**土壤普查路径和任务规划模型**

**摘 要**

土壤普查作为一项重要的基础性工作，对于摸清土壤资源状况、保障国家粮食安全和生态文明建设具有重要意义。然而，在实际普查工作中，如何高效完成多个地点的土壤采样任务，成为困扰基层工作者的现实难题。特别是在有限的时间和资源约束下，科学合理地规划采样路径和分配调查任务，将直接影响普查工作的质量和效率。

针对这一关键问题，本研究从数学建模的角度出发，创新性地将土壤采样路径优化问题转化为经典的旅行商问题（Traveling Salesman Problem, TSP）。通过建立基于地理空间关系的网络模型，我们采用haversine公式精确计算采样点之间的球面距离，该公式充分考虑了地球曲率对距离计算的影响，相较于简单的欧氏距离更能反映实际地理空间关系。在此基础上，构建的距离矩阵通过弗洛伊德（Floyd）算法进行优化求解，该算法通过动态规划的思想，能够有效处理路径规划中可能存在的中间节点优化问题，从而确保获得全局最优的采样路线。这种系统化的建模与求解方法，不仅显著提高了土壤普查外业工作的科学性，也为同类资源调查工作提供了可复用的技术路径。

问题1：根据附件1所提供的xx地区普查点的经纬度数据提取出序号为：31、44、61、83、100、115、147、158这8个点位的采样。我们假设所有点位之间可以直线通行，建立TSP（旅行商）模型，并利用弗洛伊德算法和haversine公式计算出8组地理坐标点之间的距离数据。得出工作组这一天的最优路径为115-31-100-147-83-158-61-44，且得出当天最短工作时间为 401.5328。

问题二：采用就近原则，以每天工作8个点位为基准，将附件1所有点位进行划分。用 Haversine 公式将经纬度转弧度算球面距离。点位分组则基于贪心算法，每天按就近原则选 8 个点位，直至全部分配。路径优化使用随机排列搜索，对每组生成 1000 个排列，选总时间最短的为最优路径，基于此我们得到每一天的点位最优路径，及相应的工作时间。最优路径工作时间求取最大值为425.28 分钟、最小值为318.16 分钟。

问题三：取消每天8个点位数的限制，将最多工作时间延长至8.5个小时，我们基于启发式贪心算法做采样点调度，以均衡地理空间点位工作时间。解决带时间约束的 TSP 变种问题，因为有固定工作时间、每日总工时上限等约束，我们用贪心算法与启发式规则，按最近邻选点、检查时间约束，从首点顺序遍历，不满足则开启新一天的方式计算，最后得到了24天的均衡化的方案。

问题四：我们已知办公室为工作组某一天安排的任务是序号为：31、44、61、83、100、115、147、158这8个点位的采样，且同时已知工作组在所有点位之间，都采用专车通行的方案。我们通过模拟退火算法和百度地图API求解了该问题，使得总时间最短。算法通过随机搜索和逐步降温的策略，有效避免了局部最优解，最终输出最优路径（采样点顺序）: 61-147-44-31-100-115-158-83

和完成当天工作最短的时间为480.28。

关键词：土壤普查路径优、TSP（旅行商问题）模型、Haversine 距离计算、

贪心算法与启发式调度、模拟退火算法。

**目 录**

摘要..............................................................................................................................................................1

一、 问题重述................................................................................................................................................3

二、 问题分析................................................................................................................................................3

2.1 问题一分析..................................................................................................................................... 3

2.2 问题二分析..................................................................................................................................... 3

2.3 问题三分析..................................................................................................................................... 4

2.4 问题四分析..................................................................................................................................... 4

三、 模型假设................................................................................................................................................5

四、 符号说明................................................................................................................................................5

五、 模型的建立与求解................................................................................................................................6

5.1 问题一模型的建立与求解............................................................................................................. 6

5.1.1两两点位之间的距离..........................................................................................................8

5.1.2八个点位之间的求解代码..........................................................................................................8

5.1.3问题一模型的求解与可视化分析......................................................................................9

5.2 问题二模型的建立与求解........................................................................................................... 10

5.2.1 问题二模型思路分析.......................................................................................................10

5.2.2 贪心算法与模型建立.......................................................................................................10

5.2.3 执行步骤...........................................................................................................................11

5.2.4 问题二求解结果与可视化分析.......................................................................................12

5.2.5按就近原则的代码实现....................................................................................................12

5.2.6题目总述............................................................................................................................13

5.3 问题三模型的建立与求解........................................................................................................... 13

5.3.1 问题三模型思路分析.......................................................................................................13

5.3.2 模型建立...........................................................................................................................13

5.3.3 模型求解...........................................................................................................................14

5.3.4 方案呈现...........................................................................................................................15

5.4 问题四模型的建立与求解........................................................................................................... 16

5.4.1 问题四模型思路分析........................................................................................................16

5.4.2 模拟退火算法与模型建立................................................................................................16

5.4.3 程序运行与题目求解........................................................................................................17

5.4.4 题目总述与代码评价........................................................................................................18

六、 模型的检验和灵敏度分析.................................................................................................................19

七、模型的优缺点分析及模型的推广.......................................................................................................19

八、参考文献...............................................................................................................................................20

九、附录

1. **问题重述**

1.1问题背景

土壤普查是保障国家粮食安全的基础工作。普查农用地和部分未利用地土壤，能全面 “体检” 土壤，摸清质量家底，掌握资源情况，对保障粮食生产、守牢耕地红线意义重大。但在偏远复杂环境下，如何在有限时间与资源条件中，高效完成多地土壤采样，成亟待解决的问题。

1.2问题提出

现有 XX 地区成立土壤普查办公室，根据上级部门设定的土壤普查点位安排一个工作组进行土壤采样工作。办公室需要提前对工作组每天的土壤采样工作进行路径和任务规划。按照正常工作流程和进度，工作组的工作时间仅包括在点位工作的时间和点位之间切换路上通行的时间，但不考虑工作组从驻地到第一个点位和当天离开最后一个点位回到驻地的时间，即当天来第一个工作点位之前和最后一个工作点位之后的通行时间不算在内。

