中，在搜索路径时会浪费不必要的时间。例如：寻找“成都-北京”的最短路径时，就可以在第一层中搜索，省去了搜索小站的时间。在 VC++环境下开发了一个小型的绘图软件，可以实现基本图形的绘制、填充等操作；并可进行线形、线宽及图形填充颜色等的设置；也可设置字体、字号、字高及字体的颜色。最后编程实现了路网的分层显示。图 5-3 和 5-4 分别是第一层和第二层铁路运输网络矢量图，表 5-1 是第一层铁路运输矢量图站名对照表，图 5-5 是带

站名标注的铁路运输网络矢量图。

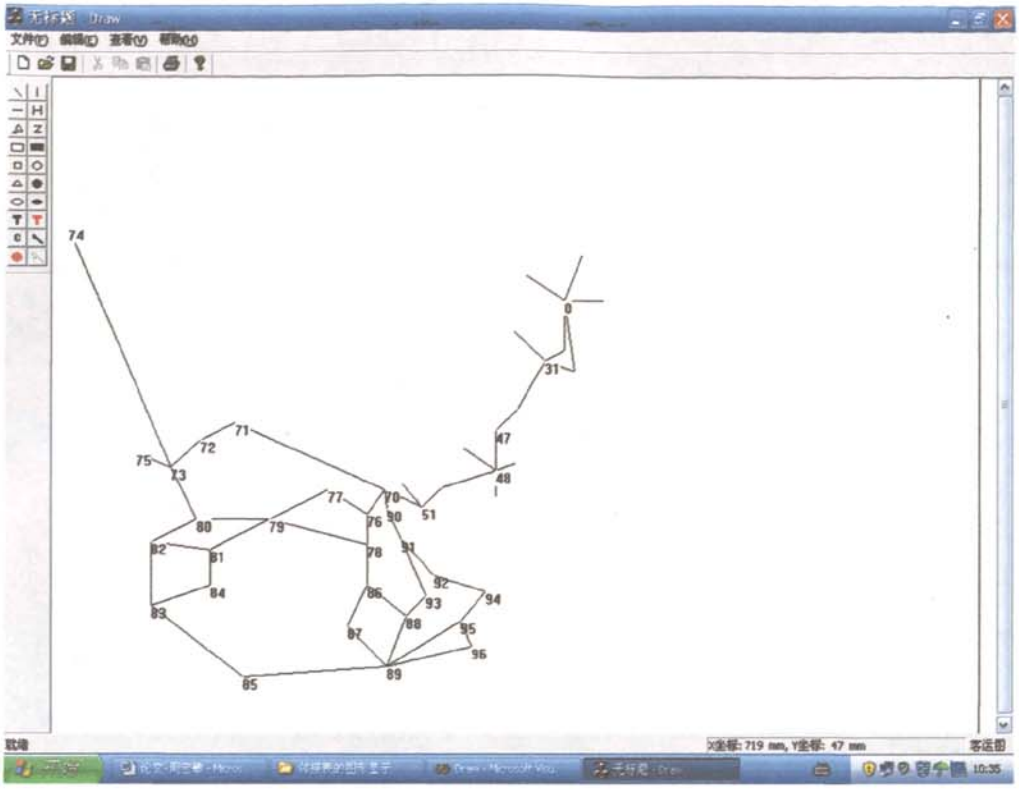

图 5-3 第一层铁路运输网络矢量图

表 5-1 第一层铁路运输矢量图站名对照表

标号	站名	标号	站名	标号	站名	标号	站名
0	哈尔滨	31	长春	48	沈阳	70	北京
71	呼和浩特	72	银川	73	兰州	74	乌鲁木齐
75	西宁	76	石家庄	77	太原	78	郑州
79	西安	80	宝鸡	81	重庆	82	成都
83	昆明	84	贵阳	85	南宁	86	武汉
87	长沙	88	南昌	89	广州	90	天津
91	济南	92	南京	93	合肥	94	上海

