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：T3837966976539

参赛组别（研究生、本科、专科、高中）：本科

所属学校（学校全称）：浙江大学

参赛队员：　队员 1 姓名：戚铭宇

　　队员 2 姓名：张承

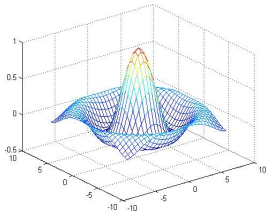
　　队员 3 姓名：陈景颖

联系方式：　Email：19357571847@163.com　联系电话：19357571847

日期：　2024 年 5 月 1 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：基于交通需求的路径规划和需求分配研究

关键词：动态规划，图论，多目标复杂规划，蒙特卡洛模拟，模拟退火算法

摘　要：

本文针对现代城市交通体系的动态适应性和灵活性需求，提出了一种融合最短路径选择与实时交通数据分析的综合路径规划策略，旨在优化交通网络的可达性和运行效率。研究围绕四个递进的问题展开，从基础的小型交通网络到复杂的大型网络，并逐步引入突发状况下网络可达性的最大化、路段容量限制、以及新路段建设的决策分析。

首先，针对单一突发状况，建立数学模型并利用模拟退火算法，为小型网络找到交通需求的最佳分配方案，确保高期望可达率。随后，模型扩展至大型网络，采用蒙特卡洛模拟处理多路段同时受阻的情形，维持高效交通流调度。在此基础上，加入路段容量约束，确保交通分配量不超载。最后，探讨了在现有网络中新增线路的策略，以进一步提升网络韧性与效率。

由于摘要篇幅受限，问题 1 到 4 的答案详见本文中“5.1.3 运行结果”，“5.2.3 运行结果”，“5.3.3 运行结果”，“5.4.3 运行结果”中的表 2 到 5。

一、问题重述

1.1 问题背景

随着城市化进程的不断加速及智能驾驶科技的日益普及，现代城市交通体系正经历前所未有的变革，对动态适应性和灵活性提出了更高要求。在此背景下，传统的静态路径规划策略——单纯依据地理距离最短来选取通行路径，已难以满足当前需求，因其未能充分考虑实时路况波动，包括交通流量的瞬息万变与不可预见的路网中断情况。因此，探索一种融合起点与终点间最短路径选择与实时道路状态评估的综合路径规划及交通需求分配策略显得尤为重要，旨在最大化整个交通网络的预期可达性与运行效率。本文致力于打破传统框架限制，通过整合最短路径算法与实时交通数据分析，提出创新解决方案，以确保在面对复杂多变的城市交通挑战时，能够实现交通流的高效调度与优化配置，进而提升城市的整体交通出行体验与可达性水平。

1.2 问题重述

①问题 1

在一个小型交通网络（图 2）的背景下，考虑到所有车辆均采用自动驾驶技术，我们需要设计一个交通需求分配方案，以最大化交通网络在任意一条路段发生突发状况时的期望可达率。交通需求数据由附件 1 提供，每一对起点终点的交通需求需被合理分配到不超过 5 条的预设路径上，优先考虑最短路径。目标是通过数学模型确定每条路径应分配的交通量，确保在任何单一路段受阻时，整个网络的交通需求仍能以最大可能的比例得到满足。

②问题 2

扩展到一个稍大规模的交通网络（图 3），面对相同的自动驾驶技术环境，问题升级为在任意 5 条路段同时发生突发状况的情况下，如何最优化交通需求分配，以实现整个网络交通需求的期望可达率最大化。交通需求数据来源于附件 2，依然遵循每对起点终点间最多 5 条路径的选择原则，且优先考虑最短路径。该问题要求建立数学模型来指导交通量在不同路径间的分配，确保即使在多条路径不可用时，网络整体的交通流畅性依旧能得到保障。

③问题 3

在交通网络 3 中，除了考虑前文所述的自动驾驶交通需求规划外，还引入了各路段的容量上限（见附件 3）。在任意 5 条路段同时发生故障的设定下，需再次优化交通需求分配，以最大化期望可达率，同时确保任一路径上的交通流量不会超出相应路段的容量限制。这要求在满足交通流约束的条件下，通过数学模型来精准分配每对(起点,终点)的交通需求到各条路径，以达到最优的交通网络性能。

④问题 4

在问题 3 的交通网络基础上，考虑新修建 6 条单向直线且长度为单位 1 的路段，以进一步优化网络的可达性和效率。新路段的选择需遵循特定规则：只能在现有网络内部节点间直接连接，不得跨越其他路段，并且假设新建路段容量无限大，不构成流量限制。通过数学模型，需确定一组新建路段方案，使得在任意 5 条路段（包括新建路段）发生故障时，整个网络的交通需求期望可达率达到最高，同时遵守原有的路段容量限制。

二、问题假设

①**交通网络结构**

所有提及的交通网络（图 2、图 3）都是明确定义的有向图，其中节点代表交汇点，边代表路段，且每条边的长度统一为 1 单位。

②**交通需求**

各(起点,终点)对之间的交通需求数据准确无误，且为已知量，存储于附件中。需求量代表从某起点到某终点的车辆数，需求量为正整数。

③**路径选择**

对于每一对(起点,终点)，最多考虑 5 条路径，路径长度以经过路段的数量来衡量。若存在多条路径且没有明显优劣之分，优先选择最短的路径。

④**突发状况概率**

任意一条路段发生突发状况的概率相等，且每次只考虑单个或特定数量路段同时发生事故的情况。问题 2 中只考虑单个路段受阻情形，因此各个路段受阻的概率均相同，样本空间有限且每个样本点概率相等，是古典概型，但由于只能出现一条受阻道路，各个路段受阻概率并不相互独立。问题 3 和 4 中考虑任意五条路段受阻情形，假设与问题 1 类似。

⑤**路段容量**

在考虑路段容量限制的情况下（问题 3 和 4），各路段的最大承载能力明确给出，超出此容量的交通量将不被允许。

⑥**新建路段**

在问题 4 中，新修路段的提议需遵循实际可行性原则，即不能跨越现有路段，仅限于网络内部节点间直接相连；且每条新建路段的容量被视为无限大，不受交通流量限制。

三、符号说明

符号	说明
$D_{i,j}$	以 i 为起点,j 为终点的（起点，终点）对的交通需求
$P_{i,j}$	以 i 为起点,j 为终点的（起点，终点）对的所有可行路径
k	$P_{i,j}$中的元素,表示某一条以 i 为起点,j 为终点的路径
$X_{i,j;k}$	路径 k 上的交通分配量
$a_{i,j;k}$	0-1 变量,表示使用路径 k 与否,若值为 1 则使用,值为 0 则不使用

