



**本科毕业设计（论文）**

**巡检线路的排班调度模型及其仿真**

学 院 **计算机学院**

专 业 **网络工程**

年级班别  **2015级3班**

学 号  **3115005488**

学生姓名 **林钟涛**

指导教师 **刘冬宁**

2019 年 5 月

**摘要**

本论文研究题目来源于2017年全国大学生数学建模竞赛题D题，要求建立数学模型并用合适的算法解决巡检线路的规划问题。化工厂巡检是一个非常重要的工作，在合适的时间内巡检各个检查点能够有效地降低化工厂出现意外危险的概率，因此设计一个合理的巡检线路非常必要，要求用尽量少的人力资源达到巡检目的。

本文将基于该化工厂问题展开研究，经分析可知，各个巡检点的巡检时间和巡检周期已定，所以只能考虑通过减少在各个巡检点之间移动的时间来达到减少人力资源的目的。由此分析建立数学模型，根据原始的稀疏图利用弗洛里德算法求得各点之间的最短路径，建立无向带权完全图，将问题转化为寻找一条或多条线路使得遍历各点的路径之和最短，并使得各个路线的工作量均衡，可知这是一个旅行商问题。本文解决旅行商问题使用了Java语言编写遗传算法，求得近似最优解。经过实践分析，本文所建立的数学模型非常有效地解决了此类旅行商问题，规划了更优秀的巡检路线。

最终本文规划的巡检线路使用了更少的人力，采用了三班轮换和线路轮换的策略，任务更加均衡，最终可在13天内实现每个工人的任务量相对均衡。在第一问不休息排班中，本文规划的路线每班4人，每天只需12人即可完成巡检任务，采用线路轮换让每个员工任务量更均衡；在第二问可休息进餐排班中，本文规划的路线每天只需14人，其中由于进餐需要1至2名替班工人，早班2点到10点需4人，安排中班10点到18点需6人，晚班18点至次日2点4人；在第三问错时上班排班模型中，经分析是一个最小哈密顿回路问题，也可以使用旅行商问题的解决方法，在不休息排班中每天需12人，可休息进餐排班中每天也只需要12人，进餐时间安排在上班时间前后，省下了进餐时间。

**关键词：**旅行商问题，最小哈密顿回路，遗传算法，巡检线路，最短路径

**Abstract**

The research topic of this thesis comes from the 2017 National College Students Mathematical Modeling Competition, title D, which requires the establishment of mathematical models and the use of appropriate algorithms to solve the planning problems of the inspection lines. Chemical plant inspection is a very important task. patrol inspection points in a suitable time can effectively reduce the probability of accidents in chemical plants. Therefore, it is necessary to design a reasonable inspection route, requiring as little manpower as possible. Resources meet the purpose of inspection.

This paper will conduct research based on the chemical plant problem. According to the analysis, the inspection time and inspection cycle of each inspection point have been determined, so we can only consider reducing the human resources by reducing the time between each inspection point. the goal of. This analysis establishes a mathematical model, uses the Floridian algorithm to obtain the shortest path between points according to the original sparse graph, establishes a complete graph of undirected weights, and transforms the problem into finding one or more lines to traverse the points. The sum of the paths is the shortest, and the workload of each route is balanced. It is known that this is a traveling salesman problem. This paper solves the traveling salesman problem and writes the genetic algorithm in Java language to obtain the approximate optimal solution. After practice analysis, the mathematical model established in this paper is very effective in solving such traveler problems and planning a better inspection route.

In the end, the inspection route planned in this paper uses less manpower, adopts the strategy of three shifts and line rotation, and the task is more balanced. In the end, the workload of each worker can be relatively balanced within 13 days. In the first no-shift schedule, the planned route of this article is 4 people per class, only 12 people can complete the inspection task every day, using line rotation to make each employee's task more balanced; in the second question, rest can eat. In the shift, the

route planned for this article only needs 14 people per day, including 1 to 2 substitute workers for meals, 4 for early shifts from 2 to 10, and 6 for mid-shifts from 10 to 18, late. From 18 o'clock to 2 o'clock in the next day; In the third shift, the work shift model, the analysis is a minimum Hamiltonian loop problem, you can also use the traveler problem solution, 12 people a day in the rest schedule, can rest in the meal schedule every day Only 12 people are needed, and the meal time is arranged before and after the working hours, saving the meal time.

**Keywords:** Traveling salesman problem, Minimum Hamiltonian loop, Genetic Algorithm, Inspection line, Shortest path

**目录**

[1. 绪论 1](#_Toc32716)

[1.1 研究背景与意义 1](#_Toc28933)

[1.2 国内外研究现状 2](#_Toc26816)

[1.2.1 国内研究现状 2](#_Toc20874)

[1.2.2 国外研究现状 4](#_Toc20649)

[1.3 本文内容与组织架构 5](#_Toc27268)

[2. 相关基础介绍与准备工作 6](#_Toc19513)

[2.1 弗洛伊德算法 6](#_Toc31351)

[2.2 遗传算法 6](#_Toc4634)

[2.3 数学建模 7](#_Toc20175)

[2.4 Java编程 7](#_Toc3523)

[2.5 IntelliJ IDEA 8](#_Toc12767)

[3. 模型与解法 9](#_Toc8720)

[3.1 问题描述 9](#_Toc32161)

[3.2 问题分析 9](#_Toc24951)

[3.3 数学模型 9](#_Toc8581)

[3.3.1 模型假设 9](#_Toc28950)

[3.3.2 模型建立 10](#_Toc7141)

[3.3.3 模型求解 11](#_Toc13542)

[3.3.4 模型分析 12](#_Toc11210)

[3.3.5 模型校验 13](#_Toc5849)

[3.4 遗传算法求解模型 13](#_Toc26152)

[3.4.1 遗传算法基本流程 13](#_Toc1975)

[3.4.2 染色体编码 15](#_Toc29123)

[3.4.3 适应度函数 15](#_Toc4071)

[3.4.4 个体选择 16](#_Toc1871)

[3.4.5 双亲进行交叉 16](#_Toc14225)

[3.4.6 子代变异 17](#_Toc32655)

[3.4.7 种群评估 18](#_Toc2231)

[3.4.8 算法终止检查 19](#_Toc18309)

[4. 实验仿真与线路规划 20](#_Toc17211)

[4.1 算法运行结果 20](#_Toc4840)

[4.1.1. 固时上班不休息和可休息模式的最优解 20](#_Toc23018)

[4.1.2. 错时上班不休息和可休息模式的最优解 21](#_Toc10104)

[4.2 算法结果分析 22](#_Toc18689)

[4.2.1. 固时上班不休息工作模式的解决方案分析 22](#_Toc12279)

[4.2.2. 固时上班可休息模式的解决方案分析 24](#_Toc6608)

[4.2.3. 错时上班不休息和可休息模式的解决方案分析 25](#_Toc15301)

[4.3 线路规划与时间安排 26](#_Toc17498)

[4.3.1. 固时不休息工作模式线路规划与排班 26](#_Toc30829)

[4.3.2. 固时可休息进餐工作模式线路规划与排班 28](#_Toc26414)

[4.3.3. 错时上班模式线路规划与排班 30](#_Toc23728)

[5. 总结 34](#_Toc20169)

[参考文献 35](#_Toc30927)

[附录A 程序源码 36](#_Toc20798)

[附录B 巡检线路规划时间安排表 42](#_Toc32281)

# 绪论

## 研究背景与意义

本论文研究题目是2017年全国大学生数学建模竞赛题目D题。化工厂是一个比较危险的地方，为了减少事故发生的概率，安排人员定时巡检各个巡检点非常必要，可以有效降低事故发生率，提高化工厂安全生产水平。现有某个化工厂有26个巡检点，巡检周期和各个巡检点之间的连通图已在附件种给出，工人在调度中心XJ-0022上班接受任务后开始按照任务进行巡检，每个巡检点至少需要一名工人巡检，要求建立数学模型，设计出巡检路线，使每个巡检点能够按照要求完成巡检，并且耗费的人力资源尽量少，在固定时间不休息上班模式、可进餐可休息模式、错时上班模式三种情况下，分别求出每班最少需要的人数，给出每个巡检线路及其时间表。

巡检线路的规划问题，类似于车辆路径规划问题[1]、物流配送问题[2]，由于存在组合爆炸，我们人为规划的路线一般无法求得令人满意的结果，因此一般都会将原问题转化为旅行商问题。

旅行商问题[3]是一个NP困难问题，在巡检点过多的情况下使用传统的算法，比如穷举法，将无法在可允许的时间范围内求得最优解，而现实生活中我们可能并不需要使用最优解，我们只要一个合理的接近最优解的解。目前解决旅行商问题已经有多种算法，主要有两大类，一类是传统的算法，一类是启发式算法，例如遗传算法[4][5]、模拟退火算法[6]、蚁群算法[7]等。对旅行商问题的研究是非常有意义的，因为在很多场合，很多问题在本质上就是旅行商问题。

本文研究的化工厂巡检排班问题截止至目前，网络上已经有多种解决方案，经过研究分析，发现解决本问题的方案主要有以下几种：基于图形启发法的数学模型、基于最短路径的数学模型、基于最小生成树的数学模型、基于哈密顿圈分割[8]的数学模型、基于蚁群算法的数学模型等，其中基于哈密顿圈和基于蚁群算法的两种数学模型有相似共同的地方，都是将问题转化为旅行商问题，而其他的数学模型则较为简单。根据这几种解决方案在最好的情况下，在固定时间不休息工作模式下每班最少需要5人，一天需要15人；在可休息可进餐工作模式下每天至少需要16人，因为进餐时间需要一名工人代替工作；在错时上班模式下，针对问题1，每天最少只需要12人，针对问题2，每天至少需要13人。