标号	站名	标号	站名	标号	站名	标号	站名
95	杭州	96	福州				

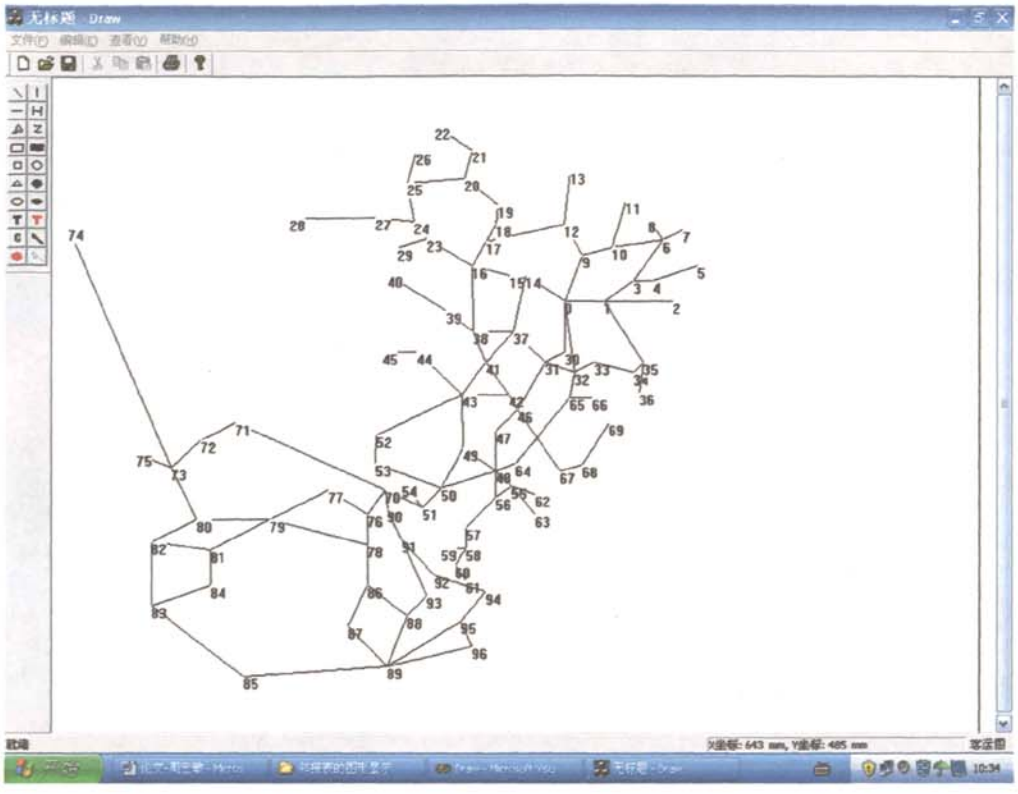

图 5-4 第二层铁路运输网络矢量图

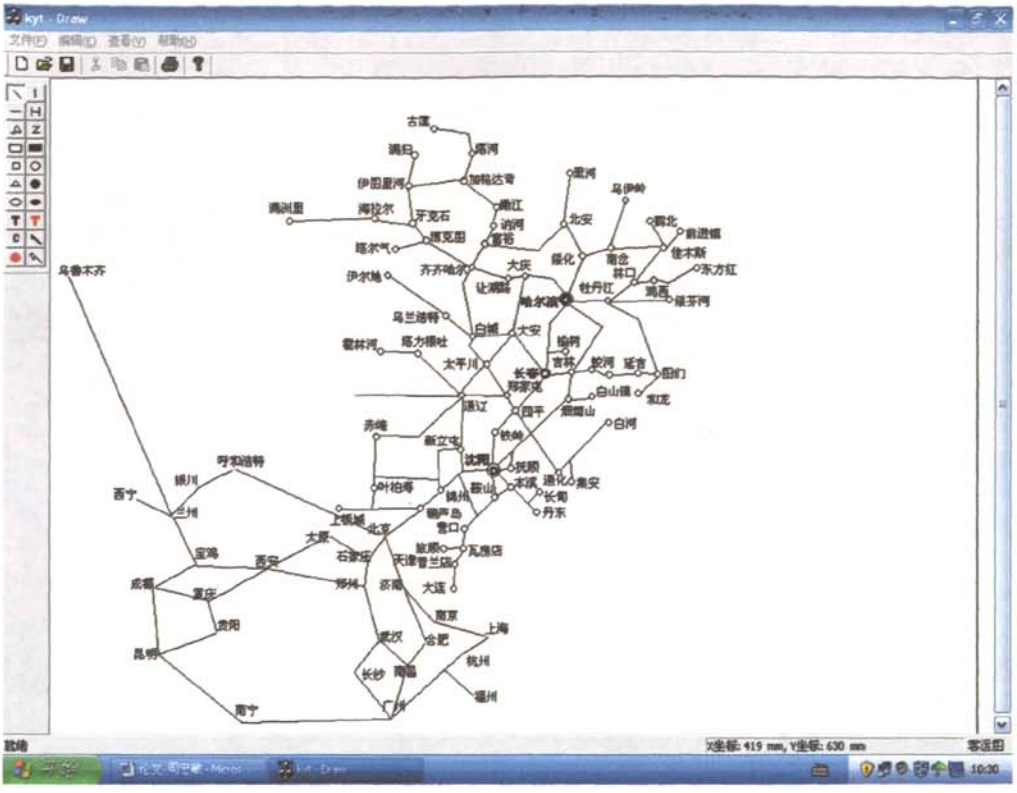

图 5-5 带站名标注的铁路运输网络矢量图

5.4 蚁群算法设计

第三章已经提到，蚁群算法自应用于旅行商问题和二次分配问题取得成功后，在许多组合优化领域都有应用。Bullnheiner 等人就曾经采用蚁群算法结合 2-opt 算法用于解决车辆路径问题。在综合考虑第四章各种改进蚁群算法的基础上，并受到某些改进算法思想的启发，本文采用如下改进方案解决问题。

5.4.1 线路构造

在改进蚁群算法的程序设计中，第一次循环采用随机性选择的策略，第二次以后采用一种确定性选择和随机性选择相结合的策略。随机选择的思想来源于自然界的蚂蚁。自然界的蚂蚁在遇到障碍物时，会随机的挑选一条路前进。但在蚁群算法中，在第一次循环时蚂蚁会选择启发值较大的边，也就

是说第一次循环时蚂蚁选择的站点不是随机的而是固定的。这个固定的站点却并不一定是最优路径包含的站点。为了避免第一次寻优的固定化，程序采取了第一次循环寻优随机化的策略。也就是先把 m 只蚂蚁放在初始站点，第一次循环时 m 只蚂蚁随机地寻找目标站点。这个策略本文也将它应用到了循环进行到一定程度后，蚂蚁找到的解都趋于固定值时，这时再次应用这一随机选择策略，让蚂蚁有重新选择其他解的可能性。

人工蚁具有记忆功能，每只蚂蚁都有相关联的数据结构，叫做禁忌表。在求解问题时，用 $tabu_k$(k=1,2,…, m)记录该蚂蚁当前已经访问过的节点，用于避免蚂蚁访问同一节点不止一次的情况。开始时，输入起始城市和目标城市，所有蚂蚁都集中在起始城市。蚂蚁 k 从起点出发，按照转移策略从和起点相关联的邻接表中选择一个城市，判断是否是目标城市，若不是蚂蚁转移到该城市，然后再从与该城市邻接的邻接表中向下选择，以此类推，直到目标城市出现。若与城市 j 相邻接的邻接表已经为空而目标城市并未出现，则蚂蚁 k 退回到上一级节点，选择与该节点邻接的另一节点，继续寻找目标城市。当目标城市出现，就构造了一条路径，计算路径长度。蚂蚁 k 搜索完后，蚂蚁 k+1 从起点出发，按照和 k 相同的方法，搜索出一条路径，得到另一个解，比较路径长度后，记录到目前为止最短的路径。直到所有蚂蚁都完成搜索，然后更新最优解上的信息素浓度。位于城市 i 的蚂蚁 k，按公式（5-1）的选择策略移动到城市 j，并将 j 加入 $tabu_k$ 中。

$$j=\begin{cases}\arg\max_{\mu\notin tabu_k}\{[\tau_{i\mu}(t)]^{\alpha}[\eta_{i\mu}(t)]^{\beta}\}, q\le p_t \\ \text{随机选择}\quad j\notin tabu_k, otherwise\end{cases} \qquad (5\text{-}1)$$

式中 $\tau_{i\mu}$—连接（i，μ）的信息素强度；

$\eta_{i\mu}$—启发式值，称为能见度（visibility），对于本文讨论的模型，在初始时，$\eta_{i\mu}=1/d_{i\mu}$；

α、β—分别反映了蚂蚁在运动过程中所积累的信息和启发信

息在蚂蚁选择路径中的相对重要性；

$\arg\max_{\mu \in tabu_k}\{[\tau_{i\mu}(t)]^{\alpha}[\eta_{i\mu}(t)]^{\beta}\}$—在与 i 相关联的边集中，$\{[\tau_{i\mu}(t)]^{\alpha}[\eta_{i\mu}(t)]^{\beta}\}$最大的边的另一节点；

q——一个随机数，$0<q<1$；

p_t—确定性选择概率。

表 5-2 列出的实验结果是基本蚁群算法与在程序中加入了随机选择策略的对比，结果表明采取该策略是有效的。源站点（70 北京）目标站点（80 宝鸡）的最短路径长度为 237.078，最短路径为 70（北京）-76（石家庄）-78（郑州）-79（西安）-80（宝鸡）