由此，提出以下四个问题：

问题一：假设工作组在所有点位之间可以按照直线通行，速度为20千米/小时，在31、44、61、83、100、115、147、158八个点位中找出最短路径并计算最短用时。

问题二：假设工作组在所有点位之间可以按照直线通行，速度为20千米/小时，以附件一的点位划分为准，采取就近原则，得出每一天工作组的最优工作路径和工作时间，及其最大值、最小值。

问题三：考虑具体实际情况，现允许工作组工作时间延长为 8.5h，判断问题二中所给的方案是否存在工作时间超时或不均衡的现象。如果取消点位数目的限制，在所给时限下设计均衡化的方案。

问题四：在31、44、61、83、100、115、147、158八个点位中，根据附件一和电子地图提供的数据，求出工作的最优路径（采样点顺序）以及工作的最短时间。

1. **问题分析**

2.1问题一分析

根据题目要求，首先在附件中找到所求的8个点位，使用哈弗辛公式计算每对点位之间的距离，将距离最近的两个点位聚合成一个新的簇。之后对于每个簇内的点位，计算它们之间的直线距离，并将它们连接起来形成一条路径。

最优路径则可通过建立弗洛伊德的模型和旅行商问题（Traveling Salesman Problem, TSP）模型，根据计算出的各点之间的距离，得出最优路线，进而求得最优路径长度；

所耗费的时间等于最优路径长度除以速度20km/h，当天工作最短的时间等于所耗费的时间加完成工作所需的时间。

2.2问题二分析

根据题目所要求的就近原则，若要做到两点间距离最短，首先初始化未访问的点位集合。每天选择8个点位，优先选择距离当前点位最近的下一个未访问点位。通过贪心策略逐步构建每天的点位组，直到所有点位被分配完毕。对于每一天的点位组，生成一定数量（`num\_permutations=1000`）的随机排列。计算每种排列的总时间（包括行驶时间和工作时间）。然后选择总时间最短的排列作为当天的最优路径。这是一种启发式算法，适用于小规模问题（每组最多8个点位），但可能会导致一些误差。随后我们对该问题建模，建立旅行商问题（TSP）的变种模型：目标是最小化每天的总时间（行驶时间 + 工作时间）。由于点位分组后每组规模较小（≤8），采用随机排列搜索来近似求解TSP。因此采用旅行商方法。使其任意两点之间只存在唯一路径，且路径长度唯一，通过TSP函数求和，输出总距离。通过计算各个工作日最优路径的长度,我们就可以得到完成全部采样任务所需的最大工作时间和最小工作时间。

通过分析这些最大值和最小值,我们就可以评估工作量的均衡性。如果最大值和最小值相差很大,说明工作量分配存在不平衡的情况。这时,我们可以适当调整任务划分的方式,例如采用更复杂的聚类算法,或者手动调整某些工作日的点位安排,以期达到更加均衡的工作量分配。

2.3问题三分析

前两小问中，固定点位的任务分配存在超市或时间不均匀的问题，所以放宽点位工作时间的限制，故针对问题三，取消每天8个点位的限制，但增加工作总时长限制，将所有点位划分为若干天的工作任务。在此基础上，对每个工作日的路径进行优化，得到最优路径及其对应的工作时间，尽快完成任务。

通过分析所有最优路径的最大值和最小值，可评估工作量的均衡性。最大值和最小值相差不大时可认为工作量较为均衡。我们采取启发式贪心算法进行规划求解。

贪心算法：首先计算每个点位之间的距离，让每日工作时间在时间上限之内（8.5h），在每一步做选择时，都采用当前状态下最优的选择，从而希望得到全局最优解。从第一个点开始找到每一个对应的最近的下一个的未访问点，直到当日工作时间达到上限，然后开启新的一天进行采样安排。如果加上当前点位的工作时间和行程时间后，总工作时间不超过每日最大工作时间，则将该点位加入当前天的日程安排；否则，将当前天的日程安排添加到schedule中，并开启新的一天，将当前点位作为新一天的第一个点位。找到下一个距离当前点位最近的未访问点位，将其作为下一个访问点位，并标记为已访问。遍历schedule中的每一天，输出每天需要访问的点位信息。

通过分析这些最大值和最小值，我们就可以评估工作量的均衡性。如果最大值和最小值相差很大，说明工作量分配存在不平衡的情况。这时，我们可以适当调整任务划分的方式，人工划分工作安排，以达到更加均衡的工作量分配。

2.4问题四分析

问题四中专车通行即指考虑实际情况，两点之间不再是直线，而是八个工作点之间的实际距离，将旅行商（TSP）模型和具体实际道路相结合，应用模拟退火算法的创新路线规划方法，建立目标函数和约束对象来求解。对于专车通行的情况，利用网络百度地图所提供的API驾车路线，获得附录一中规定的八个点位之间的实际路线和行驶时间。

模拟退火算法：随机选择初始地点和初始温度。利用当前工作时间对解进行扰动，产生新解。计算新解与旧解的目标函数值：如果新解更优，则接受新解。如果新解不优，则以一定概率接受新解（即存在“跳出局部最优”的可能性），概率依赖于当前温度和解的优劣。随着每次迭代，逐渐减少所需时间，直到达到终止条件。

基于上述模型，我们可以更灵活地安排每天的工作任务，确保在时间限制下实现任务的最佳分配，从而提高工作效率并解决工作不均衡或超时等问题，给出工作组这一天的最优路径（采样点的顺序）并成功计算出完成当天工作最短的时间。