在研究多种解决方案时，发现现存解决方案都有一个共性，就是数学模型要么过于复杂，要么过于简单，解决方法和模型的关联程度较低，更多的方案是直接采用图形启发法设计巡检线路，由于巡检点数较少，采用此方法也无可厚非，但是如果有另一个化工厂也需要规划路径，那么，但是出于严谨的角度我们应该增强数学模型的普适性和解法的强逻辑性，让解法每一步都有理可循。因此，建立具有普适性的数学模型，设计更合理的求解方法非常必要。

在本文中，我将采用遗传算法和图形启发法相结合的方式规划出更加节省人力的巡检线路，既满足了化工厂安全的需求，又满足了节约人力资源的目的。通过参考文献，取各家之长结合自己的想法重新建立了数学模型，下面分析问题。该化工厂巡检线路规划中最核心的要求是使用尽量少的人力资源完成巡检任务，各个巡检点的带权连通图和各点巡检时间已给出，所有巡检点的最小巡检周期为35分钟，该类巡检点共有18个，其余8个巡检点周期均大于35分钟。由此分析可知，每条巡检线路的周期不能大于35分钟才能满足巡检要求。因此只要每个工人在35分钟周期内巡检更多的巡检点就可以了，而工人线路巡检消耗的时间有路程时间和节点巡检时间，节点巡检时间是一定不能省的，因此只能考虑节省路程时间。综上分析，我们只要能节省巡检26个节点路程消耗的时间就能使用更少的工人完成巡检任务，分析至此，可知原问题已经转化为旅行商问题或多人旅行商问题，需要计算所有巡检点两两之间的最短距离，这个任务可以使用弗洛伊德算法完成。最后，通过遗传算法编程解决旅行商问题，再计算每条巡检线路的巡检时间，结合图论启发的方式做轻微调整。

## 国内外研究现状

### 国内研究现状

目前本文所研究的题目已有了多种解决方案，下面我将分析各种解决方案的思路和存在的问题，对多种解决方案的理论依据在国内外的研究现状做一个综述，并提出自己的更优秀的解决方案。

谢晓敏、王正的《化工厂安全巡检线路的排班研究》[9]一文中采用了弗洛伊德算法找出各个巡检点之间的最短路径，并按照巡检周期是否为35分钟将26个巡检点划分为两类，一类巡检周期35分钟的有18个巡检点，另一类是巡检周期大于35分钟的8个特殊巡检点，对两类巡检点分别采用蚁群算法求出最小哈密顿回路，其中第一类巡检点又划分为4个区域，最终规划的线路的路程消耗时间至少为105分钟。文中对问题一规划出的巡检线路每天需要15人；对问题二规划的线路是建立在第一问基础之上，将周期大于35的穿插到第一问线路中，需要16人或17人每天；对问题三，原有模型已不适合，重新规划后固定时间上班排班需要14人，可休息进餐排班需要15人。

《化工厂安全巡检线路的排班研究》一文中建立的模型采用了区域划分、分而治之的思想，对不同区域采用蚁群算法求解，虽然解决了问题，但是建立的模型和弗洛伊德算法、蚁群算法之间的关联程度有所欠缺，最后需要人为修正路径，因为该模型划分的不同区域之间路径重合过多，所以在路程上消耗的时间也就相对较多。

在《巡检线路排班优化模型》[10]一文中，作者采用了基于最小生成树建立多旅行商模型，利用弗洛伊德算法求最短路径，使用MATLAB工具求解模型，最后人为规划线路。对问题一规划的路线需要15人每天；对问题二规划的线路需要16人，替班工人在12点和6点先后顶替其他工人的工作，让其他工人进餐；对问题三采用错时上班，固定时间上班排班需要12人每天，可休息可进餐排班需要多一名替班工人，需13人每天。

该文中建立的模型和解法相对灵活，较好的解决了巡检线路的规划问题，同时巡检人员的数量相对较少。但是解法出现误差的可能性大，因为人为划分线路，数据的读入可能会容易出现不精确的情况。

在《“巡检线路的排班”问题解析》[11]文中，作者首先采用了LINGO程序得到了精确的路程耗时68分钟和检查耗时67分钟，共135分钟，经计算得知至少需要4个工人，作者考虑增加一名工人，由此划分5条线路。作者采用TSP建立模型，由启发式方法根据TSP图形划分了5条路线。对第一问需要工人15人每天；第二问需要工人16人；第三问中，固时上班需要12人，可休息进餐需要13人每天。

该文中作者对排班调度分析较为详细，基于TSP建立的图形启发式模型简单有效，问题分析到位。在解决问题三时，作者提出7小时换班制，会出现每天换班时间不一样的情况，提前了三小时，这里可以优化。

旅行商问题具有非常久的历史了，最早出现类似的问题是1759年欧拉图提出的骑士环游问题。目前解决旅行商问题主要有两大类算法，一类是精确算法，如动态规划法、分支定界法等；一类是近似算法，如遗传算法、蚁群算法、模拟退火算法、神经网络等。

综合多方文献资料，现对各种解法做总结。多数解法都是围绕多旅行商问题[12]划分为单旅行商问题建立模型，采用了不同的解法，因为巡检节点数只有26个，大多数使用了图论启发法，有的直接基于最短路径、最小生成树划分路径。多数建立的模型复杂，解法却比较简单。

本文将基于图论法建立TSP模型，使用遗传算法解决旅行商问题，基于任务均衡划分每条线路的巡检节点数量，期望使用更少的人力资源完成巡检任务。

### 国外研究现状

旅行商问题又称为旅行推销员问题，也称为货郎担问题，最早起源于1959年，由Dantzig等人提出。该问题可表述为图论里面的哈密顿回路问题，即给出一个带权完全图，找出一条权值之和最小的哈密顿回路。该问题已经被证明是一个NP难问题，因为要求权值之和最小的哈密顿回路，需要找出所有哈密顿回路分别计算权值之和，可知要对图中的所有的顶点做全排列，随着顶点数的增加，会产生组合爆炸，计算机将无法在可允许时间内得到结果。在上世纪50年代和60年代，国外一些生物学家尝试将生物进化理论应用于模拟计算，在70年代和80年代，Holland最先提出并发展了遗传算法的概念，将达尔文的生物进化理论转化为计算机语言，建立了计算机0和1的序列与染色体的联系，用计算机模拟实现了生物进化的选择、变异、交叉、优胜劣汰等理论。

在2010年，英国研究发现了蜜蜂这种生物能够轻而易举地解决旅行商问题的能力，蜜蜂每天都要在各个花丛之间采蜜，这相当于一个带权网络。英国学者的这项研究发现很快吸引了大批的数学研究者的注意，在此之前有很多国内外学者都曾研究过旅行商问题，取得了令人瞩目的成就。蜜蜂解决这个问题表现出的非凡的能力，如果能够理解蜜蜂解决这个问题的方法，那么对我们人类的生产和生活将会有非常大的帮助。因为旅行商问题应用的领域非常广阔，生活中很大部分的规划问题都属于旅行商问题，例如道路交通规划、物流配送、车辆路径、电路板设计、电力线路巡检[13]等等。

美国在2006年的某次航空任务中需要设计一个天线，以满足本次航空任务要求，NASA（美国航空航天局）编写了一个遗传算法，利用遗传算法的进化本质，进化出了满足所有要求的天线设计，比如信号质量、天线重量、尺寸等要求。

## 本文内容与组织架构

本论文为了解决化工厂巡检线路的规划问题，参考了多方文献资料，研究了旅行商问题的应用场景和解决方法，研究了遗传算法在解决旅行商问题的有效性，以遗传算法为核心建立数学模型并求解、仿真测验。

下文介绍论文结构及其内容：

第一章：绪论，介绍选题背景及其研究意义，阐述国内外研究现状，以及介绍论文组织结构。

第二章：相关基础介绍，主要介绍本文使用的算法、数学建模等基础概念，详细介绍遗传算法的编写要素等。

第三章：建立数学模型，描述化工厂问题，并为化工厂问题建立数学模型。

第四章：算法实验及其仿真，完成算法并测试仿真，测试结果的准确性，规划出线路并给出巡检时间表。

第五章：总结全文。

# 相关基础介绍与准备工作

## 弗洛伊德算法

Floyd算法是由弗洛伊德创造的求多源点最短路径的算法，适用于稠密图，类似的求取最短路径的算法有迪杰斯塔拉算法，单源点，适用于稀疏图。

在本文中，原网络带权图是一个稀疏图，通过分析将问题转化为旅行商问题，需要求取各个巡检点之间的最短路径，假设工人从一个巡检点到另一个巡检点都是走最短路径。

在效率上，求取所有节点之间的最短路径这虽然可以通过n次调用迪杰斯塔拉算法，但是效率不如弗洛伊德算法。弗洛伊德算法通过三层循环，采用松弛技术，代码简单，代码核心在状态转移方程:

distance[i,j]=min{distance[i,k]+distance[k,j],distance[i,j]}，

本文中将编写Java代码实现弗洛伊德算法求最短路径。

## 遗传算法

遗传算法的研究于上世纪九十年代迅速兴盛起来，该算法是由达尔文的自然选择进化论得到启发创造出来的，模拟了自然界中生物选择进化的过程，在算法中表现为不断进化寻找最优解的过程。

