

**硕士研究生课程作业**

课程名称 计算机视觉技术

题 目 一种求解TSP问题的遗传算法设计与实现

学 院 计算机科学与工程学院

专业名称 计算机技术

姓 名 刘震

学 号 2023600042

授课时间 2023-2024年2学期

一种求解TSP问题的遗传算法设计与实现

**摘 要：**旅行商问题(TSP)是典型的算法组合优化问题，依据该算法的显著优势，其在社会科学的众多领域，均得到广泛应用。与此同时，随着该问题规模的上升，解空间出现了爆炸式增长等问题。本文以遗传算法为问题去求解算法，并对TSP的求解结果进行分析和探讨。本文首先通过以选题背景和研究意义为切入点进行讨论，引出遗传算法原理、特点和算法描述，然后对TSP进行建模，并且给出求解TSP的遗传算法设计与实现，在此之后组织了测试数据进行实验仿真，结果符合预期的效果。最后，对全文进行总结，并指出研究的不足。

**关键词：**旅行商问题，MATLAB仿真，算子，遗传算法

# 1引言

巡回旅行商问题(TSP)是一种易于描述但是难以处理解决的组合优化问题, 该问题具体是指：一位旅行商从某一城市出发，对若干个城市进行单独一次访问，最后回至出发点的城市，该旅行商需要通过合理安排路径，使得旅行的总路程实现最短的结果。根据图论术语，假设有一个图 *g* ＝（ *a* \**b*），其中 *a* 是顶点集，*b* 是边集，设距离矩阵 *d*＝（ *dij*）由两个顶点 *i* 和*j* 之间的距离组成。旅行路线选择问题就是计算一条能够通过所有顶点的同时还可以保证每个顶点只通过一次的最短路线，且最后必须返回出发城市。

从理论层面，最简单有效的方法就是穷尽列举法，将旅行商的所有路径可能性全部列举出来，然后逐一比较或计算权值，最后找到最优解。然而，这种穷举的方式既费时又费力，并不可取。所以，人们又提出了各种解决TSP问题的优化算法，例如，模拟退火、神经网络、蚁群算法、免疫算法、禁忌搜索算法等[1]。

显然，TSP的关键是在给定的交通网络图中，如何规划行程，获取最优的路径。然而，随着旅行商访问城市数目的增加，问题的解空间呈指数级增长，传统的精确算法要在较短时间内取得最优解往往比较困难。遗传算法是一种典型的仿生算法，其全局搜索能力极强。近年来，在很多问题的算法优化求解中，遗传算法得以广泛使用。本文采用遗传算法对TSP进行优化、求解，以期获取高质量的问题解决方案。首先，对遗传算法和TSP进行理论概述分析，在此基础上构建解决TSP问题的遗传算法模型。接着进行算法模型分析,分模块进行建立数据迭代方式。编写代码实现功能完成系统开发。最后,对系统进行模块测试和系统性测试，形成结论,并撰写论文。

# 2理论概述

## 2.1 TSP问题描述

TSP是典型的NP完全问题。可描述为：旅行商选择并计算一条能够通过所有顶点的同时还可以保证每个顶点只通过一次的最短路线，且最后必须返回出发城市。换一种描述方式即，计算一条距离最短的遍历n个城市的路线，或者说搜索，其中标识对n个城市的编号的排列

 （3.1）

其中表示城市之间的距离，表示累加所有距离之和，min 表示取最小数值。

TSP问题不仅是旅行商问题，而且生活和科学研究中存在着许多其他NP完全问题也可以归属于TSP问题，即无法找到固定的算法能在多项式时间内 计算出最优解。例如油路问题、公共汽车路线和产品展示位置问题等。这类问题的求解意义重大。

