五一数学建模竞赛

承　诺　书

我们仔细阅读了五一数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与本队以外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其它公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们愿意承担由此引起的一切后果。

我们授权五一数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

参赛题号（从 A/B/C 中选择一项填写）：B

参赛队号：T3837016407919

参赛组别（研究生、本科、专科、高中）：本科

所属学校（学校全称）：中国海洋大学

参赛队员：　队员 1 姓名：陈小宝

队员 2 姓名：姜常琳

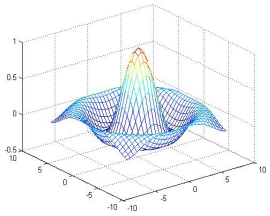
队员 3 姓名：张峻萌

联系方式：Email：qqqq0246810@outlook.com　联系电话：13627245707

日期：　2024　年　5　月　3　日

(除本页外不允许出现学校及个人信息)

五 一 数 学 建 模 竞 赛



题 目：交通网络中路径规划方案的研究

关键词：遗传算法；混合整数规划；分配交通量；期望可达率

摘　要：

本文针对城市规划问题中的交通运输分配量问题进行研究，使用混合整数规划模型求解最大期望可达率，及其对应的交通分配量。使用 MATLAB 工具中的遗传算法对模型进行求解，并结合实际情况，帮助城市规划者确定合理的交通运输分配量。

问题一中，要求规划出起点终点分别为 1 和 9、3 和 7 时的至多 5 条路径及其对应的交通分配量，使得交通网络中的期望可达率最大化。为此，我们建立了混合整数规划模型，并结合 MATLAB，使用遗传算法编程求解，得出了理想的五条路径以及其交通分配量，如正文中的表格所示。

问题二则是问题一的复杂化。针对一个更为复杂的交通网络，更换了起点与终点，修正第一问的模型，调整遗传算法的相关参数，对其进行求解。除此之外，在本问中，适应度函数需要评估任意 5 条路段出现突发状况时的期望可达率。我们的求解结果详见正文。

问题三是在问题二的基础上增加了路径容量的约束，对于问题二中建立的模型只需要增加约束条件即可，也就是需要在原有的模型基础上加入路段容量限制，并确保在任意 5 条路段出现突发状况时，网络中所有交通需求的期望可达率最大化。对此，可用 MATLAB 添加约束的方式求解。

问题四则是在问题三模型的基础上，考虑新修建的 6 条路段，并优化这些新建路段的位置，以最大化在任意 5 条路段出现突发事故时网络中所有交通需求的期望可达率。使用遗传算法来确定新建路段的最优位置和优化交通量的分配。

一、问题背景与重述

1.1 问题背景

随着经济的发展和社会的不断进步，城市化进程持续推进，人们对于出行安全性和多样化出行方式的要求逐渐提高。未来新城将自动驾驶技术作为其主导交通模式，将其整合到未来新城的交通需求规划中，以期建立更高效、更可持续的新型城市交通网络。然而，对于交通需求的不同分配会影响城市交通运行效率，交通可达率不同也会对市民出行造成不同影响。因此，对于未来新城背景下的交通需求规划与可达率问题的研究显得尤为重要。

1.2 问题提出

根据附件中给出的不同交通网络的交通需求和路段容量，建立数学模型，研究解决以下问题。

(1)分析计算某小型交通网络在一定条件下期望可达率最大时的规划路径及其分配交通量。

(2)分析计算某大型交通网络在一定条件下期望可达率最大时的规划路径及其分配交通量。

(3)分析计算某大型交通网络在一定条件下期望可达率最大时的规划路径及其分配交通量，各路段上的交通量不能超过路段容量。

(4)分析计算新建路段方案在交通需求期望可达率最大，且交通量不超过现有路段的容量时的五种方案及其可达率。

二、问题分析

2.1 问题一的分析

问题一要求在一个小型交通网络中，找到相应起点与终点的至多 5 条路径以及不同路径所对应的交通分配量。为解决此问题，我们可以建立相关优化模型。找到目标函数与限制条件，我们需要在满足相关限制条件的情况下使期望可达率最大化。为此，可以应用遗传算法并用 MATLAB 进行实现，通过迭代求得最优解。

2.2 问题二的分析

问题二我们面对的是一个复杂的大型交通网络，仍然需要为相应起点与终点之间规划出相应路径并找到对应的交通分配量。除此之外，问题二出现突发状况的路段也从一条变成了五条。因此，问题二可以看做复杂版的问题一，仍然应用遗传算法求得最优解。

2.3 问题三的分析

问题三的网络与问题二相同，但是增加了路段容量的上限这一限制条件，是对交通分配量的限制。对于问题三，我们可以将其看做复杂版的问题二，仍然使用遗传算法，但是在算个体适应度的时候给最终的目标函数添加一个罚函数，通过迭代求得最优解。

2.4 问题四的分析

对于问题四，我们需要在问题二网络的基础上新修建六条满足一定要求的路段，使得任意五条路段出现突发状况时，网络中所有交通需求的期望可达率最大。问题四虽然继续使用遗传算法，但是复杂度较前三问有了较大提升。

三、模型假设

1. 假设各路段出现事故是相互独立的。
2. 假设网络中的节点和路段构成的拓扑结构不发生改变，保持原有的交通网络形态和连接方式。
3. 假设在任何路段上，只有通过预设路径的车辆会对该路段产生交通量，无其他非路径车辆造成的干扰。
4. 在模型中，始终保持整个交通网络的可用性和可靠性应当是一个优先考虑的目标，这关系到整个交通网络在面对突发情况时的适应性。
5. 假设模型中不存在其他外部干扰，如天气、突发灾害、非法停车等因素，这些因素可能对交通流动产生影响。
6. 假设模型中假设路段具有一定的容忍度，即在超出最大容量后依然可以承载有限的流量，防止交通网络在局部出现堵塞。

四、符号说明

符号	意义
$AR_{(s,t)}$	交通需求可达率
F_r	分配到路径 r 上的交通量
$T(s,t)$	从 s 到 t 的总交通需求量
$EAR_{(s,t)}$	期望可达率
C_e	路段e的容量上限
$a_{r,(s,t)}$	判断是否使用 r 的决策函数

五、模型建立与求解

5.1 问题一模型的建立与求解

5.1.1 问题一的分析

首先，根据题目意思，我们定义交通需求可达率的计算公式。给定一个起点和终点对(s,t)，交通需求可达率是从起点 s 到终点 t 的已分配交通量中，能够实际到达目的地的交通量比例。计算公式为：

$$AR_{(s,t)} = \frac{\sum_{r \in R_{(s,t)}} F_{r,(s,t)} \cdot a_{r,(s,t)}}{T_{(s,t)}} \tag{5-1-1}$$

其中 $R(s,t)$是从起点 s 到终点 t 的所有可能路径的集合，F_r是分配到路径 r 上的交通量，$T(s,t)$是从起点 s 到终点 t 对(s,t)的总交通需求量。

由于每个路段出现突发状况的概率相同，且假设这些事件是独立的，那么交通需求的期望可达率可以通过计算所有可能的突发状况下可达率的平均值来得到。

在交通网络中，假设每个路段出现突发状况的概率为 P，并且这些事件是独立的。对于给定的(起点,终点)对(s,t)，其交通需求可达率 $AR_{(s,t)}$取决于网络中各路段的状态以及各路径分配的交通量。

首先，定义 $R(s,t)$为(起点,终点)对 (s,t) 的所有可能路径的集合。对于每一条路径 $r \in R(s,t)$，定义 F_r 为分配到路径 r 上的交通量，$T(s,t)$为(起点,终点)对(s,t)的总交通需求量。