要理解遗传算法我们需要先理解达尔文的自然选择学说里面的相关基础概念，比如种群、个体、染色体、基因、表现型、基因型、选择、变异等。生物中的种群是由染色体类型相同的许许多多个体组成，每个个体都至少有一条染色体，染色体由多个基因组成，性状表现由基因决定，生物在产生后代时会发生基因的重组，变异等，更加适应环境的个体将会有更大的生存优势，产生后代的几率也更大，变现为优胜劣汰的大趋势，不断进化产生更加适应环境的个体。

遗传算法中许多概念与达尔文进化学说中的概念相同，算法中一个解集包含着许许多多的解，相当于种群中包含着许多个体，每个解就是一个个体，一个个体至少含有一条染色体。在本文巡检线路研究中，染色体就是一组巡检节点遍历序列，每个巡检节点相当于一个基因，这条染色体决定了该个体的表现型，即遍历巡检节点的顺序，以及该个体的适应度也由染色体的基因序列计算得到，后文将详细介绍如何计算。

## 数学建模

数学建模是对实际问题进行抽象建立数学模型，将实际问题符号化模型化，能够模拟抽象客观的现象或规律，并通过对模型进行求解达到解决实际问题的目的，是用数学语言描述实际问题的过程。数学建模有助于提高建模者的应用数学思维，提高解决实际问题的能力。

数学建模国外兴起于上世纪60年代和70年代，到了80年代，美国首先举办了大学生数学建模竞赛，我国在90年代首次举办并迅速发展，参赛人数逐年增加。随着计算机技术的迅速发展，数学建模抽象实际问题并通过计算机工具对模型进行求解，对问题的求解更加精确。建模过程需要使用多门学科知识，特别是数学这门学科，抽象、严密，应用非常广阔，计算机科学也是数学应用的一种形式。

数学建模的过程一般包括以下几个阶段：模型准备阶段、模型假设阶段、模型建立阶段、模型求解阶段、模型分析阶段、模型校验阶段。

本文中分析化工厂巡检线路建立的数学模型是基于旅行商问题的，后文将详细分析建立过程。

## Java编程

Java语言是一门面向对象的高级编程语言，没有C++语言的指针、多继承等复杂难懂的概念，一般不存在内存泄漏的风险。Java在上世纪90年代兴起至今不衰，通过Java虚拟机，Java语言实现了一次编译到处运行的理念。

Java语言具有很好的可移植性，跨平台，能在Windows环境和Linux环境等操作系统运行；具有安全性，由于没有指针，可以防止恶意修改内存指向；Java具有高性能的特点，由于是先编译后解释执行，为了提高执行效率，Java拥有将字节码转化为机器代码的程序……。

本文将化工厂巡检问题转化为旅行商问题，解决旅行商问题采用了遗传算法，实现遗传算法和求取各个巡检点之间的最短路径都采用了Java编程语言，程序代码在附录中给出。

## IntelliJ IDEA

IDEA是Java集成开发环境，由于它具有代码提示、支持git等版本工具、具有许多优秀的智能插件等优点，被业界公认为最好的Java开发工具之一。早期的Java集成开发环境如Eclipse、MyEclipse等，与IDEA比较各有特点，网络上对这两种IDE分两大派，争论各持己见，人云亦云，本人接触最多的Java IDE就是IDEA，使用起来比较习惯。

# 模型与解法

## 问题描述

题目见附录，来源于2017年全国大学生数学建模竞赛题目D题。大意为某化工厂有26个巡检点需要巡检，巡检时间、巡检周期和各巡检点之间的连通关系和路程耗时由附件已知，现需要规划巡检路线，要求使用尽量少的人力资源并按质按量完成巡检任务，分别给出在固时不休息不进餐三班倒模式、可休息可进餐三班倒模式、错失上班三班倒模式下巡检人员安排时间表。

## 问题分析

分析原问题：对26个巡检点进行线路规划，26个巡检点总共的巡检耗时68分钟，为了精确计算巡检人员的下限，可知应该尽量减少巡检线路的路程耗时，即遍历26个巡检点，耗费时间最少，由此分析得知此问题是旅行商问题。经过计算，遍历所有巡检点路程耗时最少为67分钟，加上巡检点耗时68分钟，一共135分钟，巡检点的最小巡检周期为35分钟，135/35=4，因此至少需要4人才能完成巡检任务。

下面尝试将连通网络图的26个巡检点划分为4个区域，分别判断4个区域巡检时间是否满足要求，是否能做微调整使其满足巡检任务要求。在以下建立的模型中使用Java编写遗传算法求解得出线路划分，多次运行该算法程序尝试求得最优解，并对线路划分进行判断是否满足35分钟周期的约束，如果不满足应该再尝试是否能做调整。

## 数学模型

### 模型假设

1. 假设巡检人员严格按照巡检任务安排执行巡检工作
2. 假设巡检人员不会出现旷工、迟到、早退等现象，按时按量完成工作
3. 假设巡检人员一定按照附件中的时间行走
4. 假设化工厂巡检点在巡检周期内一定不会出现安全问题
5. 假设天气等外界因素不会影响巡检任务

### 模型建立

1. 针对化工厂巡检线路安排问题一，固定时间上班，不考虑工人的休息时间，每天三班倒。可建立多人旅行商问题模型，将26个巡检点划分为若干区域，每个区域由一名工人负责巡检，每个区域都是一个单独的旅行商问题，将多人旅行商[14]划分为单人旅行商问题。考虑到每名工人的工作均衡，区域划分应该尽量均衡。由前文的分析可知，要完成本化工厂巡检至少需要的巡检人员数是4人，因此尝试将区域划分为4个，每个区域为5至7个巡检点。如果4人不足以完成巡检要求，则可以逐渐增加巡检人员数量。
2. 针对问题二，固定时间上班可休息可进餐，每天三班倒，与问题一类似，只是需要考虑休息时间和进餐时间，由前面分析知道，本化工厂有8个周期异常的巡检点，周期长于35分钟，因此可以考虑间隔周期检查，利用异常周期巡检点休息。经分析，进餐时间无法仅仅依靠异常巡检点挤出的时间，应考虑增加员工数量，进餐时间由替班工人替班巡检。
3. 针对问题三，错时上班，即工人可以在不同时间上班，在错时上班模式下，模型将可以改进为单人旅行商问题，构造一条最小哈密顿圈[15]，每个工人都是按照哈密顿圈路线巡检，在第一名工人从第一个巡检点出发开始巡检后的35分钟，第二名工人接着上班从第一个巡检点出发，以此类推，最后所有工人按照哈密顿回路进行巡检。

模型变量：

Z：巡检员工数量

MinZn：所有巡检线路路程耗时之和

D(i,j)：巡检节点i到巡检节点j的最小耗时

ak：表示某个化工厂巡检点标号k

AREA(i)：化工厂部分巡检点的集合，集合元素为ak

模型建立：

AREA(i)={a1,a2,…,an}， (3.1)

i=1,2,…,Z；ak∈{1,2,3,…,26}；AREA(1)∩AREA(2)∩…∩ARAE(N)=Ф

F(k)=， (3.2)

k=1,2,…,Z；a1,an∈AREA(k)；

MinZn=，i∈1,2,…,Z (3.3)

H(k)=F(i)/MinZn，i,k∈1,2,…,Z (3.4)

公式（3.1）表示将26个巡检点划分为Z个区域，每个区域由一个工人负责巡检；公式（3.2）表示对划分的每个区域计算路程耗时，每条线路都是一个哈密顿回路；公式（3.3）计算了所有区域路程耗时之和，即每条哈密顿回路路程耗时之和，问题最后转化为MinZn取得最小值；公式3.4计算了各条线路的路程耗时与各线路总路程耗时之和的比例关系，可以通过比较此值判断工作均衡程度。

针对化工厂巡检问题一和问题二，固定时间上班不休息不进餐的三班倒模式和固定时间上班可休息可进餐的三班倒模式，可以转化为多人旅行商问题，由之前的分析可知最少需要4人，那么尝试将巡检点划分成4个部分，若不考虑工作均衡的问题，由多人旅行商的解决方案划分的某些线路可能会无法完成巡检任务，有些路线每周期总工作时间超出35分钟，有些路线工作量则太少，导致有些超出预期的线路需要增加一名工人才能完成巡检要求，这显然不是很符合我们的目的，而网络上有些解决方案也确实是这样的。在此我们考虑工作的均衡，将每条线路的节点数量控制在5-7个，这样就能解决工作均衡的问题又能让每条线路巡检时间趋向于饱和，最后可以根据网络图适当的调整，就可以设计出更加优秀的路线。

### 模型求解

1. 针对问题一，根据巡检人员下限4人，使用Java编程语言编写遗传算法，将26个巡检点划分为4个区域，考虑每名工人的工作均衡，每个区域巡检点数分别为

5、7、7、7，

或6、6、7、7，

分别对这两类划分方式编写适应度函数，经过实践分析，第一种划分方式具有更好的划分结果，算法进化出的线路基本满足35分钟的巡检周期约束，可以对巡检线路做微调整使其满足巡检要求，于是决定采用第一种划分方式求解模型。下一章节将详细介绍如何根据这类划分方式编写遗传算法解决问题一[16][17]。