$P'_{i,j;m,n}$	当路段 m-n 发生事故时,$P_{i,j}$中仍然能到达的路径构成的的子集
$m-n$	路段 m − n
t	问题 234 中用蒙特卡洛法模拟情况的序数
St	表示第 t 次模拟时五条路径发生事故的情况
N	蒙特卡洛法模拟的总次数
$P'_{i,j;St}$	第 t 次模拟时,$P_{i,j}$在 St 情况下仍能到达的路径构成的子集
C_{m-n}	路段 m-n 的容量上限
$NP_{i,j}$	考虑新建路段后，以 i 为起点,j 为终点的所有可行路径
$NP'_{i,j;St}$	考虑新建路段后，$NP_{i,j}$在 St 情况下仍能到达的路径构成的子集

表 1 模型建立中需要用到的记号

四、问题分析

4.1 问题 1 的分析

问题 1 要求在小型交通网络中，通过数学建模找到交通需求分配方案，使得当交通需求的期望可达率最大化。这里假设只有一个路段发生事故，对模型较小情形下所有可能性概率相等，且可以穷举，然后遍历每一种可能性求出条件期望，在对条件期望求平均得到总体期望可达率作为目标函数进行线性规划。实际代码中使用了模拟退火算法以提高代码效率。

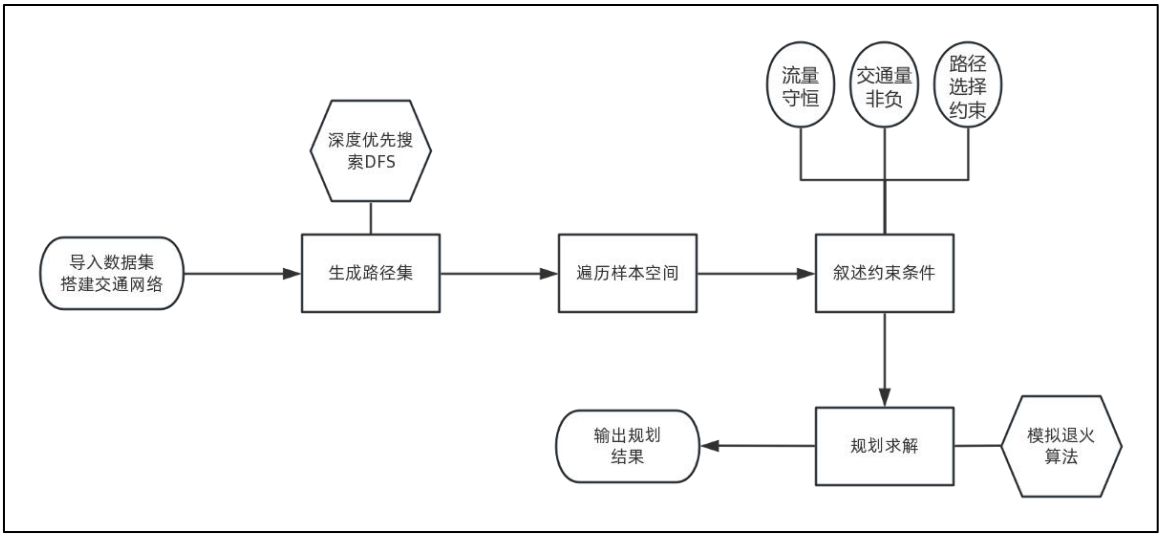


图 1 解决问题 1 的流程图

4.2 问题 2 的分析

问题 2 与问题 1 核心思路没有改变，但从小型网络转向大型网络，模型变大后无法遍历每一种可能性，需要提高效率以求解。这里运用了蒙特卡洛模拟以较多次情形模拟代替，并使用模拟退火算法提高规划效率，防止陷入局部最优，使用更少的时间找到较优的解。

4.3 问题 3 的分析

在问题 2 的基础上追加约束条件即可，即每一个路段有容量上限，经过某路段的交通量综合不超过路段容量。规划复杂性进一步加大，需要运用模拟退火算法以及蒙特卡洛法进行高效计算，多层次模拟、使用不同的算法可以减少蒙特卡洛模拟次数，减少程序运算次数。

4.4 问题 4 的分析

在问题 2 基础上追加决策变量，即建造六条新单向路段，程序复杂度进一步增大，需要用到多目标复杂规划以及优化算法。

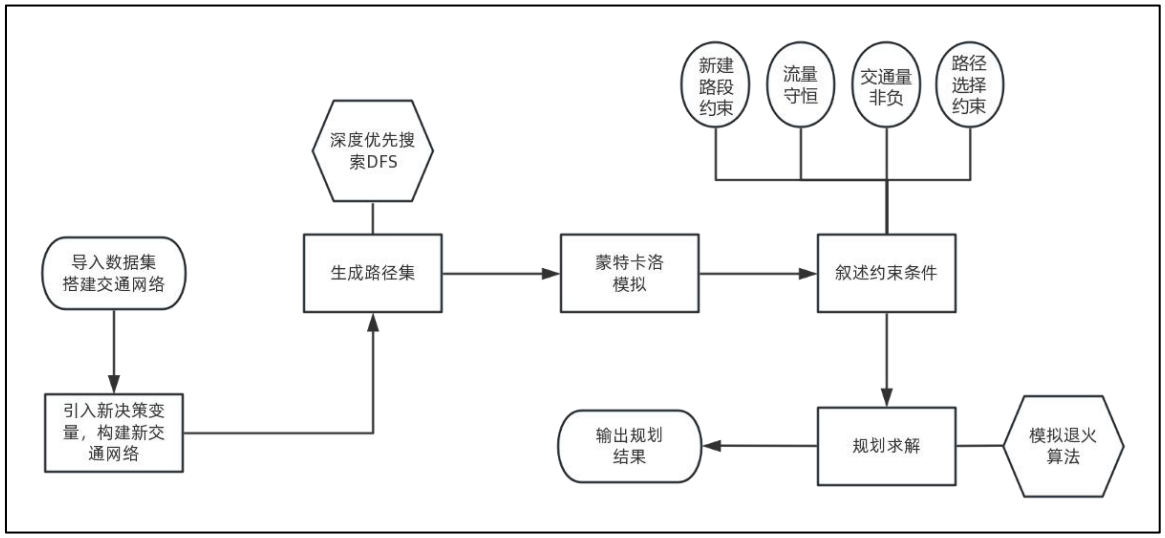


图 2 解决问题 4 的流程图

五、模型的建立与求解

5.1 问题 1 模型的建立与求解

5.1.1 交通网络 2 期望可达率模型

①**条件交通需求可达率**（Conditional Accessibility Rate，以下用 CR 代替）

条件交通需求可达率，指的是某一确定的网络事故条件下，所有（起点，终点）对的交通总需求中实际能到达的比例.根据问题 1 题意,要求只有一个路段发生事故时的可达率最高，不妨假定有且仅有路段 m-n 发生事故，这时原先的路径集中可能有部分路无法通行，此处用全体交通需求中能达到的交通量比上总的需求量作为计

算公式：

$$CR(m,n)=\frac{\sum_{k\in P'_{i,j;m,n}}\sum_{D_{i,j}>0}X_{i,j;k}a_{i,j;k}}{\sum_{D_{i,j}>0}D_{i,j}}$$

公式中分母中求和号表示对所有（起点，终点）对需求量为正的项求和，即为附件 1 中所有需求量相加，分子中外层求和号表示对所有不依赖路段 m-n 的路径项求和，分子即为当路段 m-n 发生事故条件下能到达终点的需求量.