## 2.2遗传算法[2]

## 2.2.1 算法特点

遗传算法可解释为基于自然选择和基因遗传学原理的随机并行搜索算法, 是一种寻求全局最优解而不需要任何初始化信息的高效优化方法。此问题的解集类比于一个种群, 通过不断地选择、交叉、变异等遗传操作, 使解的质量趋于最优。该算法不仅全局寻优能力强、适应性强，而且对于解决非线性问题具有较强的鲁棒性，并且其对问题没有特定限制，计算过程简单、对搜索空间没有特殊要求、易于与其他算法结合等都是其优秀的独特特点。在函数优化、图像处理、自动控制、经济预测等领域均有大规模应用。遗传算法在求解NP完全问题方面是一种效果显著的全局方法，特别适合传统算法不好解决的非确定性多项式选择问题。

(1) 遗传算法几乎对待解决问题没有要求，而且能明确如何构建目标函数；

(2) 启发性的知识智能搜索算法是遗传算法所采用，其在计算高度空间复杂问题上比早期的算法有更好的效果；

(3) 遗传算法是全局化优化，其对问题参数或者群体的变量进行迭代优化，而不是个体参数；

(4) 遗传算法初始化时，随机产生选择、交叉、变异算子，这样能扩大算法适用的不确定问题的规模；

遗传算法3个最重要的算子：交叉算子、选择算子和变异算子。遗传算法由五个基本元素组成：

(1) 对参数进行编码；

(2) 设定初始种群大小；

(3) 适应度函数的设计；

(4) 遗传操作设计；

(5) 控制参数设定(包含种群、进化代数、交叉变异率等)；

## 2.2.2遗传算法步骤

这些遗传算法都有通过模仿生物遗传和进化过程中选择、交叉、变异机理的，来完成对问题最优解的自适应求解过程的共同特点。

在遗传算法中，待解决问题存在各种各样的可能性。而它的每一种可能性，可以理解为一条染色体中，每条染色体是一个个体，若干个个体组成一个群体，即所有可能的解。算法启动后，可能会随机生成一些个体（即初始解），每个个体根据一个目标函数进行评估，并分配相应的拟合优度值来表示个体对环境的适应程度。基于这个拟合优度值，对个体进行选择、复制、杂交、变异等，最终重组产生“最佳拟合环境”（即最优解）的个体。

遗传算法的中心思想：首先确定问题的解范围，对每个点进行编码(即一个可行解)。然后从解范围中选择几个可能的解决方案被分组到初始组中，并计算下一次拟合。对于每一个个体函数值，使用一个复制算子来保证拟合优度值较高的个体有更多的复制机会，评估新组输出最优解。它符合相关要求。

图示, 示意图

描述已自动生成

图 2.1 遗传算法流程图

# 3遗传算法的算法设计

## 3.1算法仿真概述

遗传算法的目标为：定义总体需求作为用户与开发人员之间相互理解的基础，并且对算法提供有关性能要求，得到初步设计和对用户的影响的信息，并作为开发人员设计和实现的基础，以进行整体验证和验证为基础，进行系统介绍，系统功能需求讲解和解决方案。

当前TSP问题已成为当今社会的经典问题。每年有成千上万的研究者想方设法地解决该经典问题，并坚持验证算法的性能。值得注意的是，遗传算法的设计是 基于用户的实际需求进行细化分析，明确遗传算法的功能细节，为后续设计编码提供业务层面的支撑。