在没有任何路段失效的情况下，交通需求可达率 $AR_{(s,t)}$为 1，因为所有分配的交通量都能到达目的地。但是，当某个路段失效时，只有那些不经过该路段的路径上的交通量能够到达目的地。

为了计算交通需求的期望可达率$EAR_{(s,t)}$，需要考虑所有路段可能的状态组合，并计算每种状态下(起点, 终点)对 (s,t)的可达率。由于每个路段出现突发状况的概率为 P，不出现突发状况的概率为 $1-P$，我们可以使用概率论中的期望公式来计算 $EAR_{(s,t)}$。

对于(起点, 终点)对 (s,t)，其对应的交通需求的期望可达率 $EAR_{(s,t)}$可以用以下公式表示：

$$EAR_{(s,t)} = \sum_{r \epsilon R_{(s,t)}} (1-P)\frac{F_{r',(s,t)}a_{r',(s,t)}}{T_{(s,t)}} + P\frac{F_{r^*,(s,t)} \cdot a_{r^*,(s,t)}}{T_{(s,t)}} \quad (5\text{-}1\text{-}2)$$

通过这个公式，我们可以计算出每条路径r在不出现任何路段失效的情况下的可达率为 1，与实际情况相符。在此基础上，乘以该路径上分配的交通量F_r与总交通需求量 $T(s,t)$的比例，然后将所有路径的可达率相加，便可以计算出从起点 s 到终点 t 的交通需求期望可达率。

在接下来解决问题的过程中，我们的思路主要如下流程图所示。

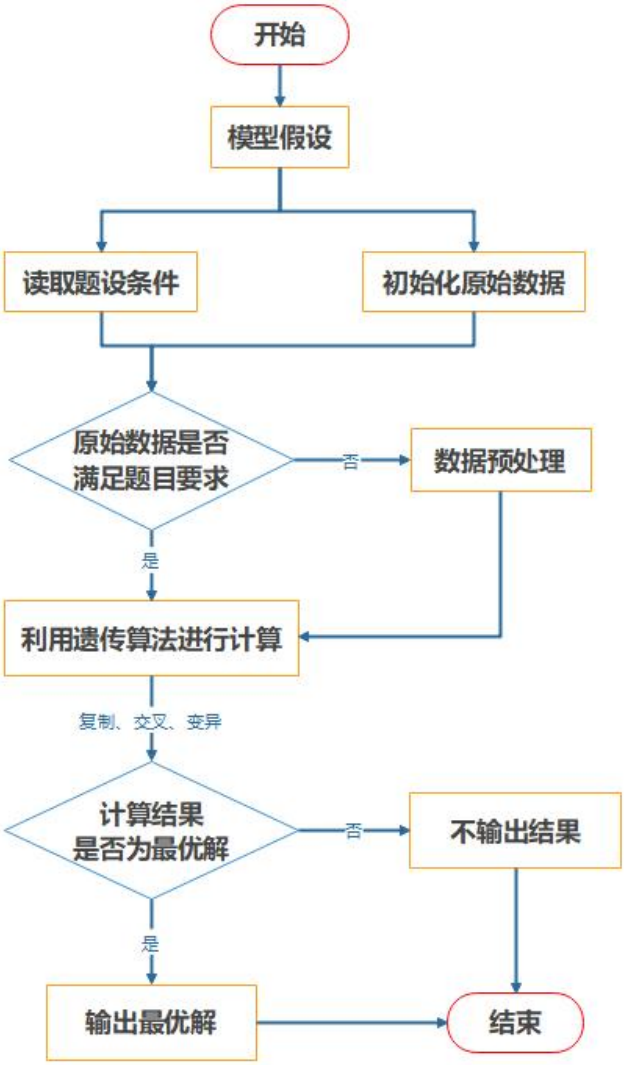


图 1：问题一思路流程图

5.1.2 问题一的简化案例分析

为了解决问题一，我们先建立一个较为简单的模型进行分析计算，再针对本问题对该模型进行推广。

首先，我们建立一个简单交通网络，如下图所示。

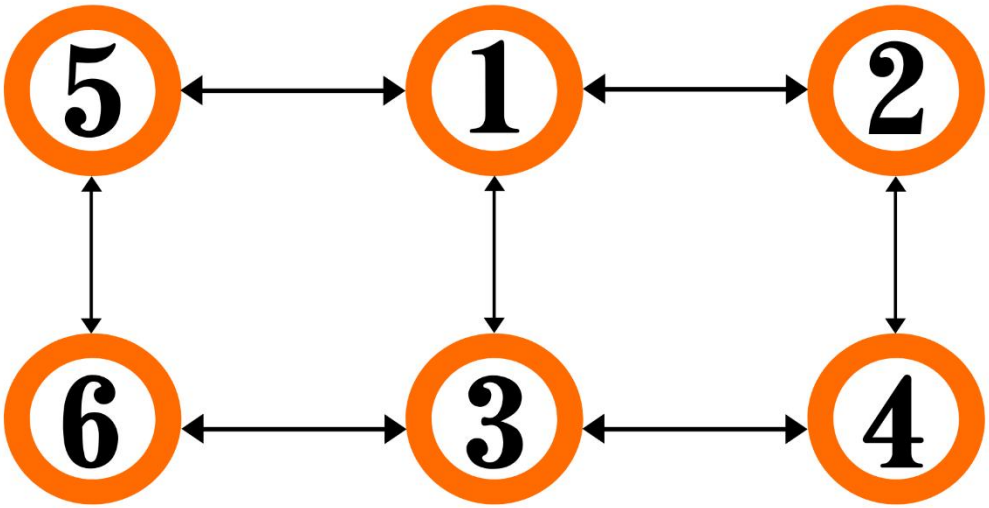


图 2：简单交通网络示意图

现在我们要给出各(起点,终点)对之间交通需求分配到对应路径上的交通量，使得网络中任意 1 条路段出现突发状况时(每个路段出现突发状况概率相同)，网络中所有交通需求的期望可达率最大。对于这张简单交通网络，我们假设需要考虑两个(起点,终点)对，分别为(1,6)和(1,4)。为了增加模型的鲁棒性，我们假设其交通需求量不相等，分别为 300 和 100。显然，我们可以对于此交通网络列出如下表格：

表 1：不同（起点，终点）对需求量与可能路径示意图

(起点,终点)对	需求量	可能的路径
(1,6)	300	1-3-6 1-5-6
(1,4)	100	1-2-4 1-3-4

接下来进行计算。由交通网络图可知，图中共有 14 条路段，故假设每条路段出现突发状况的概率为$\frac{1}{14}$。设 3-6 分配交通量为x_1,3-4 分配交通量为x_2，各条路段分配交通量如下图所示。