②**总体交通需求的期望可达率**（Total Expected Rate，以下用 TR 代替）

在只有一个路段发生事故的题设条件下，不同的路段发生事故是等可能性的，由于问题 1 中节点数较少，因此样本空间中所有的样本点可以穷举，且所有样本点概率均相等，是古典概型.图 2 交通网络中一共 24 条有向路段，每一路段发生事故可能性为 1/24，对 5.1.1（1）中的条件可达率求期望可以得到总体期望可达率.

$$TR=\sum_{m-n}CR(m,n)*P(m-n\text{发生事故})$$
$$=\sum_{m-n}CR(m,n)/24$$

公式中求和号表示对遍历所有路段 m-n 的 CR(m,n)求和.

③**具体规划模型**

综上，我们以交通网络 2 的总体期望可达率作为目标函数，最大化这一函数。并依照题意设置约束条件：

- 目标函数

$$Max\ TR=\frac{\sum_{m-n}\sum_{k\in P'_{i,j;m,n}}\sum_{D_{i,j}>0}X_{i,j;k}a_{i,j;k}}{24*\sum_{D_{i,j}>0}D_{i,j}}$$

- 约束条件 1：交通量守恒

$$\sum_{k\in P_{i,j}}X_{i,j;k}a_{i,j;k}=D_{i,j}$$

- 约束条件 2：分配量非负

$$X_{i,j;k}\geq 0\ ,\ \forall k\in P_{i,j}$$

- 约束条件 3：路径选择有限

引入 0-1 变量 a 以表示是否使用路径 k，若 a 为 1 则使用，若 a 为 0 则不使用，每个(起点,终点)对之间使用的路径数不超过 5，因此固定 i,j,对 k 求和不超过 5. 另外为了方便输出结果，若某一参数的 a 值为 0，规定同样参数下的 X 值也为 0.

$$a_{i,j;k}=0,1\ ;\forall k \in P_{i,j}$$

$$\sum_{k\in P_{i,j}} a_{i,j;k} \leq 5$$

$$若a_{i,j;k}=0,则X_{i,j;k}=0$$

5.1.2 模型的求解

模拟退火算法（Simulated Annealing，简称 SA）源自固体物质的退火原理，是一种以概率为基础的优化技术。退火过程首先涉及将固体物质加热至高温，后使之逐渐冷却。与模拟退火相比，遗传算法同样是一种有效的全局优化方法，利用自然选择的机制来在解的种群中寻找最优解，但我们注意到它在某些情况下更容易过早收敛，可能需要更多的调参来保持种群多样性。

基于我们的测试和比较，模拟退火算法以其更优的鲁棒性和稳定性在最终求解中被选定。通过精心设置的降温策略，模拟退火算法能够更有效地逃离局部最优解，并有望在给定时间内找到全局最优解或其良好近似。在复杂和噪声较大的优化环境中，模拟退火算法展示了它作为一个求解器的可靠性和实用性，成为我们工程问题优化的首选技术。

算法步骤

Step1：创建路径集 paths1

首先读取附件 1 数据转换为边权矩阵生成 m1，读取图 2 数据并转换为路径矩阵生成 a1，通过深度优先搜索（DFS）策略，找出网络中从指定起点到终点的所有可能路径保存至 paths1。

Step2：创建函数 objective_function1 和 modify

objective_function1 函数中约定了模型的目标函数 TR，并随机生成一条故障路段。Modify 函数中规定模型的 3 个约束条件。

Step3：模拟退火算法求期望可达率最优解

创建模拟退火算法及问题所需的基本参数，根据多次调整结果设置初始温度为 1000 以确保结果最优。随机生成初始值并代入目标函数获得当前解，扰动产生新值代入获得新解。当新解比当前解大时则接受新解，否则按 metropolis 准则接受新解。重复该过程至温度降为 0，算法将获得最优解或接近最优解的状态。

5.1.3 运行结果

如下所示，图 3 和图 4 是不同温度参数下的优化过程图。对比可知，随着迭代

次数的增长，目标函数的值在逐步减小，并趋于平稳。这不仅反映了我们所使用的算法具有出色的稳定性，更重要的是，它表明我们通过持续的迭代优化，已经找到了目标函数的近似最优解。

一般来说，稳定性是算法性质的一种重要指标，它衡量了在不同的输入或小的初始条件变化下，算法表现的一致性。在这个案例中，稳定性体现在：即使在多次迭代后，目标函数的值都稳步收敛，没有出现明显的波动或者急剧的变化。这一结果表明，无论我们在何种程度上改变初始输入或者调整参数，算法都能够稳定地收敛到一个最优解，这是一种非常理想的性质。

此外，在我们调整温度参数后，图 3 和图 4 的结果非常接近，甚至可以说几乎相同。这意味着我们的算法具有极高的鲁棒性。鲁棒性是另一个评判算法性能的重要指标，它衡量的是算法在应对不确定因素，如噪声或者误差时的稳定性。在这里，即使我们更改了算法的参数，或者说在不同的场景下运行同一算法，我们的结果依然在一个非常可接受的范围内，没有出现较大的变化。这一结果表明，我们的算法对于各种变化和不确定性具有非常高的容忍度，能在各种条件下保持其高效的性能。

