医疗物资的选址与存储问题

摘要

1998 年长江中下游的特大洪涝灾害与 2008 年的里氏 8.0 级地震, 严重破坏了人们的家园. 自 2020 年以来, 新冠病毒在全世界范围之内肆虐，造成了 2.5 亿感染数的同时, 至今仍影响着人们正常的生产生活. 因此, 建立一个医疗物资仓储基地, 对于自然灾害发生时的应急响应和疫情常态化防控,具有重大意义.

对于问题 1, 首先根据给出的四川省各个地级市经纬度，绘出了各个医疗目的地的经纬度，精确绘制出了其在地图上的位置。使用 Haversine 公式求得球面上两点之间的距离，并将距离和最小化的目标转化为正向目标，得到了优化函数 $Y(center)$，目标转化为求解 $Y(center)$ 最大取值的地点。采用人工鱼群智能算法的思想，根据鱼群在觅食、跟随等等行为之中体现出的群体智能，求解最大食物浓度函数的地点，规划了使距离和最小的地址，并对于不同的 epoch 所得到的结果进行了比较。在第七部分，也使用了 Matlab 中的 fmincon 和传统欧氏距离的方法对于所得到的结果进行了验证。查阅资料可以得到，医疗物资存储基地的最佳选址地点在我国四川省资阳市境内（位置 104.624^0E，30.1314^0N）。

对于问题 2，关于实现总价值最大的目标，建立了背包问题的 0-1 规划的数学模型。通过引入 0-1 类型的数组来表示医疗物品是否被选择，根据医疗存储的体积进行限制，建立了待优化的目标函数。并使用 Matlab 对于这个 0-1 规划的问题进行了求解,得到了具体的医疗物品选取的方案.得到最终的结果是：最大的总价值为 4471，选取物品的方案在附录 1 之中给出。

最后, 给出了模型的优缺点和改进方法.

关键字：鱼群智能算法，选址问题，0-1 规划，Haversine 距离

一、问题重述

1.1 问题背景

通过已知的四川省主要城市中医院的经纬度以及医疗物资价值的信息，拟定医疗物资存储基地的方案，使其到各个城市的飞行总距离之和最短，且存储的医疗物资的总价值最大.

1.2 需要解决的问题

问题 1：附件 1 给出了四川省内 18 个地级市以及 3 个自治州及其医院的经纬度坐标.需要在四川省境内确定一个医疗物资的存储基地的位置，使得该基地到其他城市的飞行距离之和最短.

问题 2：在基地简称之后，需要存放医疗物资.附件 2 之中提供了每件医疗物资的体积和价值.假设医疗物资的数量为 100，基地能够存储 3000 吨的物资.需要拟定一种储存的策略，使得医疗物资放入基地的总价值最大.

二、问题分析

2.1 问题 1

考虑到无人机的飞行高度较地球半径而言可以忽略不计，于是我们可以将两地之间飞行距离视作地球表面上两点的连线。所以需要通过几何公式的推导得到球的表面上两点的距离。同时，需要将距离和取负，使优化的目标进行正向化。然后，采用群体智能算法之中的鱼群算法，将正向化之后的目标函数作为“食物浓度函数”，以寻找最大值。

2.2 问题 2

一个物件只有“取”和“不取”这两种状态，所以可以采用一组 0-1 的变量来表示这个状态。引入体积的限制，建立一个带有线性约束的优化模型。通过 Lingo 软件求解即可得到最终的方案。

三、模型假设与约定

1　假定在飞行过程之中，直升机的飞行路线是球面上的两点连线.忽略地形、气象等实际因素对于飞行路线的影响.

2　忽略直升机的飞行高度.以地球表面两点距离进行计算

3　为方便计算，地球视为均匀球体.