## 3.2 算法实现步骤

Step1：选择决策变量和约束条件。也就是说，它决定了个体表型X 和问题的解决空间。

Step2：建立优化模型。即选择目标函数的类型(计算目标函数的最大值还是计算目标函数的最小值)及其数学描述或量化方法。

Step3：确定了被标注算法可行解的染色体编码方法，与此同时也确定出遗传算法的搜索空间和个体的基因型X等。

Step4：决定如何排毒。也就是说，它决定了从个体基因型X 到个体表型X的相应转化方法。

Step5：决定如何定量评估个体适应度。就是确认函数f(X)的值到个体拟合优度F(X)的优化方式。

Step6：设计遗传算子。就是确定出变异运算、交叉运算、选择运算等遗传算子的操作方式的细节。

Step7：确定算法的相关的运行参数。即确定该算法的M、T、Pc、Pm等参数。

## 3.2.1算法个体编码[3]

在遗传算法中，必须要把问题中的条件和文字描述变成计算机能读懂的编码数字，这样才能在后续遗传算法的算子中进行计算和迭代，这个步骤我们称之为编码。

针对不同的问题，截止到目前为止没有一个通用的编码方案。当然这个研究方向也是遗传算法的重要发展方向。因此我们只能针对具体问题，采用不同的编码方案。

## 3.2.2适应度函数[3]

遗传算法基本上不使用外部信息来研究进化过程，而是基于适应度函数的搜索，使用每个种群的个体适应度值。因此，自适应函数的选择直接影响遗传算法的收敛速度，其对于寻找最优解非常重要。一般来说，适应度函数由目标函数组成与目标函数值范围的映射变换称为匹配级比例变换。科研人员在自然界中生物遗传以及进化研究实验中,也会使用适应度一词来衡量一种生物对生活环境的迎合程度。更适应自然环境的生物自然会有更多的生存机会，而不适应的物种几乎没有机会，甚至灭绝。同理能力强体力好的人，繁衍下一代时，将优点进行遗传。一般过程：

1. 在破译个体密码字符串后,可以获得个体的表型。
2. 根据判断算法在优化问题方面的类型，由指定函数值对映转换规则计算出适应度。

## 3.2.3选择算子

选择算子（也称为重播算子）在遗传算法中用于为一组个体执行最合适的任务的生存。低健康是遗传的，并且传给下一代的机会很小。在遗传算法中，选择操作是一种遗传操作，它确定亲代种群中的哪些个体将以特定方式从亲代种群遗传给下一代。

选择基于个人的健康评估。选择的主要目的是避免基因缺失，提高全局收敛性和计算效率。选择算子的常用计算方法有：比例选择、最优守恒策略、确定的抽样选择、以及随机选择。

## 3.2.4交叉算子[1]

自然界中，交配这样才能繁衍下一代物种。遗传算法使用交叉来模拟交配，因此交叉又是这三个算子中最重要且关键的部分，它的好坏直接决定了迭代产生新个体的优劣，并最终决定了此遗传算法的功能和性能问题。

如前文所述，路线表示的编码方法要求每个城市可以访问一次，而基本遗传算法的交叉操作生成的个体通常不能满足这一条件。因此，我们又可以提出一种被称为重排操作的方法来解决这类问题。顺序交叉法就是其中的一种。

## 3.2.5变异算子[4]

在求解TSP问题时，变异算子的设计比交叉算子灵活得多，可以使用任何带有局部搜索的运算符作为变异运算符。生物界中的物种要获得进化的能力，就是通过细胞分裂复制的环境，有意或无意出现的偶然因素导致复制时出现差错。这些差错导致下一代物种与上一代之间有差别，我们叫做基因突变或者物种变异。表现出来的现象就是，新的个体与上一代之间存在一定的差别。例如长颈鹿的脖子原来没那么长，但是经过进化后，脖子越来越长。同样变异算子也是在产生下一代时出现的必不可少的运算。变异算子使得算法朝着最优化问题不停迭代，直至找到最优结果。变异算子主要作用包括两方面：一是改善局部搜索功能，二是保证群体多种可能，防止出现早熟停止进化迭代。

## 3.3算法问题具体描述

本文仿真以20个城市为例，假定20个城市的位置坐标如下：

表3.1 数据表

|  |  |
| --- | --- |
| 站点序号 | X,Y |
| 1 | 13.47,56.10 |
| 2 | 16.47,54.44 |
| 3 | 20.09,52.54 |
| 4 | 22.39,53.37 |
| 5 | 25.23,57.24 |
| 6 | 22.00,56.05 |
| 7 | 20.47,57.02 |
| 8 | 17.20,56.29 |
| 9 | 16.30,57.38 |
| 10 | 14.05,58.12 |
| 11 | 16.53,57.38 |
| 12 | 21.52,55.59 |
| 13 | 19.41,57.13 |
| 14 | 10.09,55.55 |
| 15 | 12.09,52.55 |
| 16 | 15.09,56.55 |
| 17 | 17.09,53.55 |
| 18 | 18.09,54.55 |
| 19 | 21.09,55.55 |
| 20 | 25.09,56.55 |