1. 针对问题二，本问题是在问题一的基础上增加休息进餐的约束，本质上与问题一区别不大，因此可以使用问题一的解决方案，对每条线路查看是否能挤出时间满足此条件。首先是休息时间，每两个小时需要休息5至10分钟，经过观察，有8个巡检点周期大于35分钟，其中7个巡检点周期大于70分钟，因此如果这些巡检点能够分部到每条巡检路线中，就能够利用这些巡检周期大于70分钟的巡检点，每两个周期检查一次，就可以有休息时间了。其次是进餐时间，12点和6点需要进餐，每个工人进餐时间需要30分钟，使用异常周期巡检点时间显然不够，因此只能考虑增加至少一名工人，在吃饭时间顶替吃饭工人完成巡检任务。
2. 针对问题三，错时上班模式，在固定上班模式中，通过划分区域的旅行商模型，经过算法仿真得知，路程上耗费的时间比单旅行商消耗时间多，但是通过微调整线路，时间上的差别进步一步缩小了。本模型解法与问题一解法有相同的地方，也是通过解决旅行商问题，但是不用划分区域，只要构造一条最小哈密顿圈就可以，因为工人错时上班。
3. 总结模型解法：可见本文建立的模型非常适用于该化工厂的三个问题，解法也相对统一，只是单人或多人旅行商问题的区别，以及算法上的略微差异，确切的说是根据染色体计算适应度值时不同。因此遗传算法非常适合解决旅行商问题模型。

### 模型分析

本模型建立过程基于分步分析、问题转化的方式，一步步将化工厂巡检线路的排班问题转化为旅行商问题，建立起原问题与旅行商问题的联系，并且建立的TSP模型完全适用于原问题的三个小问题。问题转化过程严谨，将原有的稀疏图使用弗洛伊德算法求得所有巡检点之间的最短路径转化为完全图，再转化为旅行商问题。

目前针对本化工厂排班问题已经有解决方案，也有相应的模型，比如基于最短路径的模型、基于最小生成树的模型等，这类模型相对简单，适用于较少巡检点的情况，缺少可扩展性，换了一个化工厂可能就不适用了。

本文的TSP模型与基于最小生成树、最短路径构建的模型具有以下优点：

1. 模型适用范围更广，可适用于交通规划、车辆路径规划、物流配送等优化问题；
2. 模型求解方式多样，可使用遗传算法、蚁群算法等进化算法求解，可使用数学工具MATLAB等；
3. 模型可扩展性强，当化工厂巡检点改变时，模型不变，只需稍微改变算法；
4. 模型与解法联系紧密，线路规划不用人为划分，可以使用算法进化巡检线路；
5. 模型适用于更多巡检点，求得的解更精确

当然，本TSP模型也有缺点，模型和解法都比最小生成树模型复杂。模型建立过程中通过问题转化将巡检点的巡检周期、巡检时间等都暂时忽略，不然分析之后就是带有时间戳的旅行商问题，问题太过于复杂，因此简单化考虑，在最后线路再分析是否满足巡检周期时间要求。

### 模型校验

通过模型分析结果可知，模型的适用性比较广，求解方法多样，这是优点但同时也是缺点，模型的结果比较依赖解法的结果，本文通过遗传算法进化出的解是比较符合要求的，但是也不能保证就是最优的解。总体上看，模型是与实际问题吻合的，能够在合理的时间内求得合适的解。

## 遗传算法求解模型

算法整体思想是：先根据弗洛伊德算法计算得到26个巡检点之间的最短路径，以二维矩阵方式存储，再使用遗传算法求解，遗传算法设计详细内容见后文。

### 遗传算法基本流程

遗传算法基本流程如下图所示：



图3.1 遗传算法流程图

本文算法程序结构由Floyed类、GeneticAlgorithm类、Individual类、Node类、Path类、Population类、Main类、OriginalDistance.txt、shortest.txt构成。

Floyed类：根据原始的稀疏图OriginalDistance.txt，使用弗洛伊德算法生成26个巡检点之间的最短路径生成二维矩阵，并将二维矩阵保存到shortest.txt文件中。

GeneticAlgorithm类：遗传算法的核心类，里面包含种群初始化函数、判断终止函数、个体和种群适应度评估函数、选择个体函数、交叉函数、变异函数等，都是遗传算法的核心操作。

Individual类：表示一个个体，内含一条染色体，即巡检点访问。

Node类：包含巡检点信息，如巡检点名称。

Path类：表示一条路径，内含巡检节点序列信息。

Population类：表示一个种群，内含多个个体，即解集。

Main类：启动类，创建并启动遗传算法。

### 染色体编码

根据前文遗传算法介绍可知，算法的关键一环在于染色体编码。目前染色体编码有二进制编码、格雷编码、浮点数编码、序列编码等。

针对本文研究的化工厂排班调度问题，染色体编码的设计可以根据巡检线路的巡检点序列号进行编码，这种编码非常符合旅行商问题，容易理解，简单又合理，一个染色体代表着一种巡检序列，一个编码代表一个巡检点，即基因，例如：

1 2 3 4 5 6 7……24 25 26，

表示从1号巡检点开始巡检，顺序巡检2号、3号……26号巡检点最后回到1号巡检点，从一个巡检点到另一个巡检点走的是最短路径。

每个个体都含有一条染色体，染色体的基因序列表示访问巡检点的顺序。

本文编写的算法代码中使用数组存储染色体基因，一个数组代表一条染色体，数组长度为巡检点总个数。在Individual类中，使用int类型数组表示一个染色体，数组存储的内容是化工厂巡检点的编号对应的数组下标。

在本文中，针对化工厂巡检问题一和问题二，通过巡检点划分为7,7,7,5的方式，将染色体划分为4个小段，当然是逻辑上的划分，只要在计算路程耗时的Path类中的getTimeCost()函数中，对染色体做划分，分别计算4个哈密顿圈的路程耗时，最后求和。

### 适应度函数

每个个体都有适应度值，用来衡量该个体在群体中的适应程度，本文中直接将线路耗时的倒数作为适应度函数值，路程耗时越小，适应值越大。

Fitness=1/time\_cost （4.1）

本文算法设计在GeneticAlgorithm类中有individualFitness()方法和populationFitness()方法，方法中计算个体和种群的平均适应度，采用的是公式（4.1）的计算方法，其中time\_cost是该个体解的路程消耗时间之和，方法中调用了Path类中的路程时间耗时的计算方法getTimeCost()，在Path类中不同的计算方式主要体现在这个方法的不同，如针对问题一，建立的模型求解需要划分区域，4个区域的巡检点数分为7,7,7,5，则针对每个区域计算每个哈密顿回路的时间耗时，再对4个哈密顿回路耗时求和，得出总体时间耗时。

### 个体选择

本文中算法采用的选择算子是锦标赛选择，从种群中选择若干个个体参加锦标赛，最后锦标赛获胜的个体被选择。在算法设计中，锦标赛的规模大小会影响算法的执行效率，由于每次选择亲代时都会进行锦标赛，如果锦标规模太大，则相应的算法执行时间会增加，如果锦标赛规模太小，则算法的收敛速度太慢，求得最优解的代数会增加，即算法效率也会变低。因为锦标赛本质是在随机挑选若干候选解中的最优解，即选择适应度高的个体，较大的锦标赛规模会让算法在选择过程中效率变低，这很容易理解；较小的锦标赛规模虽然在选择过程中算法效率变高，但是意味着适应度高的个体作为亲代的可能性降低了，因此得到最优解的代数会增加。不过，实践证明锦标赛规模有个相对合适的值，既能考虑到选择算子的效率，又能提高遗传算法算法的收敛速度，这个较为合适的值需要针对不同的问题不断实践才能得出，经实践，本文中锦标赛规模较为合适的值为3至5人。

### 双亲进行交叉

本文中交叉算子位于GeneticAlgorithm类中的crossoverPopulation()方法，交叉算子在前文基础介绍中已有介绍，适用于本文算法的交叉算子是顺序交叉。因为旅行商问题中每个巡检点不能重复访问，而且巡检节点的顺序很重要，不同的顺序个体的适应度值差别较大，所以亲代交叉尽量选择改变量小的算子。

交叉过程例子如下：



图3.2 交叉过程图

首先通过选择算子选出两个亲代，子代直接从亲代1获取一部分基因，如

图3.3 基因片段图

然后根据亲代2中的基因顺序，将剩余的基因添加到子代中，本文中算法从亲代2的8开始依次检查8,6,2,5,4,7,1,3是否存在于子代中，如果存在则忽略，如果不存在则顺序添加到子代染色体中。

在本算法中，为了保护种群中适应度较强的个体，采用了精英主义的思想，即将种群中若干适应度最高的个体直接加入到后代中，避免因为交叉而失去优秀的解，加快算法的收敛速度。精英个数与锦标赛的规模具有相似的地方，精英的个数越大，算法的收敛速度就更快，不过需要注意的是，精英个数太大，种群规模一定的情况下，种群中因为交叉产生的个体就相对减少，即种群的多样性减少，最终可能会让遗传算法陷入局部最优解的局面。因此需要设置一个合理的精英个数，既能加快遗传算法收敛，又能避免收敛过快陷入局部最优解的尴尬局面。当然，为了避免陷入局部最优解的局面，还有一些方法，比如增加个体变异的概念。

### 子代变异

合理的变异概率能增加种群的基因多样性，有效避免算法陷入局部最优解的局面。本文中遗传算法设计的变异算子是交换变异，因为旅行商问题的独特性，各个巡检点不能重复出现在染色体中，即基因在一条染色体中是唯一的，因此，交换变异能很好地解决这个问题。