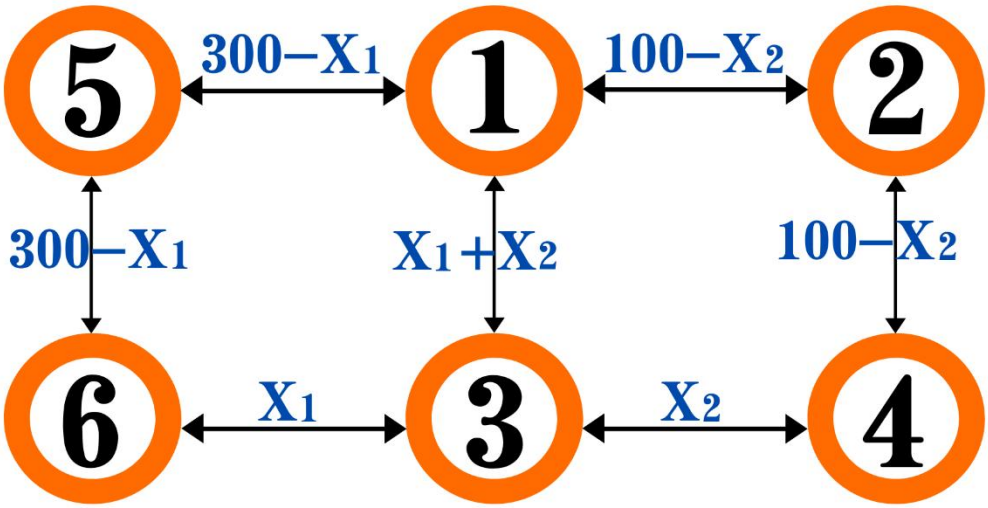


图 3：路段分配交通量示意图

则对于每条路段出现突发状况时的交通量，可以用如下公式表示：

$$X_{15} = 400 - (300 - x_1) \tag{5-3-3}$$

$$X_{56} = 400 - (300 - x_1) \tag{5-1-4}$$

$$X_{12} = 400 - (100 - x_2) \tag{5-1-5}$$

$$X_{24} = 400 - (100 - x_2) \tag{5-1-6}$$

$$X_{13} = 400 - (x_1 + x_2) \tag{5-1-7}$$

$$X_{36} = 400 - x_1 \tag{5-1-8}$$

$$X_{34} = 400 - x_2 \tag{5-1-9}$$

$$X_{51} = X_{65} = X_{21} = X_{42} = X_{31} = X_{63} = X_{43} = 400 \tag{5-1-10}$$

其中，X_{ij}表示 i-j 路段出现突发状况时整张网络的交通量。

由上述公式可知，该交通网络的期望可达率 EAR 为：

$$A = \frac{1}{14} \times (X_{15} + X_{56} + X_{12} + X_{24} + X_{13} + X_{36} + X_{34}) \tag{5-1-11}$$

$$B = \frac{1}{14} \times (X_{51} + X_{65} + X_{21} + X_{42} + X_{31} + X_{63} + X_{43}) \tag{5-1-12}$$

$$EAR = (A + B) \times \frac{1}{400} = 85.71\% \tag{5-1-13}$$

由此可知，在交通网络对称的情况下，即使交通需求量不对称，无论如何分配交通量，期望可达率 EAR 都为一常数，与问题一题设条件类似。故对于问题一而言，期望可达率也为一常数。

5.1.3 问题一模型的建立

根据上述对题目的分析接下来我们建立第一问的模型。这是一个典型的交通规划问题。为了解决这个问题，可以建立以下数学模型：

首先定义决策变量，我们假设$F_{r,(s,t)}$为从起点 s 到终点 t 经过路径 r 的交通量。而在本题中，我们希望达成的目标为使所有路径的交通需求量的期望可达率 EAR 最大。

那么我们构建如下目标函数：

$$\max f_1 = \sum_{(s,t)} \left(\sum_{r \epsilon R_{(s,t)}} (1-P) \frac{F_{r',(s,t)} a_{r',(s,t)}}{T_{(s,t)}} + P \frac{F_{r^*,(s,t)} \cdot a_{r^*,(s,t)}}{T_{(s,t)}} \right) \tag{5-1-14}$$

接下来我们寻找到在实现目标过程中的约束条件，主要有以下几个。

首先是交通量守恒，即从 s 出发并到达 t 的总交通量必须等于其总的交通需求，如下式所示：

$$\sum_{r \epsilon R_{(s,t)}} F_{r,(s,t)} = T_{(s,t)} \tag{5-1-15}$$

除此之外，我们注意到每个(起点,终点)对之间使用的路径数不超过 5，将其称之为路径选择约束，假设$a_{r,(s,t)}$为对(s,t)是否使用路径r，其中：

$$a_{r,(s,t)} = \begin{cases} 0，(s,t)未使用\ r \\ 1，(s,t)使用\ r \end{cases} \tag{5-1-16}$$

那么，根据以上分析有：

$$\sum_{r \epsilon R_{(s,t)}} a_{r,(s,t)} \leq 5 \tag{5-1-17}$$

由于分配到路径 r 上的交通量不可以超过从该起点到终点的总交通需求量，于是我们有下式：

$$F_{r,(s,t)} \leq T_{(s,t)} \cdot a_{r,(s,t)} \tag{5-1-18}$$

另外，所有交通量都非负的，即：

$$F_{r,(s,t)} \geq 0 \tag{5-1-19}$$

综上所述构建出问题一的模型。

5.1.3 问题一模型的求解

对于该模型的求解，我们可以使用遗传算法。遗传算法是一种随机并行搜索算法，其基本思想是：首先对个体进行编码，生成初始种群，然后开始进化，用适应度函数判断个体的优劣，优秀的个体将有更高的概率获得复制，交叉和变异的机会，一直到满足迭代的终止条件，从而得到最优解。

在接下来解决问题的过程中，我们对于遗传算法的思路主要如下流程图所示。

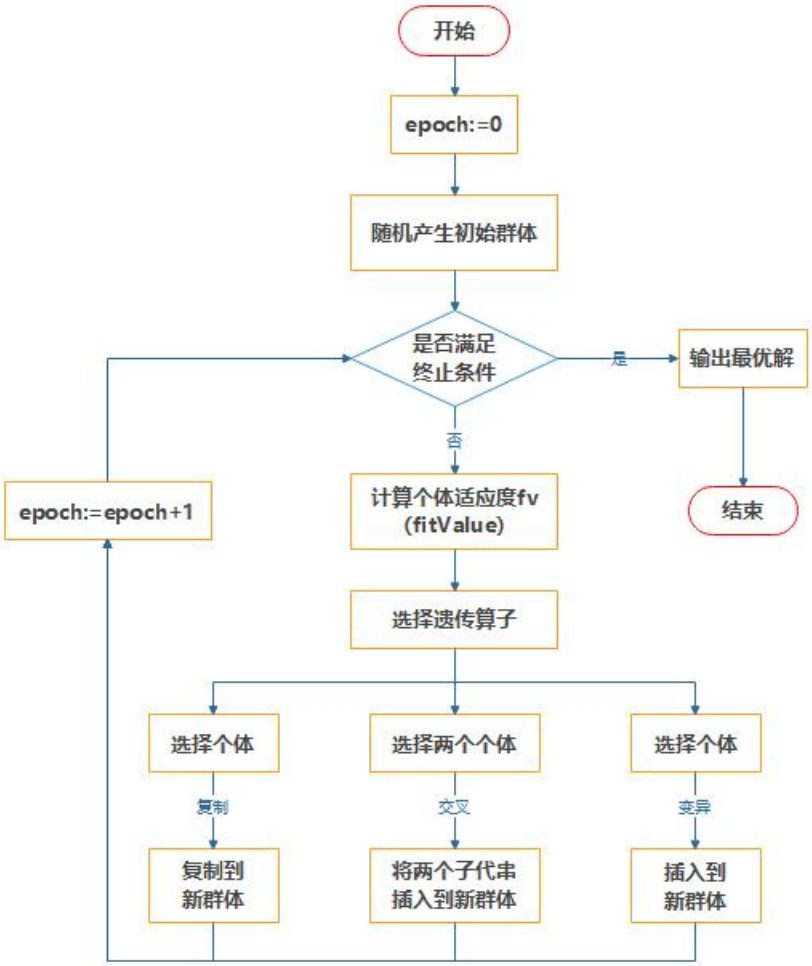


图 4：遗传算法解释图

5.1.4 问题一模型的求解结果

根据上述模型以及求解过程，我们得到了(起点,终点)对分别为(1,9)和(3,7)的情况下，相应的最优规划路径及其分配交通量，如下表所示。

表 2： 问题一结果

(起点,终点)	规划路径	分配交通量
(1,9)	1-4-5-6-9	119
	1-2-5-8-9	53
	1-2-5-6-9	2
	1-4-5-8-9	2
	1-2-3-6-9	1
(3,7)	3-6-5-4-7	59
	3-6-5-8-7	14
	3-2-5-4-7	3
	3-2-5-8-7	2

