逢山开路

摘要

对于给定的地形和必经点的路径规划问题，本文将给出具体的解决方法设计线路使其总造价尽可能低。

本题将需要设计的线路看做一条曲线，将该曲线的切线方向上的方向导数作为约束条件，将造价作为曲线积分的权值，求解得到最优方案。本文将连续曲线简化为离散的线段和进行求解。首先利用双三次插值将数据加细，尽可能还原地形来设计更精细的路线。然后我们通过定性与定量分析，分别建立了沿等高线设计路线的方法和动态规划的求解方法。沿等高线设计路线的核心思想是少爬山或下坡让总路线最短从而优化总造价。动态规划模型基于对桥梁和隧道的造价远高于普通公路的考虑。我们需要先优化桥梁和隧道的长度。确定好它们的修建位置后，路线就转化为分段求和问题。而公路路段的造价与长度成正比，通过求解公路的最短路径即可得到最小造价设计方案。我们分别使用 Dijkstra 算法和 SPFA 算法求解最短路径然后取最优解。

对于居民点，我们将整个路径划分为山脚到桥、桥、桥到居民点、居民点到隧道、隧道、隧道到矿区共六段路程。因为隧道长度大于 300 米时价格翻倍，所以将隧道的长度控制在 300 米之内。通过模型得到一种总长度为 10199.65 米，造价为 357 万元的设计方案。对于居民区，我们保持第一问的设计不变，通过对比分析四个定点确定公路经过居民区的点的坐标为 $(3600, 2400, 1150)$，利用 SPFA 算法计算得到一种总长度为 9392.63 米，造价 332 万元的设计方案。

最后，我们求解了不修建隧道、改变桥梁位置两种情况下的设计方案，造价均在 400 万元以上，说明我们的方案相对合理且有效。

关键字： 动态规划， 局部最优， 单源最短路径， 坡度

目录

一　问题重述

我们要在一山区修建公路，已知该山区各点的高程。数据显示：在 $y = 3200$ 处有一东西走向的山峰；从坐标 $(2400, 2400)$ 到 $(4800, 0)$ 有一西北——东南走向的山谷；在 $(2000, 2800)$ 附近有一山口湖，其最高水位略高于 1350 米，雨季在山谷中形成一溪流。经调查知，雨量最大时溪流水面宽度 w 与（溪流最深处的）x 坐标的关系可近似表示为:

$$w(x) = (\frac{x - 2400}{2})^{3/4} + 5 \quad (2400 \leqslant x \leqslant 4000) \tag{1}$$

公路从山脚 $(0, 800)$ 处开始，经居民点（4000，2000）至矿区（2000，4000）。已知路段工程成本及对路段坡度（上升高程与水平距离之比）的限制如表1。

表 **1**　造价与坡度要求

工程种类	一般路段	桥梁	隧道
工程成本 (元/米)	300	2000	1500(长度 ⩽300 米);3000(长度 >300 米)
对坡度的限制	<0.125	=0	<0.100

1. 给出一种线路设计方案，包括原理、方法及比较精确的线路位置（含桥梁、隧道），并估算该方案的总成本。
2. 居民点改为 $3600 \leqslant x \leqslant 4000,\quad 2000 \leqslant y \leqslant 2400$ 的居民区，公路只须经过民区即可，提出修改方案。

概括即：我们需要根据地形（高度）和工程成本来决定一条最佳的线路，本题中我们将总成本作为最终最优化的目标。

二　问题分析

在本题中，通过给定的数据表，我们可以得到该山区的地形图，如图（1，2）

通过观察该地形图，发现直接使用原有数据进行计算难以满足工程中对于不同工程种类的坡度限制要求。同时，采用在格点间修路的方式容易导致出现大角度转弯等现象。我们需要对地形数据进行优化，以满足计算需要。

同时，观察地形图发现在出发点和居民点之间存在一道谷地，在雨季会形成小溪，需要修建桥梁；在居民点和矿区之间存在一高山阻挡，且高山的坡度很大，可能需要考虑修建隧道以缩短盘山公路长度。详见图（3）。

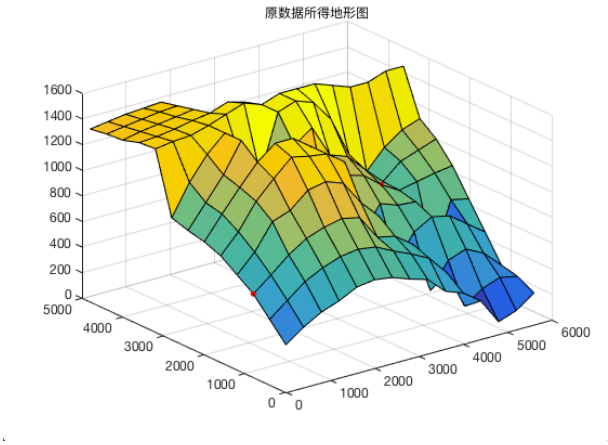

图 **1**　通过给定数据的到的山区地形图

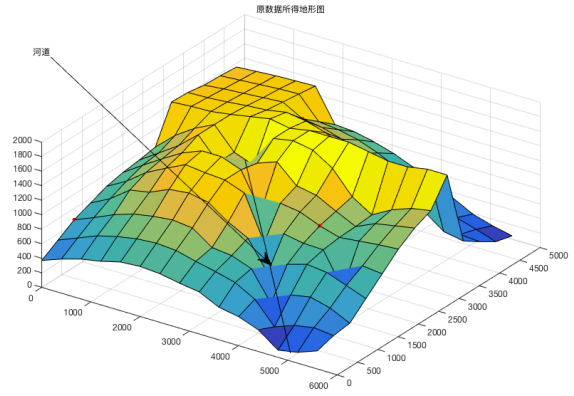

图 **2**　河道面地形图

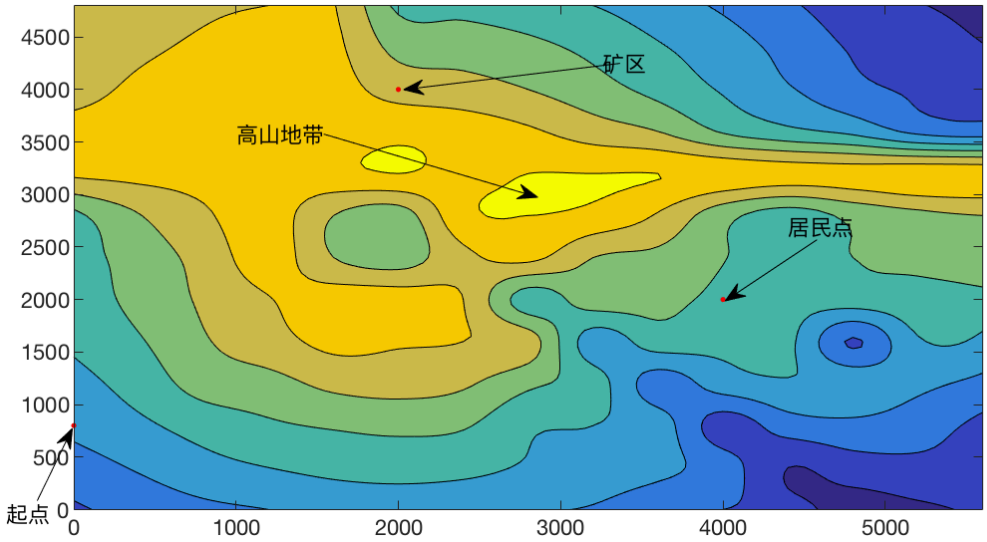

图 **3**　等高线图

三　模型建立

3.1　理想模型

本题中可将问题一般化抽象为求解最优路线的问题，对于一般给定的路径如图 (4)、公式 (2)，可以通过计算该曲线的曲线积分得到路径总长度。同时，为了满足题目条件中的坡度限制要求，分别对不同路段线路的方向导数添加限制条件 (3)，最终得到路径规划。