求旅行商经过这20个站点的最短距离。

## 3.4解题步骤设计

遗传算法一般流程图如下：

图示

描述已自动生成

图 3.1 遗传算法一般流程图

遗传算法一般操作步骤:

1编码：求解问题之前，首先分析问题对应的目标函数和变量，然后对变量进行编码。

2基因操纵：通过个体的体力评估个体的素质，确定个体如何进行交叉，变异运算。

3个体适应度函数：如何使用算法搜索对群体中每个个体的适应度值。从而找到最适合的解决方案。

遗传算法的核心流程图如下：

****

图 3-2 算法核心部分流程图

基于TSP对应排列编码的遗传算法的基本步骤：

(1)编码：先对问题进行详细分析，然后对到达站点位置信息进行编码，包括站点信息之间如何计算距离。

(2) 拟合优度函数：两个站之间的距离直接作为适应度评价函数，站点之间直线最短路径。

(3) 选择策略：具体操作是将每个站点的N个对象按距离远近降序排列，第一个对象表现最好，直接复制到下一代。下一代组中的N-1 个人应根据上一代组中N个人的适应度进行轮选择。

(4)交叉操作：在该编码方式下有几种编码方式：仅对编码进行交叉编码。

(5)变异操作：本程序中对于变异操作，采用对编码变异的方式。

**3.5算法核心代码**

基于3.1和3.2的理论，改进后建模代码如下（代码中有注释）：

|  |
| --- |
| **while** gen**<=**genmax  **%%** 计算适应度  ObjV**=**PathLength**(**D**,**Chrom**);** **%**计算路线长度  **%** fprintf**(**'%d %1.10f\n'**,**gen**,**min**(**ObjV**))**  line**([**gen**-**1**,**gen**],[**preObjV**,**min**(**ObjV**)]);**pause**(**0.0001**)**  preObjV**=**min**(**ObjV**);**  FitnV**=**Fitness**(**ObjV**);**  **%%** 选择  SelCh**=**Select**(**Chrom**,**FitnV**,**GGAP**);**  **%%** 交叉操作  SelCh**=**Recombin**(**SelCh**,**Pc**);**  **%%** 变异  SelCh**=**Mutate**(**SelCh**,**Pm**);**  **%%** 逆转操作  SelCh**=**Reverse**(**SelCh**,**D**);**  **%%** 重插入子代的新种群  Chrom**=**Reins**(**Chrom**,**SelCh**,**ObjV**);**  **%%** 更新迭代次数  gen**=**gen**+**1 **;**  End |

代码主要公式如下PathLength 为计算两个(城市点之间)点之间的距离，采用最简单直观的x和y相减。min是获取最小距离，即获取这个城市到其他城市的最小距离。Fitness计算适应度，即对每个距离进行赋值，让他进入下一步做准备。Select计算种群，适应度，选择概率。Reverse进化逆转函数下，被选择个体的个城市的距离矩阵。Mutate 变异操作。Reins 将上述重插入子代的新种群。Gen为统计并更新迭代次数，当迭代次数到最大数值时，退出算法。

# 4实验仿真

## 4.1运行准备环境[5]

本文算法及本文算法1都是运行在1台计算机的Matlab上（环境配置见表4-1）上。

**表4-1 实验环境**

|  |  |
| --- | --- |
| 名称 | 参数 |
| 处理器 | Intel(R)Core(TM)i5-4210UCPU@1.70GHz 2.39GHz |
| 内存 | 4G |
| 操作系统 | Windows 10 |
| Matlab | Matlab 2018b |
| 硬盘 | 固态512G |