	3-6-9-8-7	2

5.2 问题二模型的建立与求解

5.2.1 问题二模型的分析

问题二在问题一的基础上扩大了规模，并加入了新的条件。与问题一相比，问题二更换了较问题一更为复杂的交通网络，并设置任意 5 条路段出现突发状况。尽管问题一和问题二的目标都是最大化网络中所有交通需求的期望可达率，但在问题二中，由于需要考虑多个路段的突发状况，这可能导致可达率的计算和最大化的方式更复杂，而模型需要更加灵活和复杂，以应对多重路段故障的影响。为了解决这个问题，我们需要对问题一的模型进行修改，以适应新的条件：即在网络中任意 5 条路段出现突发状况时，最大化所有交通需求的期望可达率。

在接下来解决问题的过程中，我们的思路主要如下流程图所示。

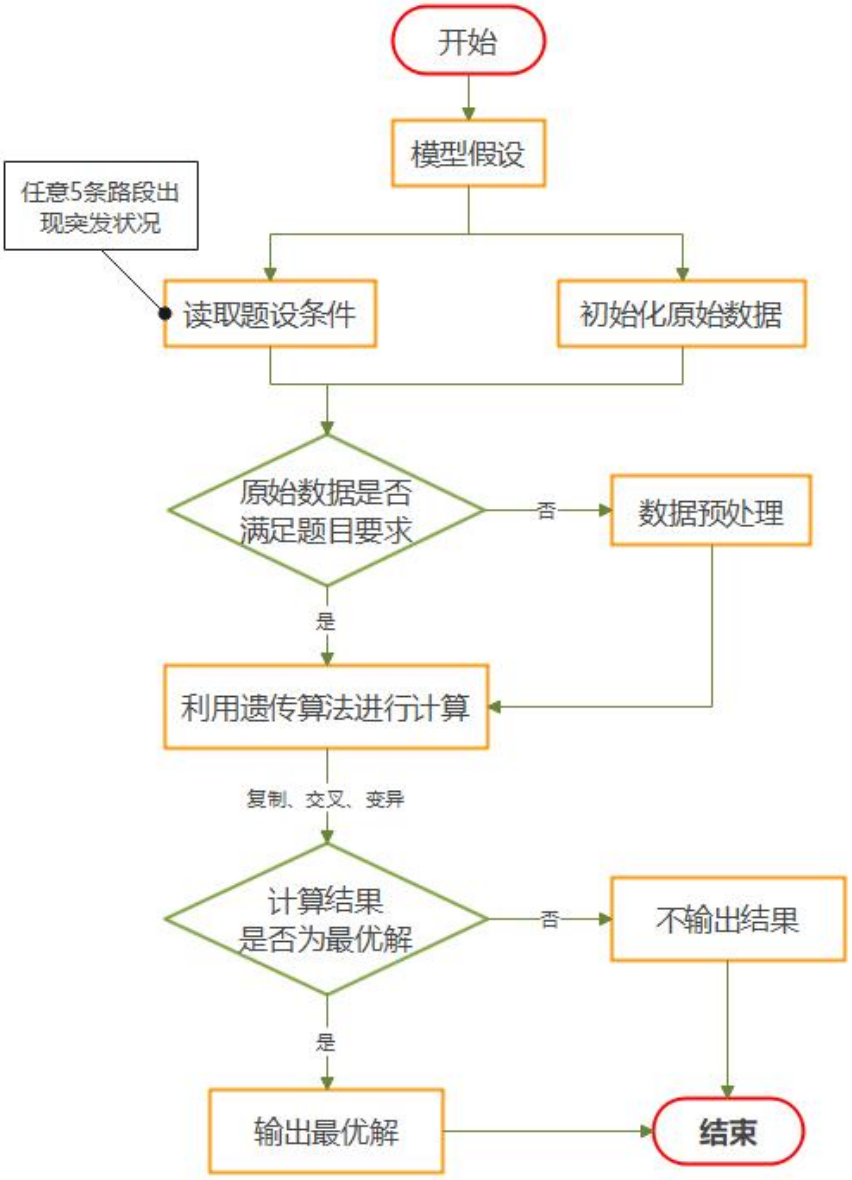


图 5：问题二思路流程图

5.2.2 问题二模型的建立

在问题一的基础上重新建立模型如下。仍然定义$F_{r,(s,t)}$为从起点 s 到终点 t 经过路径 r 的交通量，这也是我们的决策变量。此时我们的目标函数为最大化在所有可能的 5 条路段突发状况组合下，所有(起点, 终点)对的期望可达率 EAR。

由于直接考虑所有可能的 5 条路段突发状况组合会非常复杂，我们可以采用随机抽样或近似方法来估计期望可达率。一个常见的近似方法是基于场景的方法，即随机生成多组 5 条路段失效的场景，并计算每组场景下的可达率，然后取平均值作为期望可达率的估计。

假设生成了 N 组场景，其中第 i 组场景中有 5 条路段失效，这组场景下(起点, 终点)对(s,t)的期望可达率 EAR 的估计可以表示为：

$$EAR \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{(s,t)}\left(\sum_{r\epsilon R_{(s,t)}}(1-P)\frac{F_{r',(s,t)}a_{r',(s,t)}}{T_{(s,t)}}+P\frac{F_{r^*,(s,t)}\cdot a_{r^*,(s,t)}}{T_{(s,t)}}\right)\right) \tag{5-2-1}$$

$$\max f_2 = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{(s,t)}\left(\sum_{r\epsilon R_{(s,t)}}(1-P)\frac{F_{r',(s,t)}a_{r',(s,t)}}{T_{(s,t)}}+P\frac{F_{r^*,(s,t)}\cdot a_{r^*,(s,t)}}{T_{(s,t)}}\right)\right) \tag{5-2-2}$$

第二问的约束条件与第一问基本一致，首先仍然假设$a_{r,(s,t)}$为对(s,t)是否使用路径 r，其中：

$$a_{r,(s,t)} = \begin{cases} 0，(s,t)未使用\ r \\ 1，(s,t)使用\ r \end{cases} \tag{5-2-3}$$

将各约束条件整理为如下的方程组

$$\begin{cases} \sum_{r\epsilon R_{(s,t)}} F_{r,(s,t)} = T_{(s,t)} \\ \sum_{r\epsilon R_{(s,t)}} a_{r,(s,t)} \le 5 \\ F_{r,(s,t)} \le T_{(s,t)}\cdot a_{r,(s,t)} \\ F_{r,(s,t)} \ge 0 \end{cases} \tag{5-2-4}$$

综上所述构建出问题二的模型。

5.2.3 问题二模型的求解

对于该模型的求解，与问题一类似，我们可以继续使用遗传算法。然而，相对问题一，我们需要针对问题二的复杂性，对模型进行相应的调整和优化，以确保算法能够有效地搜索解空间并找到最佳解决方案。由于问题二的交通网络规模更大，且(起点,终点)对的数量更多，搜索空间变大，染色体长度增加。除此之外，在问题二中，适应度函数需要评估任意 5 条路段出现突发状况时的期望可达率，这会使该问的模型更复杂，解空间也更大。为了应对更复杂的问题二，遗传算法的初始种群可能需要更具多样性，以确保覆盖更广泛的解空间。