$$l:\begin{cases} x = & x(t) \\ y = & y(t) \\ z = & f(x,y) \end{cases} \qquad l=\int_l ds \tag{2}$$

$$\frac{dz}{d\bar{l}} \leqslant K \tag{3}$$

其中 K 为不同种类路段的坡度要求。

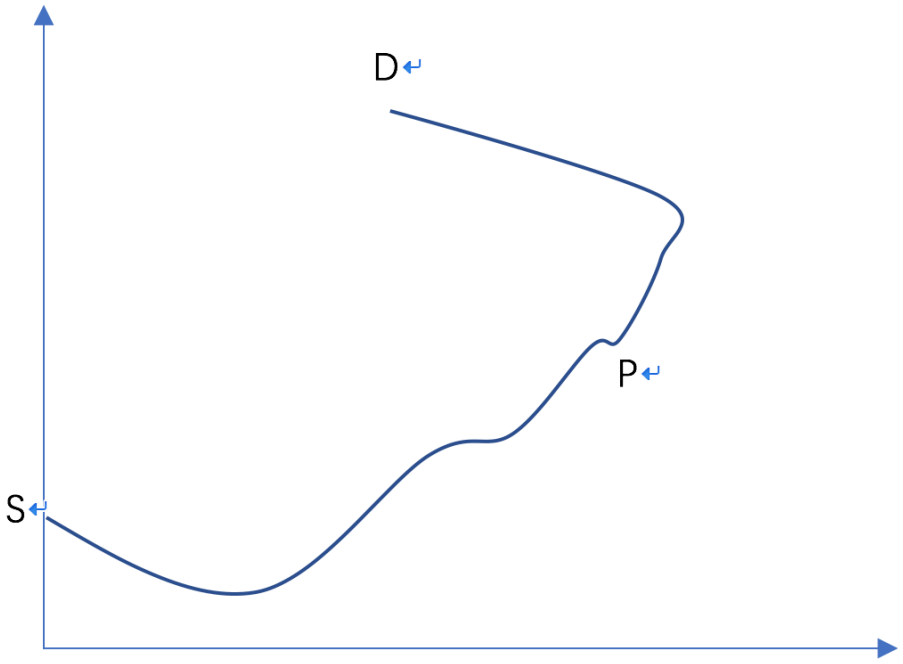

图 **4**　示意图

3.2　求解方案

针对河流和高山影响，若绕过河流则无法实现目标设定，因此桥梁必须修建。

若考虑修建隧道，则可将全程分为 6 个部分（起点—桥东侧，桥梁，桥西侧-居民点，居民点-隧道南侧，隧道，隧道北侧-矿区），并且设置 4 个控制点以分别确定大桥和隧道的修建位置。

若不考虑修建隧道，可将全程分为 4 个部分（起点—桥东侧，桥梁，桥西侧-居民点，居民点-矿区）。

3.3　符号声明

- S：起点位置 $(0, 800)$
- B_1：大桥在峡谷西侧的起始位置（未定）
- B_2：大桥在峡谷西侧的起始位置（未定）
- P：居民区 $600 \leqslant x \leqslant 4000, 2000 \leqslant y \leqslant 2400$ ／居民点 $(4000, 2000)$
- T_1：隧道在山峰南侧的起始位置（未定）
- T_2：隧道在山峰北侧的结束位置
- D：矿区位置 $(2000, 4000)$

3.4 基本假设

1. 满足坡度要求，工程上即能够实现修路修隧道或者架桥。
2. 将地图抽象为离散的点，以简化模型。
3. 无重大地质灾害。
4. 路线近似曲线，但相邻细分的点之间均用直线相连。

四 模型求解前准备

4.1 地形数据插值优化

针对题目给定地形数据量不足，采用双三次插值（立方卷积插值）的方法对原有数据进行加密，使得各个格点间距离减小，便于模型计算时使用。原有数据点间距为 400 米，我们将其间距插值为 50 米，将原有每个方格面积缩小为原面积的 $\frac{1}{64}$。

双三次插值（立方卷积插值）[2] 创造的数据点的数值来源于该点附近的 (4×4) 个数据点，精确度较高。

定义高程数据点集合为 F，可以通过公式（4）得到，

$$F(i+v, j+u) = ABC \tag{4}$$

其中

$$A = F(1+u,:) + F(u,:) + F(1-u,:) + F(2-u,:)$$

$$B = \begin{bmatrix} F(i-1,j-2) & F(i,j-2) & F(i+1,j-2) & F(i+2,j-2) \\ F(i-1,j-1) & F(i,j-1) & F(i+1,j-1) & F(i+2,j-1) \\ F(i-1,j) & F(i,j) & F(i+1,j) & F(i+2,j) \\ F(i-1,j+1) & F(i,j+1) & F(i+1,j+1) & F(i+2,j+1) \end{bmatrix}$$

$$C = F(1+v,:) + F(v,:) + F(1-v,:) + F(2-v,:)$$

对数据进行插值计算的具体实现详见附录，经过插值之后，得到了新的山区地形图如图（5）所示

4.2 对河流宽度的计算

通过对图（5）的观察，可以看出：峡谷的谷底基本为直线，峡谷两侧基本对称。谷底方程可近似为：$x + y = 4800 \quad (2400 \leqslant x \leqslant 4800)$ 结合公式（1）可得：

河岸西侧方程：

$$4800 - x - y = \frac{\sqrt{2}}{2}\omega\left(\frac{x - y + 4800}{2}\right) \quad (2400 \leqslant x \leqslant 4800) \tag{5}$$