表 5-2 北京到宝鸡的实验结果

实验次数	改进前结果	改进后结果
1	237.078（最短路径）	237.078（最短路径）
2	237.078（最短路径）	237.078（最短路径）
3	303.144 70（北京）-71（呼和浩特）-72（银川）-73（兰州）-80（宝鸡）	237.078（最短路径）
4	237.078（最短路径）	237.078（最短路径）
5	237.078（最短路径）	237.078（最短路径）
6	303.144 70（北京）-71（呼和浩特）-72（银川）-73（兰州）-80（宝鸡）	237.078（最短路径）
7	237.078（最短路径）	237.078（最短路径）
8	237.078（最短路径）	237.078（最短路径）
9	237.078（最短路径）	237.078（最短路径）

实验次数	改进前结果	改进后结果
10	237.078（最短路径）	237.078（最短路径）

5.4.2 信息素的更新

本文采取的是信息素全局更新规则，一次循环中只有最优路径的蚂蚁才进行信息素更新，更新公式为（5-2）、（5-3）

$$\tau_{ij}(t+n)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t,t+n) \qquad (5\text{-}2)$$

$$\Delta\tau_{ij}(t,t+n)=\begin{cases}\dfrac{Q}{L_{best}}, & \text{如果最优路径包括}(i,j)\\ 0, & \text{否则}\end{cases} \qquad (5\text{-}3)$$

式中 ρ—信息素挥发的系数；

$\Delta\tau_{ij}$—信息素增量；

Q—是一个常数；

L_{best}—最优路径长度。

在一次循环中仅让最好个体所走路径上的信息量作调整，这样可以加快算法的收敛速度，但是这样容易陷入局部最优，出现停滞现象。为了避免算法收敛到非全局最优解而停滞，本文采用最值蚁群算法的思想，对路径上的信息素浓度进行限制，具体做法是把信息素浓度限制在$[t_{min},t_{max}]$之间，超出这个范围的值被强制设为 t_{min} 或者 t_{max}，这样就避免了信息素的无限制累加和可能出现的信息素为零的现象。t_{max} 的设置是一个渐进的过程，在算法启动时将所有支路上的信息素浓度初始化为最大值（程序中 t_{max} 值为 10）[63]，每次迭代后，按挥发系数 ρ 降低痕迹浓度，只有最佳路径上的支路才允许增加其浓度，并保持在高水平上。t_{min} 的值可以根据经验选取（程序中 t_{min} 值为 0.025）[63]。这种策略可以有效地避免某条路径上的信息量远大于其余路径，避免了所有的蚂蚁都集中到同一条路径上，从而使进一步搜索更优解的行为得以继续。

5.4.3 约束条件

目前铁路运输网络繁忙，必须要考虑全局优化问题，交通平衡分为用户最优平衡 UE（User Optimum Equilibrium）和系统最优平衡 SO（System Optimum Equilibrium）。一般来说系统最优平衡是系统规划者所期望得到的一种网络交通流的最优化分布状态，即所谓全局最优。传统的路径优化偏重于最短路径，这对单个用户无疑是有效的，但若在整个交通网中使用就会发生问题。例如，在一个周期，大量用户精确搜索到一条最短路径，此路径就会被大量使用，造成拥挤直至道路阻塞；到下一周期，此路径对这些用户来说可能不是最短的，就会造成在这一周期内该路径的空闲，如此便会造成路网的震荡。这表明简单的最短路径算法有时无法满足合理配置交通流量的需要。考虑了流量约束后，蚁群算法计算得到的解对单个用户来说并不一定是最短的，但对整个系统来说却是最合理的。

用蚁群算法求解带约束条件的最短路径问题普遍采用自适应路径选择的方法，也就是采用不同的信息素更新或者挥发规则。本文采用的方法是调整 d_{ij}'，在一轮寻优周期，第一次寻优时 d_{ij}'的值为 d_{ij}（d_{ij} 是两城市之间的距离），寻找到最优路径后，若没有选择进入新一轮的寻优，则最短路径所包括的边的距离按公式（5-4）调整。

$$d_{ij}' = d_{ij} + \lambda d_{ij} \quad (0<\lambda<1) \qquad (5\text{-}4)$$

在选择了调整 d_{ij}'后，蚁群算法的一个重要参数 η_{ij} 将受到影响。η 是边(i,j)的能见度（visibility），是一个启发式值，反映由城市 i 转移到城市 j 的启发程度。对于本文讨论的模型，$\eta_{ij}=1/d_{ij}$，通常情况下这个量在蚂蚁系统的运行中是不变的。但由于本文采用调整 d_{ij}'的策略，η_{ij} 在循环初始时的值为 $1/d_{ij}$，一次寻优找到最优路径后，最优路径中所包括边的 η_{ij} 就调整为 $\eta_{ij}=1/d_{ij}'$。

程序的具体实现是，设三个图，其中一个是全局变量图 g，另外两个分

别是初始图 chushi、找到最优路径后d_{ij}'需要改变的图 gaibian。在输入起始站点-目标站点找出最优路径后，程序会提示是否还有用户要寻找路径以及提示下面的寻找是否是在同一周期内，若选择是在一个周期内，全局变量图 g 指向 gaibian；若选择不是，则进入新一周期的寻优，全局变量图 g 指向 chushi。设一个记录边信息的结构体数组，用它来传递图中边的信息，这样就可以实现图的遍历、记录与复制操作。同时由于两站点d_{ij}'值的动态调整，找到的最优路径长度会在再次寻优时增长，这样选择另一条次优路径的可能性就会增大，进而实现系统最优平衡。

5.5 实验与结果

采用 VC++6.0 为编程工具，模拟实际铁路运输图，参数的取值见下文。编程实现了能达到系统最优平衡的最优路径的求解。输入起点 14（大庆）终点 76（石家庄）程序运行结果如下：最优路径长度是：333.616，最优路径是：14（大庆）-37（大安）-41（太平川）-43（通辽）-49（新立屯）-50（锦州）-51（葫芦岛）-70（北京）-76（石家庄），界面如图 5-6 所示。