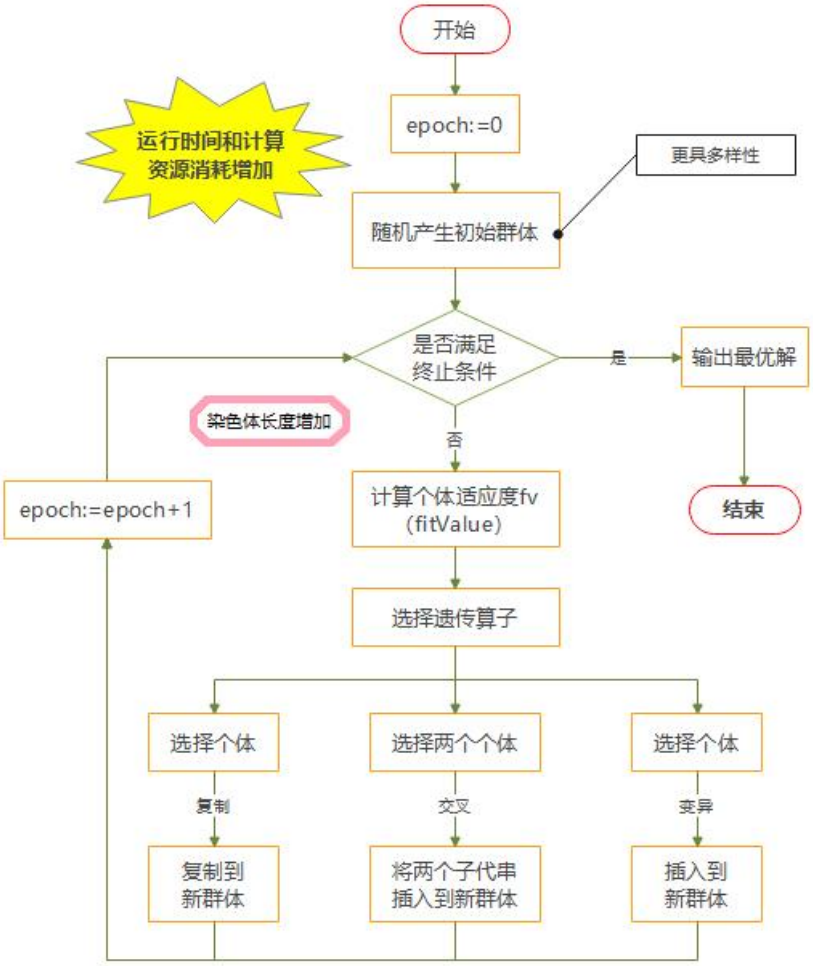


图 6：问题 2 遗传算法模型求解图

5.2.4 问题二模型的求解结果

根据上述模型以及求解过程，我们得到了(起点,终点)对分别为(27,6)和(19,25)的情况下，相应的最优规划路径及其分配交通量，如下表所示。

表 3：问题二结果

(起点,终点)	规划路径	分配交通量
(27,6)	27-35-41-8-9-5-6	41
	27-35-41-10-9-5-6	26
	27-35-28-8-9-5-6	18
	27-35-41-10-16-5-6	6
(19,25)	19-10-11-12-7-33-25	88
	19-20-11-12-7-33-25	63

	19-10-41-35-27-33-25	15
	19-10-41-35-27-30-25	8

5.3 问题三模型的建立与求解

5.3.1 问题三模型的分析

问题三是在问题二的基础上增加了路径容量的约束，对于模型只需要增加约束条件即可，也就是需要在原有的模型基础上加入路段容量限制，并确保在任意 5 条路段出现突发状况时，网络中所有交通需求的期望可达率最大化。我们继续使用遗传算法，但是在算个体适应度的时候给最终的目标函数添加一个罚函数，让不符合路径容量的交通量分配方案获得复制的概率更低，通过迭代求得最优解。

5.3.2 问题三模型的建立

仍然定义从起点 s 到终点 t 经过路径 r 的交通量$F_{r,(s,t)}$为本问的决策变量。

需要在考虑路段容量限制的情况下，最大化期望可达率 EAR，本问的目标函数表达式如下：

$$\max f_3 = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{(s,t)}\left(\sum_{r\epsilon R_{(s,t)}}(1-P)\frac{F_{r',(s,t)}a_{r',(s,t)}}{T_{(s,t)}} + P\frac{F_{r^*,(s,t)}\cdot a_{r^*,(s,t)}}{T_{(s,t)}}\right)\right) \tag{5-3-1}$$

其相应的约束条件如下所示。

首先，交通量守恒。从 s 出发并到达 t 的总交通量必须等于起点,终点对(s,t)之间的交通需求为：

$$\sum_{r\epsilon R_{(s,t)}} F_{r,(s,t)} = T_{(s,t)} \tag{5-3-2}$$

其次，要满足非负约束，即所有交通量都非负：

$$F_{r,(s,t)} \geq 0 \tag{5-3-3}$$

第三，需要满足路段容量限制，确保各路段上的交通量不超过其容量上限：

$$\sum_{e\epsilon r,(s,t)} F_{r,(s,t)}\cdot a_{r,(s,t)} \leq C_e \tag{5-3-4}$$

其中，C_e表示路段e的容量上限。

第四为路径选择约束，即每个(起点,终点)对之间使用的路径数不超过 5。假设

$$a_{r,(s,t)} = \begin{cases} 0，(s,t)未使用\ r \\ 1，(s,t)使用\ r \end{cases} \tag{5-3-5}$$

根据以上约束，我们有：

$$\sum_{r\epsilon R_{(s,t)}} a_{r,(s,t)} \leq 5 \tag{5-3-6}$$

其中，a为(起点,终点)对(s,t)是否使用路径r的决策函数。

最后，需满足路径规划交通量约束：

$$F_{r,(s,t)} \leq T_{(s,t)} \cdot a_{r,(s,t)} \tag{5-3-7}$$

综上所述构建出问题三的模型。

5.3.3 问题三模型的求解

对于该模型的求解，与问题二类似，我们可以继续使用遗传算法。然而，相对问题二，我们需要针对问题三的限制条件，对模型进行相应的调整，以确保算法能够有效地搜索解空间并找到最佳解决方案。由于问题三的交通网络路段上新增了路段交通量限制，所以为超出该限制条件的交通量分配方案添上一个惩罚项，超出限制越多，则该方案被选中的概率越小。除此之外，问题三需要计算每条路段上的总交通量，所以我们使用了一个循环来累加每条经过该路段的路径的交通量。

5.3.4 问题三模型的求解结果

根据上述模型以及求解过程，我们得到了(起点,终点)对分别为(32,9)和(17,8)的情况下，相应的最优规划路径及其分配交通量，如下表所示。

表 4：问题三结果

(起点,终点)	规划路径	分配交通量
(32,39)	32-24-7-12-11-10-19-18-2-15-13-39	40
	32-24-7-12-11-20-19-18-2-15-13-39	33
	32-24-7-12-11-10-16-36-3-42-38-39	32
	32-24-7-12-11-10-16-18-2-15-13-39	28
(17,8)	17-39-13-15-2-18-16-10-41-8	67
	17-39-38-42-3-36-6-7-9-8	42
	17-14-13-15-2-18-16-10-41-8	17

5.4 问题四模型的建立与求解

5.4.1 问题四模型的分析

对于问题四，需要在问题三模型的基础上，考虑新修建的 6 条路段，并优化这些新建路段的位置，以最大化在任意 5 条路段出现突发事故时网络中所有交通需求的期望可达率。我们仍然选择遗传算法，不过需要添加新路段的建立与选择机制，并且在得到最优解后输出这六条路段。