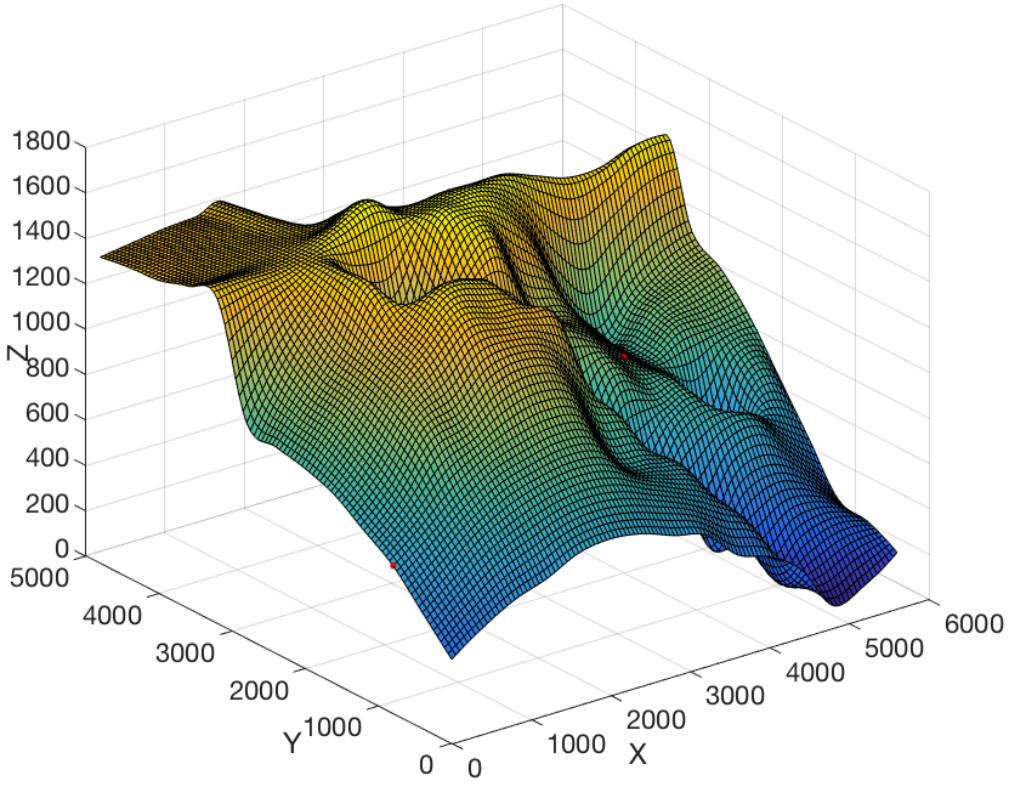

图 **5**　插值后得到的山区地形图

河岸东侧方程:

$$x+y-4800=\frac{\sqrt{2}}{2}\omega\left(\frac{x-y+4800}{2}\right)\quad(2400\leqslant x\leqslant 4800)\tag{6}$$

五　问题一分析与求解

5.1　方案一

通过插值数据得到更为精确的等高线图, 为了保证公路的坡度符合要求，沿等高线修建公路，同时为了降低桥梁和隧道的建设成本，通过绕路的方式实现桥梁和隧道长度的最小化。可以得到修建路线图如下图（6）所示:

该模型仅可作为粗略的估计方法，沿等高线修路直观、简便、但非常不精确，只能得到大概的路线以及费用估计值。同时，若要在坡度变化较大的地方通过等高线方法修路，必须使用等高线密度足够大的等势图。由于无法求得具体的路线长度，所以无法估计该修建方法所需的花费。

5.2　方案二

本题核心问题为求解从起点 S 到居民点 P 再到矿区 D 两段路经的最短路径，但由于桥梁和隧道的影响，直接求解存在极大困难。因此根据对于地形雨同路段费用系数的分析先确定桥梁和隧道的修建位置 [1]。

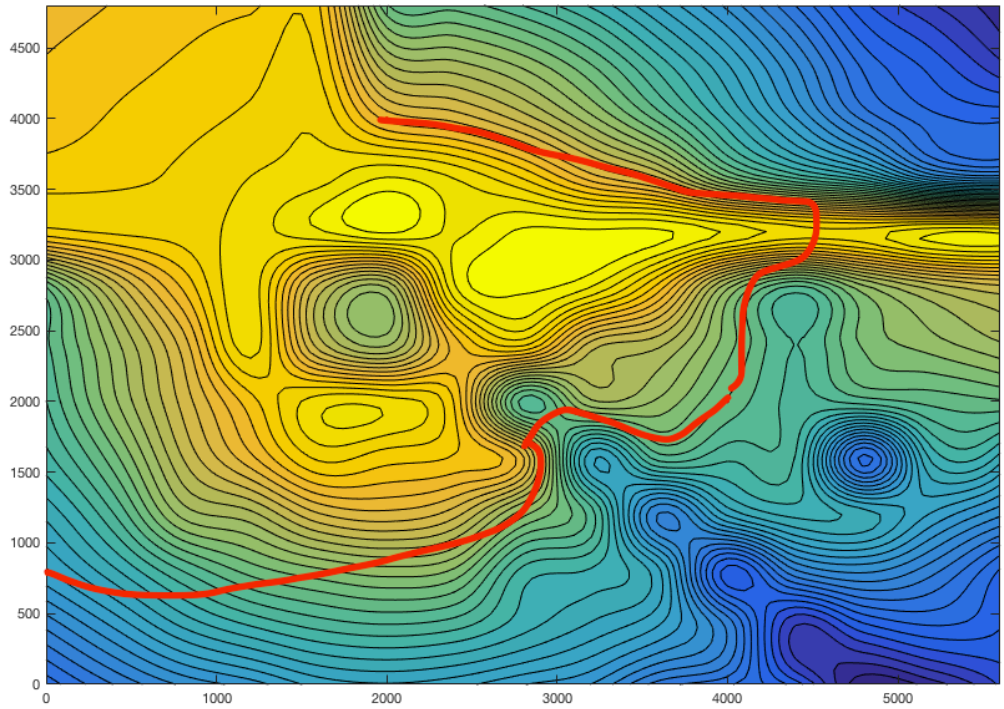

图 **6**　公路修建路线图

在确定大桥和隧道的修建位置之后，剩余四段路均为普通公路。由于桥梁和隧道的建设费用已经确定，若使得工程总费用最小，问题转化为求解四段公路的最短路径。

由于最短路径 Dijkstra 算法基于二维平面上的带权图，所以需要对插值后的数据进行处理，以便使用最短路径算法求解公路修建路径。

5.2.1　桥梁位置选择

通过公式（5，6）可知，小溪的宽度随 x 的增加呈递增关系。再从小溪左右公路对坡度要求，我们暂时确定小溪的位置在点之间，因为小溪的直线方程从点开始为：

$$P_{start} = \begin{cases} x = & 2800 + 400t \\ y = & 2000 - 400t \\ z = & 900 - 200t \end{cases}$$

通过计算发现，在小溪上游与下游修建桥梁的长度差距在 50 米以内，但在最上游修建桥梁虽然可以使得桥长最短，但会导致第一段普通公路严重绕行，在下游修桥更会导致桥长和第一段公路长度的大大加长，因此我们先假设在小溪中心高程 $z = 800$ 的地方修桥。在这样的高程上，我们找到小溪上对应的点为 $B(3000, 1800, 800)$，由此算出溪流的宽度：$\omega(3000) = 77.084M$。

由于桥梁的建设要求坡度为零，从这方面考虑，则桥的两个落点只能在点 $B_{11}(2800, 1600, 1300)$, $B_{21}(3200, 2000, 1100)$ 这两点与 A 点的两条直线上确定，为了使得桥的宽度最接近于小溪宽度，通过计算，我们求得桥的两端点为 $B_1(2978, 1788, 855)$,