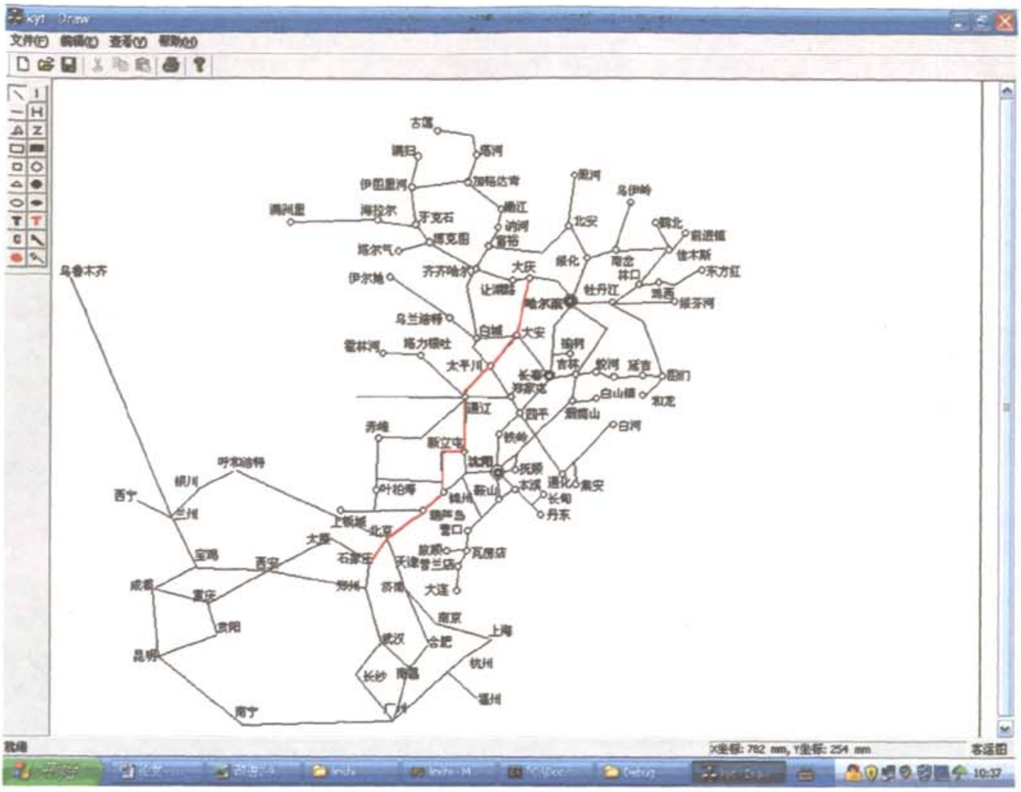

图 5-6 同一周期第一次寻找大庆到石家庄的最优路径

程序提示：是否还需要寻找最优路径，若还要寻找，程序进一步提示：是否是同一周期的寻找，在选择“是” 后。若继续输入起始站点是 14（大庆），目的站点是 76（石家庄），此时第一次寻找到的最优路径所包括的边的 d_{ij}’值已经根据公式 5-4 进行了调整，第一次寻找到的最优路径的长度已经由最初的 333.616 变为第二次寻找的 443.71，此时，另一条路径的长度已经比 443.71 更优，所以程序的最终运行结果是：最优路径长度是：431.428，最优路径是：14（大庆）-37（大安）-31（长春）-46（四平）-47（铁岭）-48（沈阳）-50（锦州）-51（葫芦岛）-70（北京）-76（石家庄），界面如图 5-7 所示。

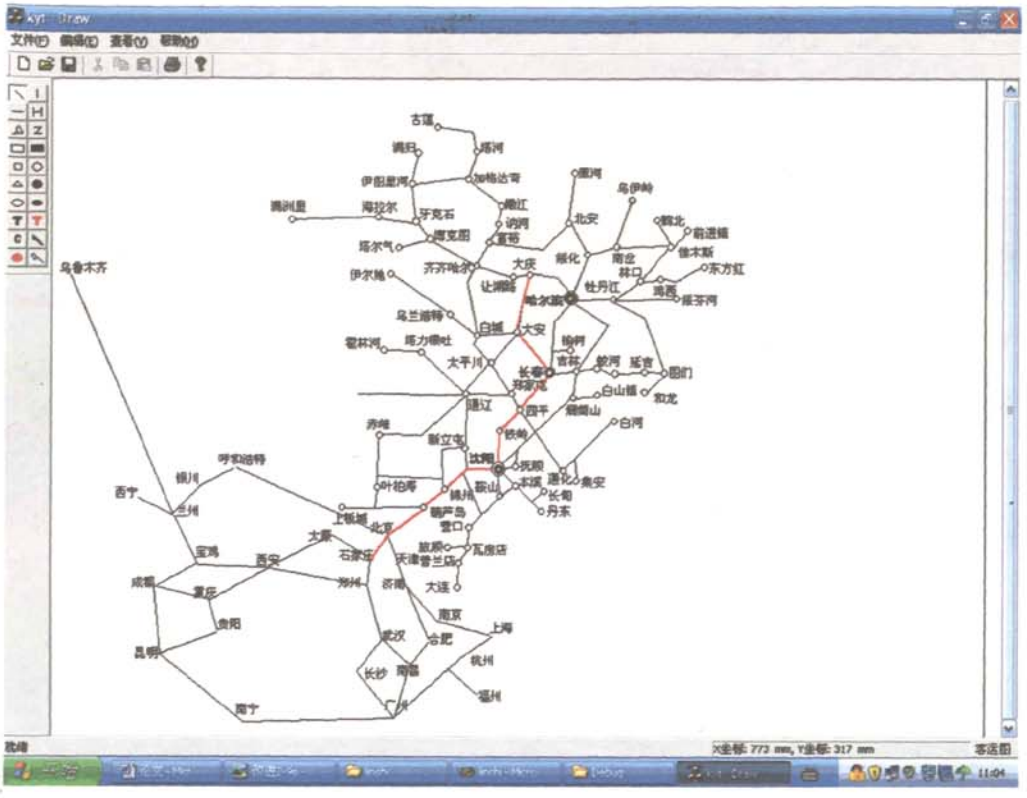

图 5-7 同一周期第二次寻找大庆到石家庄的最优路径

继续寻优依然是上述起点和终点查询结果变为（509.246）： 14（大庆）-15（让湖路）-16（齐齐哈尔）-38（白城）-41（太平川）-43（通辽）-49（新立屯）-50（锦州）-51（葫芦岛）-70（北京）-76（石家庄）；第四次输入相同的起点和终点，仍然是同一周期最优路径的寻找，查询结果变为（558.010）：14（大庆）-0（哈尔滨）-32（吉林）-65（烟筒山）-64（抚顺）-48（沈阳）-50（锦州）-51（葫芦岛）-70（北京）-76（石家庄）。第五次找到的最优路径与第一次找到的相同，但路径长度已经变为：706.093

在一个周期内寻找乌鲁木齐（74）-贵阳（84），找到的最优路径长度为461.665，最优路径为：74（乌鲁木齐）-73（兰州）-80（宝鸡）-82（成都）-81（重庆）-84（贵阳）。在另一新的周期内寻找银川（72）-贵阳（84），找到的最优路径长度为 244.127，最优路径为 72（银川）-73（兰州）-80（宝