交换过程例子如下：



图3.3 交换变异图

具体做法是：精英个体不变异，针对需要变异的子代，对每个基因根据变异概率随机跟另一个基因做交换。由此可见，该变异方式较为简单直观，算法实现也简单明了，因为交换变异得跨度可能比较大，产生的基因更具有多样性，不过，大部分变异应该都是不利变异，这与现实世界中生物变异一致。

基因变异有利于帮助遗传算法探索解集附近的解，如果变异是有利的，则会表现在该个体适应度变高，产生一个新的优解。最优解和局部最优解之间，一般会存在介于中间的解集，在算法陷入局部最优解时，变异能够帮助我们探索局部最优解附近的解集，有一定概率会变异出最优解附近的解集。在理论上，只要算法运行的代数足够长，变异就会产生比局部最优解更好的解。因此变异能够帮助算法摆脱局部最优解的困局，实现多样性。

### 种群评估

本文中评估种群适应度是基于个体适应度的，对种群中的每个个体计算适应度值并求和，然后与种群个体数求比值。

populationFitness=sumFitness/poppulationSize

个体的适应度已由前面分析，是根据路程耗时取倒数计算得到的，种群适应度与个体适应度联系紧密。评估种群主要看种群适应度，这与种群中的每个个体相关。在算法程序的GeneticAlgorithm类中有populationFitness()函数，计算种群的适应度。

### 算法终止检查

遗传算法终止检查可以有多种方式，本文中采用的方式是最大代数终止，即选定最大代数，当算法达到最大代数就停止。当然，除了最大代数终止，如果你知道算法的最优解，设置终止条件是找到最优解就停止也是可以的。

# 实验仿真与线路规划

## 算法运行结果

### 固时上班不休息和可休息模式的最优解

针对化工厂巡检线路规划问题一和问题二，属于多人旅行商问题，通过区域划分的遗传算法设计的线路，经过多次实践，许多与最优解接近的在此我并不一一列举了，比如某个解与最优解相比，线路之中两个巡检点位置不同导致的路程耗时大了一点点。

算法最后进化出的最优解为：

线路一：22->21->4->23->24->20->19

线路二：2->1->3->5->7->6->14

线路三：10->11->13->16->18->15->12

线路四：9->25->26->17->8

线路区域划分如下图：



图4.1 旅行商线路划分图

四条线路巡检路程耗时分别为18分钟、16分钟、19分钟、16分钟，一共69分钟；若加上巡检点耗时，各线路耗时分别为：35分钟、34分钟、38分钟、29分钟，一共136分钟；

由此可见，线路一、线路二、线路四都是满足35分钟的最小巡检周期要求的，而线路三则相对特殊，38分钟的周期，与35分钟为周期只差了3分钟。后文将分析如何解决这个小问题。

### 错时上班不休息和可休息模式的最优解

针对问题三错时上班模型，遗传算法的设计与问题一和问题二基本一样，只是在Path类中的getTimeCost()方法不同，因为不用对染色体进行划分，是完完全全的旅行商问题。经过多次运行，最后得到的进化后的最优解为：

22->20->19->21->4->2->1->3->5->7->6->14->10->11->13->16->18->15->12->26->25-> 17->8->9->24->23->22

路程共耗时67分钟，巡检点共耗时68分钟，一共135分钟。

线路如下图：



图4.2 最短哈密顿回路图

## 算法结果分析

上一小节我们已经通过算法求解得到了初步的路线规划，本节将分析算法得到的线路是否满足我们的巡检要求。分别分析三个问题的解决方案。

### 固时上班不休息工作模式的解决方案分析

针对问题一，不考虑员工休息情况下，线路一、线路二、线路四均满足35分钟为周期的隐性约束，只有线路三38分钟，比35分钟超出了3分钟，如果因为这多出的3分钟而增加一名巡检工人的话，显然是有点可惜的。对此，我最先想到的是能否利用线路三中的异常巡检点做调整。

经观察，10号巡检点巡检周期为120分钟，巡检时间需2分钟，13号巡检点巡检周期是80分钟，巡检时间需5分钟。由于两个异常巡检点周期均大于70分钟，如果能够交替周期检查10号巡检点和13号巡检点，看能否挤出3分钟时间，如果能则方案可行，否则不可行。可惜的是，10号巡检点巡检时间是2分钟，小于3分钟，因此周期至少36分钟，正常巡检周期的巡检点将会超过1分钟的非安全期，如果不严格按照巡检时间的话，此方案还是可行的，但是如果严格的话，此方案不可行。

继续分析，线路四的工作时间为29分钟，线路三的工作时间为38分钟，线路四有6分钟的冗余时间，线路三则过量3分钟，能否考虑两者平衡？该如何平衡？经过长时间的思考和分析，考虑能否将线路三和线路四合并为一条线路，走最短哈密顿回路。

如下图：



图4.3 线路三、四合并的哈密顿回路图

将线路三和线路四合并为一条哈密顿回路，巡检时间一共为71分钟，两人巡检这一条线路，如果每个巡检点都按照35分钟的周期巡检，则又会过量1分钟。不过好在这条线路的异常巡检点中有个17号巡检点，巡检周期为480分钟，即只需要8小时巡检一次，因此规定在这条线路巡检的工人上班第一时间去检查17号，之后巡检时便跳过这一个巡检点，这样算来巡检一周需要69分钟，两人在这条线路上巡检就满足巡检要求了。

因此，在不休息工作模式下，每班次只需要4人即可完成巡检要求，每天三班，共需要12人。

现在分析每个工人工作量的均衡，线路一、线路二分别是35分钟、34分钟为周期，8个小时为一班，可以巡检13至14趟，因此线路一和线路二每班次工作时间最多相差14分钟；线路三和线路四是合并的，两人的工作量一样，两趟69分钟，每班次可巡检7圈，因此与线路一工作时间最多相差7分钟。

可见每名员工的工作量基本持平，若要考虑工作量的绝对均衡，工人上班可以采用线路轮换、三班倒的方式，即可在一定时间内实现绝对的均衡。

详细排班安排见附录。

### 固时上班可休息模式的解决方案分析

问题二是要在问题一基础上解决员工休息和吃饭时间，下面分析每条线路是否满足休息时间。

线路一：该线路中21号巡检点巡检周期为80分钟，因此可以在间隔一个周期巡检一次，从中挤出休息时间。但是21号巡检点位于线路一的必经节点上，只能利用巡检时间3分钟做休息，如果严格要休息5分钟以上，则必须考虑与线路二做整合。利用线路二中的5号和7号巡检点的时间，去检查线路一的4号节点，就可以为线路一提供11分钟的休息时间。

线路二：该线路中2号巡检点巡检周期为50分钟，小于两周期70分钟，直接把它当成正常巡检点考虑，而5号、7号巡检点巡检周期分别为720分钟、80分钟，都大于70分钟，因此可以从中挤出休息时间。

线路三和线路四合并：13号巡检点巡检周期80分钟，巡检时间是5分钟，可以利用5分钟时间安排休息，此外还有10号、25号巡检点巡检周期都大于70分钟。

分析至此，休息时间可安排。进餐时间是12点和6点，每人每次30分钟，只能考虑增加至少一名替班工人，在进餐时间替班。考虑到进餐时间，每天应增加两名工人，一名负责替班线路一和线路二，一名负责替班线路三和线路四，进餐时间就可以控制在1小时内完成。如果只增加一名替班工人，则4人的进餐时间一共是两小时，若将替班人员的进餐时间考两次进去则进餐时间是两个半钟，时间跨度太大，因此两个替班工人是比较合理的，进餐时间可以控制在一个小时。

因此，在可休息可吃饭工作模式下，可做如下安排：

早班从2点开始至10点，巡检人数需要4人；中班从10点开始至18点，需要增加一至两名工人替班工人，共5人或6人；晚班从18点开始至次日2点，需要4人。如此安排可以减少由于12点和下午6点进餐时的替班工人数量。每天需要13人或14人，考虑到进餐时间跨度问题，最好使用两名替班工人，因此每天需14人。

各班次详细时间安排见附录。

### 错时上班不休息和可休息模式的解决方案分析

问题三是在错时上班模式下，由前面的分析和计算可知，建立的最小哈密顿圈共耗时135分钟。

1. 重新考虑不休息工作模式

在最小哈密顿圈路线中，第一名工人从8点开始上班并检查22号巡检点，根据最小哈密顿圈顺序检查20、19、21、4、2、1、3等巡检点；在8：35时，第二名工人上班并检查22号巡检点，并根据最小哈密顿圈顺序检查20、19、21、4、2、1等巡检点；第三名工人在9：10上班并检查22号巡检点……；第四名工人在9:45上班并检查22号巡检点……在10:15时，第一名工人回到22号巡检点，完成一个回路的检查，继续顺序检查。

因此对问题一，错时上班每班次需要的人数是4人，每天需要12人，需要工人数量与之前的固定上班时间一样。

1. 重新考虑可休息工作模式

在之前的基础上，需要考虑工人的休息和进餐时间。

首先是休息时间，由之前的分析可知，每次完成一圈巡检，工人有5分钟的冗余时间，如之前的例子：第一名工人在10:15分会回到22号巡检点，上次检查22号巡检点的时间是9：45，每个巡检点35分钟检查一次，可计算下次要在10:20前再次检查，因此每个回到22号巡检点的工人都有5分钟的休息时间，即每工作135分钟可以休息5分钟，满足要求。