$B_2(3036, 1836, 855)$，则桥的实际高程为 855 米，桥的实际宽度为 82 米。桥梁修建位置如下图（7）所示。

5.2.2 隧道位置选择

由题目所给数据容易看出，修建隧道的成本很高，特别是当隧道长度超过 300 米时，成本会激增 1 倍，达到普通公路修建成本的 10 倍，因此我们应当尽量选取宽度在 300 米以内的山峰处修建隧道，通过等高图（3）可以发现在 $x = 4400M$ 处的山峰非常尖锐，非常适合于修建隧道以节约成本，因此在以下的计算中，我们固定隧道的 x 坐标为 $x = 4400M$。

为了使得隧道长度小于 300 米，我们需要通过修建盘山公路的方式将公路高度抬升到一定高度，对于水平隧道

$$Z_c = 1500 - \frac{300 \times 650}{400 \times (1 + \frac{650}{600})} = 1266$$

由于隧道的坡度限制条件为 $\alpha \leqslant 0.100$ 通过数据表可知矿区海拔高度为 1320 米，考虑修建一条南高北低的隧道，同时令隧道坡度小于临界值 0.100，通过计算得到了隧道南北两侧的定位点 $T_1(4400, 3050, 1341)$，$T_2(4400, 3350, 1329)$ 修建的隧道位置如图（8）所示。

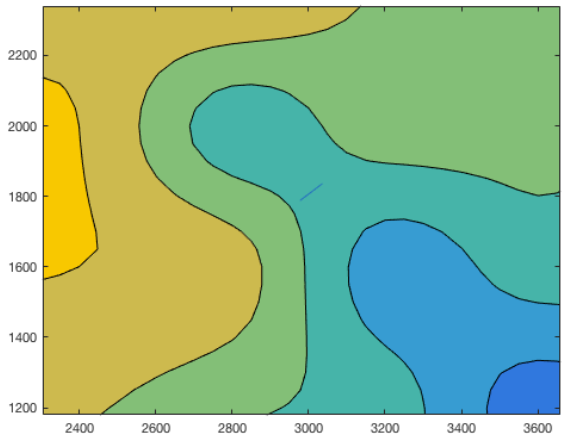

图 **7** 桥梁修建位置

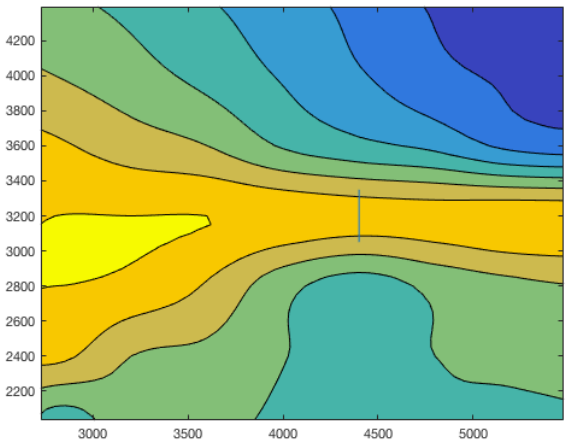

图 **8** 隧道修建位置

5.2.3 普通公路修建位置选择-Dijkstra 算法

对于普通公路，我们采用适用于稠密图且复杂度仅为 $o(nlogn)$ 的 Dijkstra 最短路径算法进行求解。该算法的核心思想是以起始点为中心向外层层扩展，直到扩展到终点为止。

动态规划 Dijkstra 算法的设计思路

1. 首先，引入一个辅助向量 D，它的每个分量 $D[i]$ 表示当前所找到的从起始点 v（即源点 v）到其它每个顶点 v_i 的长度。
2. D 的初始状态为：若从 v 到 v_i 有弧（即从 v 到 v_i 存在连接边），则 $D[i]$ 为弧上的权值（即为从 v 到 v_i 的边的权值）；否则置 $D[i]$ 为 ∞。

   显然，长度为 $D[j] = MinD|v_i \in V$ 的路径就是从 v 出发到顶点 v_j 的长度最短的一条路径，此路径为 (v, v_j)。
3. 找到对于下一条长度次短的一条边，也就是找到从源点 v 到下一个顶点的最短路径长度所对应的顶点，且这条最短路径长度仅次于从源点 v 到顶点 v_j 的最短路径长度。

   假设该次短路径的终点是 v_k，则可想而知，这条路径要么是 (v, v_k)，或者是 (v, v_j, v_k)。它的长度或者是从 v 到 v_k 的弧上的权值，或者是 $D[j]$ 加上从 v_j 到 v_k 的弧上的权值。
4. 一般情况下，假设 S 为已求得的从源点 v 出发的最短路径长度的顶点的集合，则可证明：下一条次最短路径（设其终点为 x）要么是弧 (v, x)，或者是从源点 v 出发的中间只经过 S 中的顶点而最后到达顶点 x 的路径。

   因此，下一条长度次短的的最短路径长度必是 $D[j] = MinD[i]|v_i \in V - S$，其中 $D[i]$ 要么是弧 (v, v_i) 上的权值，或者是 $D[k](v_k \in S)$ 和弧 (v_k, v_i) 上的权值之和。

针对本问题的处理

在本问题中，我们拥有的并不是二维平面上的带权值图，而是三维空间上的高程图，因此我们需要通过一种算法将三维空间中的高度信息转化为二维图中的边权值信息。

考虑到普通公路修建是需要满足坡度条件：$\alpha \leqslant 0.125$，我们考虑在将高程信息表中网格通过双三次插值的方法将网格从 400×400 加密到 10×10，以便于每个顶点找到前往附近 8 各顶点的通路。

将插值之后的每个格点视作二维平面图的顶点，我们针对每个顶点邻接的 8 顶点计算其权值。邻接方式如下矩阵所示：

$$\begin{bmatrix} P(i-1,j-1) & P(i-1,j) & P(i-1,j+1) \\ P(i,j-1) & P(i,j) & P(i,j+1) \\ P(i+1,j-1) & P(i+1,j) & P(i+1,j+1) \end{bmatrix}$$

考虑到普通公路的坡度限制之后，我们将两顶点间边的坡度大于 0.125 的边看作非通路（∞），通过如下公式 () 对每个顶点的边权值进行计算，具体实现详见附录。

$$d_{ij} = \begin{cases} [(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2]^{1/2} & \frac{|\Delta z|}{[(\Delta x)^2+(\Delta z)^2]^{1/2}} \leqslant 0.125 \\ \infty & \frac{|\Delta z|}{[(\Delta x)^2+(\Delta z)^2]^{1/2}} > 0.125 \end{cases} \tag{7}$$