5.4.2 问题四模型的建立

第四问决策变量仍然为从起点 s 到终点 t 经过路径 r 的交通量，记为$F_{r,(s,t)}$。

假设x_{ij}为是否在节点 i 和节点 j 之间新建路段的决策函数：

$$x_{ij}=\begin{cases}0，不在\ i\ 与\ j\ 之间修建\\1，在\ i\ 与\ j\ 之间修建\end{cases} \tag{5-4-1}$$

目标为最大化期望可达率 EAR，同时考虑路段容量限制。相应的目标函数如下式：

$$max f_4=\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{(s,t)}\left(\sum_{r\epsilon R_{(s,t)}}(1-P)\frac{F_{r',(s,t)}a_{r',(s,t)}}{T_{(s,t)}}+P\frac{F_{r^*,(s,t)}\cdot a_{r^*,(s,t)}}{T_{(s,t)}}\right)\right) \tag{5-4-2}$$

本题的约束条件有如下几个。

首先，需要满足交通量守恒。从 s 出发并到达 t 的总交通量必须等于(起点,终点)对(s,t)之间的交通需求，其公式表达为：

$$\sum_{r\epsilon R_{(s,t)}}F_{r,(s,t)}=T_{(s,t)} \tag{5-4-3}$$

其次为非负约束，即所有交通量都非负：

$$F_{r,(s,t)}\geq 0 \tag{5-4-4}$$

然后要满足路段容量限制：确保各路段上的交通量不超过其容量上限，其数学公式形式表达为：

$$\sum_{e\epsilon r,(s,t)}F_{r,(s,t)}\leq C_e \tag{5-4-5}$$

接下来是新建路段约束，即需要确保新建路段的位置是合法的。

$$x_{ij}=\begin{cases}0，i\ 与\ j\ 之间路段不合法\\1，i\ 与\ j\ 之间路段合法\end{cases} \tag{5-4-6}$$

路径选择约束为每个(起点,终点)对之间使用的路径数不超过 5。假设

$$a_{r,(s,t)}=\begin{cases}0，(s,t)未使用\ r\\1，(s,t)使用\ r\end{cases} \tag{5-4-7}$$

根据以上约束，我们有

$$\sum_{r\epsilon R_{(s,t)}}a_{r,(s,t)}\leq 5 \tag{5-4-8}$$

上式中，$a_{r,(s,t)}$为(起点,终点)对(s,t)是否使用路径 r 的决策性函数。

最后需要满足路径规划交通量约束，即：

$$F_{r,(s,t)}\leq T_{(s,t)}\cdot a_{r,(s,t)} \tag{5-4-9}$$

综上所述，构建出问题四的模型。

5.4.3 问题四模型的求解

由于这个问题涉及到新建路段的位置选择以及交通量的分配，因此求解过程会相对复杂。可以分阶段进行求解：

首先，随机选择新建路段的位置（即确定x_{ij}的值）。然后，在确定了新建路段的位置后，使用遗传算法来优化交通量的分配。在多次选择不同的路段后，比较每一次得到的最优函数值，留下得分最高的交通量分配方案，并输出新建的路段。

5.4.4 问题四模型的求解结果

根据上述模型以及求解过程，我们得到了新建路段方案在交通需求期望可达率最大，且交通量不超过现有路段的容量时的五种方案及其可达率，如下表所示。

表 5：问题四结果

	新建路段1	新建路段2	新建路段3	新建路段4	新建路段5	新建路段6	可达率
方案 1	36-5	32-33	14-39	39-14	19-11	29-26	95.862
方案 2	27-7	15-38	32-7	16-9	2-36	16-19	95.477
方案 3	6-1	42-13	42-2	33-30	27-25	7-27	92.073
方案 4	17-13	12-27	35-27	36-2	0-36	36-18	91.342
方案 5	18-10	6-22	15-3	33-30	36-5	24-33	91.024

六、模型优缺点

6.1 模型优点

（1）跳出局部最优解：遗传算法是在搜索空间内全局搜索最佳解，所以不会陷入局部最优。可以通过保留多个解来提高搜索范围，并在演化过程中不断优化解决方案。

（2）高效性：遗传算法在搜索空间中并行地进行多个解的搜索，因此在本题中可以更快地找到潜在的最优路径。

（3）较强的适应性：遗传算法能够适应问题的变化和复杂性，比如题目要求中路径交通量的大小变化，可以通过基因交叉和变异的操作，在搜索空间中进行探索和优化，从而找到更好的解决方案。

（4）无需特定领域知识：遗传算法不需要启发式的规则，只需要目标函数和适应度函数即可。

6.2 模型缺点

（1）参数设置困难：遗传算法中存在大量参数，如群体大小、交叉概率、变异概率等，这些参数对算法的性能产生影响。优化算法的性能高度依赖于参数的选择和调整，而这一过程相对复杂。

（2）收敛速度慢：遗传算法通常需要较多的迭代次数来达到收敛，特别是在解空间较大或复杂问题上，比如第二个问题中复杂的大型交通网络。

（3）计算资源需求较高：由于遗传算法需要维护一个群体中多个个体，并对其进行评估、选择、交叉和变异等操作，因此可能需要较大的计算资源和存储空间。

七、参考文献

[1]李德刚.综合运输网中的通道分析与系统配置研究[D].西南交通大学,2006.

[2]汤洪，徐旺.基于改进遗传算法的资源优化调度方法[J].现代电子技术,2024,47(09):169-172.DOI:10.16652/j.issn.1004-373x.2024.09.030.

[3]张杨阳,张革佚,贺娜,等.基于改进型遗传算法的多目标配送线路优化仿真[J].物流工程与管理,2023,45(10):33-37+3.

[4]汪松泉.遗传算法在组合优化中的应用研究[D].安徽大学,2010.
[5]罗翼飞.考虑动态流量分配的非机动车道决策优化模型[D].大连交通大学,2023.DOI:10.26990/d.cnki.gsltc.2023.000175.
[6]周云鹏,题正义.遗传算法在组合优化中的应用[J].辽宁工程技术大学学报,2005,(S1):283-285.

附录

附录 1 代码

第一问所用代码

```
clear;clc
% 参数
load m1.mat;load paths1.mat;load a1.mat;
% 定义遗传算法参数
popsize = 20; % 群体大小 可视为有 20 条染色体
num_vars = size(paths1,1); % 决策变量个数
bits_per_var = 8; % 每个决策变量所需的二进制位数
chromlength = num_vars * bits_per_var; % 染色体长度（个体长度）
pc = 0.6; %交叉概率
pm = 0.1; %变异概率
G = 5 ; %迭代次数

% 定义问题参数
p = 1/num_vars; % 每条边的故障概率
xlim = [0,max(max(m1))]; % 决策变量上下限
k = 1; % 故障路段数量
n = sum(sum(a1)); % 图中路段数

y = zeros(1,G); % 记录每代最优个体对应的函数值
pop = round(rand(popsize, chromlength)); %随机产生初始群体
decpop = bintodec(pop ,popsize, num_vars, bits_per_var, xlim); % 计算初代解对应十进制
[fx,decpop] = objective_function1(decpop,p,paths1,n,m1); % 计算初代解的函数值
pop = dectobin(decpop, num_vars, bits_per_var, xlim);

for j = 2 : G
fitvalue = fx ; % 适应度映射
newpop = copyx(pop,fitvalue,popsize); %复制
newpop = crossover(newpop, pc, popsize,chromlength ); %交叉
newpop = mutation(newpop,pm, popsize,chromlength); %变异
newdecpop = bintodec(newpop ,popsize, num_vars, bits_per_var, xlim);
[new_fx,new_decpop] = objective_function1(decpop,p,paths1,n,m1); %计算新解目标函数
newpop = dectobin(new_decpop, num_vars, bits_per_var, xlim);
%计算新群体中每个个体的适应度
new_fitvalue = new_fx; %计算新群体中每个个体的适应度
index = find(new_fitvalue > fitvalue) ;
pop(index, :) = newpop(index,:) ; % 更新得到最新解
```

```
fx(index, :) = new_fx(index,:) ;
% 找出更新后的个体最优函数值
[bestindividual,bestindex] = max(fx) ;
y(j)=bestindividual; % 记录每一代的最优函数值
x = new_decpop(bestindex,:) ;
end
[ymax, max_index] = max(y);
disp(['最优解为：', num2str(ymax)]);
disp(['最优交通量为：', num2str(x)]);
```