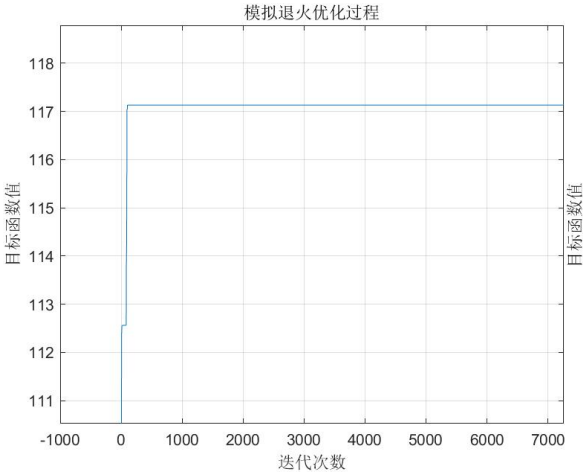


图 3 问题 1 模拟退火优化过程图 1

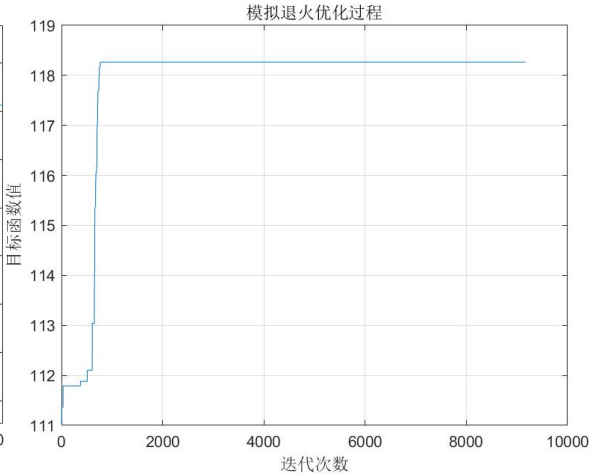


图 4 问题 1 模拟退火优化过程图 2

如下所示，表 2 为以 1 为起点，9 为中点的（起点，终点）对和以 3 为起点，7 为中点的（起点，终点）对的规划路径和分配交通量表。图 5 是按分配交通量赋值的(1,9)(3,7)的 5 条路径图。

(起点,终点)	规划路径	分配交通量
(1,9)	Path1：1-2-5-4-7-8-9	19
	Path2：1-2-5-6-9	8
	Path3：1-4-5-2-3-6-9	35
	Path4：1-4-5-6-9	76

	Path5:1-4-7-8-5-6-9	8
(3,7)	Path1：3-6-5-2-1-4-7	52
	Path2：3-2-5-6-9-8-7	38
	Path3：3-6-5-8-7	26
	Path4：3-2-1-4-7	20
	Path5：3-6-9-8-7	3

表 2 问题 1 结果

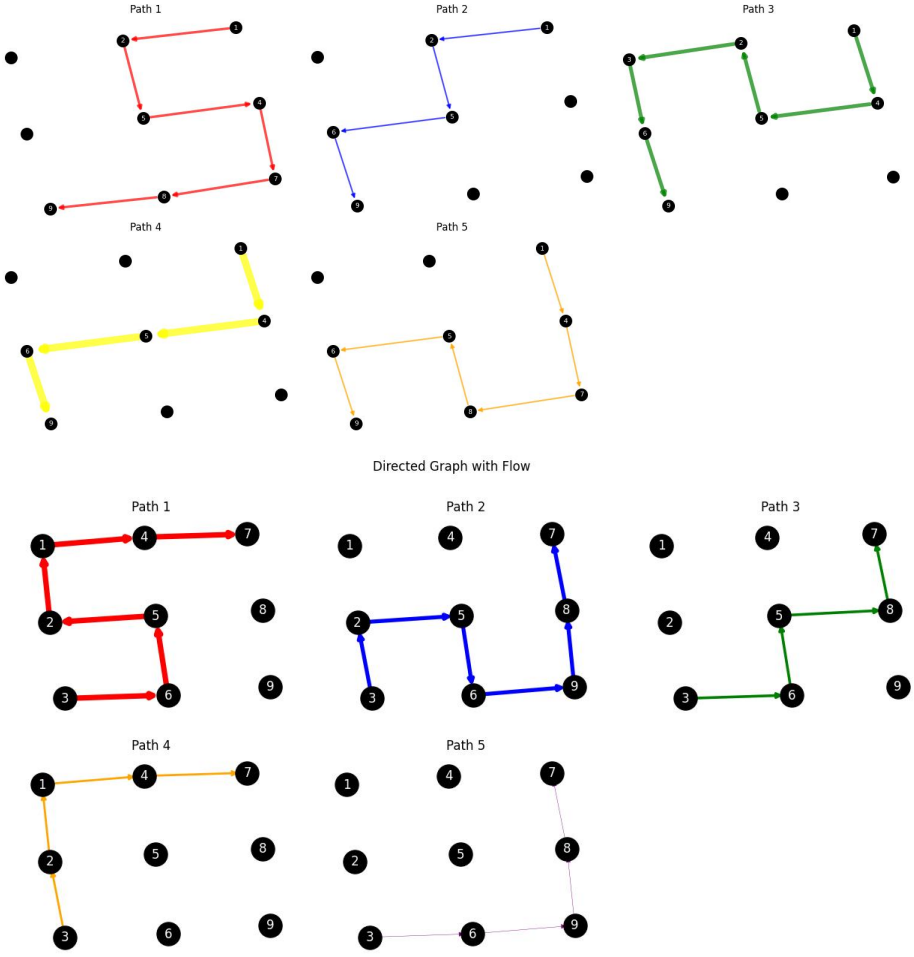


图 5 问题 1（1,9）与（3,6）路径交通需求分配结果图

5.2 问题 2 模型的建立与求解

5.2.1 交通网络 3 期望可达率模型

①公式的修正

问题 2 没有改变问题 1 的核心原理与公式，只需在问题 1 基础上修正网络图以及条件假设.由于交通网络 3 节点的增加，且条件变为任意五条路段发生事故，在 132 条路段中挑选 5 条约有三亿种可能性，这几乎是无穷且不可遍历的，因此采用蒙特卡洛法,模拟较多次数的随机情形并求平均值作为最终的期望可达率,这里模拟 N 次.

$$CR(St)=\frac{\sum_{k\in P'_{i,j;St}}\sum_{D_{i,j}>0}X_{i,j;k}a_{i,j;k}}{\sum_{D_{i,j}>0}D_{i,j}}\ ,\quad t=1,2,\ldots$$

$$TR=\sum_{t=1}^{N}CR(St)/N$$

②具体规划模型

综上，我们以交通网络 2 的总体期望可达率作为目标函数，最大化这一函数。并依照题意设置约束条件：

- 目标函数

$$Max\ TR=\frac{\sum_{t=1}^{N}\sum_{k\in P'_{i,j;St}}\sum_{D_{i,j}>0}X_{i,j;k}a_{i,j;k}}{N*\sum_{D_{i,j}>0}D_{i,j}}$$

- 约束条件 1：交通量守恒

$$\sum_{k\in P_{i,j}}X_{i,j;k}a_{i,j;k}=D_{i,j}$$

- 约束条件 2：分配量非负

$$X_{i,j;k}\geq 0\ ,\ \forall k\in P_{i,j}$$

- 约束条件 3：路径选择有限

引入 0-1 变量 a 以表示是否使用路径 k，若 a 为 1 则使用，若 a 为 0 则不使用，每个(起点,终点)对之间使用的路径数不超过 5，因此固定 i,j,对 k 求和不超过 5.另外为了方便输出结果，若某一参数的 a 值为 0，规定同样参数下的 X 值也为 0.