**三、模型假设**

本文提出以下合理假设：

1.假设每天的天气不会对路程行驶有影响。

2.假设每天的天气不会对工作点工作产生影响。

3.假设每天行驶的路程的时间都是一致的。

4.假设地球是一个近似的球体且不考虑地球的曲率。

5.假设采样工作结束以后能够立即上车出发到另一个点位，且是直线行驶。

6.假设行驶过程中不会发生任何的意外情况，如车辆故障，交通堵塞等一系列的造成时间延长的事件。

**四、符号说明**

|  |  |  |
| --- | --- | --- |
| 符号 | 含义 | 备注 |
| Lons | 经度 |  |
| Lats | 纬度 |  |
| work\_time | 工作时间 |  |
| travel\_time | 工作时间 |  |
| total\_time | 总时间 |  |
| Distance | 路径 |  |
| initial\_route | 初始路径 |  |
| R | 地球半径 |  |
| Lujing1\2\3 | 问题一、二、三的最优路径结果 |  |
| max\_work\_time | 工作时间上限 |  |
| next\_point | 下一个最近的未访问点 |  |
| Dlon | 经度差 |  |
| Dlat | 纬度差 |  |
| i\j | 循环变量 | 用于遍历附件中的每一个数据 |
| T | 初始温度 | 模拟退火算法中使用 |
| T\_min | 终止温度 | 设为1 |
| Alpha | 温度衰减系数 |  |
| max\_iter | 最大迭代次数 |  |
| simulated\_annealing | 函数调用 |  |

**五、模型建立及求解**

5.1问题一模型的建立与求解

5.1.1两两点位之间的距离

由于题设所给出的点均基于地球上点的经纬度，所以采用haversine公式计算地球上两点之间球面距离。

haversine公式：d=2arcsin(sin2(2Δφ​)+cosφ1​cosφ2​sin2(2Δλ​)​)。

d表示两点之间的球面距离；r为地球半径，通常取平均半径约6371千米；φ1、φ2分别是两点的纬度，λ1、λ2分别是两点的经度，Δφ是纬度之差，Δλ是经度之差。

|  |  |  |  |
| --- | --- | --- | --- |
| 点1 | 点2 | 经度差Δ*λ* | 纬度差Δ*φ* |
| 61 | 100 | 0.024843 | 0.002940 |
| 61 | 115 | 0.029466 | 0.006007 |
| 61 | 147 | 0.022868 | 0.001943 |
| 61 | 158 | 0.004543 | 0.010992 |
| 83 | 100 | 0.012539 | 0.004362 |
| 83 | 115 | 0.017162 | 0.013309 |
| 83 | 147 | 0.010564 | 0.005359 |
| 83 | 158 | 0.007761 | 0.003690 |
| 100 | 115 | 0.004623 | 0.008947 |
| 100 | 147 | 0.001975 | 0.000997 |
| 100 | 158 | 0.020300 | 0.008052 |
| 115 | 147 | 0.006598 | 0.007950 |
| 115 | 158 | 0.024923 | 0.016999 |
| 147 | 158 | 0.018325 | 0.009049 |

下表是8个点位之间的经纬度差

|  |  |  |  |
| --- | --- | --- | --- |
| 点1 | 点2 | 经度差Δ*λ* | 纬度差Δ*φ* |
| 1 | 44 | 0.031202 | 0.004554 |
| 31 | 61 | 0.027160 | 0.004255 |
| 31 | 83 | 0.014856 | 0.011557 |
| 31 | 100 | 0.002317 | 0.007195 |
| 31 | 115 | 0.002306 | 0.001752 |
| 31 | 147 | 0.004292 | 0.006198 |
| 31 | 158 | 0.022617 | 0.015247 |
| 44 | 61 | 0.004042 | 0.000299 |
| 44 | 83 | 0.016346 | 0.007003 |
| 44 | 100 | 0.028885 | 0.002641 |
| 44 | 115 | 0.033508 | 0.006306 |
| 44 | 147 | 0.026910 | 0.001644 |
| 44 | 158 | 0.008585 | 0.010693 |
| 61 | 83 | 0.012304 | 0.007302 |

根据haversine公式以及每两点之间的经纬度差计算连点之间的直线距离

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| d | 31 | 44 | 61 | 83 | 100 | 115 | 147 | 158 |
| 31 | 0 | 1840.38 | 3988.54 | 6100.65 | 7006.13 | 7183.32 | 5942.4 | 5319.33 |
| 44 | 1840.38 | 0 | 2198.17 | 4530.76 | 5793.5 | 6444.19 | 6567.85 | 6437.42 |
| 61 | 3988.54 | 2198.17 | 0 | 2518.45 | 4143 | 5310.62 | 7048.12 | 7507.48 |
| 83 | 6100.65 | 4530.76 | 2518.45 | 0 | 1870.06 | 3463 | 6702.74 | 7779.92 |
| 100 | 7006.13 | 5793.5 | 4143 | 1870.06 | 0 | 1728.41 | 5618.58 | 7003.54 |
| 115 | 7183.32 | 6444.19 | 5310.62 | 3463 | 1728.41 | 0 | 4161.04 | 5697.41 |
| 147 | 5942.4 | 6567.85 | 7048.12 | 6702.74 | 5618.58 | 4161.04 | 0 | 1632.74 |
| 158 | 5319.33 | 6437.42 | 7507.48 | 7779.92 | 7003.54 | 5697.41 | 1632.74 | 0 |

 5.1.2八个点位之间的求解代码

 表5.1.3理想状态下的最短距离和最短时间

 

5.1.3问题一模型的求解与可视化分析

最优路径（点位序号顺序）:

 115 31 100 147 83 158 61 44

总工作时间（分钟）: 387

总通行时间（分钟）: 14.5328

总时间（分钟）: 401.5328

图5.1.4 为最优路径示意图

 

最优解的总工作时间为401.5328分钟(存在较小误差),约6小时40分钟,如果考虑实际的道路情况,实际的工作时间可能会略有增加,但应该仍然在一个工作日内完成。

总说，求解代码通过TSP和Floyd算法,我们成功地找到了一条相对最优的路径,大幅缩

短了总距离和工作时间。这种方法不仅能够提高工作效率,也能较好地控制整体的工

作成本。这个最优解是基于haversined地球两点直线距离公式计算的,如果考虑实际的道路网络和交通状况,可能会存在一些偏差。在实际应用中,我们可以进一步利用高德地图API等服务,获取更加精确的行驶距离和时间信息,从而得到更加贴近实际的最优方案。