鸡）-82（成都）-81（重庆）-84（贵阳）。但在同一周期内输入这两个查询项目，先查询乌鲁木齐（74）-贵阳（84），再查询银川（72）-贵阳（84），程序给出的结果是：乌鲁木齐到贵阳的路径长度为 461.665，查询到的路径为：74（乌鲁木齐）-73（兰州）-80（宝鸡）-82（成都）-81（重庆）-84（贵阳）；银川到贵阳的路径长度为 257.579，找到的路径为：72（银川）-73（兰州）-80（宝鸡）-79（西安）-81（重庆）-84（贵州）。之所以出现这种结果是因为在同一周期的寻找，考虑了约束条件。同一周期内程序给出的贵阳到银川的路径虽不是最短的，但却给出了考虑流量问题的备选路径。运行界面如图 5-8 所示：

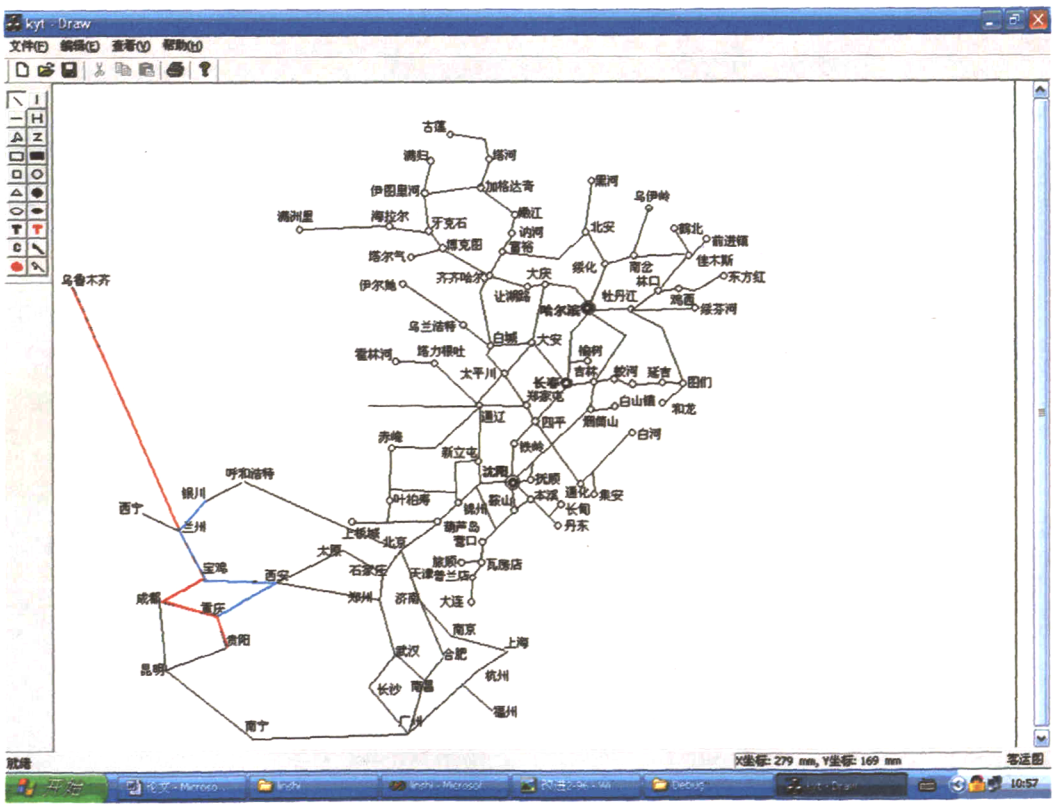

图 5-8 同一周期找到的乌鲁木齐到贵阳与银川到贵阳的路径

5.6 算法中的参数选择

根据实验发现算法中的参数取不同的值对结果有很大的影响。Q 为蚂蚁

循环一周时释放在所经过路径上的信息素总量。总信息量Q越大，则在蚂蚁已经走过的路径上的信息素累积就越快，可以加强蚁群搜索时的正反馈性能，有助于算法的快速收敛。在本文的数据规模下Q值取100获得的实验结果令人满意。α值的大小表明结点上的信息量受重视的程度，α值越大，蚂蚁选择以前选过的点的可能性越大，但过大会使搜索过早陷入局部最小点，本文实验时α取值为3。β的大小表明启发式信息受重视的程度，β越大，表明选择路径时越依赖启发式信息，程序中β取值为2。信息素挥发系数ρ的大小直接关系到蚁群算法的全局搜索能力及其收敛速度，取值太大或太小，运行结果都不理想，ρ取值趋近于1时，挥发太快，路径上的痕迹基本上无积累，不能很好的在蚂蚁之间传递信息，ρ取值趋近于0时，则挥发很小或几乎不挥发，此时以前积累的信息比例太大，不易更新，也不可能全面地传递信息，故效果也不好，实验表明，ρ的取值在0.5-0.9之间，效果较好，程序中ρ取为0.8。m为蚂蚁的数目，单个蚂蚁在一次循环中所经过的路径，表现为问题可行解集中的一个解，m个蚂蚁在一次循环中所经过的路径，则为可行解的一个子集。显然，子集越大越可以提高算法的全局搜索能力以及算法的稳定性，但过大会使信息正反馈的作用不明显，收敛速度减慢；反之，子集较小，收敛速度会加快，但本文的实验处理问题的规模比较大，子集过小会使算法的全局性能降低，算法的稳定性能差，容易出现过早停滞现象。通过实验，m的理想值在100-300之间，程序中取为200。本文有一参数λ，λ的选择依据是约束条件的强弱，约束条件越严那么λ的取值就越大，程序中λ的值为0.3。程序终止的条件，可通过设定循环次数或根据进化的趋势及问题的要求来决定，本文采用设定循环次数。蚁群算法中各个参数的作用实际上是紧密耦合的，一个参数值的选取有赖于其他参数的配置[64]。Q、α、β、ρ可以用实验方法确定最优组合，已经公布的一些例子的经验结果为：①$0\leqslant\alpha\leqslant5$；②$1\leqslant\beta\leqslant5$；③$0.1\leqslant\rho\leqslant0.99$,取0.7左右为佳；④$10\leqslant Q\leqslant1000$。

结论与展望

1、结论

多种运输方式的激烈竞争，货主对运输方式的选择余地越来越大，对运输服务质量的要求越来越高。可靠的运输能力，是使货运营销目标得以实现的根本保证。为了适应货运市场营销的需要，铁路一方面要继续加快建设，提高路网的综合运输能力；另一方面要科学地进行运输组织，根据货运市场的需求优化运力配置。否则就建立不了正确、迅速、安全、廉价的作业体制。本文在对带流量约束的铁路车辆路径问题研究后得出如下主要结论：

（1）根据铁路运输中带流量约束的线路优化问题的实际情况，在对其分类、模型建立以及求解方法进行详细分析、讨论和对比后，决定采用蚁群算法来求解问题。

（2）对铁路运输网络矢量图的分层有助于搜索速度的提高。

（3）分析了许多典型改进蚁群算法，编程实现并进行对比。在吸收一些典型改进蚁群算法的思想后，针对研究的问题本文采取了以下改进措施：①确定性选择和随机性选择相结合的选择策略；②信息素全局更新规则；③对信息素浓度进行限制。实验表明这些改进是有效的。