第一问目标函数

```
function [obj,fpop] = objective_function1(decpop,p,paths1,n,m1)
nn = size(decpop,1);
obj = zeros(nn,1); % 目标函数
fpop = [];
for k = 1:nn
pop = decpop(k,:);
pop = Modify(pop,paths1,m1); % 修正解使其满足约束条件
aa = zeros(size(m1)); % 统计路段交通量
% 随机设置故障路段
one_b = ones(n,1);
b = diag(one_b) .* p;
b(b == 0) = 1 - p;
AR = zeros(1,n);
% 遍历每对源-目的地
for i = 1:numel(paths1)
path = paths1{i};
m = m1(path(1),path(end)); % 对应交通量
mm = pop(1,i); % 随机交通量
AR(1,i) = mm/m;
for j = 1:size(path,2) - 1
aa(path(j),path(j+1)) = aa(path(j),path(j+1)) + mm/m; % 将交通量赋值到路段中
end
end
aa = aa';
a_a = aa(aa ~= 0);
% 目标函数值
f = mean(b * a_a);
fpop = [fpop;pop];
obj(k,1) = f;
end
end
```

第二问所用代码

```
clear;clc
% 参数
load m2.mat;load paths2.mat;load a2.mat;
% 定义遗传算法参数
popsize = 20; % 群体大小 可视为有 20 条染色体
num_vars = size(paths2,1); % 决策变量个数
bits_per_var = 8; % 每个决策变量所需的二进制位数，由决策变量上限决定
chromlength = num_vars * bits_per_var; % 染色体长度（个体长度）
pc = 0.6; %交叉概率
pm = 0.1; %变异概率
G = 5 ; %迭代次数

% 定义问题参数
p = 1/num_vars; % 每条边的故障概率
xlim = [0,max(max(m2))]; % 决策变量上下限
k = 1; % 故障路段数量
n = sum(sum(a2)); % 图中路段数
N = 10; % 生成 N 组场景

y = zeros(1,G); % 记录每代最优个体对应的函数值
pop = round(rand(popsize, chromlength)); %随机产生初始群体
decpop = bintodec(pop ,popsize, num_vars, bits_per_var, xlim); % 计算初代解对应十进制
[fx,decpop] = objective_function2(decpop,p,paths2,n,m2,N,a2); % 计算初代解的函数值
pop = dectobin(decpop, num_vars, bits_per_var, xlim);

for j = 2 : G
fitvalue = fx ; % 适应度映射
newpop = copyx(pop,fitvalue,popsize); %复制
newpop = crossover(newpop, pc, popsize,chromlength ); %交叉
newpop = mutation(newpop,pm, popsize,chromlength); %变异
newdecpop = bintodec(newpop ,popsize, num_vars, bits_per_var, xlim);
[new_fx,new_decpop] = objective_function2(decpop,p,paths2,n,m2,N,a2); %计算新解目标函数
newpop = dectobin(new_decpop, num_vars, bits_per_var, xlim);
%计算新群体中每个个体的适应度
new_fitvalue = new_fx; %计算新群体中每个个体的适应度
index = find(new_fitvalue > fitvalue) ;
pop(index, :) = newpop(index,:) ; % 更新得到最新解
fx(index, :) = new_fx(index,:) ;
% 找出更新后的个体最优函数值
[bestindividual,bestindex] = max(fx) ;
y(j)=bestindividual; % 记录每一代的最优函数值
```

```
x = new_decpop(bestindex,:) ;
end
[ymax, max_index] = max(y);
disp(['最优解为：', num2str(ymax)]);
disp(['最优交通量为：', num2str(x)]);

indices = [];
% 遍历 paths2 的每一行
for i = 1:length(paths2)
% 检查当前行的第一个元素和最后一个元素
if paths2{i}(1) == 27 && paths2{i}(end) == 6
% if paths2{i}(1) == 19 && paths2{i}(end) == 25
% 如果是，则将该行的索引添加到 indices 向量中
indices = [indices; i];
end
end
```

第二问目标函数

```
function [obj,fpop] = objective_function2(decpop,p,paths2,n,m2,N,a2)
nn = size(decpop,1);
obj = zeros(nn,1); % 目标函数
fpop = [];
for k = 1:nn
pop = decpop(k,:);
pop = Modify(pop,paths2,m2); % 修正解使其满足约束条件
aa = zeros(size(a2)); % 统计路段交通量
% 随机设置故障路段
b = zeros(N,n);
for t = 1:N
cols = randsample(n, 5, true);
b(t, cols) = 1;
end
b(b == 1) = p;
b(b == 0) = 1 - p;
AR = zeros(1,n);
% 遍历每对源-目的地
for i = 1:numel(paths2)
path = paths2{i};
m = m2(path(1),path(end)); % 对应交通量
mm = pop(1,i); % 随机交通量
AR(1,i) = mm/m;
for j = 1:size(path,2) - 1
aa(path(j),path(j+1)) = aa(path(j),path(j+1)) + mm/m; % 将交通量赋值到路段中
end
```

```
end
aa(isnan(aa)) = 0;
logical_idx = a2' == 1;
aa = aa';
a_a = aa(logical_idx);
% 目标函数值
f = mean(b(1,:) * a_a);
fpop = [fpop;pop];
obj(k,1) = f;
end
end
```

第三问代码

```
clear;clc
% 参数
load m2.mat;load m3.mat;load paths2.mat;load a2.mat;
% 定义遗传算法参数
popsize = 20; % 群体大小 可视为有 20 条染色体
num_vars = size(paths2,1); % 决策变量个数
bits_per_var = 8; % 每个决策变量所需的二进制位数，由决策变量上限决定
chromlength = num_vars * bits_per_var; % 染色体长度（个体长度）
pc = 0.6; %交叉概率
pm = 0.1; %变异概率
G = 5 ; %迭代次数

% 定义问题参数
p = 1/num_vars; % 每条边的故障概率
xlim = [0,max(max(m2))]; % 决策变量上下限
n = sum(sum(a2)); % 图中路段数
N = 10; % 生成 N 组场景

y = zeros(1,G); % 记录每代最优个体对应的函数值
pop = round(rand(popsize, chromlength)); %随机产生初始群体
decpop = bintodec(pop ,popsize, num_vars, bits_per_var, xlim); % 计算初代解对应十进制
[fx,decpop] = objective_function3(decpop,p,paths2,n,m2,N,a2,m3); % 计算初代解的函数值
pop = dectobin(decpop, num_vars, bits_per_var, xlim);

for j = 2 : G
fitvalue = fx ; % 适应度映射
newpop = copyx(pop,fitvalue,popsize); %复制
newpop = crossover(newpop, pc, popsize,chromlength ); %交叉
newpop = mutation(newpop,pm, popsize,chromlength); %变异
```

```
newdecpop = bintodec(newpop ,popsize, num_vars, bits_per_var, xlim);
[new_fx,new_decpop] = objective_function3(decpop,p,paths2,n,m2,N,a2,m3); %计算新解目标函数
newpop = dectobin(new_decpop, num_vars, bits_per_var, xlim);
%计算新群体中每个个体的适应度
new_fitvalue = new_fx; %计算新群体中每个个体的适应度
index = find(new_fitvalue > fitvalue) ;
pop(index, :) = newpop(index,:) ; % 更新得到最新解
fx(index, :) = new_fx(index,:) ;
% 找出更新后的个体最优函数值
[bestindividual,bestindex] = max(fx) ;
y(j)=bestindividual; % 记录每一代的最优函数值
x = new_decpop(bestindex,:) ;
end
[ymax, max_index] = max(y);
disp(['最优解为：', num2str(ymax)]);
disp(['最优交通量为：', num2str(x)]);

indices = [];
% 遍历 paths2 的每一行
for i = 1:length(paths2)
% 检查当前行的第一个元素和最后一个元素
% if paths2{i}(1) == 32 && paths2{i}(end) == 39
if paths2{i}(1) == 17 && paths2{i}(end) == 8
% 如果是，则将该行的索引添加到 indices 向量中
indices = [indices; i];
end
end
```