其次是进餐时间，进餐时间不可能从节点巡检时间挤出来的。

因此有两种方案：方案一，考虑增加替班工人，进餐时间由替班工人代替巡检；方案二，将中班的上班时间调整为12点至18点，工人在12点前吃完午饭后再上班，18点下班后再吃晚饭。因此可做如下安排：

早班大概是3、4点至11点，需要4名工人；中班总体大概在12点至18点，需要4名工人；晚班大概19点至次日3点，需要4名工人。

错时上班一天最少需要12人。

中班员工错时上班，4名工人工作量都是7小时左右；晚班4名员工工作量都是9小时左右；早班比较特殊，先上班的两名员工工作量是9小时左右，后上班的两名员工工作量是7小时左右。考虑工作量的均衡，采用轮换工作制，可在6天内实现所有员工工作量均衡。

详细时间安排见附录。

## 线路规划与时间安排

### 固时不休息工作模式线路规划与排班

由前文分析可知，在不休息工作模式下，每天需要工人12名，分为早班、中班、晚班，每班次4名。在此工作模式下，三个班次的时间安排相差不大，每班次工作时间大约是8小时，现给出中班部分时间安排表，早班、晚班按照中班时间表推算即可，详细时间安排可见附录。

**表4.1 不休息模式下线路一、二部分时间安排表**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 轮次 | 1 | 2 | 3 | …… | 12 | 13 | 14 |
| 线路一 |  |  |  |  |  |  |
| 22 | 10:00 | 10:35 | 11:10 | 16:25 | 17:00 | 17:35 |
| 23 | 10:03 | 10:38 | 11:13 | 16:28 | 17:03 | 17:38 |
| 24 | 10:07 | 10:42 | 11:17 | 16:32 | 17:07 | 17:42 |
| 20 | 10:13 | 10:48 | 11:23 | 16:38 | 17:13 | 17:48 |
| 19 | 10:18 | 10:53 | 11:28 | 16:43 | 17:18 | 17:53 |
| 21 | 10:26 | 11:01 | 11:36 | 16:51 | 17:26 | 18:01 |
| 4 | 10:30 | 11:05 | 11:40 | 16:55 | 17:30 | 18:05 |
| 22 | 10:35 | 11:10 | 11:45 | 17:00 | 17:35 | 18:07  下班 |
|  |  |  |  |  |  |  |
| 线路二 |  |  |  |  |  |  |
| 2 | 10:06 | 10:41 | 11:16 | 16:31 | 17:06 | 17:41 |
| 1 | 10:10 | 10:45 | 11:20 | 16:35 | 17:10 | 17:45 |
| 3 | 10:16 | 10:51 | 11:26 | 16:41 | 17:16 | 17:51 |
| 5 | 10:20 | 10:55 | 11:30 | 16:45 | 17:20 | 17:55 |
| 7 | 10:24 | 10:59 | 11:34 | 16:49 | 17:24 | 17:59 |
| 6 | 10:30 | 11:05 | 11:40 | 16:55 | 17:30 | 18:05 |
| 14 | 10:34 | 11:09 | 11:44 | 16:59 | 17:34 | 18:09 |
| 2 | 10:40 | 11:15 | 11:50 | 17:05 | 17:40 | 18:12  下班 |

以上是线路一和线路二的部分时间那排表。

线路三和线路四合并为一个哈密顿圈，巡检一圈需要69分钟，需要两个工人在圈上交替巡检，假设两名工人分别是甲、乙，则规定：

1. 甲10:00上班首要任务是去检查17号巡检点，从22号巡检点出发，走最短路径到达17号巡检点，此时是10：08，检查完毕后走最短路径到达26号巡检点，此时10:14，开始检查26号巡检点并依次巡检15、12、18、16、13、11、10、8、9、25最后回到26，继续下一轮巡检；
2. 乙10:00从22号巡检点接受任务后走最短路径到达11号巡检点，此时10：15，依次检查11、10、8、9、25、26、15、12、18、16、13回到11，继续下一轮巡检。

以上安排恰好能让甲、乙完成巡检任务，例如乙在10：15检查完11号巡检点，乙下次到达时间是69分钟后是11:24，而11号巡检点在10:50需要检查一次，而甲能在10:49赶到该巡检点，其余巡检点推理类似。

具体安排如下表：

**表4.2 不休息模式下线路三、四部分时间安排表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 线路三、四 |  | 1 | 2 | …… | 6 | 7 |
| 甲 | 17 | 10:08 |  |  |  |
| 26 | 10:14 | 11:23 | 15:59 | 17:08 |
| 15 | 10:22 | 11:31 | 16:07 | 17:16 |
| 12 | 10:26 | 11:35 | 16:11 | 17:20 |
| 18 | 10:32 | 11:41 | 16:17 | 17:26 |
| 16 | 10:37 | 11:46 | 16:22 | 17:31 |
| 13 | 10:42 | 11:51 | 16:27 | 17:36 |
| 11 | 10:49 | 11:58 | 16:34 | 17:43 |
| 10 | 10:54 | 12:03 | 16:39 | 17:48 |
| 8 | 11:03 | 12:12 | 16:49 | 17:58 |
| 9 | 11:11 | 12:20 | 16:56 | 18:05 |
| 25 | 11:18 | 12:27 | 17:03 | 18:12 |
|  |  |  |  |  | 18:14下班 |
| 乙 | 11 | 10:15 | 11:24 | 16:00 | 17:09 |
| 10 | 10:20 | 11:29 | 16:05 | 17:14 |
| 8 | 10:29 | 11:38 | 16:14 | 17:23 |
| 9 | 10:37 | 11:46 | 16:22 | 17:31 |
| 25 | 10:44 | 11:53 | 16:29 | 17:38 |
| 26 | 10:49 | 11:58 | 16:34 | 17:43 |
| …… | …… | …… | …… | 18:16下班 |

因此每班需要4名工人，每天三班需要12名工人。

### 固时可休息进餐工作模式线路规划与排班

问题二是在问题一基础上增加可休息可进餐的条件，由前面分析可知，线路一不能满足5分钟的休息时间，而线路二由于5、7号巡检点可以挤出10分钟休息时间，可以考虑将线路二挤出的时间留给线路一的工人作为休息时间；线路三和线路四合并的最小哈密顿圈中，13号巡检点周期是80分钟，检查时间是5分钟，这5分钟可作为工人的休息时间。



图4.4 可休息模式下线路一、二调整图

观察巡检连通图可知，如果能让线路二的工人在不巡检5、7号巡检点的情况下，去巡检线路一的4号巡检点，就能让线路一的工人得到休息时间。

吃饭时间需要安排替班人员，考虑到工人工作的均衡问题，干脆让替班工人也一起上班，在不进餐时间跟普通工人轮流巡检。

线路一和线路二考虑增加一名替班工人，与原来的两名工人轮流巡检线路一和线路二，在饭点时就可以轮换进餐。

线路三和线路四合并，也考虑增加一名替班工人，与原来的两名工人轮换巡检，进餐时间则轮换进餐。

因此，早班2:00至10:00，需要4人；中班10:00至18:00，需要6人，只考虑在中午12点进餐，18点在下班后进餐；晚班18:00至次日2:00，进餐后上班，需要4人。每天共需14人。

具体安排是：

**表4.3 可休息模式下线路一部分时间安排表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 线路一 | 1 | 2 | 3 | 4 | …… | 14 |
| 21 | 10:02 | 10:37 |  | 11:46 | 17:36 |
| 22 | 10:07 | 10:42 | 休息10min  11:16 | 11:51 | 17:41 |
| 20 | 10:11 | 10:46 | 11:20 | 11:55 | 17:45 |
| 19 | 10:16 | 10:51 | 11:25 | 12:00 | 17:50 |
| 23 | 10:23 | 10:58 | 11:32 | 12:07 | 17:57 |
| 24 | 10:27 | 11:02 | 11:36 | 12:11 | 18:01 |
| 4 | 10:34 | NO CHECK！ | 11:43 | 12:18 | NO CHECK！ |
| 21 | 10:37 | 11:46 | 12:21 |

**表4.4 可休息模式下线路二部分时间安排表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 线路二 | 1 | 2 | 3 | 4 | …… | 14 |
| 3 | 10:07 | 10:42 | 11:17 | 11:52 | 17:42 |
| 6 | 10:11 | 10:46 | 11:21 | 11:56 | 17:46 |
| 14 | 10:15 | 10:50 | 11:25 | 12:00 | 17:50 |
| 2 | 10:21 | 10:56 | 11:31 | 12:06 | 17:56 |
| 1 | 10:25 | 11:00 | 11:35 | 12:10 | 18:00 |
| 5(或4) | 10:32 | 11:08  CHECK 4! | NO CHECK! | NO CHECK!  休息11min | 18:04  CHECK 4 |
| 7(或4) | 10:36 | 11:44 |

**表4.5 可休息模式下线路三、四部分时间安排表1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 线路三、四 | 1(甲) | 2(乙) | …… | 5(乙) | 6(丙) | 7(甲) |
| 17 | 10:08 | NO CHECK | …… | NO CHECK | NO CHECK | NO CHECK |
| 26 | 10:14 | 11:23 | 14:51 | 16:01 | 17:10 |
| 15 | 10:22 | 11:31 | 14:59 | 16:09 | 17:18 |
| 12 | 10:26 | 11:35 | 15:03 | 16:13 | 17:22 |
| 18 | 10:32 | 11:41 | 15:09 | 16:19 | 17:28 |
| 16 | 10:37 | 11:46 | 15:14 | 16:24 | 17:33 |
| 13 | 10:42 | 11:51  休息6min | 15:19  休息6min | 16:29 | 17:38 |
| 11 | 10:49 | 11:59 | 15:27 | 16:36 | 17:45 |
| 10 | 10:54 | 12:04 | 15:32 | 16:41 | 17:50 |
| 8 | 11:03 | 12:13 | 15:42 | 16:51 | 18:00 |
| 9 | 11:11 | 12:21 | 15:49 | 16:58 | 18:07 |
| 25 | 11:18 | 12:28 | 15:56 | 17:05 | 18:14 |
| 26 | 11:23  甲吃饭30min | 12:33  乙吃饭30min | 16:01 | 17:10 | 18:16  甲下班 |