（4）考虑了流量约束，目的就是要获得网络交通流的最优化分布状态，这样就能避免某路径同一时间被大量使用，造成拥挤直至道路堵塞。考虑了约束条件后的蚁群算法计算得到的解并不一定是最短的，而是给出了考虑流量约束的备选路径。本文采用的方法是动态地调整 d_{ij}'，通过大量实验发现这种调整是有效的。

（5）针对论文研究的实际问题，多次实验后在论文的最后总结了蚁群算法中各参数的选取规则。

2、展望

随着优化方法及理论的发展，蚁群算法这种寻优思想具有十分光明的前

景，更多深入细致的工作还有待开展。举例如下：

（1）蚁群算法中的参数设定有很多经验因素，目前还没有很好的规律，只能依照试探。怎样根据不同的问题抽象出一些有共性的特征，依据这些特征配置合适的参数，是一个值得研究的问题。

（2）对蚁群算法而言，在基本原理已经明确的条件下，重要的就是开发出求解问题的算法模型，使问题的求解更加切实有效。而在工程实际的应用中，算法模型的收敛和算法的复杂度是值得深入研究的问题。

蚁群算法在带约束条件的铁路 VRP 上的这一成功尝试再次表明了该算法在组合优化领域是具有强大竞争力的。这种模仿自然界生物的新型寻优算法无疑具有十分光明的前景。随着深入的研究，该算法必将成为一种强有力的寻优手段和工具，将会在越来越多的领域中得到运用。

致谢

两年多的研究生学习即将告一段落，虽然仅仅是两年多的时光，但是我自己却觉得过得很充实。记得刚开始读研时，连自己想从事哪方面的研究都不清楚，如今即将从交大毕业了，我最想感谢的是我的导教楼新远副教授。他严谨的治学使我在学业上获得了很大的收获；悉心的关怀使我在生活上得到了很大的帮助。由于我以前所学的专业与现在的计算机应用技术相差较远，因此刚入学时我学习很吃力，而楼老师却一直鼓励我、支持我，并根据我的专业背景为我选择了路径寻优这一有着光明前途的研究方向，在楼老师的耐心点拨、细心指导下，我的科研能力逐步提高，并且在今天得以顺利完成我的论文。回忆起这两年多来楼老师对我的关怀与帮助，我要在这里对楼老师说一声“谢谢您”，祝您事业百尺竿头，更进一步。

我还要感谢和我一起学习的同学们，在与你们的日常交往中，通过不同学术观点的交流，我获得了很多知识，扩大了自己的眼界。同学们，谢谢你们，祝你们在以后的岁月里都能大展宏图，实现自己的梦想。

最后，衷心感谢我的家人多年来对我辛苦的栽培，有了你们的支持我才能够专心于自己的学业，我一定会把你们的关心作为鼓励，踏实工作，努力向前。

参考文献

[1] 黄方林．现代铁路运输概论．西南交通大学出版社，2002：26-29

[2] 杨文鹏，贺兴时，杨选民．新编运筹学教程——模型、解法及计算机实现．陕西科学技术出版社，2005：12-14

[3] 严余松，蒋葛夫．智能铁路系统与枢纽车流组织优化．西南交通大学出版社，2001：16-18

[4] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. Management Science, 1959：81-89

[5] J.A.G.Willard. Vehicle routing using r-optimal tabu search: [M. Sc. dissertation]. London: The Imperial College. 1989

[6] Gendreau, Hertz and Laporte. A fast composite heuristic for the symmetric traveling salesman problem. INFORMS Journal on computing. 1994

[7] G. Barbarosoglu and D. Ozgur. A tabu search algorithm for the vehicle routing problem. Computers and Operations Research. 1999, 26：255-270

[8] [美]森尼尔·乔普瑞，彼得·梅因德尔著．李丽萍译．供应连管理——战略、规划与运营．社会科学文献出版社，2003：53-54

[9] L Schrage. Formulation and structure of more complex/realistic routing and scheduling problem. Networks, 1981：63-68

[10] D Sariklis and S Power. A heuristic method for the open vehicle routing problem. Journal of the Operational Research Society. 2000, 51：564-573

[11] J Brandao. A tabu search algorithm for open routing problem. European journal of Operational Research, 2005：128-132

[12] 李军，郭耀煌．物流配送车辆优化调度理论与方法．北京：中国物资出版社，2001

[13] 李军．车辆调度问题的分派启发式算法．系统工程理论与实践．1999（1）：27-33

[14] 李军．车辆调度问题的网络启发式算法．系统工程．1999（2）：66-71

[15] 李大为．车辆路径问题的求解方法研究．系统工程．2004

[16] 张涛，王梦光．遗传算法和3-OPT结合求解带有能力约束的VRP．东北大学学报，1999

[17] 肖朋等. 车辆路径问题的单亲遗传算法. 计算机技术与自动化，2000

[18] 霍雪丽等. 车辆路径问题（VRP）的蚂蚁搜索算法. 系统工程学报，2004

[19] 魏俊华，王安麟. 基于分段遗传编码的配送路径优化. 工业工程与管理，2005

[20] 尹晓峰，杜艳萍. 车辆路径问题的蚁群算法研究. 太原科技大学学报，2005

[21] 谢秉磊. 随机车辆路径问题研究. 西南交通大学. 2003

[22] N. Christofides. Exact algorithms for the vehicle routing problem based on spanning the shortest path relaxtion[J] . Mathematical programming. 1981，20：255-282

[23] M. Fisher. Optimal solution of vehicle routing problems using minimum k-trees[J] . Operation Research. 1994，42（4）：624-626

[24] G. Laporte. A tabu search heuristic for the vehicle routing problem. Centre de recherché surles transpors. 1991

[25] J. Lenstra. Complexity of vehicle routing and scheduling problem. Networks. 1981，11：221-227

[26] S. Eilon. Distribution management：mathematical. Modeling and practical analysis[M]. 1974，22：340-349

[27] N. Christofides. The vehicle routing problem. RAIRO. 1976，10：55-70

[28] E. Taillard. Parallel interactive search method for vehicle routing problems[J] . Networks. 1993，23：661-673

[29] Rao. Ants Can Solbe Constraint Satisfaction Problem[J] . In IEEE Transactions on Evolutionary Computation. 2002，6：347-358

[30] Desrocher Karl Hartl. D-Ants：Savings Based Ants divide and conquer the vehicle routing problem. Computers & Operations Research. 2004，31：563-591

[31] Benders. On an integer program for a delivery problem[J] . Oprrations Research. 1962，12：300-304