5.2问题二模型的建立与求解

5.2.1关于问题二

题目要求采用就近原则计算工作组一天的最优路径，分析附件所给出的数据，首先使用Haversine公式计算出每两点位间的距离，对点位进行初始化，其次采用贪心算法，每天选择八个点位，优先选择距离当前点位最近的下一个未访问点位，逐步构建每天的点位组，直到所有点位被分配完毕。

5.2.2贪心算法与模型建立

从问题的某个初始解出发，采用循环语句，当可以向求解目标前进一步时，就根据局部最优策略，计算每个组每天的最优路径和时间，得到一个部分解，缩小问题的范围或规模。将所有的局部解综合起来，得到问题的最终解。

5.2.3执行步骤

（1）读取数据

（2）提取经度、纬度和工作时间

（3）定义地球半径

（4）计算距离矩阵

（5）按就近原则划分每天8个点位

（6）计算每个组每天的最优路径和工作时间

5.2.4题目求解与可视化分析

将算法所得的路径依次分组，每八个点位一组，总分27组，余6个，经过数据分析后得出局部最优解。

每日最优路径及工作时间表

|  |  |  |
| --- | --- | --- |
| **天数** | **最优路径** | **工作时间（分钟）** |
| 第 1 天 | 57 1 110 78 218 19 217 11 | 420.34 |
| 第 2 天 | 166 67 4 157 2 142 69 179 | 392.29 |
| 第 3 天 | 222 221 3 135 81 177 14 37 | 404.27 |
| 第 4 天 | 52 56 5 10 149 123 71 150 | 405.21 |
| 第 5 天 | 106 148 99 12 21 6 212 211 | 401.36 |
| 第 6 天 | 171 62 28 42 7 23 172 50 | 399.30 |
| 第 7 天 | 8 54 82 15 63 129 83 158 | 387.31 |
| 第 8 天 | 162 104 75 41 35 107 32 9 | 388.24 |
| 第 9 天 | 195 191 25 216 128 215 143 13 | 385.30 |
| 第 10 天 | 16 124 105 194 151 47 167 120 | 405.36 |
| 第 11 天 | 17 119 132 165 164 141 49 169 | 387.28 |
| 第 12 天 | 184 185 176 198 197 40 153 18 | 388.31 |
| 第 13 天 | 60 61 44 202 200 20 204 203 | 399.28 |
| 第 14 天 | 127 170 173 55 163 154 207 22 | 401.25 |
| 第 15 天 | 214 213 39 175 24 45 131 122 | 385.27 |
| 第 16 天 | 205 206 115 31 100 147 140 26 | 376.30 |
| 第 17 天 | 79 145 103 116 144 113 160 27 | 395.52 |
| 第 18 天 | 53 70 155 156 29 72 138 102 | 415.43 |
| 第 19 天 | 30 189 188 187 186 85 152 126 | 425.28 |
| 第 20 天 | 134 192 193 190 97 51 33 89 | 375.45 |
| 第 21 天 | 66 220 114 95 219 178 208 34 | 394.35 |
| 第 22 天 | 64 108 117 109 121 65 58 36 | 382.28 |
| 第 23 天 | 38 98 48 77 101 199 201 130 | 396.05 |
| 第 24 天 | 43 68 137 76 94 86 91 88 | 404.40 |
| 第 25 天 | 209 210 46 92 80 174 73 133 | 391.53 |
| 第 26 天 | 168 87 125 59 111 182 161 74 | 409.34 |
| 第 27 天 | 96 159 90 146 84 196 139 136 | 390.32 |
| 第 28 天 | 180 118 112 93 183 181 | 318.16 |

每天的最优路径和工作时间的最大值和最小值如下：

 根据问题二，将222个数据分为28个组，所有最优路径工作时间的最大值：425.28 分钟。

所有最优路径工作时间的最小值：318.16 分钟。

5.2.5按就近原则的代码实现

下图为部分matlab代码实现，完整代码见附录

5.2.6题目总述

本题要求采用就近原则，将所给222个点位全部划分，根据附件1 中的所给信息，建立贪心算法，使用给出每一天的点位最优路径及相应的工作时间，并对最优路径工作时间求取最大值和最小值。本题将问题理想化，认为各点位之间路径为直线且匀速通行，可能会导致结果的误差，可能存在其他算法，如动态规划算法、分治算法等，能够更好地解决问题。

5.3 问题三模型的建立与求解

5.3.1 问题三模型思路分析

根据问题三题目要求，现每天的工作时间不得超过8.5个小时，且如果取消每天8个点位数的限制，在工作时限内请给出均衡化的方案，使得尽快完成土壤采样工作。

 解决问题三我们需要将222个点位合理的分配每天的工作任务，且在时间限制的情况下，尽量平衡各天的工作量。

5.3.2模型建立

由附件一可得每个点位的工作时间，根据问题一的haversine公式和车速可得点位之间的周转时间，再用“贪心算法”可进一步求解每天的最优工作路径。

5.3.3模型求解

为了求解该222个整数点位的合理规划问题，可以采用一些常见的优化算法，如贪心算法、启发式规则等。

通过贪心算法每一步选择当前最优的局部解，从当前点位出发，选择最近的未访问点位作为下一个目标。此题以点位1、2、3、5、6、7等分别为每天的第一个点位，每天内将点位的移动时间和工作时间累加，若超过当天上限8.5个小时则开启新的一天。也可通过启发式规则，初始化从第一个点位开始，按顺序计算时间。若无法满足时间约束，则终止当前天的分配，剩余点位分配到后续天数。