**表4.6 可休息模式下线路三、四部分时间安排表2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线路三、四 | (乙) | （丙） | （甲） | …… | （甲） | （乙） | （丙） |
| 26 | - | 10:49 | 11:58  丙吃饭30min | …… | 15:26 | 16:35 | 17:45 |
| 15 | - | 10:57 | 11:06 | 15:34 | 16:43 | 17:53 |
| 12 | - | 11:01 | 12:10 | 15:38 | 16:47 | 17:57 |
| 18 | - | 11:07 | 12:16 | 15:44 | 16:53 | 18:03 |
| 16 | - | 11:12 | 12:21 | 15:49 | 16:58 | 18:08 |
| 13 | - | 11:17 | 12:26 | 15:54 | 17:03  休息6min | 18:13 |
| 11 | 10:15 | 11:24 | 12:33 | 16:01 | 17:11 | 18:18  丙下班 |
| 10 | 10:20 | 11:29 | 12:38 | 16:06 | 17:16 | - |
| 8 | 10:29 | 11:38 | 12:47 | 16:15 | 17:25 | - |
| 9 | 10:37 | 11:46 | 12:55 | 16:23 | 17:33 | - |
| 25 | 10:44 | 11:53 | 13:02 | 16:30 | 17:40  乙下班 | - |

### 错时上班模式线路规划与排班

错时上班模式下，由前文的分析可知，第一名工人在10:00上班，第二名工人在10:35分上班，第三名工人在11:10上班，第四名工人在11:45上班。

1. 在错时上班模式下，重新解答问题一。

假设四名工人分别称为为甲、乙、丙、丁，中班时间10:00至18:00

甲10:00上班，19:05下班，工作时长9小时05分钟；

乙10:35上班，19:40下班，工作时长9小时05分钟；

丙11:10上班，17:55下班，工作时长6小时45分钟；

丁11:45上班，18:30下班，工作时长6小时45分钟。

晚班从18:00开始至次日2:00，时间安排与中班相差8小时；早班以此类推。

每班需要4名工人，每天三班共需12名工人，与之前求解的固定上班时间模式下的工人数量一样。

部分时间安排如下：

**表4.7 错时上班不休息模式下部分员工时间安排表**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线路 | 员工1 | 员工2 | 员工3 | 员工4 | …… | 员工1 | 员工2 |
| 22 | 10:00 | 10:35 | 11:10 | 11:45 | …… | 16:50 | 17:25 |
| 20 | 10:04 | 10:39 | 11:14 | 11:49 | 16:54 | 17:29 |
| 19 | 10:09 | 10:44 | 11:19 | 11:54 | 16:59 | 17:34 |
| 21 | 10:17 | 10:52 | 11:27 | 12:02 | 17:07 | 17:42 |
| 4 | 10:21 | 10:56 | 11:31 | 12:06 | 17:11 | 17:46 |
| 2 | 10:26 | 11:01 | 11:36 | 12:11 | 17:16 | 17:51 |
| 1 | 10:30 | 11:05 | 11:40 | 12:15 | 17:20 | 17:55 |
| 3 | 10:36 | 11:11 | 11:46 | 12:21 | 17:26 | 18:01 |
| 5 | 10:40 | 11:15 | 11:50 | 12:25 | 17:30 | 18:05 |
| 7 | 10:44 | 11:19 | 11:54 | 12:29 | 17:34 | 18:09 |
| 6 | 10:50 | 11:25 | 12:00 | 12:35 | 17:40 | 18:15 |
| 14 | 10:54 | 11:29 | 12:04 | 12:39 | 17:44 | 18:19 |
| 10 | 11:03 | 11:38 | 12:13 | 12:48 | 17:53 | 18:28 |
| 11 | 11:07 | 11:42 | 12:17 | 12:52 | 17:57 | 18:32 |
| 13 | 11:12 | 11:47 | 12:22 | 12:57 | 18:02 | 18:37 |
| 16 | 11:19 | 11:54 | 12:29 | 13:04 | 18:09 | 18:44 |
| 18 | 11:25 | 12:00 | 12:35 | 13:10 | 18:15 | 18:50 |
| 15 | 11:29 | 12:04 | 12:39 | 13:14 | 18:19 | 18:54 |
| 12 | 11:33 | 12:08 | 12:43 | 13:18 | 18:23 | 18:58 |
| 26 | 11:43 | 12:18 | 12:53 | 13:28 | 18:33 | 19:08 |
| 25 | 11:48 | 12:23 | 12:58 | 13:33 | 18:38 | 19:13 |
| 17 | 11:51 | 12:26 | 13:01 | 13:36 | 18:41 | 19:16 |
| 8 | 11:54 | 12:29 | 13:04 | 13:39 | 18:44 | 19:19 |
| 9 | 12:02 | 12:37 | 13:12 | 13:47 | 18:52 | 19:27 |
| 24 | 12:08 | 12:43 | 13:18 | 13:53 | 18:58 | 19:33 |
| 23 | 12:11 | 12:46 | 13:21 | 13:56 | 19:01 | 19:36 |
| 22 | 12:15 | 12:50 | 13:25 | 14:00 | 19:05 | 19:40 |

由此可见，在此时间安排表下错时上班，四名工人的工作时长是不同的，甲乙两名工人工作时间接近9小时，丙丁工作时间不足7小时。考虑到每名工人的工作均衡，可以采用轮换制度，即次日上班丙丁先上班，甲乙后上班，实现甲乙和丙丁工作轮换的目的，即可在两天内实现每名工人的工作均衡。

1. 在错时上班模式下，重新解答问题二。

错时上班每个工人走的路线完全一样，135分钟一个周期，35x4=140分钟，因此每次回到出发点都有5分钟的冗余时间，可以作为休息时间。

假设四名工人分别称为为甲、乙、丙、丁，

甲11:00上班，17:55下班，工作时长6小时55分钟；

乙11:35上班，18:30下班，工作时长6小时55分钟；

丙12:10上班，19:05下班，工作时长6小时55分钟；

丁12:45上班，19:40下班，工作时长6小时55分钟。

中班的员工工作量一样，每个工人都是巡检3趟，一趟135分钟加上中间的10分钟休息时间。

晚班四名员工上班时间分别为18:00、18:35、19:10、19:45，员工上班前吃晚饭，下班时间分别为3:15、3:50、4:25、5:00，工作时长都是9小时15分钟。

早班特殊点，四名员工上班时间分别为3:20、3:55、4:30、5:05，下班时间分别为11:25、12:00、12:15、12:50，前两名工人和后两名工人的工作时长分别为6小时55分钟、8小时55分钟，工作量有差别。

考虑所有员工的工作量均衡，可以采用三班倒制度下结合轮换方式，可在6天内实现所有员工工作量均衡。

每班需要4名工人，每天三班共需12名工人。由于将员工的饭点安排在了上班时间之外，中班时间大概为12点左右到18点左右，省下了替班员工，比之前求解的固定上班时间可休息进餐模式下的工人数量少2人。

因此，错时上班可以节省人力资源，并且每个巡检点都以最小周期35分钟考虑，不用像固时上班一样，在规划线路基础上要考虑每个不同巡检周期的巡检点用来节省休息时间。

部分时间安排表如下标：

**表4.8 错时上班可休息进餐部分时间表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 线路 | 员工1 | 员工2 | …… | 员工1 | | 员工2 |
| 22 | 饭后上班  11:00 | 饭后上班  11:35 | …… | 15:40 | | 16:15 |
| 20 | 11:04 | 11:39 | 15:44 | | 16:19 |
| 19 | 11:09 | 11:44 | 15:49 | | 16:24 |
| 21 | 11:17 | 11:52 | 15:57 | | 16:32 |
| 4 | 11:21 | 11:56 | 16:01 | | 16:36 |
| …… | …… | …… | …… | …… | …… | |
| 12 | 12:33 | 13:08 | …… | 17:13 | | 17:48 |
| 26 | 12:43 | 13:18 | 17:23 | | 17:58 |
| 25 | 12:48 | 13:23 | 17:28 | | 18:03 |
| 17 | 12:51 | 13:26 | 17:31 | | 18:06 |
| 8 | 12:54 | 13:29 | 17:34 | | 18:09 |
| 9 | 13:02 | 13:37 | 17:42 | | 18:17 |
| 24 | 13:08 | 13:43 | 17:48 | | 18:23 |
| 23 | 13:11 | 13:46 | 17:51 | | 18:26 |
| 22 | 13:15  休息5min | 13:50  休息5min | 17:55  下班吃饭 | | 18:30  下班吃饭 |
| - | - | - |  | 工作量6:55 | | 工作量6:55 |

详细安排见附录。

# 总结

本文以2017年大学生数学建模竞赛题目D题为研究出发点，引导研究比较合适的解决方法，最后决定以旅行商问题为基础建立数学模型，以遗传算法为核心求解模型，最终通过Java语言编程实现了化工厂排班调度模型的算法仿真，设计得到了任务相对均衡的化工厂巡检线路，并对各条线路的巡检时间安排做出了详细的规划。