## 4.2仿真结果

用简单的遗传算法模拟建模，解决基本的TSP调度问题。

已知20个城市相互之间的距离和坐标位置，某一旅行商从某个城市出发访问每个城市一次且只能一次，最后回到出发城市，如何安排才能使其所走路线最短。

假定20个站位置为3.2.1章节，入参：

**表4-2 实验初始数据**

|  |  |
| --- | --- |
| 站点序号 | X,Y |
| 1 | 13.47,56.10 |
| 2 | 16.47,54.44 |
| 3 | 20.09,52.54 |
| 4 | 22.39,53.37 |
| 5 | 25.23,57.24 |
| 6 | 22.00,56.05 |
| 7 | 20.47,57.02 |
| 8 | 17.20,56.29 |
| 9 | 16.30,57.38 |
| 10 | 14.05,58.12 |
| 11 | 16.53,57.38 |
| 12 | 21.52,55.59 |
| 13 | 19.41,57.13 |
| 14 | 10.09,55.55 |
| 15 | 12.09,52.55 |
| 16 | 15.09,56.55 |
| 17 | 17.09,53.55 |
| 18 | 18.09,54.55 |
| 19 | 21.09,55.55 |
| 20 | 25.09,56.55 |

代码中的参数：

X =[13.47,51.10

16.47,54.44

20.09,52.54

22.39,53.37

25.23,57.24

22.00,56.05

20.47,57.02

17.20,56.29

16.30,57.38

14.05,58.12

16.53,57.38

21.52,55.59

19.41,57.13

10.09,55.55

12.09,52.55

15.09,56.55

17.09,53.55

18.09,54.55

21.09,55.55

25.09,56.55];

基本流程如下：

1 初始化20个城市地址坐标。

2 建立遗传算法的初始化入参(种群大小，交叉变异概率，城市之间的距离算法等)。

3 进行遗传算法具体参见（图 3-2）。

4 画出随机到20个城市的坐标并计算距离，图4.1并计算出路径。

5 画出通过遗传算法后的最优路径 图4.3并计算出路径。

6 画出遗传算法过程中迭代图[8]。

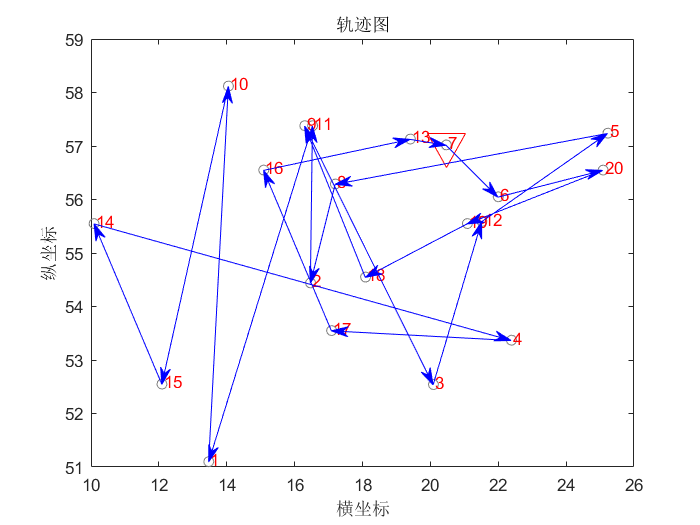


图4.1 随机路径

初始种群中的一个随机值:

7—>6—>20—>19—>18—>9—>3—>12—>5—>8—>2—>11—>1—>10—>15—>14—>4—>17—>16—>13—>7

总距离：92.5323

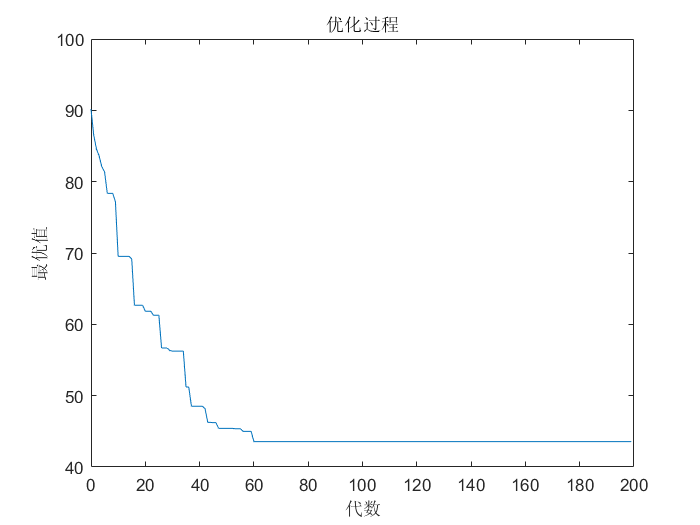


图4.2 使用遗传算法迭代次数

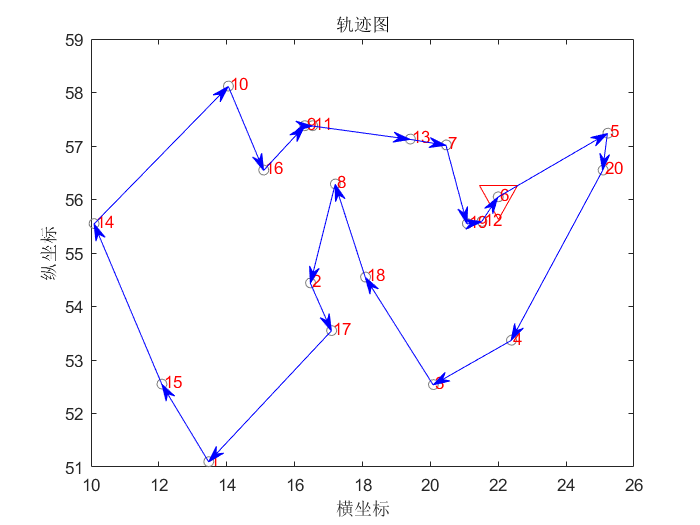


图4.3 遗传算法最优路径调度

最优解为：

6—>5—>20—>4—>3—>18—>8—>2—>17—>1—>15—>14—>10—>16—>9—>11—>13—>7—>19—>12—>6

总距离：43.5549

## 4.3结果分析

如上研究结果进行分析：

图4.1是不采用算法，采用随机路线进行解决TSP问题，其得出的路线总距离为：92.5323。

图4.2 遗传算法总共迭代200次，但是80次左右就已经得到最优解了。

图 4.3 总距离从92.5323变成43.5549，减少了近一半路程。

由此数据结果我们可以看出优化前后的路径长度得到很大的改进。特别是在优化路径数据上，得到了大量的压缩。可以得出结论：仿真结果各项指标符合预期[9]。

# 5总结

在该项目中，本人设计并完成了一个遗传算法来求解一个TSP 模型。模型求解结果表明，改进后的遗传算法能较好地解决TSP问题，使系统运行更加优化。该研究为后续各系统的优化计算以及使用该算法的系统的整体规划和运行提供依据具有重要的意义。

然而，该研究方法也具有一定的局限性，比如时间尺度的选择，理想环境下的模拟等，没有考虑输入参数对具体问题和系统的影响。我们相信,通过不断的改进和优化,它会发展成为一个更加完善的系统。

参考文献

[1] 李薇. 遗传算法及其在TSP问题中的应用研究[D]. 贵州: 贵州大学,

2018.4.1.

[2] 薛宏智. 遗传算法在TSP上的应用及改进[D]. 西安长安大学, 2018 .

[3] 彭丹平. 遗传算法在TSP问题上的应用[D]. 南京东南大学, 2019.5.20.

[4] 赵雪梅. 遗传算法及其在TSP问题求解中的应用[J]. 四川兵工学201930(11): 22-26.

[5] 雷英杰等编著. MATLAB遗传算法工具箱及应用[M]. 西安: 西安电子科技大学出版社, 2005.4

[6] 王小平等编著. 免疫遗传算法在TSP求解中的应用[J]. 计算机应用于软件, 2006. 05.

[7] 周明, 孙树栋编著. 遗传算法原理及应用[Z]. 西北工业大学, 1999.6.

[8] 谢胜利等. 求解TSP问题的一种改进的遗传算法[J]. 计算机工程与应用, 2002 (8), 58~245.

[9] 王俊海. TSP问题的一种高效Memetic算法[J]. 交通与计算机, 2002, 20 (1) .