$$a_{i,j;k}=0,1\ ;\forall k\in P_{i,j}$$

$$\sum_{k \in P_{i,j}} a_{i,j;k} \leq 5$$

$$若 a_{i,j;k}=0, 则 X_{i,j;k}=0$$

5.2.2 模型的求解

蒙特卡洛算法（Monte Carlo method）由大量的随机抽样和统计计算的方法组成，通过程序随机抽样来模拟或者近似实现对问题域中特定事件发生的概率，从而寻找问题的解。

在我们的案例中，当五条任意路段出现故障时，重新生成全部新的路径将会花费大量的计算资源，计算量过于庞大，效率不高。因此，我们采用了蒙特卡洛方法来解决这个问题。给定一个或多个故障场景，我们在每个场景中随机选取五个路段，将其标识为“故障”状态。这样，每个故障场景都对应了一种特定的现实交通状况。通过考虑海量不同的故障场景，我们尽可能模拟了更多可能的交通状况，并在这些状况下计算平均的交通效率。这种方式有效地降低了计算量，也使我们可以更精确地评估当前的交通流量分配方案的性能。

算法步骤

Step1：创建路径集 paths2

首先读取附件 2 数据转换为边权矩阵生成 m2，读取图 2 数据并转换为路径矩阵生成 a2，通过深度优先搜索（DFS）策略，找出网络中从指定起点到终点的所有可能路径保存至 paths2。

Step2：创建函数 objective_function1 和 modify

objective_function1 函数中约定了模型的目标函数 TR，并通过蒙特卡洛算法随机生成五条故障路段。Modify 函数中规定模型的 3 个约束条件。

Step3：模拟退火算法求期望可达率最优解

创建模拟退火算法及问题所需的基本参数，根据多次调整结果设置初始温度为 1000 以确保结果最优。随机生成初始值并代入目标函数获得当前解，扰动产生新值代入获得新解。当新解比当前解大时则接受新解，否则按 metropolis 准则接受新解。重复该过程至温度降为 0，算法将获得最优解或接近最优解的状态。

5.2.3 运行结果

如下所示，表 3 为以 27 为起点，6 为中点的（起点，终点）对和以 19 为起点，25 为中点的（起点，终点）对的规划路径和分配交通量表。图 6 和图 7 分别是按分配交通量为路径赋权的六条路径图。

(起点,终点)	规划路径	分配交通量
(27,6)	Path1:27-34-8-13-12-11-10-6	59
	Path2:27-34-28-36-29-9-10-6	20

	Path3:27-34-28-36-42-9-10-6	12
(19,25)	Path1:19-17-11-42-36-13-8-25	2
	Path2:19-20-11-42-36-13-8-25	6
	Path3:19-24-20-11-12-13-8-25	8

表 3 问题 2 结果

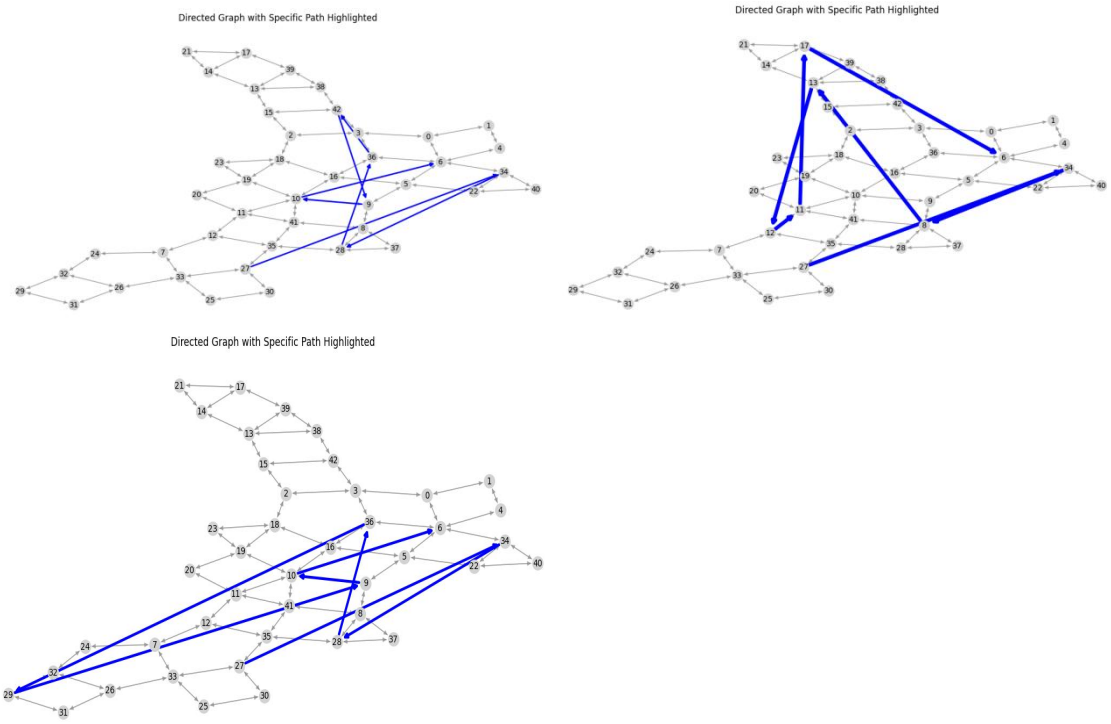
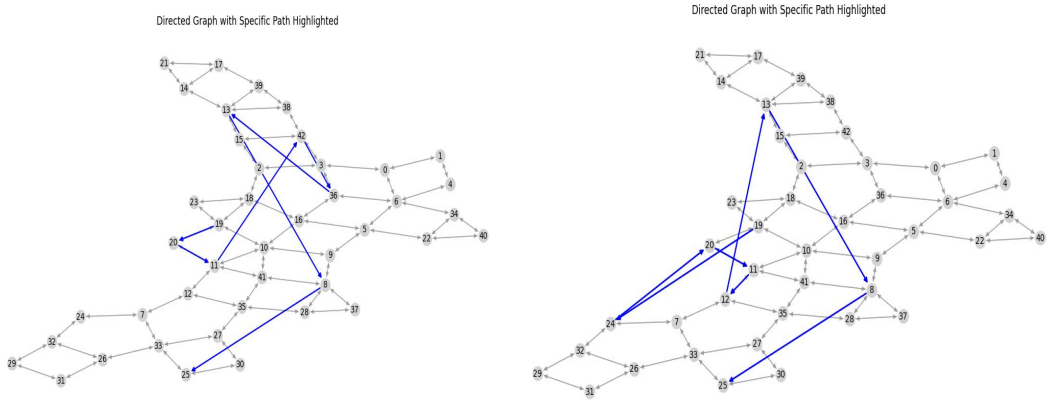