5.2.4 结果与分析

通过计算，得到工程设计方案如下图（9）所示，工程费用如表（2）所示。

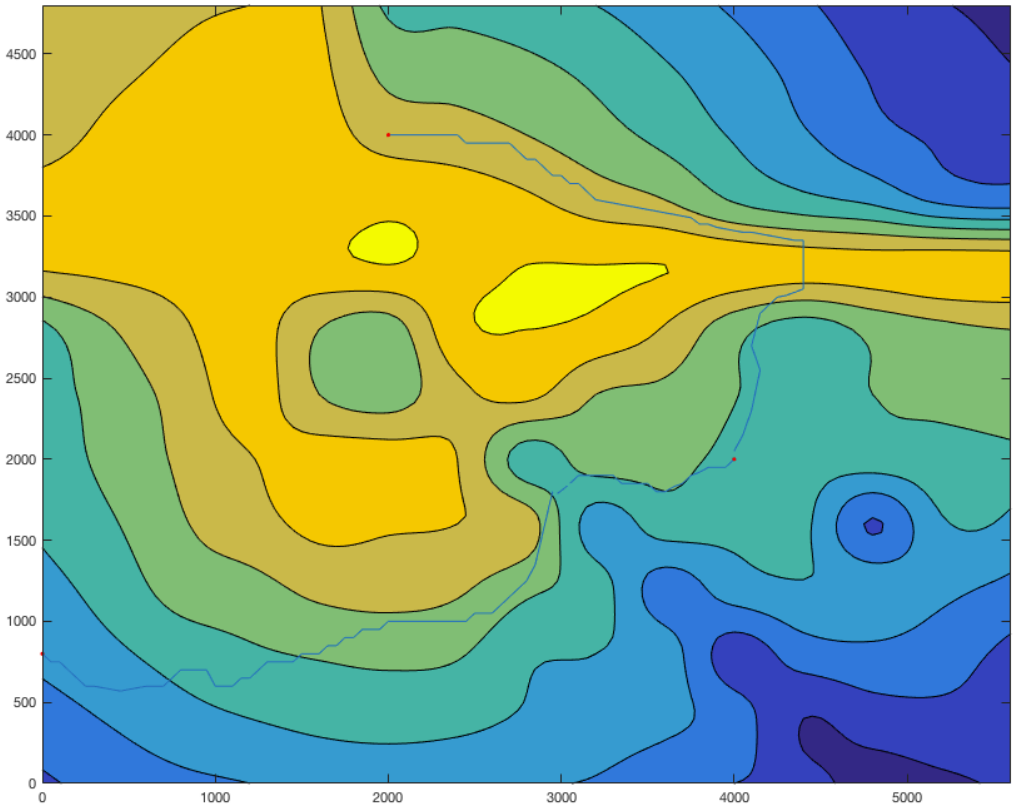

图 **9**　结果路线图

表 **2**　工程费用表一

项目	桥梁	隧道	普通公路	总计
长度	82 米	304.3 米	9813.35 米	10199.65 米
造价	164000 元	462900 元	2944005 元	3570905 元

六　问题二模型与求解

6.1　方案一

与第一问中的情况相似，沿等高线修路的方法只能得到粗略的线路方向，如下图（10）所示：

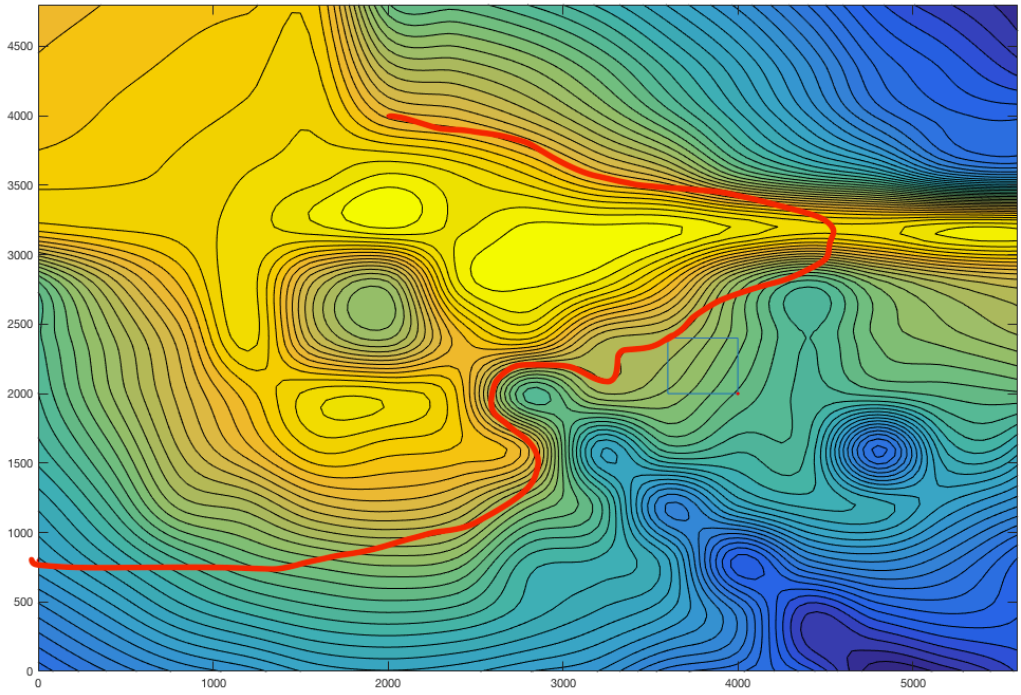

图 **10**　公路修建线路图

6.2　方案二

对于第二问，题目条件仅仅改变了居民点的范围，因此只需要针对第一问模型中从桥梁东岸到居民点和从居民点到隧道南侧入口的两段普通公路进行修改即可。

6.2.1　动态规划 SPFA 算法

考虑到分段之后，每一段路可以抽象为单源路径，故考虑采用单源最短路算法 SPFA 对第二问进行求解，其在 Bellman-ford 算法的基础上加入队列优化，减少了冗余的松弛操作，极大提高了算法效率。有助于更快的求解出所需的结果。

算法设计思路

通过建立一个队列，初始时队列里只有起始点，再建立一个表格记录起始点到所有点的最短路径（该表格的初始值要赋为极大值，该点到他本身的路径赋为 0）。然后执行松弛操作，用队列里有的点作为起始点去刷新到所有点的最短路，如果刷新成功且被刷新点不在队列中则把该点加入到队列最后。重复执行直到队列为空。最终得到最短路径所经过的各个顶点的列表。