针对本论文研究的化工厂巡检线路排班的题目，本人查阅了相关的资料，了解到针对本问题的解决方案有基于最短路径、最小生成树、启发式算法等，本人得到启发采用了以旅行商问题为基础模型，遗传算法为解法，现对本文模型和解法做出总结分析：

旅行商问题是一个经典的问题，本文中的化工厂巡检排班问题也可以考虑为旅行商问题，建立的数学模型能够适用于更多的巡检点，算法程序也比较灵活，如果化工厂巡检点增多了或者巡检点连通图改变了，只需修改对应的两个距离二维矩阵的文件和算法中计算路程耗时的函数即可。

旅行商问题模型的解法不仅仅可以遗传算法求解，还可以使用蚁群算法、模拟退火算法、神经网络算法等求解，因此本文建立的针对化工厂排班的旅行商模型求解方式多样。不过，模型的缺点是在针对工作均衡的区域划分问题上，巡检点数量的划分具有多样性，本文的划分是7、7、7、5，当然也可以划分为6、6、7、7，相应的计算路程耗时的函数要修改。本文简单考虑了工作均衡问题直接将巡检点数划分为7、7、7、5和6、6、7、7，通过实践之后决定采用了7、7、7、5的划分方式。因此，把握巡检点数量的划分与工作均衡问题之间的平衡是一个需要优化的点。

虽然遗传算法在解决旅行商问题上表现良好，但是最终在规划的线路上做出调整以及时间安排上，由于线路时间冗余较少，部分线路根本没有冗余时间，甚至超出了35分钟的最小巡检周期，做出调整需要花费一定的思考时间。

总体上看，本文达到了消耗最少的人力资源完成巡检任务的目的，研究和实践了遗传算法在解决旅行商问题的有效性和适用性。

# 参考文献11

1. 谢桂芩.车辆路径问题建模与多目标进化算法[D].广东工业大学，2012.
2. 黄尚.遗传算法在物流配送中的研究与应用[D].广东工业大学，2013.
3. Gilbert Laporte .The Traveling Salesman Problem: An overview

of exact and approximate algorithms[J]. European Journal of

Operational Research 59 (1992) 231-247 North-Holland.

1. 唐文琦,曾干敏,刘泽宇.浅谈遗传算法及其部分改进算法[J].科技风,2019(12):57.
2. 李岩,袁弘宇,于佳乔,张更伟,刘克平.遗传算法在优化问题中的应用综述[J].山东工业技术,2019(12):242-243+180.
3. 庞峰. 模拟退火算法的原理及算法在优化问题上的应用[D].吉林大学,2006.
4. 陈星宇. 蚁群算法理论及其应用研究[D].湖南师范大学,2008.
5. 刘楠.巡检线路的哈密顿圈分割模型及算法[J].甘肃科学学报,2018,30(03):15-18.
6. 谢晓敏,王正.化工厂安全巡检线路的排班研究[J].南通职业大学学报,2018,32(02):69-71+74.
7. 段振华,杨海铭,李循鹏,文雷华.巡检线路排班优化模型[J].南方职业教育学刊,2018,8(04):61-66.
8. 薛毅.“巡检线路的排班”问题解析[J].数学建模及其应用,2018,7(01):52-62+79.
9. 卢厚清,王辉东,黄杰,李波.任务均分的多旅行商问题[J].系统工程,2005(02):19-21.
10. Dickens, J.,Teleka, R.. The use of the Travelling Salesman Problem To optimise power line inspections[P]. ,2013.
11. 代坤,鲁士文,蒋祥刚.基于遗传算法的多人旅行商问题求解[J].计算机工程,2004(16):139-140+145.
12. 刘楠.巡检线路的哈密顿圈分割模型及算法[J].甘肃科学学报,2018,30(03):15-18.
13. Jun Li,Qirui Sun,MengChu Zhou,Xianzhong Dai. A New Multiple Traveling Salesman Problem and Its Genetic Algorithm-Based Solution[P]. ,2013.
14. Wei Zhou, Yuanzong Li. An improved genetic algorithm for multiple traveling salesman problem[P]. Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on,2010.

# 附录A 程序源码

1. Floyed类：用于求出所有巡检点之间的最短距离

public class Floyed {  
 public static int[][] getShortestPath(int[][] cost, int nodeNumber) {  
 int[][] shortest = new int[nodeNumber][nodeNumber];  
 int[][] bridge = new int[nodeNumber][nodeNumber];  
 for(int i=0; i<nodeNumber; i++){  
 for(int j=0; j<nodeNumber; j++){  
 shortest[i][j] = cost[i][j];  
 bridge[i][j] = -1;  
 }  
 }  
 for(int k=0; k<nodeNumber; k++){  
 for(int i=0; i<nodeNumber; i++){  
 for(int j=0; j<nodeNumber; j++){  
 if (i==j) {  
 continue;  
 }  
 if(shortest[i][k] + shortest[k][j] < shortest[i][j]){  
 shortest[i][j] = shortest[i][k] + shortest[k][j];  
 bridge[i][j] = k;  
 }  
 }  
 }  
 }  
 return shortest;}  
}

1. GeneticAlgorithm类：遗传算法核心功能，包括选择、交叉、变异方法等。

public class GeneticAlgorithm {  
 private int populationSize;  
 private double crossoverRate;  
 private double mutationRate;  
 private int pickCount;  
 private int tournamentSize;  
 public GeneticAlgorithm(int populationSize, double crossoverRate, double mutationRate, int pickCount, int tournamentSize) {  
 this.populationSize = populationSize;  
 this.crossoverRate = crossoverRate;  
 this.mutationRate = mutationRate;  
 this.pickCount = pickCount;  
 this.tournamentSize = tournamentSize;  
 }  
 public Population initPopution(int chromosomeLength){  
 Population population = new Population(populationSize, chromosomeLength);  
 return population;  
 }  
 public boolean isEnd(int generationCount, int maxGenerations){  
 if(generationCount < maxGenerations){  
 return false;  
 }  
 return true;  
 }  
 public double individualFitness(Individual individual, Node[] nodes){  
 Path path = new Path(individual,nodes);  
 double fitness = 1.0/path.getTimeCost();  
 individual.setFitness(fitness);  
 return fitness;  
 }  
 public void populationFitness(Population population, Node[] nodes){  
 double sumFitness = 0.0;  
 for(Individual individual : population.getIndividuals()){  
 sumFitness += this.individualFitness(individual, nodes);  
 }  
 double populationFitness = sumFitness/population.size();  
 population.setPopulationFitness(populationFitness);  
 }  
 public Individual selectParent(Population population){  
 Population tournament = new Population(tournamentSize);  
 population.shuffle();  
 for(int i=0; i<tournamentSize; i++){  
 Individual individual = population.getIndividual(i);  
 tournament.setIndividual(i, individual);  
 }  
 return tournament.getFittest(0);  
 }  
 public Population crossoverPopulation(Population population) {  
 Population newPopulation = new Population(population.size());  
 for (int populationIndex = 0; populationIndex < population.size();

populationIndex++) {  
 Individual parent1 = population.getFittest(populationIndex);  
 if (this.crossoverRate > Math.random() && populationIndex >= this.pickCount) {  
 Individual parent2 = this.selectParent(population);  
 int sonChromosome[] = new int[parent1.getChromosome().length];  
 Arrays.fill(sonChromosome, -1);  
 Individual son = new Individual(sonChromosome);  
 int index1 = (int) (Math.random() \* parent1.getChromosome().length);  
 int index2 = (int) (Math.random() \* parent1.getChromosome().length);  
 final int start = Math.min(index1, index2);  
 final int end = Math.max(index1, index2);  
 for (int i = start; i < end; i++) {  
 son.setGene(i, parent1.getGene(i));  
 }  
 for (int i = 0; i < parent2.getChromosome().length; i++) {  
 int parent2Gene = i + end;  
 if (parent2Gene >= parent2.getChromosome().length) {  
 parent2Gene -= parent2.getChromosome().length;  
 }  
 if (son.containsGene(parent2.getGene(parent2Gene)) == false) {  
 for (int j = 0; j < son.getChromosome().length; j++){   
 if (son.getGene(j) == -1) {  
 son.setGene(j,parent2.getGene(parent2Gene));  
 break;  
 }  
 }  
 }  
 }  
 newPopulation.setIndividual(populationIndex, son);  
 } else {  
 newPopulation.setIndividual(populationIndex, parent1);  
 }  
 }  
 return newPopulation;  
 }  
 public Population mutatePopulation(Population population){  
 Population newPopulation = new Population(this.populationSize);  
 for (int i = 0; i < population.size(); i++) {  
 Individual individual = population.getFittest(i);  
 if (i >= this.pickCount) {  
 for (int geneIndex = 0; geneIndex < individual.getChromosome().length; geneIndex++) {  
 if (this.mutationRate > Math.random()) {  
 int newGeneIndex = (int) (Math.random() \* individual.getChromosome().length);  
 int gene1 = individual.getGene(newGeneIndex);  
 int gene2 = individual.getGene(geneIndex);  
 individual.setGene(geneIndex, gene1);  
 individual.setGene(newGeneIndex, gene2);  
 }  
 }  
 }  
 newPopulation.setIndividual(i, individual);  
 }  
 return newPopulation;  
 }  
}

# 附录B 巡检线路规划时间安排表