**Design and implementation of a genetic algorithm for TSP**

**Author: LiuZhen Director: Zhang Yuzhou**

**Abstract:** Traveling Salesman Problem (TSP) is a typical combinational optimization Problem. Based on its obvious advantages, TSP has been widely used in many fields of social science. Such as circuit board printing, airline planning of civil aviation, construction and design of highway network, communication node strategy setting in the network, logistics goods distribution, supermarket goods on and off shelves and other situations. At the same time, with the increase of the scale of the problem, the solution space explodes. This paper takes genetic algorithm as the problem to solve the algorithm, and analyzes and discusses the solution results of TSP. In this paper, the principle, characteristics and algorithm description of genetic algorithm are introduced by discussing the topic background and research significance. Then the TSP is modeled, and the design and implementation of genetic algorithm for solving TSP are given. After that, test data are organized for experimental simulation, and the results are in line with the expected results. Finally, the thesis is summarized and the deficiencies of the research are pointed out.

**Key words:** Travel agent problem; MATLAB simulation; operator; genetic algorithm

# 附 录 1

代码如下：

clear

clc

close all

X =[13.47,51.10

16.47,54.44

20.09,52.54

22.39,53.37

25.23,57.24

22.00,56.05

20.47,57.02

17.20,56.29

16.30,57.38

14.05,58.12

16.53,57.38

21.52,55.59

19.41,57.13

10.09,55.55

12.09,52.55

15.09,56.55

17.09,53.55

18.09,54.55

21.09,55.55

25.09,56.55];%个城市坐标位置,可以换成load CityPosition1.mat

NIND=100; %种群大小

MAXGEN=200;

Pc=0.9; %交叉概率

Pm=0.05; %变异概率

GGAP=0.9; %代沟(Generation gap)

D=Distanse(X); %生成距离矩阵

N=size(D,1); %(34\*34)

%% 初始化种群

Chrom=InitPop(NIND,N);

%% 在二维图上画出所有坐标点

% figure

% plot(X(:,1),X(:,2),'o');

%% 画出随机解的路线图

DrawPath(Chrom(1,:),X)

pause(0.0001)

%% 输出随机解的路线和总距离

disp('初始种群中的一个随机值:')

OutputPath(Chrom(1,:));

Rlength=PathLength(D,Chrom(1,:));

disp(['总距离：',num2str(Rlength)]);

disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')

%% 优化

gen=0;

figure;

hold on;box on

xlim([0,MAXGEN])

title('优化过程')

xlabel('代数')

ylabel('最优值')

ObjV=PathLength(D,Chrom); %计算路线长度

preObjV=min(ObjV);

while gen<MAXGEN

%% 计算适应度

ObjV=PathLength(D,Chrom); %计算路线长度

% fprintf('%d %1.10f\n',gen,min(ObjV))

line([gen-1,gen],[preObjV,min(ObjV)]);pause(0.0001)

preObjV=min(ObjV);

FitnV=Fitness(ObjV);

%% 选择

SelCh=Select(Chrom,FitnV,GGAP);

%% 交叉操作

SelCh=Recombin(SelCh,Pc);

%% 变异

SelCh=Mutate(SelCh,Pm);

%% 逆转操作

SelCh=Reverse(SelCh,D);

%% 重插入子代的新种群

Chrom=Reins(Chrom,SelCh,ObjV);

%% 更新迭代次数

gen=gen+1 ;

end

%% 画出最优解的路线图

ObjV=PathLength(D,Chrom); %计算路线长度

[minObjV,minInd]=min(ObjV);

DrawPath(Chrom(minInd(1),:),X)

%% 输出最优解的路线和总距离

disp('最优解:')

p=OutputPath(Chrom(minInd(1),:));

disp(['总距离：',num2str(ObjV(minInd(1)))]);

disp('-------------------------------------------------------------')

Distince

%% 计算两两城市之间的距离

%输入 a 各城市的位置坐标

%输出 D 两两城市之间的距离

function D=Distanse(a)

row=size(a,1);

D=zeros(row,row);

for i=1:row

for j=i+1:row

D(i,j)=((a(i,1)-a(j,1))^2+(a(i,2)-a(j,2))^2)^0.5;

D(j,i)=D(i,j);

end

end

Fitness

%% 适配值函数

%输入：

%个体的长度（TSP的距离）

%输出：

%个体的适应度值

function FitnV=Fitness(len)

FitnV=1./len;