通过该两种算法最后得到合理分配的工作点位如下图5.3.3所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 天数 | 工作点位 | 工作时间 | 中转时间 | 总时间 |
| 1 | 1, 57, 110, 78, 218, 11, 19, 202, 200 | 476.0 | 20.88 | 496.88 |
| 2 | 2, 142, 157, 4, 67, 166, 179, 69, 73 | 443.0 | 18.28 | 461.28 |
| 3 | 3, 221, 222, 135, 81, 177, 37, 14, 198 | 457.0 | 15.42 | 472.42 |
| 4 | 5, 10, 149, 71, 123, 150, 56, 52, 159 | 456.0 | 13.22 | 469.22 |
| 5 | 6, 212, 211, 21, 106, 148, 12, 99, 144 | 455.0 | 19.27 | 474.27 |
| 6 | 7, 42, 23, 172, 50, 62, 171, 28, 208 | 448.0 | 24.96 | 472.96 |
| 7 | 8, 54, 82, 63, 15, 129, 158, 83, 147 | 435.0 | 19.09 | 454.09 |
| 8 | 9, 32, 107, 75, 41, 35, 104, 162, 79 | 438.0 | 12.73 | 450.73 |
| 9 | 13, 128, 215, 143, 216, 25, 195, 191, 146 | 442.0 | 17.25 | 459.25 |
| 10 | 16, 124, 105, 151, 194, 47, 167, 120, 30 | 457.0 | 20.63 | 477.63 |
| 11 | 17, 119, 132, 165, 141, 164, 49, 169, 156 | 434.0 | 17.84 | 451.84 |

5.3.3每天的工作点位及时间

附上5.3.3表格

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 12 | [18, 153, 176, 40, 197, 185, 184, 59, 125] | 448.0 | 18.45 | 466.45 |
| 13 | [20, 203, 204, 201, 101, 199, 43, 68, 137] | 444.0 | 18.67 | 462.67 |
| 14 | [22, 207, 154, 55, 173, 163, 170, 127, 113] | 450.0 | 19.80 | 469.80 |
| 15 | [24, 175, 39, 213, 214, 131, 45, 122, 183] | 433.0 | 20.33 | 453.33 |
| 16 | [26, 140, 115, 31, 100, 206, 205, 53, 70] | 432.0 | 19.93 | 451.93 |
| 17 | [27, 160, 116, 77, 103, 64, 117, 109, 65] | 435.0 | 25.02 | 460.02 |
| 18 | [29, 72, 138, 102, 93, 87, 168, 161, 74] | 447.0 | 22.67 | 469.67 |
| 19 | [33, 89, 51, 152, 187, 188, 186, 85, 189] | 468.0 | 20.72 | 488.72 |
| 20 | [34, 130, 178, 114, 220, 66, 95, 219, 180] | 456.0 | 31.01 | 487.01 |
| 21 | [36, 58, 121, 98, 48, 108, 145, 38] | 378.0 | 55.43 | 433.43 |
| 22 | [44, 61, 60, 46, 92, 133, 80, 174, 217] | 438.0 | 24.64 | 462.64 |
| 23 | [76, 91, 88, 86, 94, 196, 136, 139, 193] | 428.0 | 15.62 | 443.62 |
| 24 | [84, 90, 209, 118, 210, 96, 112, 134, 192] | 444.0 | 33.22 | 477.22 |
| 25 | [97, 190, 126, 111, 182, 181, 155] | 372.0 | 90.52 | 462.52 |

即前24天每天勘察9个工作点位为最合适的土壤采集方案，同时总工作天数尽可能的减少，总工作天数相比问题二，此方案总天数缩减到了25天时间。

5.3.4方案呈现

根据上述模型的建立和求解已经得到一个较为合理的工作方案，再通过Matlab绘图，使得我们得到的方案更加可视化，以便直观的了解。

下图5.3.4为工作方案的具体呈现.

图5.3.4具体工作路径



5.4问题四的模型建立与求解：

5.4.1关于问题四

在这个土壤采样优化问题中，要求在每天工作时间不超过8.5小时的情况下，尽可能多地完成土壤采样点。与实际路况相结合，利用附件中的数据和地图所给的实际路程车程，我们可以利用启发式算法，例如模拟退火算法，来解决这个问题。

5.4.2模拟退火算法与模型建立

首先通过使用高德地图的 API，预测出两点之间的实际最小通行时间

1. 将每个采样点的工作时间作为解的一部分，解是一个包含所有采样点的序列。
2. 调用`simulated\_annealing`函数求解最优路径和最短总时间
3. 绘制最短距离路径图以及利用哈弗辛公式计算距离
4. 随机选择一个采样点的序列作为初始解，设定初始温度值T，T值足够高（1000）以便探索解空间；
5. 使用初始解计算第一天的工作时间，并确定解是否满足时间限制（不超过8.5h），然后通过随机交换序列中两个采样点的位置产生新解；
6. 计算新解的适应度，判断是否满足时间限制，根据Metropolis准则决定是否接受新路径（即使新路径更差，也有一定概率接受，避免陷入局部最优）。
7. 逐步降低温度，减少接受劣解的概率。
8. 温度降至T\_min时停止。
9. 不断迭代生成新解，直到达到停止条件

通过模拟退火算法，我们可以在允许的工作时间内尽可能多地完成土壤采样点，平衡每天的工作量，提高工作效率。该算法能够帮助我们找到一个符合时间限制的最优解。

5.4.3程序运行与题目求解

以下图一为最优采样路径图，图二为主要程序（完整程序见附录）

图一

图二

5.4.4题目总述及代码评价

这个代码解决的是一个路径优化问题，目的是在给定的多个采样点中找到最优的访问顺序，使得完成所有采样工作所需的总时间最短。总时间包括在各采样点的工作时间和在采样点之间移动的旅行时间。代码通过模拟退火算法求解了一个带工作时间的旅行商问题（TSP），目标是找到访问所有点位的最优顺序，使得总时间最短。算法通过随机搜索和逐步降温的策略，有效避免了局部最优解，最终输出最优路径和对应的最短时间。