4　选取方案的价值总和具有线性和叠加性

四、变量的声明与定义

变量符号	变量意义
i, j, k	形参
m	四川省城市数量
Cities	城市位置 矩阵
Pos	医疗物资存储基地位置
R_0	地球半径
λ, φ	经纬度
n	待选取医疗物品的数量
V_0	最大存储体积
select	医疗物品选取的状态变量
$\mathbf{P}^n$	医疗物品的价值向量
$\mathbf{V}^n$	医疗物品的体积向量

五、基本模型的建立

5.1 求解球面之中两点之间的距离

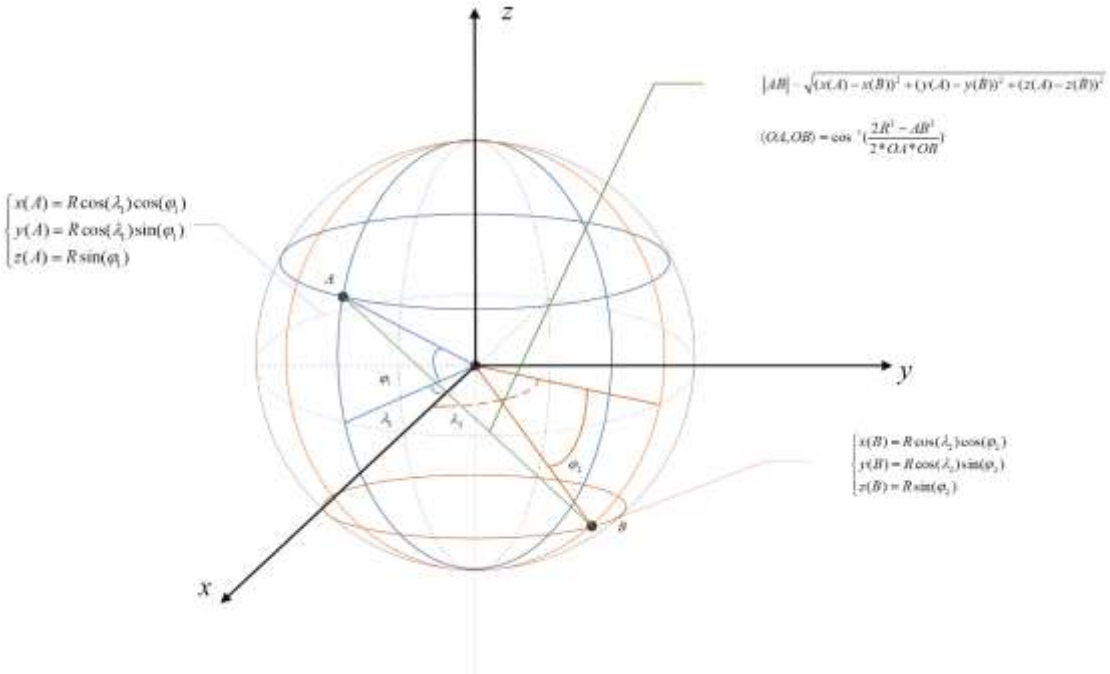


图 5-1 求地球上两点的表面距离示意图

根据几何知识，求过 A,B 两点圆弧的长度可以通过如下的形式进行表示：

$$\left|AB\right| = \langle OA,OB\rangle \cdot R_{外接} = R\langle OA,OB\rangle \quad \text{..... (5.1.1)}$$

所以问题转化为求解两个半径 $\langle OA,OB\rangle$ 之间的角度，以及求解球 O 的表面上 AB 两点外接圆的半径.

在圆面上，已知经度 λ 、纬度 φ 和球面半径 R ，可以写出点 (x,y,z) 的坐标为：

$$(x,y,z) = (R\cos(\lambda)\cos(\varphi), R\cos(\lambda)\sin(\varphi), R\sin(\varphi)) \quad \text{..... (5.1.2)}$$

根据两点之间的距离公式，圆上的两个点 A,B 两点的距离为：

$$\begin{aligned}\left|AB\right|^2 &= (x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2 \\ &= 2R^2(1-\cos(\lambda_1)\cos(\lambda_2)\cos(\varphi_1-\varphi_2) - \sin(\lambda_1)\sin(\lambda_2))\end{aligned} \quad \text{..... (5.1.3)}$$