沿用第一问中对于数据的处理以及获得的地形图各顶点的邻接矩阵进行计算。

6.2.2　分段点的确定

通过图（11）可以发现，若使得普通公路尽量减少绕行，应当从居民区左上角一点处通过，其余各个分段点与第一问保持不变。

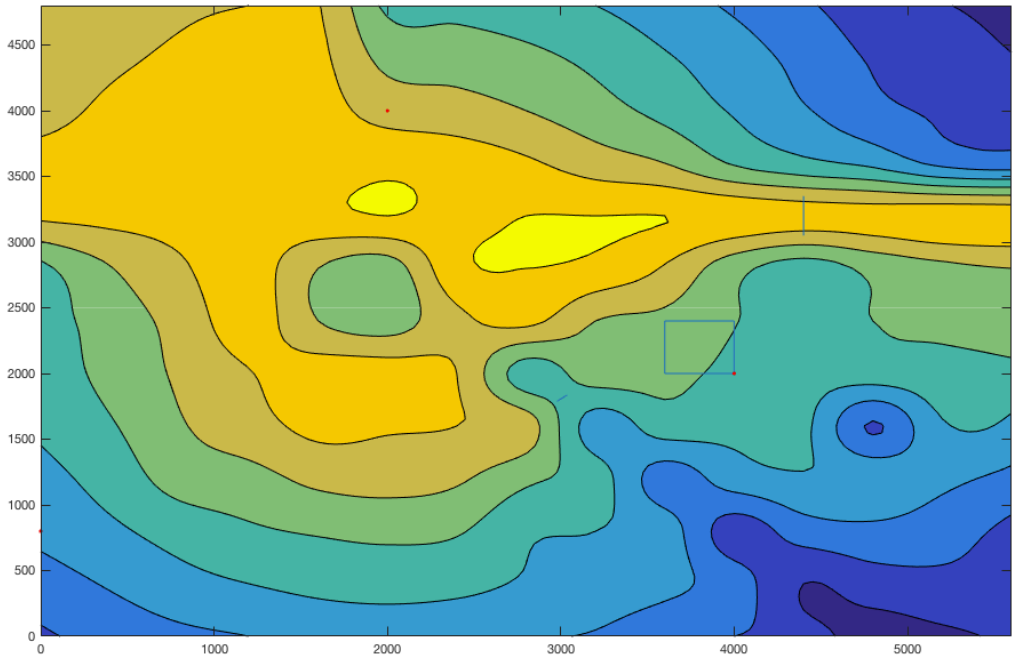

图 **11**　居民区位置

6.2.3　结果与分析

通过 SPFA 算法分段计算最短路径，得到工程设计方案如下图（12）所示，工程费用如表（3）所示。

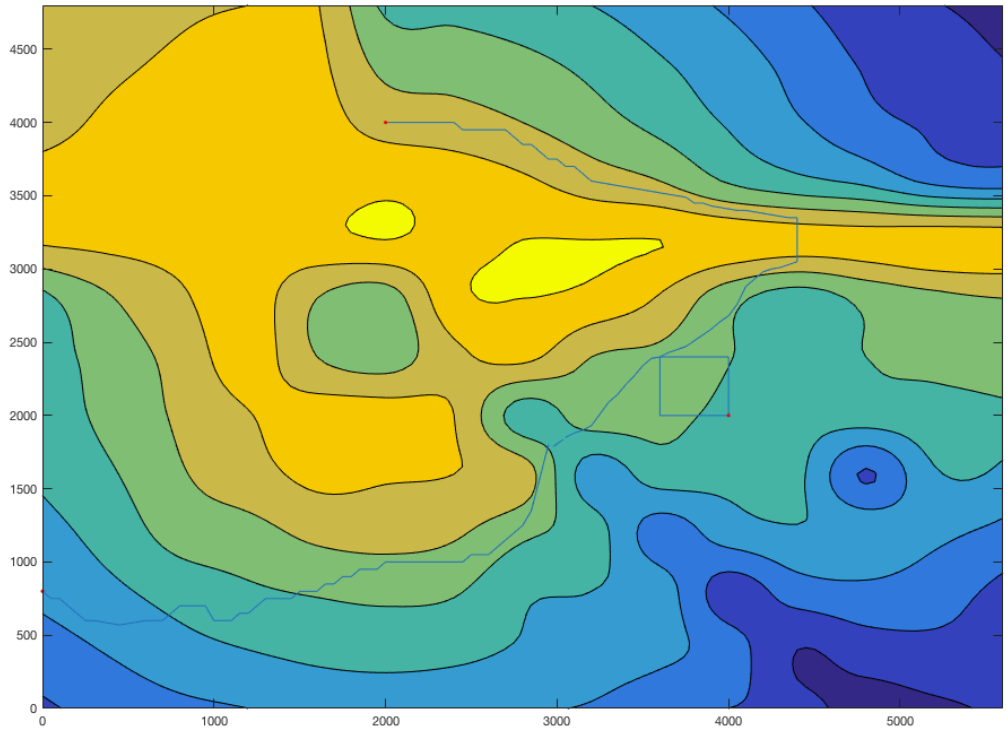

图 **12**　结果路线图

表 **3**　工程费用表二

项目	桥梁	隧道	普通公路	总计
长度	82 米	304.3 米	9006.33 米	9392.63 米
造价	164000 元	462900 元	2701899 元	3328799 元

七　方案合理性分析

由于求解所有可能情况存在计算上的极大困难，因此我们优先采用了固定桥梁隧道位置的方法对方案与求解进行简化，但这种简化使得我们的方案只能确保为可行方案，但无法保证方案对成本足够节约。因此我们通过分析改变桥梁位置、采取修建盘山路来避免修建隧道这两种不同方案，来验证我们方案的成本是否相对较优。

对于需要验证的两种方案，在求解时除确定桥梁隧道位置外，均采用最短路径的方法进行求解。

7.1　不修建隧道

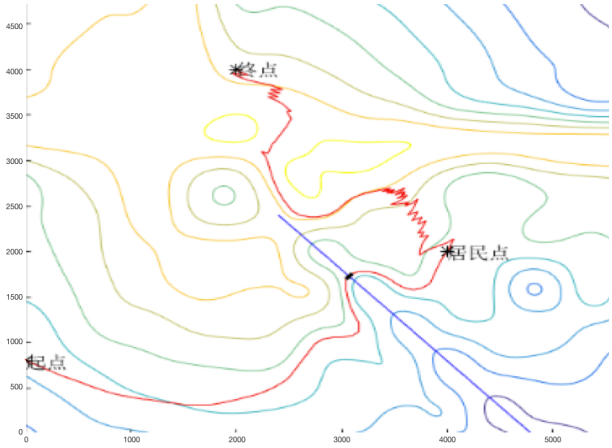

图 **13**　居民点

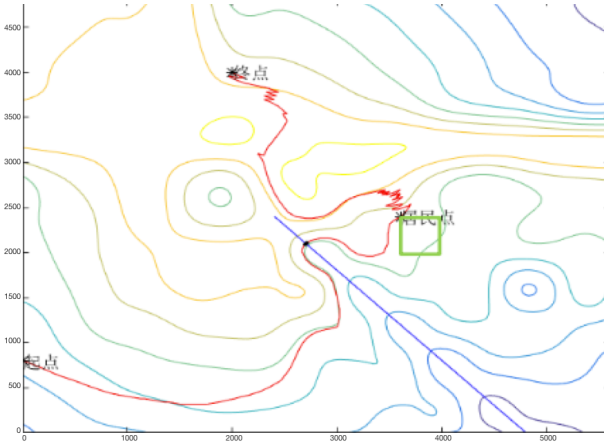

图 **14**　居民区

表 **4**　不修建隧道时工程总费用

方案	费用
方案一：居民点，不修建隧道	4126064 元
方案二：居民区（过点 (3600, 2000)），不修建隧道	3679765 元

7.2 改变桥梁位置

修改桥梁位置到点 $B_1(3200,1200)$,$B_2(3200.1600)$ 之间，桥长为 110.737 米。计算发现居民点变为居民区对于该种桥梁修建方案无影响。进而修改隧道修建位置到 $x=4600$ 处，隧道长度可缩短为 280.00 米。计算所得路线规划如下图 (15) 所示，总造价为 4626928 元。调整为居民区后路径与造价不变。