六、模型的检验、灵敏度分析

1. 方法选择说明：

四个问题均建立旅行商模型，分别使用了弗洛伊德算法、贪心算法和模拟退火算法，均是在题目限制条件下我们所想到的较为优化的算法；

1. 数据处理过程：

主要使用了题目中附件一所给出的多个地区的经纬度和工作时间，问题四中我们自行在百度地图API中收集数据用于matlab程序编写和计算，每一个问题中都对数据进行预处理和初始化，分析后并未发现缺少值和异常值，故直接采用数据进行计算；

1. 结果呈现与总结：

每一个问题中都附带了matlab实行程序后所生成的路径规划图，更加直观的展示出了问题的结果，总结模型检验的整体结论后，模型在处理一些实际问题时，仍未将一切可能发生的实际情况纳入其中，但对实验结果数据的影响较小，所以我们认为模型已基本通过检验。

1. 模型的优缺点分析及模型的推广
2. 优点：

总体采用旅行商模型进行题目的分析与求解，该模型作为一个经典的组合优化问题，描述简单直观，有广泛的代表性和应用范围，有激发算法研究的能力，为后续四个问题中所采用的弗洛伊德算法、贪心算法和模拟退火算法打下了解决问题的基础，同时促进相关数学理论的发展。

1. 缺点：

旅行商模型会随着城市密度的增加，问题的复杂度呈指数型增长，精确求解算法往往在合适的时间中无法找到最优解；模型通常假设城市之间的距离是固定的，然而，在实际旅行中，城市之间的距离可能会受到交通状况、路况等因素的影响而发生变化，而且旅行商可能有特定的访问顺序要求，就导致该模型会缺乏灵活性。

1. 推广：

可将二维的问题进行三维化，如旅行商需要在三维空间中规划路线，访问各个城市，同时考虑空间中的距离和其他因素；以此类推可将三维问题四维化，如给每个城市添加时间窗限制，规定旅行商需要在特定范围内到达每个城市以满足客户需求。

还可与动态规划问题相结合，现实中城市的位置、数量或旅行成本等因素在旅行过程中会发生动态变化，旅行商需要根据实时信息（如导航等）调整路线。

1. 参考文献

[1] 韩忠民．知经纬度计算两点精确距离［J］．科技传播， 2011(6) :196，194．

[2]刘哲,周波,余澜婷,等.融合 Pearson 相似度与最小生成树的 K-means 算法[J].

南昌工程学院学报,2022,41(06):91-96.

[3]张庭辉.基于 NetworkX 的复杂网络特征实现及其在生物网络中的应用[D].华

南理工大学,2016.

1. 王鹏杰,陶怡,朱凯,赵晨杰.基于改进蚁群算法的AGV路径规划研究[J/OL].计算机测量与控制:11.
2. 陈焱,陆杰,李大鹏.A\*蚁群融合的复合启发式路径规划算法[J/OL].无线电通信技术:9.
3. 赵强,沈正平,史春云,叶青.利用Floyd算法优化设计旅游路线——以苏州市为例[J].地理空间信息,2024(03):95-98.
4. 倪建云,吴杰,薛晨阳,谷海青.融合改进A\*和蚁群算法的机器人路径规划[J/OL].天津理工大学学报:8.
5. 杨晓明,杨在旭,宋宗莹.基于虚拟编组技术的单线重载铁路通过能力计算方法研究 [J/OL].铁道运输与经济:11
6. 附录

**问题一**

% 读取数据

data = readtable('附件1：xx地区.xlsx');

% 目标点位序号

target\_points = [31, 44, 61, 83, 100, 115, 147, 158];

% 筛选目标点位数据

target\_data = data(ismember(data.XuHao, target\_points), :);

% 提取经度、纬度和工作时间

lons = target\_data.JD;

lats = target\_data.WD;

work\_time = target\_data.Work;

% 计算距离矩阵

num\_points = height(target\_data);

distance\_matrix = zeros(num\_points, num\_points);

for i = 1:num\_points

for j = i+1:num\_points

distance\_matrix(i,j) = haversine(lats(i), lons(i), lats(j), lons(j));

distance\_matrix(j,i) = distance\_matrix(i,j);

end

end

% 初始路径

initial\_route = 1:num\_points;

% 使用 2 - opt 算法优化路径

optimal\_route = two\_opt(initial\_route, distance\_matrix);

% 将索引转换为实际的点位序号

optimal\_point\_route = target\_data.XuHao(optimal\_route);

% 计算通行时间（单位：小时）

travel\_time = sum(distance\_matrix(sub2ind(size(distance\_matrix), optimal\_route(1:end - 1), optimal\_route(2:end))));

travel\_time = travel\_time + distance\_matrix(optimal\_route(end), optimal\_route(1));

travel\_time = travel\_time / 20;

% 计算工作时间（单位：小时）

total\_work\_time = sum(work\_time) / 60;

% 计算总时间（单位：小时）

total\_time = travel\_time + total\_work\_time;

% 输出结果

fprintf('最优路径（采样点顺序）: ');

disp(optimal\_point\_route);

fprintf('完成当天工作最短的时间（小时）: %.2f\n', total\_time);

% 绘制采样位点和最优路径

figure;

hold on;

plot(lons, lats, 'ko', 'MarkerFaceColor', 'k', 'DisplayName', '采样位点');

for i = 1:num\_points - 1

start\_idx = optimal\_route(i);

end\_idx = optimal\_route(i + 1);

plot([lons(start\_idx), lons(end\_idx)], [lats(start\_idx), lats(end\_idx)], 'b-');

end

% 连接最后一个点和第一个点

start\_idx = optimal\_route(end);

end\_idx = optimal\_route(1);

plot([lons(start\_idx), lons(end\_idx)], [lats(start\_idx), lats(end\_idx)], 'b-');

xlabel('经度');

ylabel('纬度');

title('土壤采样最优路径');

legend;

grid on;

% 2 - opt 算法

function route = two\_opt(route, distance\_matrix)

improved = true;

while improved

improved = false;

for i = 2:numel(route) - 2

for j = i + 1:numel(route)

new\_route = route;

new\_route(i:j) = fliplr(route(i:j));