由余弦定理可以得到 OA,OB 的夹角 $\langle OA,OB\rangle$ ：

$$\langle OA,OB\rangle = \cos^{-1}\left(\frac{OA^2+OB^2-AB^2}{2\cdot OA\cdot OB}\right) = \cos^{-1}\left(1-\frac{AB^2}{2R^2}\right) \quad \text{..... (5.1.4)}$$

整理得到：

$$\left|AB\right| = R\cos^{-1}(\cos(\lambda_1)\cos(\lambda_2)\cos(\varphi_1-\varphi_2) + \sin(\lambda_1)\sin(\lambda_2))^{1} \quad \text{..... (5.1.5)}$$

当 AB 两点的距离相对球的半径较小时， $\cos^{-1}\left(1-\frac{AB^2}{2R^2}\right) = 0.999\cdots \to 1$ ，在计算中存在着舍入误差，可能会存在着精度的问题[2].在 Haversine 公式之中，进行了三角恒等变形： $\text{haversine}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$ ，于是有：

$$hav\left(\frac{\left|AB\right|}{R}\right) = hav(\lambda_2-\lambda_1) + \cos(\lambda_1)\cos(\lambda_2)hav(\Delta\varphi) \quad \text{..... (5.1.6)}$$

$$\left|AB\right| = 2r\sin^{-1}\left(\sqrt{hav(\lambda_2-\lambda_1) + \cos(\lambda_1)\cos(\lambda_2)hav(\Delta\varphi)}\right) \quad \text{..... (5.1.7)}$$

从而得出了第一问之中飞行距离之和最小的优化模型：

$$\min D = \sum_{i=1}^{m} Dist(Pos, Cities(i)) \quad \text{..... (5.1.7)}$$

5.2 背包问题的模型建立

设置向量 *select* ，其第 i 个元素的值表示第 i 个物品是否被选中

$$select(i) = \begin{cases} 1, & \text{if i is chosen;} \\ 0, & \text{otherwise;} \end{cases} \quad \text{..... (5.2.1)}$$

于是选中的物品占据仓库的体积为：

$$V=\sum_{i=1}^{n}\mathbf{select}_i\cdot\mathbf{V}_i=\mathbf{select}\bullet\mathbf{V}^T \quad \text{.....(5.2.2)}$$

同时，医疗物资存储基地具有一定的体积限制V_0：

$$V=\mathbf{select}\bullet\mathbf{V}^T\le V_0 \quad \text{.....(5.2.3)}$$

选中的物品具有的总价值为：

$$P=\sum_{i=1}^{n}\mathbf{select}_i\cdot\mathbf{P}_i=\mathbf{select}\bullet\mathbf{P}^T \quad \text{.....(5.2.4)}$$