第三问目标函数：

```
function [obj,fpop] = objective_function3(decpop,p,paths2,n,m2,N,a2,m3)
nn = size(decpop,1);
obj = zeros(nn,1); % 目标函数
fpop = [];
for k = 1:nn
pop = decpop(k,:);
pop = Modify(pop,paths2,m2); % 修正解使其满足约束条件
aa = zeros(size(a2)); % 统计路段 AR
a_m3 = zeros(size(a2)); % 统计路段交通量
% 随机设置故障路段
b = zeros(N,n);
for t = 1:N
cols = randsample(n, 5, true);
b(t, cols) = 1;
```

```
end
b(b == 1) = p;
b(b == 0) = 1 - p;
AR = zeros(1,n);
% 遍历每对源-目的地
for i = 1:numel(paths2)
path = paths2{i};
m = m2(path(1),path(end)); % 对应交通量
mm = pop(1,i); % 随机交通量
AR(1,i) = mm/m;
for j = 1:size(path,2) - 1
aa(path(j),path(j+1)) = aa(path(j),path(j+1)) + mm/m; % 将交通量赋值到路段中
a_m3(path(j),path(j+1)) = a_m3(path(j),path(j+1)) + mm;
end
end
aa(isnan(aa)) = 0;
logical_idx = a2' == 1;
aa = aa';
a_a = aa(logical_idx);
% 罚函数
a_diff = a_m3 - m3;
if sum(sum(a_diff > 0))
penalty = sum(sum((a_m3 - m3) .* (a_diff > 0)));
else
penalty = 0;
end
% 目标函数值
f = mean(b * a_a) - 100000 * penalty;
fpop = [fpop;pop];
obj(k,1) = f;
end
end
```

第四问所用代码

```
clear;clc
% 参数
load m2.mat;load m3.mat;load paths2.mat;load a2.mat;load a4.mat;
% 定义遗传算法参数
popsize = 20; % 群体大小 可视为有 20 条染色体
num_vars = size(paths2,1); % 决策变量个数
bits_per_var = 8; % 每个决策变量所需的二进制位数，由决策变量上限决定
chromlength = num_vars * bits_per_var; % 染色体长度（个体长度）
pc = 0.6; %交叉概率
pm = 0.1; %变异概率
```

```
G = 5 ; %迭代次数

% 定义问题参数
p = 1/num_vars; % 每条边的故障概率
xlim = [0,max(max(m2))]; % 决策变量上下限
k = 1; % 故障路段数量
n = sum(sum(a2)); % 图中路段数
N = 10; % 生成 N 组场景

y = zeros(1,G); % 记录每代最优个体对应的函数值
pop = round(rand(popsize, chromlength)); %随机产生初始群体
decpop = bintodec(pop ,popsize, num_vars, bits_per_var, xlim); % 计算初代解对应十
进制
[fx,decpop,links_indices] =
objective_function4(decpop,p,paths2,n,m2,N,a2,m3,a4); % 计算初代解的函数值
pop = dectobin(decpop, num_vars, bits_per_var, xlim);

for j = 2 : G
fitvalue = fx ; % 适应度映射
newpop = copyx(pop,fitvalue,popsize); %复制
newpop = crossover(newpop, pc, popsize,chromlength ); %交叉
newpop = mutation(newpop,pm, popsize,chromlength); %变异
newdecpop = bintodec(newpop ,popsize, num_vars, bits_per_var, xlim);
[new_fx,new_decpop,new_links_indices] =
objective_function4(decpop,p,paths2,n,m2,N,a2,m3,a4); %计算新解目标函数
newpop = dectobin(new_decpop, num_vars, bits_per_var, xlim);
%计算新群体中每个个体的适应度
new_fitvalue = new_fx; %计算新群体中每个个体的适应度
index = find(new_fitvalue > fitvalue) ;
pop(index, :) = newpop(index,:) ; % 更新得到最新解
fx(index, :) = new_fx(index,:) ;
if max(new_fitvalue) > max(fitvalue)
links_indices = new_links_indices;
end
% 找出更新后的个体最优函数值
[bestindividual,bestindex] = max(fx) ;
y(j)=bestindividual; % 记录每一代的最优函数值
x = new_decpop(bestindex,:) ;
end
[ymax, max_index] = max(y);
disp(['最优解为: ', num2str(ymax)]);
[row,column] = find(a4 == 1);
selected_rows = row(links_indices);
selected_columns = column(links_indices);
```

```
fx = objective_function4(x,p,paths2,n,m2,N,a2,m3,a4); % 最终结果
```

第四问目标函数

```
function [obj, fpop,links_indices] = objective_function4(decpop, p, paths2, n, m2,
N, a2, m3, a4)
nn = size(decpop,1);
obj = zeros(nn,1);
fpop = [];
% 增加新路段的选择机制
% 从可选择的路段中随机选择 6 个
[row,column] = find(a4 == 1);
links_indices = randperm(length(row), 6);
for i = 1:length(links_indices)
a2(row(links_indices(i)), column(links_indices(i))) = 1;
end
% 更新路段数目包含新建路段
n = n + 6;
for k = 1:nn
pop = decpop(k,:);
pop = Modify(pop, paths2, m2);
aa = zeros(size(a2));
a_m3 = zeros(size(a2));
% 随机设置故障路段，现在包括新建路段
b = zeros(N, n);
for t = 1:N
cols = randsample(n, 5, true);
b(t, cols) = 1;
end
b(b == 1) = p;
b(b == 0) = 1 - p;
AR = zeros(1,n);
% 遍历每对源-目的地
for i = 1:numel(paths2)
path = paths2{i};
m = m2(path(1), path(end));
mm = pop(1, i);
AR(1, i) = mm / m;
for j = 1:size(path, 2) - 1
aa(path(j), path(j+1)) = aa(path(j), path(j+1)) + mm / m;
a_m3(path(j), path(j+1)) = a_m3(path(j), path(j+1)) + mm;
end
end
aa(isnan(aa)) = 0;
logical_idx = a2' == 1;
```

```
aa = aa';
a_a = aa(logical_idx);
if size(a_a,1) > 138
a_a = a_a(1:138,:);
elseif size(a_a,1) < 138
num_rows_to_add = 138 - size(a_a,1);
a_a = [a_a; zeros(num_rows_to_add, size(a_a,2))];
end
% 罚函数
a_diff = a_m3 - m3;
penalty = sum(sum((a_m3 - m3) .* (a_diff > 0)));
% 目标函数值
f = mean(b * a_a) - 100000 * penalty;
fpop = [fpop; pop];
obj(k, 1) = f;
end
end
```