图 6 问题 2（27,6）的 path1,path2,path3 路径图



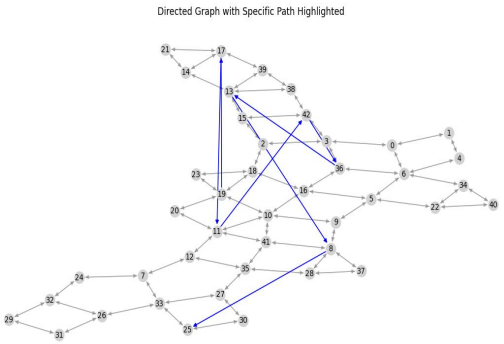


图 7 问题 2（19,25）的 path1,path2,path3 路径图

5.3 问题 3 模型的建立与求解

5.3.1 考虑路段容量下的交通网络 3 期望可达率模型

①补足修正条件

考虑到路段容量有限，遍历每条路段，对其上经过路径的交通分配量求和，要求不超过路段容量。

②具体规划模型

综上，我们以交通网络 2 的总体期望可达率作为目标函数，最大化这一函数。并依照题意设置约束条件：

- 目标函数

$$Max\ TR = \frac{\sum_{t=1}^{N}\sum_{k\in P'_{i,j;St}}\sum_{D_{i,j}>0} X_{i,j;k}a_{i,j;k}}{N*\sum_{D_{i,j}>0} D_{i,j}}$$

- 约束条件 1：交通量守恒

$$\sum_{k\in P_{i,j}} X_{i,j;k}a_{i,j;k} = D_{i,j}$$

- 约束条件 2：分配量非负

$$X_{i,j;k} \geq 0\ ,\ \forall k \in P_{i,j}$$

- 约束条件 3：路径选择有限

引入 0-1 变量 a 以表示是否使用路径 k，若 a 为 1 则使用，若 a 为 0 则不使用，每个(起点,终点)对之间使用的路径数不超过 5，因此固定 i,j,对 k 求和不超过 5. 另外为了方便输出结果，若某一参数的 a 值为 0，规定同样参数下的 X 值也为 0.

$$a_{i,j;k}=0,1\ \ ;\forall k \in P_{i,j}$$

$$\sum_{k\in P_{i,j}} a_{i,j;k} \le 5$$

$$若a_{i,j;k}=0, 则X_{i,j;k}=0$$

- 约束条件 4：路段容量有限

$$\sum_{\substack{k经过m-n且\\ k是可到达的路径}} X_{i,j;k}a_{i,j;k} \le C_{m-n}$$

5.3.2 模型的求解

算法步骤

Step1：创建边权矩阵 m3

首先读取附件 3 数据转换为边权矩阵生成 m2。

Step2：创建函数 objective_function3 和 modify

objective_function1 函数中约定了模型的目标函数 TR，并通过蒙特卡洛算法随机生成五条故障路段，并约定了路径容量的上限。Modify 函数中规定模型的 4 个约束条件。

Step3：模拟退火算法求期望可达率最优解

创建模拟退火算法及问题所需的基本参数，根据多次调整结果设置初始温度为 1000 以确保结果最优。随机生成初始值并代入目标函数获得当前解，扰动产生新值代入获得新解。当新解比当前解大时则接受新解，否则按 metropolis 准则接受新解。重复该过程至温度降为 0，算法将获得最优解或接近最优解的状态。

5.3.3 运行结果

(起点,终点)	规划路径	分配交通量
(32,39)	32-24-7-12-11-10-16-36-3-42-38-39	3
(17,8)	Path1：17-37-7-6-10-11-42-12-36-13-8	6
	Path2:17-37-7-6-10-11-42-36-13-8	2
	Path3:17-19-20-21-12-13-8	11

表 4 问题 3 结果

5.4 问题 4 模型建立与求解

5.4.1 考虑新建路段下的交通网络 3 期望可达率模型

①**找到可新建的路段集**

根据题目描述，新建路段起点和终点必须是交通网络中的任意两个节点，并且新建路段不能跨越其他路段；新建路段在网络内部修建。体现在图上，在图 3 中只有在四边形或五边形中才能修建路段，且六条路段间不能互相交叉。当用 Matlab 解析交通图，将图转化为矩阵形式，用 Matlab 工具寻找 4-圈与 5-圈（指图论中，由四或五个节点组成的简单环），找到所有能新建路段的（起点，终点）对。

②**增加决策变量**

增加新的 0-1 决策变量 R，意义为在(i,j)起点终点对建造新的路段与否，若建造赋值 1，不建造赋值 0。对于无法建造路段的起点终点对，不引入决策变量 R，仅对①中能建造路段的（起点，终点）对引入决策变量 R，以减少计算。

另以建造六条新路段作为新的约束条件。这边并不考虑新建路段是否相互交叉（例如在一个四边形内新建两条对角线会产生交叉），是为了：不引入过多的约束条件，增加计算量与编程量；题目要求给出多个方案，对于产生交叉情况的结果可以手动排除。

③**修正目标函数**

考虑到新的路段会改变交通网络形态，需要对目标函数做出修正。

④**具体规划模型**

- 目标函数

$$Max\ TR = \frac{\sum_{t=1}^{N}\sum_{k\in NP'_{i,j;St}}\sum_{D_{i,j}>0}X_{i,j;k}a_{i,j;k}}{N*\sum_{D_{i,j}>0}D_{i,j}}$$

- 约束条件 1：新建路段数量

$$\sum_{m-n\in W} R_{m,n} = 6$$

$$R_{m,n} = 0,1\ ;\ \forall m-n \in W$$

- 约束条件 2：交通量守恒

考虑新建道路后，$P_{i,j}$修正为$NP_{i,j}$.

$$\sum_{k\in NP_{i,j}} X_{i,j;k}a_{i,j;k} = D_{i,j}$$

- 约束条件 3：分配量非负

$$X_{i,j;k} \geq 0\ ,\ \forall k \in NP_{i,j}$$

- 约束条件 4：路径选择有限

引入 0-1 变量 a 以表示是否使用路径 k，若 a 为 1 则使用，若 a 为 0 则不使用，每个(起点,终点)对之间使用的路径数不超过 5，因此固定 i,j,对 k 求和不超过 5. 另外为了方便输出结果，若某一参数的 a 值为 0，规定同样参数下的 X 值也为 0.