% 直接计算原路径和新路径的成本

original\_cost = sum(distance\_matrix(sub2ind(size(distance\_matrix), route(1:end - 1), route(2:end))));

original\_cost = original\_cost + distance\_matrix(route(end), route(1));

new\_cost = sum(distance\_matrix(sub2ind(size(distance\_matrix), new\_route(1:end - 1), new\_route(2:end))));

new\_cost = new\_cost + distance\_matrix(new\_route(end), new\_route(1));

if new\_cost < original\_cost

route = new\_route;

improved = true;

end

end

end

end

end

% 定义 Haversine 函数计算两点间距离（单位：千米）

function d = haversine(lat1, lon1, lat2, lon2)

R = 6371; % 地球半径，单位：千米

dLat = deg2rad(lat2 - lat1);

dLon = deg2rad(lon2 - lon1);

lat1 = deg2rad(lat1);

lat2 = deg2rad(lat2);

a = sin(dLat / 2)^2 + cos(lat1)\*cos(lat2)\*sin(dLon / 2)^2;

c = 2\*atan2(sqrt(a), sqrt(1 - a));

d = R \* c;

End

结果：lujing1

最优路径（采样点顺序）: 31

 100

 147

 83

 158

 44

 61

 115

完成当天工作最短的时间（小时）: 6.83

**问题二**

% 读取数据

data = readtable('附件1：xx地区.xlsx');

% 提取经度、纬度和工作时间数据

JD = data.JD;

WD = data.WD;

Work = data.Work;

% 定义地球半径（单位：千米）

EARTH\_RADIUS = 6371.0;

% 计算距离矩阵

num\_points = height(data);

distances = zeros(num\_points, num\_points);

for i = 1:num\_points

for j = 1:num\_points

lat1 = deg2rad(WD(i));

lon1 = deg2rad(JD(i));

lat2 = deg2rad(WD(j));

lon2 = deg2rad(JD(j));

dlon = lon2 - lon1;

dlat = lat2 - lat1;

a = sin(dlat/2)^2 + cos(lat1)\*cos(lat2)\*sin(dlon/2)^2;

c = 2\*atan2(sqrt(a), sqrt(1 - a));

distances(i, j) = EARTH\_RADIUS \* c;

end

end

% 按照就近原则划分点位，每天8个点位

num\_days = ceil(num\_points / 8);

groups = cell(num\_days, 1);

visited = false(1, num\_points);

for day = 1:num\_days

group = [];

if all(visited) == false

if isempty(group)

start = find(visited == false, 1);

else

unvisited = find(visited == false);

last\_visited = group(end);

[~, idx] = min(distances(last\_visited, unvisited));

start = unvisited(idx);

end

group = [group, start];

visited(start) = true;

while length(group) < 8 && any(visited == false)

unvisited = find(visited == false);

current = group(end);

[~, idx] = min(distances(current, unvisited));

next\_point = unvisited(idx);

group = [group, next\_point];

visited(next\_point) = true;

end

end

groups{day} = group;

end

% 计算每个组（每天）的最优路径及工作时间

optimal\_routes = cell(num\_days, 1);

route\_times = zeros(num\_days, 1);

num\_permutations = 1000; % 生成的随机排列数量

for day = 1:num\_days

group = groups{day};

num\_group\_points = length(group);

min\_time = Inf;

best\_route = [];

for p = 1:num\_permutations

route = group(randperm(num\_group\_points));

travel\_time = 0;

for i = 1:num\_group\_points - 1

travel\_time = travel\_time + distances(route(i), route(i + 1)) / 20;

end

work\_time = sum(Work(route));

total\_time = travel\_time + work\_time;

if total\_time < min\_time

min\_time = total\_time;

best\_route = route;

end

end

optimal\_routes{day} = best\_route;

route\_times(day) = min\_time;

end

% 输出结果

figure;

hold on;

colors = 'bgrcmyk'; % 定义颜色列表

markers = '.osd^v<>'; % 定义标记样式列表

legend\_entries = cell(num\_days, 1); % 用于存储图例条目

for day = 1:num\_days

fprintf('第 %d 天的最优路径：', day);

fprintf('%d ', data.XuHao(optimal\_routes{day}));

fprintf('，工作时间：%.2f 分钟\n', route\_times(day));

route = optimal\_routes{day};

route\_JD = JD(route);

route\_WD = WD(route);

color = colors(mod(day - 1, length(colors)) + 1);

marker = markers(mod(day - 1, length(markers)) + 1);

scatter(route\_JD, route\_WD, 'filled', 'MarkerFaceColor', color);

plot(route\_JD, route\_WD, ['-' color marker]);

legend\_entries{day} = sprintf('第 %d 天，工作时间: %.2f 分钟', day, route\_times(day));

end

hold off;

title('所有天的采样路径');

xlabel('经度');

ylabel('纬度');

grid on;

legend(legend\_entries);

fprintf('所有最优路径工作时间的最大值：%.2f 分钟\n', max(route\_times));

fprintf('所有最优路径工作时间的最小值：%.2f 分钟\n', min(route\_times));