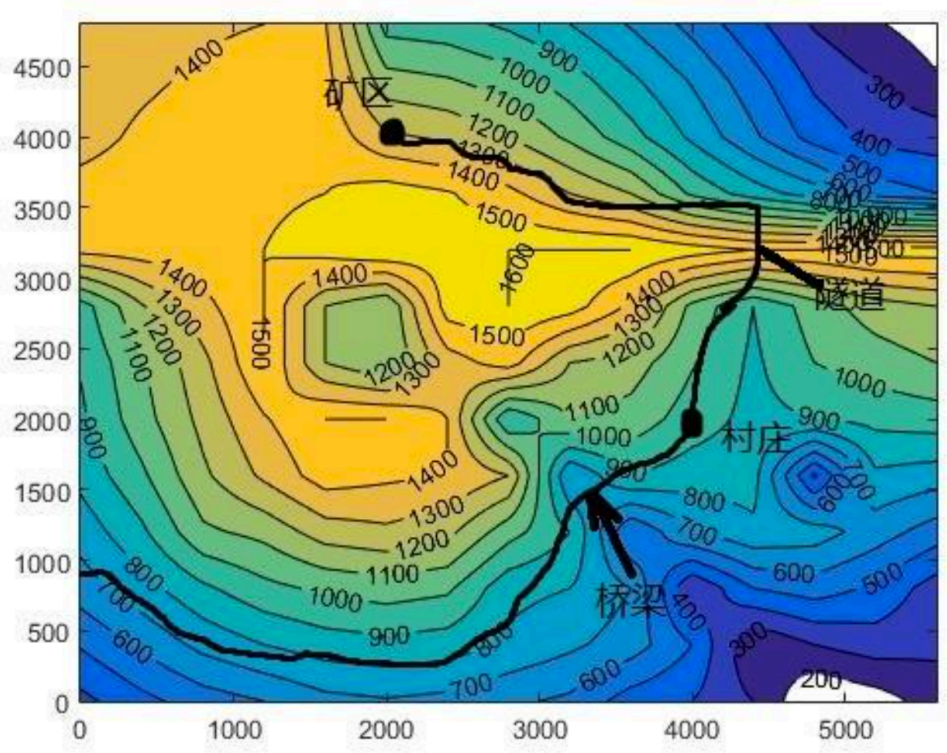

图 **15** 方案路线图

通过对两种情况的分析，可以看出，我们的方案相对于不修隧道、修长桥短隧道的方案更优。可以基本保证方案相对较优。

八 模型评价与改进方向

8.1 模型的评价

8.1.1 优点

1. 通过双三次插值的方式，尽可能精确的获得了计算所需的必要数据。
2. 对于模型一，简单直观，便于直接观察设计修建方案。
3. 对于模型二，Dijkstra 算法与 SPFA 算法的应用，通过动态规划的方式实现各段公路的最优化求解。

8.1.2 不足

1. 对于模型一，无法精确的进行计算，只能简单观察和设计路径，使用价值一般。
2. 对于模型二，主观定义桥梁和隧道的修建位置使得整体的解无法保证最优。
3. 在模型二中，使用最短路径算法存在计算困难的问题，因此细分网格点只能精确到 (100×100) 米的网格进行计算，结果不够精确。且在部分转弯处出现超大角度转弯，不符合实际。

8.2 模型的改进方向

1. 将桥梁位置、隧道位置、居民区边界选点全部纳入动态规划的范围之内，避免主观确定位置带来的影响。
2. 改进最短路径算法，换用 C++ 实现，提高计算效率，避免过长时间的计算。
3. 先通过原有网格点进行动态规划搜索，实现较低复杂度下的全局动态规划。确定控制点之后在相邻控制点间继续细分搜索区域，避免了数据量过大导致的计算不可能问题，又增加了模型精确度。

九 参考文献

[1] 孙长银, 杨峤, 黄琼. 逢山开路的数学模型 [J]. 三峡大学学报 (人文社会科学版), 1998(1):25-28.

[2] 冯仁忠, 查理. 局部构造 C^2 连续的三次 B 样条插值曲线和双三次插值曲面 [J]. 图学学报, 2005, 26(6):110-117.

[3] https://en.wikipedia.org/wiki/Shortest_Path_Faster_Algorithm

[4] https://en.wikipedia.org/wiki/Dijkstra

十 附录：相关程序

10.1 地型相关图形绘制

```
x = [0 400 800 1200 1600 2000 2400 2800 3200 3600 4000 4400 4800 5200 5600];
y = [4800 4400 4000 3600 3200 2800 2400 2000 1600 1200 800 400 0];
z = [1350 1370 1390 1400 1410 960 940 880 800 690 570 430 290 210 150;
1370 1390b 1410 1430 1440 1140 1110 1050 950 820 690 540 380 300 210;
1380 1410 1430 1450 1470 1320 1280 1200 1080 940 780 620 460 370 350;
1420 1430 1450 1480 1500 1550 1510 1430 1300 1200 980 850 750 550 500;
1430 1450 1460 1500 1550 1600 1550 1600 1600 1600 1550 1500 1500 1550 1550;
950 1190 1370 1500 1200 1100 1550 1600 1550 1380 1070 900 1050 1150 1200;
910 1090 1270 1500 1200 1100 1350 1450 1200 1150 1010 880 1000 1050 1100;
880 1060 1230 1390 1500 1500 1400 900 1100 1060 950 870 900 930 950;
830 980 1180 1320 1450 1420 1400 1300 700 900 850 840 380 780 750;
```

```
740 880 1080 1130 1250 1280 1230 1040 900 500 700 780 750 650 550;
650 760 880 970 1020 1050 1020 830  800 700 300 500 550 480 350;
510 620 730 800 850 870 850 780 720 650 500 200 300 350 320;
370 470 550 600 670 690 670 620 580 450 400 300 100 150 250];


surf(x,y,z);
colormap;
hold on;
plot3(0,800,650,r.,LineWidth,20);
plot3(4000,2000,950,r.,LineWidth,20);
plot3(2000,4000,1320,r.,LineWidth,20);
hold off;

figure;
x1= 0:50:5600;
y1= 0:50:4800;
[x2,y2]=meshgrid(x1,y1);
t11=interp2(x,y,z,x2,y2,bicubic);
surf(x1,y1,t11);
colormap;
hold on;
plot3(0,800,650,r.,LineWidth,20);
plot3(4000,2000,950,r.,LineWidth,20);
plot3(2000,4000,1320,r.,LineWidth,20);
plot3([2978 3036],[1788 1836],[855 855]);
hold off;

figure;
contourf(x1,y1,t11,40);
hold on;
plot3(0,800,650,r.,LineWidth,20);
plot3(4000,2000,950,r.,LineWidth,20);
plot3(2000,4000,1320,r.,LineWidth,20);
hold off;

figure;
pcolor(x1,y1,t11);
shading interp
hold on;
plot3(0,800,650,r.,LineWidth,20);
plot3(4000,2000,950,r.,LineWidth,20);
plot3(2000,4000,1320,r,LineWidth,20);
hold off;
```