$$a_{i,j;k}=0,1 \ ;\forall k \in NP_{i,j}$$

$$\sum_{k\in P''_{i,j}} a_{i,j;k} \leq 5$$

$$若a_{i,j;k}=0,则X_{i,j;k}=0$$

- 约束条件 5：路段容量有限

$$\sum_{k经过m-n，m-n路段原本存在且k是可到达的路径} X_{i,j;k}a_{i,j;k} \leq C_{m-n}$$

5.4.2 模型的求解

算法步骤

Step1：创建边权矩阵 m3

首先读取附件 3 数据转换为边权矩阵生成 m2。

Step2：创建函数 objective_function3 和 modify

objective_functionl 函数中约定了模型的目标函数 TR，并通过蒙特卡洛算法随机生成五条故障路段，并约定了路径容量的上限。Modify 函数中规定模型的 4 个约束条件。

Step3：模拟退火算法求期望可达率最优解

创建模拟退火算法及问题所需的基本参数，根据多次调整结果设置初始温度为 1000 以确保结果最优。随机生成初始值并代入目标函数获得当前解，扰动产生新值代入获得新解。当新解比当前解大时则接受新解，否则按 metropolis 准则接受新解。重复该过程至温度降为 0，算法将获得最优解或接近最优解的状态。

5.4.3 运行结果

	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	4-3	21-34	17-0	7-39	35-21	19-31	0.95
方案 2	14-41	28-26	20-33	15-20	27-19	33-13	0.93
方案 3	17-6	26-19	5-25	25-17	14-22	26-28	0.93

方案 4	17-40	13-27	26-15	23-31	8-16	34-2	0.92
方案 5	14-23	14-0	32-13	11-8	33-16	39-25	0.92

表 5 问题 4 结果

六、模型及算法评估

6.1 模型评估

6.1.1 模型优点

①精确提取出问题所要求的函数与约束条件
②算式精简易懂

6.1.2 模型缺点

①算式虽然简单但是逻辑复杂，较难改写成代码的形式
②既涉及连续型变量的规划，又加入了 0-1 变量规划，复杂程度提高

6.2 算法评估

6.1.1 算法优点

①通过使用模拟退火算法，能在较大的解空间中寻找到一个全局优化解，能避免陷入局部最优的情况；可以自行设置参数，调整优化的精度与时间
②在问题 2,3,4 的求解中使用蒙特卡洛算法，在保证结果渐进最优的同时有效减少了计算量

6.1.2 算法缺点

①求解复杂度较高，涉及多个函数文件以及多层逻辑嵌套，代码编写过程需要耗费较长的时间，难度较大
②对参数的设置比较敏感，不恰当的参数设置会给求解带来一定困扰

七、参考文献

[1]Boris S. Kerner , Introduction to Modern Traffic Flow Theory and Control , Berlin Heidelberg ; Springer Nature , 2009
[2]Dušan Teodorović ; Katarina Vukadinović , Traffic Control and Transport Planning , Dordrecht ; Springer
[3]Shigang Chen ; Min Chen ; Qingjun Xiao , Traffic Measurement for Big Network Data , Springer Cham
[4]蒋正威,曹一家,孙维真,基于 01 整数规划的多目标最优 PMU 配置算法,电力系统保护与控制,2008(21):12-17 ; 2008
[5]李眩，童百利，方婷婷，基于模拟退火思想的最优最差蚁群算法求解的 TSP 问题，山西师范大学学报(自然科学版)，2023,37(02)；2023

附件清单

1. MATLAB 程序 Q1.m

2. MATLAB 程序 Q2.m

3. MATLAB 程序 Q3.m

4. Python 程序 test.py

1. MATLAB 程序 Q1.m

Q1

```
clear; clc;
% 参数
load m1.mat; load paths1.mat; load a1.mat;
% 定义模拟退火参数
num_vars = size(paths1,1); % 决策变量个数
xlim = [0, max(max(m1))]; % 决策变量上下限
G = 10 ; %迭代次数
T_initial = 100; % 初始温度
cooling_rate = 0.99; % 冷却率
T_final = 0.01; % 最终温度

% 定义问题参数
p = 1/num_vars; % 每条边的故障概率
n = sum(sum(a1)); % 图中路段数

% 初始化解
current_solution = round(rand(1, num_vars) * (xlim(2) - xlim(1)) + xlim(1));
[current_fx,current_solution] =
objective_function1(current_solution,p,paths1,n,m1); % 计算初始解的函数值

% 初始化最优解
best_fx = current_fx;
best_solution = current_solution;

% 用于存储每次迭代最佳目标函数值的数组，用于可视化
fx_history = zeros(1, G);

% 模拟退火循环
T = T_initial;
iteration = 0;
while T > T_final
for j = 1 : G
```

```
iteration = iteration + 1;
% 生成新解
new_solution = current_solution;
% 选择一个决策变量进行随机扰动，生成新的解
var_index = randi(num_vars);
new_solution(var_index) = round(new_solution(var_index) + (rand - 0.5) * 2 *
(xlim(2) - xlim(1)) * T / T_initial);
new_solution = max(min(new_solution, xlim(2)), xlim(1));
[new_fx,new_solution] = objective_function1(new_solution,p,paths1,n,m1); % 计
算新解目标函数
% 接受判定
delta = new_fx - current_fx;
if delta < 0 || (exp(-delta / T) > rand)
current_solution = new_solution; % 更新当前解
current_fx = new_fx;
end
% 更新最优解
if current_fx > best_fx
best_fx = current_fx; % 更新最优函数值
best_solution = current_solution; % 更新最优解
end
% 更新函数值历史记录
fx_history(iteration) = best_fx;
end
T = T * cooling_rate; % 降温
end

disp(['最优解为：', num2str(best_solution)]);
disp(['最优目标函数值为：', num2str(best_fx)]);

% 可视化收敛过程
figure;
plot(fx_history);
xlabel('迭代次数');
ylabel('目标函数值');
title('模拟退火优化过程');
grid on;

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;
CR = 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
CR(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

function pop = Modify(pop, paths1, m1)
% 初始化一个变量来跟踪当前连续组的和
current_sum = 0;
% 初始化一个空数组来存储每个连续组的和
group_sums = [];
% 初始化一个空数组来存储每个组的索引范围
group_indices = {};
start_idx = 1;
% 遍历除了最后一个数组以外的所有数组
for i = 2:length(paths1)-1
% 取出当前数组和下一个数组
current_cell = paths1{i};
next_cell = paths1{i+1};
% 路径规划交通量约束
m = m1(current_cell(1),current_cell(end)); % 对应交通量
if pop(i) > m
pop(i) = m;
end
```

```
% 检查当前数组的结尾数字是否与下一个数组的开头数字相同
if current_cell(1) == next_cell(1) && current_cell(end) == next_cell(end)
% 如果相同，将当前 decpop 的元素加到当前组的和中
current_sum = current_sum + pop(i);
else
% 如果不同，将当前组的和添加到 group_sums 数组中，并记录组的索引范围
group_sums = [group_sums; current_sum];
group_indices{end+1} = start_idx:i;
start_idx = i + 1;
current_sum = pop(1,i); % 将当前 decpop 的元素设置为新组的和（如果它开始新的组）
end
end
% 处理最后一个组
current_sum = current_sum + pop(i+1);
group_sums = [group_sums; current_sum];
group_indices{end+1} = start_idx:length(pop);

% 交通量守恒约束
for i = 2:length(group_sums)
m = m1(paths1{group_indices{i}(1)}(1), paths1{group_indices{i}(end)}(end));
adjustment = m - group_sums(i);
% 获取该组中所有元素的索引
idxs = group_indices{i};
max_iterations = 100; % 设置最大迭代次数以避免可能的无限循环
iter = 0;
while adjustment ~= 0 && iter < max_iterations
iter = iter + 1;
% 随机选择多个索引进行一次性调整
rand_idxs = randperm(length(idxs), min(length(idxs), 10)); % 一次处理最多 10 个
随机元素
for j = 1:length(rand_idxs)
idx = idxs(rand_idxs(j));
if adjustment < 0 && pop(idx) > 0 % 需要减少总量
decrement = min([-adjustment, pop(idx),
round(abs(adjustment)/length(rand_idxs))]);
pop(idx) = round(pop(idx) - decrement);
adjustment = adjustment + decrement;
elseif adjustment > 0 % 需要增加总量
increment = min([adjustment, round(adjustment/length(rand_idxs))]); % 增加的值
按比例分配
pop(idx) = round(pop(idx) + increment);
adjustment = adjustment - increment;
end
end
```

```
end
end

% 再次确保不超过 m 的限制
for i = 2:length(paths1)
current_cell = paths1{i};
m = m1(current_cell(1), current_cell(end));
if pop(i) > m
pop(i) = m;
end
end
end
```