结果：lujing2

第 1 天的最优路径：57 1 110 78 218 19 217 11 ，工作时间：420.34 分钟

第 2 天的最优路径：166 67 4 157 2 142 69 179 ，工作时间：392.29 分钟

第 3 天的最优路径：222 221 3 135 81 177 14 37 ，工作时间：404.27 分钟

第 4 天的最优路径：52 56 5 10 149 123 71 150 ，工作时间：405.21 分钟

第 5 天的最优路径：106 148 99 12 21 6 212 211 ，工作时间：401.36 分钟

第 6 天的最优路径：171 62 28 42 7 23 172 50 ，工作时间：399.30 分钟

第 7 天的最优路径：8 54 82 15 63 129 83 158 ，工作时间：387.31 分钟

第 8 天的最优路径：162 104 75 41 35 107 32 9 ，工作时间：388.24 分钟

第 9 天的最优路径：195 191 25 216 128 215 143 13 ，工作时间：385.30 分钟

第 10 天的最优路径：16 124 105 194 151 47 167 120 ，工作时间：405.36 分钟

第 11 天的最优路径：17 119 132 165 164 141 49 169 ，工作时间：387.28 分钟

第 12 天的最优路径：184 185 176 198 197 40 153 18 ，工作时间：388.31 分钟

第 13 天的最优路径：60 61 44 202 200 20 204 203 ，工作时间：399.28 分钟

第 14 天的最优路径：127 170 173 55 163 154 207 22 ，工作时间：401.25 分钟

第 15 天的最优路径：214 213 39 175 24 45 131 122 ，工作时间：385.27 分钟

第 16 天的最优路径：205 206 115 31 100 147 140 26 ，工作时间：376.30 分钟

第 17 天的最优路径：79 145 103 116 144 113 160 27 ，工作时间：395.52 分钟

第 18 天的最优路径：53 70 155 156 29 72 138 102 ，工作时间：415.43 分钟

第 19 天的最优路径：30 189 188 187 186 85 152 126 ，工作时间：425.28 分钟

第 20 天的最优路径：134 192 193 190 97 51 33 89 ，工作时间：375.45 分钟

第 21 天的最优路径：66 220 114 95 219 178 208 34 ，工作时间：394.35 分钟

第 22 天的最优路径：64 108 117 109 121 65 58 36 ，工作时间：382.28 分钟

第 23 天的最优路径：38 98 48 77 101 199 201 130 ，工作时间：396.05 分钟

第 24 天的最优路径：43 68 137 76 94 86 91 88 ，工作时间：404.40 分钟

第 25 天的最优路径：209 210 46 92 80 174 73 133 ，工作时间：391.53 分钟

第 26 天的最优路径：168 87 125 59 111 182 161 74 ，工作时间：409.34 分钟

第 27 天的最优路径：96 159 90 146 84 196 139 136 ，工作时间：390.32 分钟

第 28 天的最优路径：180 118 112 93 183 181 ，工作时间：318.16 分钟

所有最优路径工作时间的最大值：425.28 分钟

所有最优路径工作时间的最小值：318.16 分钟

**问题四**

% 读取数据

data = readtable('附件1：xx地区.xlsx');

% 筛选出指定序号的点位数据

selected\_indices = [31, 44, 61, 83, 100, 115, 147, 158];

selected\_data = data(ismember(data.XuHao, selected\_indices), :);

% 计算距离矩阵

num\_points = height(selected\_data);

distance\_matrix = zeros(num\_points, num\_points);

for i = 1:num\_points

for j = i+1:num\_points

lat1 = selected\_data.WD(i);

lon1 = selected\_data.JD(i);

lat2 = selected\_data.WD(j);

lon2 = selected\_data.JD(j);

distance = haversine(lat1, lon1, lat2, lon2);

distance\_matrix(i, j) = distance;

distance\_matrix(j, i) = distance;

end

end

% 获取各点位的工作时间

work\_times = selected\_data.Work;

% 设置模拟退火算法参数

T = 1000;

T\_min = 1;

alpha = 0.99;

max\_iter = 1000;

% 运行模拟退火算法

[best\_path, best\_total\_time] = simulated\_annealing(distance\_matrix, work\_times, T, T\_min, alpha, max\_iter);

% 输出最优路径和最短工作时间

optimal\_indices = selected\_data.XuHao(best\_path);

fprintf('最优路径（采样点顺序）: ');

disp(optimal\_indices);

fprintf('完成当天工作最短的时间: %.2f\n', best\_total\_time);

% 绘制最优路径图

lats = selected\_data.WD(best\_path);

lons = selected\_data.JD(best\_path);

% 闭合路径

lats = [lats; lats(1)];

lons = [lons; lons(1)];

figure;

plot(lons, lats, '-o');

xlabel('经度');

ylabel('纬度');

title('最优采样路径');

grid on;

% 定义Haversine公式计算两点间距离

function distance = haversine(lat1, lon1, lat2, lon2)

% 将十进制度数转换为弧度

lon1 = deg2rad(lon1);

lat1 = deg2rad(lat1);

lon2 = deg2rad(lon2);

lat2 = deg2rad(lat2);

% Haversine公式

dlon = lon2 - lon1;

dlat = lat2 - lat1;

a = sin(dlat/2).^2 + cos(lat1).\*cos(lat2).\*sin(dlon/2).^2;

c = 2.\*atan2(sqrt(a), sqrt(1 - a));

r = 6371; % 地球平均半径，单位为公里

distance = c.\*r;

end

% 定义模拟退火算法求解TSP问题

function [best\_path, best\_total\_time] = simulated\_annealing(distance\_matrix, work\_times, T, T\_min, alpha, max\_iter)

num\_points = size(distance\_matrix, 1);

current\_path = randperm(num\_points);

current\_distance = sum(distance\_matrix(current\_path, circshift(current\_path, -1, 2)), 'all');

current\_work\_time = sum(work\_times(current\_path));

current\_total\_time = current\_distance + current\_work\_time;

best\_path = current\_path;

best\_total\_time = current\_total\_time;

while T > T\_min

for iter = 1:max\_iter

% 随机交换两个位置

i = randi(num\_points);

j = randi(num\_points);

new\_path = current\_path;

new\_path([i, j]) = new\_path([j, i]);

new\_distance = sum(distance\_matrix(new\_path, circshift(new\_path, -1, 2)), 'all');

new\_work\_time = sum(work\_times(new\_path));

new\_total\_time = new\_distance + new\_work\_time;

delta = new\_total\_time - current\_total\_time;

if delta < 0 || rand < exp(-delta/T)

current\_path = new\_path;

current\_total\_time = new\_total\_time;

if new\_total\_time < best\_total\_time

best\_path = new\_path;

best\_total\_time = new\_total\_time;

end

end

end

T = T \* alpha;

end

end