10.2 地形图抽象化为邻接矩阵

```
x = [0 400 800 1200 1600 2000 2400 2800 3200 3600 4000 4400 4800 5200 5600];
y = [4800 4400 4000 3600 3200 2800 2400 2000 1600 1200 800 400 0];
z = [1350 1370 1390 1400 1410 960 940 880 800 690 570 430 290 210 150;
     1370 1390b 1410    1430 1440 1140 1110 1050 950 820 690 540 380 300 210;
     1380 1410 1430 1450    1470 1320 1280 1200 1080 940 780 620 460 370 350;
     1420 1430 1450 1480    1500 1550 1510 1430 1300 1200 980 850 750 550 500;
     1430 1450 1460 1500    1550 1600 1550 1600 1600 1600 1550 1500 1500 1550 1550;
     950    1190 1370 1500 1200 1100 1550 1600 1550 1380 1070 900 1050 1150 1200;
```

```
      910    1090 1270 1500 1200 1100 1350 1450 1200 1150 1010 880 1000 1050 1100;
      880    1060 1230 1390 1500 1500 1400 900 1100 1060 950 870 900 930 950;
      830    980 1180 1320 1450 1420 1400 1300 700 900 850 840 380 780 750;
      740    880 1080 1130 1250 1280 1230 1040 900 500 700 780 750 650 550;
      650    760 880 970 1020 1050 1020 830  800 700 300 500 550 480 350;
      510    620 730 800 850 870 850 780 720 650 500 200 300 350 320;
      370    470 550 600 670 690 670 620 580 450 400 300 100 150 250];

x1= 0:10:5600;
y1= 0:10:4800;
[x2,y2]=meshgrid(x1,y1);
t11=interp2(x,y,z,x2,y2,'identifierstyle''identifierstyle bicubic identifierstyle');

k = 68400;
lie = 285;
alpha = 1
dis = zeros(k,k);
for i = 2:284
    for j = 2:239
        for k = 2:284
            for m = 2:239
                dx1 = abs(x1(i) - x1(i-k));
                dy1 = abs(y1(j) - y1(j-m));
                dz1 = abs(t11(j,i) - t11(j-m,i-k));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,(j-m)*lie+i-k) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,(j-m)*lie+i-k) = inf;
                end
                dx1 = abs(x1(i) - x1(i+k));
                dy1 = abs(y1(j) - y1(j+m));
                dz1 = abs(t11(j,i) - t11(j+m,i+k));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,(j+m)*lie+i+k) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,(j+m)*lie+i+k) = inf;
                end
                dx1 = abs(x1(i) - x1(i-k));
                dy1 = abs(y1(j) - y1(j+m));
                dz1 = abs(t11(j,i) - t11(j-m,i+k));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,(j+m)*lie+i-k) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,(j+m)*lie+i-k) = inf;
                end
                dx1 = abs(x1(i) - x1(i+k));
                dy1 = abs(y1(j) - y1(j-m));
                dz1 = abs(t11(j,i) - t11(j+m,i-k));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,(j-m)*lie+i+k) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,(j-m)*lie+i+k) = inf;
                end
```

```
                dx1 = abs(x1(i) - x1(i-k));
                dy1 = abs(y1(j) - y1(j));
                dz1 = abs(t11(j,i) - t11(j-m,i));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,(j-m)*lie+i) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,(j-m)*lie+i) = inf;
                end
                dx1 = abs(x1(i) - x1(i+k));
                dy1 = abs(y1(j) - y1(j));
                dz1 = abs(t11(j,i) - t11(j+m,i));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,(j+m)*lie+i) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,(j+m)*lie+i) = inf;
                end
                dx1 = abs(x1(i) - x1(i));
                dy1 = abs(y1(j) - y1(j-m));
                dz1 = abs(t11(j,i) - t11(j,i-k));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,j*lie+i-k) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,j*lie+i-k) = inf;
                end
                dx1 = abs(x1(i) - x1(i));
                dy1 = abs(y1(j) - y1(j+m));
                dz1 = abs(t11(j,i) - t11(j,i+k));
                alpha1 = dz1/(dx1^2+dy1^2)^0.5;
                if alpha1 <= alpha
                    dis(j*lie+i,j*lie+i+k) = (dx1^2+dy1^2+dz1^2)^0.5;
                else
                    dis(j*lie+i,j*lie+i+k) = inf;
                end
            end
        end
    end
end
```

10.3 Dijkstra 算法实现

```
function [ distance path] = Dijk( W,st,e )
%DIJK Summary of this function goes here

n=length(W);
D = W(st,:);
visit= ones(1,n); visit(st)=0;
parent = zeros(1,n);
path =[];
for i=1:n-1
    temp = [];
    for j=1:n
        if visit(j)
```

```
            temp =[temp D(j)];
        else
            temp =[temp inf];
        end
    end

    [value,index] = min(temp);
    visit(index) = 0;

    for k=1:n
        if D(k)>D(index)+W(index,k)
            D(k) = D(index)+W(index,k);
            parent(k) = index;
        end
    end
end

distance = D(e);
t = e;
while t~=st && t>0
    path =[t,path];
    p=parent(t);t=p;
end
path =[st,path];
end
```

10.4 SPFA 算法实现

```
#include <cstdio>
#include <iostream>
using namespace std;
struct node{
    int x;
    int value;
    int next;
};
node e[60000];
int visited[1000000],dis[500000],st[1000000],queue[1000000];
int main(){

    int n,m,u,v,w,start,h,r,cur;
    freopen("identifierstyle\"identifierstylec identifierstyle.identifierstylein identifierstyle\",identifierstyle\"identifierstyler identifierstyle\",stdin);
    freopen("identifierstyle\"identifierstylec identifierstyle.identifierstyleout identifierstyle\",identifierstyle\"identifierstylewidentifierstyle\",stdout);
    while(scanf("identifierstyle\"identifierstyle%identifierstyledidentifierstyle%identifierstyledidentifierstyle\",&n,&m) != EOF){

        for(int i = 1; i <= 1000000; i++){

            visited[i]=0;
            dis[i]=-1;
            st[i]=-1;
        }
        for(int i = 1; i <= m; i++){

            cin>>u>>v>>w;
            e[i].x = v;
```

```
28              e[i].value = w;
29              e[i].next = st[u];
30              st[u] = i;
31          }
32          start = 1;
33          visited[start] = 1;
34          dis[start] = 0;
35          h = 0;
36          r = 1;
37          queue[r] = start;
38          while(h != r){
39
40              h = (h + 1) % 1000;
41              cur = queue[h];
42              long int tmp = st[cur];
43              visited[cur] = 0;
44              while(tmp != -1){
45
46                  if (dis[e[tmp].x] < dis[cur] + e[tmp].value){
47
48                      dis[e[tmp].x] = dis[cur] + e[tmp].value;
49                      if(visited[e[tmp].x] == 0){
50
51                          visited[e[tmp].x] = 1;
52                          r = (r + 1) % 1000;
53                          queue[r] = e[tmp].x;
54                      }
55                  }
56                  tmp = e[tmp].next;
57              }
58          }
59          cout<<dis[n];
60      }
61      return 0;
62  }
```