2.MATLAB 程序 Q2.m

Q2

```
% 参数设置
load m2.mat; load paths2.mat; load a2.mat;

% 初始化模拟退火算法参数
num_vars = size(paths2, 1); % 决策变量个数
xlim = [0, max(max(m2))]; % 决策变量上下限
G = 10000; % 迭代次数
T_initial = 10000; % 初始温度
T_final = 1e-3; % 结束温度
alpha = 0.99; % 温度降低系数
N = 10; % 生成 N 组场景
p = 1/num_vars; % 每条边的故障概率
n = sum(sum(a2)); % 图中路段数
maxLength = 5;% 添加 maxLength 参数值为 5

% 计算初始解的函数值
[current_fx, current_solution] = objective_function2(current_solution, p,
paths2, n, m2, N, a2);
% 同 Q1……

 % 确保新解的值
[new_fx, new_solution] = objective_function2(new_solution, p, paths2, n, m2, N,
a2);
% 同 Q1……

% 绘制模拟退火优化过程中目标函数值的变化
% 同 Q1……

% objective_function2 函数
```

```
% 同 Q1……

%蒙特卡洛算法设置路障
failures = zeros(n, N);
% 对于每个场景
for scenario = 1:N
% 随机选择 5 个不同的路段索引作为故障路段
failed_indices = randperm(n, 5);
% 初始化当前场景的所有路段为非故障状态
failures(:, scenario) = 0;
% 将选中的 5 个路段设置为故障状态
failures(failed_indices, scenario) = 1;
end
% 同 Q1……
end

% Modify 函数同 Q1……
end
```

3. MATLAB 程序 Q3.m

Q3

```
clear; clc;
% Parameters
load m2.mat; load m3.mat; load paths2.mat; load a2.mat;
% Define parameters for Simulated Annealing
num_vars = size(paths2, 1); % Decision variable number
T_init = 100; % Initial temperature
T_final = 0.01; % Final temperature
alpha = 0.9; % Cooling rate
max_iteration = 200; % Max number of iterations per temperature
G = 1000 ; % Total number of temperature changes
maxLength = 5;

% Define problem parameters
p = 1/num_vars;
xlim = [0, max(max(m2))];
n = sum(sum(a2));
N = 10;

% Initialize
% 同 Q1……
[obj_state,state_decpop] = objective_function3(state_decpop, p, paths2, n, m2,
N, a2, m3);
best_obj = obj_state; best_decpop = state_decpop;
```

```
% 同 Q1……
[obj_new,new_decpop] = objective_function3(new_decpop,p,paths2,n,m2,N,a2,m3);
% 同 Q1……
end
function [obj,fpop] = objective_function3(decpop,p,paths2,n,m2,N,a2,m3)
% 同 Q2……
% 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;
% 同 Q2……
end

% Modify 函数同 Q1……
end
```

4. Python 程序 Q4. py

Q4

```
import pandas as pd
import networkx as nx
import itertools
import random

def evaluate_network(H, demand_dict):
    total_demand = 0
    reachable_demand = 0
    for (origin, destination), demand in demand_dict.items():
        if nx.has_path(H, origin, destination):
            path_length = nx.dijkstra_path_length(H, origin, destination)
            reachable_demand += demand / (path_length + 1)
            total_demand += demand
    return reachable_demand / total_demand if total_demand else 0

demand_data = pd.read_excel("附件 2.xlsx")
capacity_data = pd.read_excel("附件 3.xlsx")

G = nx.DiGraph()
for _, row in capacity_data.iterrows():
    G.add_edge(row['路段起点 (start node)'], row['路段终点 (end node)'],
capacity=row['路段容量 (link capacity)'])

demand_dict = {(row['起点 (origin)'], row['终点 (destination)']): row['需求量
(demand)']
               for _, row in demand_data.iterrows()}

all_nodes = list(G.nodes())
```

```
possible_new_edges = [(u, v) for u, v in itertools.permutations(all_nodes, 2)
if not G.has_edge(u, v) and u != v]

num_samples = 1000  # Number of samples for evaluation
sampled_combinations  =  [random.sample(possible_new_edges,  6)  for  _  in
range(num_samples)]

results = []
for edges in sampled_combinations:
    H = G.copy()
    H.add_edges_from(edges, capacity=float('inf'))  # Assume infinite capacity
for new road segments
    accessibility = evaluate_network(H, demand_dict)
    results.append((edges, accessibility))

results.sort(key=lambda x: x[1], reverse=True)
top_five_solutions = results[:5]

for i, (edges, accessibility) in enumerate(top_five_solutions, start=1):
    print(f"Solution {i}:")
    for j, edge in enumerate(edges, start=1):
        print(f"  New Road Segment {j}: {edge[0]}-{edge[1]}")
    print(f"  Accessibility: {accessibility}")
```