[32] S. Martello. Algorithms and Computer Implementations. Chichester：Wiley. 1990

[33] A. Van Breedam. An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related,customer-related and time-related constraints：[Ph.d.dissertation] . Unibersity of Antwerp. 1994

[34] A. Alfa. A 3-opt based simulated annealing algorithm for vehicle routing

problems. Computers & Industrial Engineering. 1991，21：635-639

[35] C. Rego. A subpath ejection method for the vehicle routing problem. Management Science. 1998，44：1447-1459

[36] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research. 1964，12：568-581

[37] B . Gillett . A generalized assignment heuristic for the vehicle routing problem. Networks. 1981，11：109-124

[38] N. Christofides，A. Mingozzi. The vehicle routing problem. Combinatorial Optimization. 1979:315-338

[39] M. Gendreau. A tabu search heuristic for the vehicle routing problem [M].Montreal: Publication .Centre de recherché surtes transpors, 1991

[40] 王小平，曹立明. 遗传算法—理论、应用与软件实现[M] . 西安交通大学出版社. 2002

[41] M.M.Syslo，N.Deo and L.S.Kowalik. Discrete Optimization Algorithms with Pascal Programs. New Jersey：Prentice-Hall，Inc. 1983

[42] M.Dorigo. The ant system：optimization by a colony of cooperating agents. IEEE Transactions on system ,Man and Cybernetics[B] . 1996，26（1）：1-13

[43] Wu QH，Zhang JH. Ant ant colony algorithm with mutation features. Jounal of Computer Research & Development. 1999，36（10）：1240-1245

[44] M.Dorige，V.Maniezzo，A.Colorni. Ant system：Optimization by a colony of cooperating agents. IEEE Trans. 1996，26（1）：28-41

[45] 马良，项培军. 蚂蚁算法在组合优化中的应用. 管理科学学报. 2001，2：32-37

[46] D.S.Hochbaum. Approximate algorithms for NP-hard problem[M] . PWS Publishing Company. 1997：181-184

[47] G.Clarke，J.W.Wright. Scheduling of vehicles form a central depot to a number of delivery points[J] . Operations Research. 1964，12：568-581

[48] 丁建立，陈增强，袁著祉. 基于混合蚂蚁算法的网络资源均衡与优化. 仪器仪表学报. 2003：592-594

[49] 毕军，付梦印，张宇河. 一种改进的蚁群算法求解最短路径问题. 计算机工程与

应用．2003，3：107-109

[50] 贾刚力，杨家本．自适应调整信息素的蚁群算法．信息与控制．2002，31（3）：198-201

[51] 徐婕，詹士昌．动态调整信息素的蚁群算法．汉中师范学院学报．2003，21（2）：31-35

[52] M.Luca，Gambardella，M.Dorigo．Ant-Q：an reinforcement learning approach to the traveling salesman problem[A] ．Proc.of 12th Machine Learning Conf.[C].France：Morgan Kaufmann．1995,252-260

[53] T.Stutzle，H.Hoos．Max-Min ant system．Future Generation Computer System．2000，8：889-914

[54] Thomas Stuztle，Holger Hoos．Improvements on the ant system．Artificial Neural Networks and Gentic Algorithms．1998:245-249

[55] Thomas Stuztle，Holger Hoos．Max-Min ant system and local search for the traveling salesman problem[A] ．Proc IEEE International Conference on Evolutionary Computation．1997:309-314

[56] Thomas Stuztle．Max-Min ant system．Elservier Science．1999

[57] Thomas Stuztle，Holger Hoos．Max-Min ant system and local search for combinatorial optimization problem[M] ．Advances and Trends in Local Search Paradigms for Optimization,kluwer,boston．1999：313-329

[58] Reimann Marc，F.Richard．D-Ants:Savings Basede Ants divide and conquer the vehicle routing problem．Computers & Operations Research．2004，31：563-591

[59] B.Bullnheimer，R.F.Hartl．An improved ant system algorithm for the vehicle routing problem．Annals of Operations Research．1999，89：319-328

[60] A.Colorini，M.Dorigo．Heuristics from nature for hard combinatorial optimization problems．International Trans Operational Research．1996，3（1）：1-21

[61] M.Dorigo L.M.Gambardella．A study of some properties of ant system．Proceedings of the PPSN 44th International Conference on Parallel Problem Solving from Nature．1996:656-665

[62] B.Wu，Z.Z.Shi．An ant colony algorithm based partition algorithm．Chinese Journal of

Computers. 2001, 24 (12): 1328-1333

[63] 李十勇. 蚁群算法及其应用. 电子图书. 2004, 9: 78-96

[64] 詹士昌, 徐婕, 吴俊. 蚁群算法中有关算法参数的最优选择. 科技通信. 2003, 3 (4): 281-386

附录

改进蚁群算法核心代码：

```
class ant
{
public:
int ChooseNextCity(int max);double prob[iCityCount];
int m_iCityCount;void addcity(int city);
int tabu[iCityCount];int AllowedCity[iCityCount];
void Clear();void UpdateResult();
double m_dLength,m_dShortest;
int move(int max);ant();
};
int ant::ChooseNextCity(int max)
{
int j=10000;
double temp=0;
int curCity=tabu[m_iCityCount-1];
if(max==1||p>p0)
{
  edgenode *p=g->adjlist[curCity].firstedge;
  int nextpoint[iCityCount];int m0=0,m1;
  while(p!=NULL)
     {
         nextpoint[m0++]=p->adjvex;
         p=p->next;
     }
     m1=(int)(rand()%m0);
      if((AllowedCity[nextpoint[m1]]==1))
          j=nextpoint[m1];
      return j;
}
else
{
edgenode *p=g->adjlist[curCity].firstedge;
while(p!=NULL)
{
  if((AllowedCity[p->adjvex]==1))
  temp+=pow((1.0/p->distance),beta)*pow((p->m_dTrial),alpha);
  p=p->next;
}
p=g->adjlist[curCity].firstedge;
double sel=0;
while(p!=NULL)
{
  if((AllowedCity[p->adjvex]==1))
  {
     prob[p->adjvex]=pow((1.0/p->distance),beta)*pow((p->m_dTrial),alpha)/temp;
     sel+=prob[p->adjvex];p=p->next;
```

```
    }
 else
  {
    prob[p->adjvex]=0;p=p->next;
  }
}
p=g->adjlist[curCity].firstedge;
double mRate=rnd(0,sel);
double mSelect1=0;
while(p!=NULL)
{
  if((AllowedCity[p->adjvex]==1))
  {
    mSelect1+=prob[p->adjvex];
    if(mSelect1>=mRate) {j=p->adjvex;break;}
    p=p->next;
  }
  else
  {
   prob[p->adjvex]=0;p=p->next;
  }
  if (j==10000)
  {
   p=g->adjlist[curCity].firstedge;
   edgenode *q;double mSelect=0;
   while(p!=NULL)
   {
     if(prob[p->adjvex]>mSelect)
      {
        mSelect=prob[p->adjvex];q=p;p=p->next;
      }
     else p=p->next;
   }
if (mSelect!=0)
{
j=q->adjvex;
}
}
return j;
}
void ant::UpdateResult()
{
int i=0;edgenode *p;
do
{
    p=g->adjlist[tabu[i++]].firstedge;
    while(p->adjvex!=tabu[i])
    p=p->next;m_dLength+=p->distance;
}while(p->adjvex!=aimcity);
}
int ant::move(int max)
```

```
{
int j=ChooseNextCity(max);
if (j==10000)
return 0;
else
{
addcity(j);return 1;
}
}
class project
{
public:
void UpdateTrial();
double m_dLength;
void initmap();
ant ants[iAntCount];
void GetAnt();
int StartSearch();
project();
};
void project::UpdateTrial()
{
int i=0;edgenode *p;
do
{
   p=g->adjlist[besttour[i++]].firstedge;
   while(p->adjvex!=besttour[i])  p=p->next;
   p->m_dTrial=rou*p->m_dTrial+Q/p->distance;
   if(m_dTrial> m_dTrial_max) m_dTrial= m_dTrial_max;
   else if(m_dTrial< m_dTrial_min) m_dTrial= m_dTrial_min;
}while(p->adjvex!=aimcity);
}
project::project()
{
 m_dLength=10e9;
}
void project::GetAnt()
{
int i=0;
for (i=0;i<iAntCount;i++)
ants[i].addcity(primcity);
}
int project::StartSearch()
{
int max=1;
int j, k;double temp; int temptour[iCityCount];
while (max<=iItCount)
{
for(j=0;j<iAntCount;j++)
{
  int l;
```

```
  while(g->adjlist[ants[j].tabu[ants[j].m_iCityCount-1]].vertex!=aimcity)
  {
  l=ants[j].move(max);
  if (l==0) {break;}
  }
}
for(j=0;j<iAntCount;j++)
{
  if (g->adjlist[ants[j].tabu[ants[j].m_iCityCount-1]].vertex==aimcity)
  ants[j].UpdateResult();
}
int t;
for(j=0;j<iAntCount;j++)
   if (g->adjlist[ants[j].tabu[ants[j].m_iCityCount-1]].vertex==aimcity)  break;
    temp=ants[j].m_dLength;
for (t=0;t<ants[j].m_iCityCount;t++)
  temptour[t]=ants[j].tabu[t];
for(j=0;j<iAntCount;j++)
 {
  if (g->adjlist[ants[j].tabu[ants[j].m_iCityCount-1]].vertex==aimcity)
  {
    if (temp>=ants[j].m_dLength)
    {
     temp=ants[j].m_dLength;k=ants[j].m_iCityCount;
     for(t=0;t<ants[j].m_iCityCount;t++)
     temptour[t]=ants[j].tabu[t];
    }
   }
 }
if(temp<m_dLength)
{
  m_dLength=temp;
  for(t=0;t<k;t++)
  besttour[t]=temptour[t];
}
UpdateTrial();
for(j=0;j<iAntCount;j++)
  ants[j].Clear();max++;
}
}
void createadjgraph(adjgraph *g)
{
    int i,j,k;edgenode *s;
    ifstream in("dingdian.txt");
    in>>g->n>>g->e;
    for(k=0;k<g->n;k++)
    {
     in>>g->adjlist[k].vertex;
     g->adjlist[k].firstedge=NULL;
    }
    for(k=0;k<g->n;k++)
```

```
    {
      in>>g->adjlist[k].x>>g->adjlist[k].y;
    }
    for(k=0;k<g->e;k++)
    {
      in>>i>>j;
      s=(edgenode *)malloc(sizeof(struct node));
      s->adjvex=j;in>>s->x>>s->y;s->m_dTrial= m_dTrial_max;
      s->distance=sqrt(pow((g->adjlist[i].x-g->adjlist[j].x),2)+pow((g->adjlist[i].y-g
      ->adjlist[j].y),2));
      s->next=g->adjlist[i].firstedge;
      g->adjlist[i].firstedge=s;
      s=(edgenode *)malloc(sizeof(struct node));
      in>>s->x>>s->y;s->adjvex=i;s->m_dTrial=1;
      s->distance=sqrt(pow((g->adjlist[i].x-g->adjlist[j].x),2)+pow((g->adjlist[i].y-g
->adjlist[j].y),2));
      s->next=g->adjlist[j].firstedge;g->adjlist[j].firstedge=s;
   }
}
void BianLi(edgnodeinform inf[],adjgraph *g2)
{
  int i,k=0;edgenode *p;
  bool visited[iCityCount][iCityCount];
  for(i=0;i<iCityCount;i++)
     for(int j=0;j<iCityCount;j++)
         visited[i][j]=false;
   for(i=0;i<g2->n;i++)
   {
     p=g2->adjlist[i].firstedge;
     while(p!=NULL)
     {
       if(visited[g2->adjlist[i].vertex][p->adjvex])
        p=p->next;
       else
       {
          inf[k].i=g2->adjlist[i].vertex; inf[k].j=p->adjvex;
          inf[k].m_dTrial=p->m_dTrial;inf[k].distance=p->distance;
          k++;
          visited[g2->adjlist[i].vertex][p->adjvex]=true;
          visited[p->adjvex][g2->adjlist[i].vertex]=true;
          p=p->next;
       }
     }
   }
}
void Copygraph(adjgraph *g1,adjgraph *g2)
{
  int k;edgenode *s;g1->n=g2->n;g1->e=g2->e;
  for(k=0;k<g1->n;k++)
  {
    g1->adjlist[k].vertex=g2->adjlist[k].vertex;
```

```
    g1->adjlist[k].firstedge=NULL;
 }
 BianLi(inf,g2);
 for(k=0;k<g1->e;k++)
 {
   s=(edgenode *)malloc(sizeof(struct node));
   s->adjvex=inf[k].j;s->m_dTrial=inf[k].m_dTrial;
   s->distance=inf[k].distance;
   s->next=g1->adjlist[inf[k].i].firstedge;
   g1->adjlist[inf[k].i].firstedge=s;
   s=(edgenode *)malloc(sizeof(struct node));
   s->adjvex=inf[k].i;s->m_dTrial=inf[k].m_dTrial;
   s->distance=inf[k].distance;s->next=g1->adjlist[inf[k].j].firstedge;
   g1->adjlist[inf[k].j].firstedge=s;
   }
}
```

攻读硕士学位期间发表的论文及科研成果

周宏敏. 带约束条件的铁路运输网络车辆路径问题研究. 2007 年信息、电子与控制技术学术会议论文集.