于是建立的背包问题的模型为：

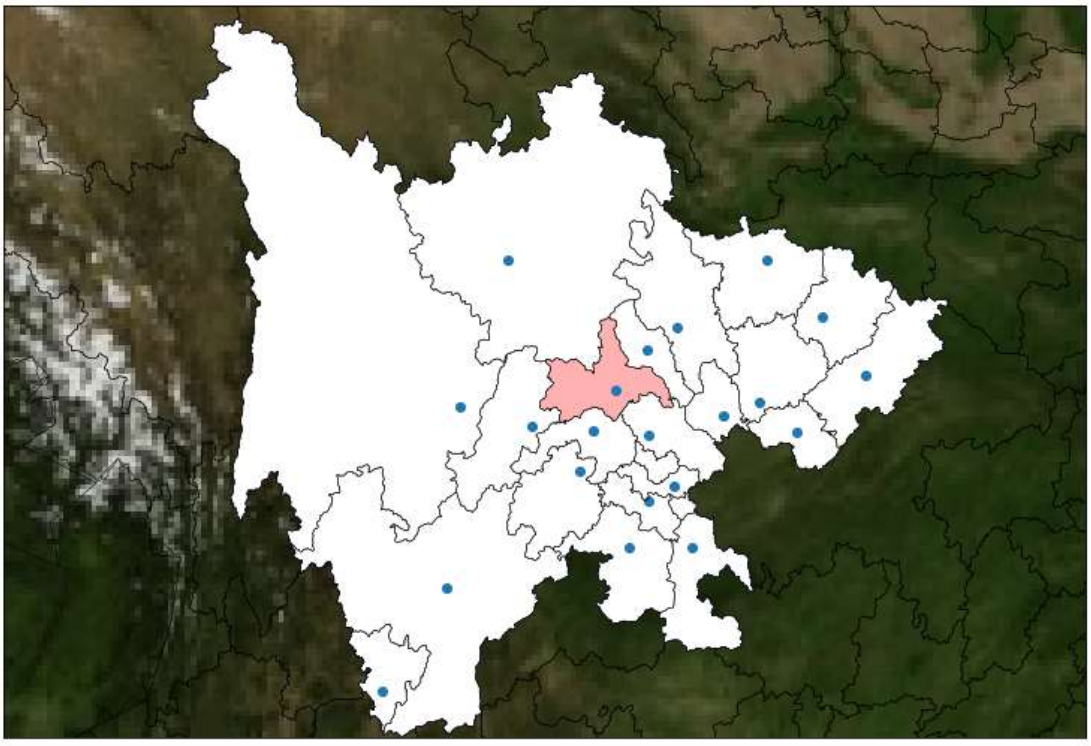
$$\max P=\mathbf{select}\cdot\mathbf{P}^T$$
$$s.t.\begin{cases}V=\mathbf{select}\cdot\mathbf{V}^T\\ \mathbf{select}_i\in\{0,1\}\end{cases} \quad \text{.....(5.2.5)}$$

求解这个模型，即可得到最大总存储价值的存储方案。

六、模型求解

6.1 求解最小距离和问题数据可视化

使用 python 之中的 numpy 库读取**附件 1** 之中的数据，并使用 Basemap 对于经纬度的数据进行可视化.得到医院位置的图如下所示.根据可视化以后的数据，



我们可以人工初步找到较为“中心”，也就是距离和较小的点.选取人工找到的中心点附近的某一邻域作为鱼群算法搜索的范围，可以加速函数的收敛.

图 6-1 四川省内城市医院位置图

6.2 基于人工鱼群优化医疗基地选址问题

6.2.1 人工鱼群算法的相关定义

人工鱼群算法是李晓磊等人于 2002 年提出的一类基于动物行为的群体智能优化算法. 该算法是通过模拟鱼类的觅食、聚群、追尾等行为在搜索域中进行寻优，是集群体智能思想的一个具体应用. 人工鱼群算法具有以下特点：具有克服局部极值、取得全局极值的较优秀的能力；算法中仅使用目标问题的函数值，对搜索空间有一定自适应能力；具有对初值与参数选择不敏感、鲁棒性强、简单易实现、收敛速度快和使用灵活等特点. 我们使用人工鱼群算法，来求解医疗物资存储的最佳地点.[3]

有一关于向量 $\mathbf{X}$ 的函数 $Y = f(\mathbf{X})$，希望找到函数 Y 的最大值.使用鱼群算法，不妨将每一条鱼的状态都用向量 $\mathbf{X} = (x_1, x_2, \cdots, x_n)$ 来进行表示，则欲寻优函数即可表示某一鱼个体所在的位置的食物浓度.

鱼群算法中所需要标记的变量如下表所示.

表 6-2 鱼群算法变量定义与声明

变量符号	变量声明
$fish_i$	标记第 i 条鱼
$Visual$	鱼感知范围的大小
$d_{ij} = \|x_i - x_j\|$	个体的鱼 x_i 与 x_j 之间的距离
try_num	尝试执行“寻找”的次数
δ	拥挤度因子
$step$	鱼移动的步长

6.2.2 鱼个体行为描述

如图 6-2 所示，在鱼个体的身上可以体现多种行为，如觅食行为、居群行为、追尾行为等.通过这四种行为，鱼群能够前往食物较为丰富的地方，并且能够形成群体来抵御外界的威胁；但同时，由于拥挤度因子 δ 的存在，鱼个体之间不会过于拥挤，能够防止食物不足等后果的发生.

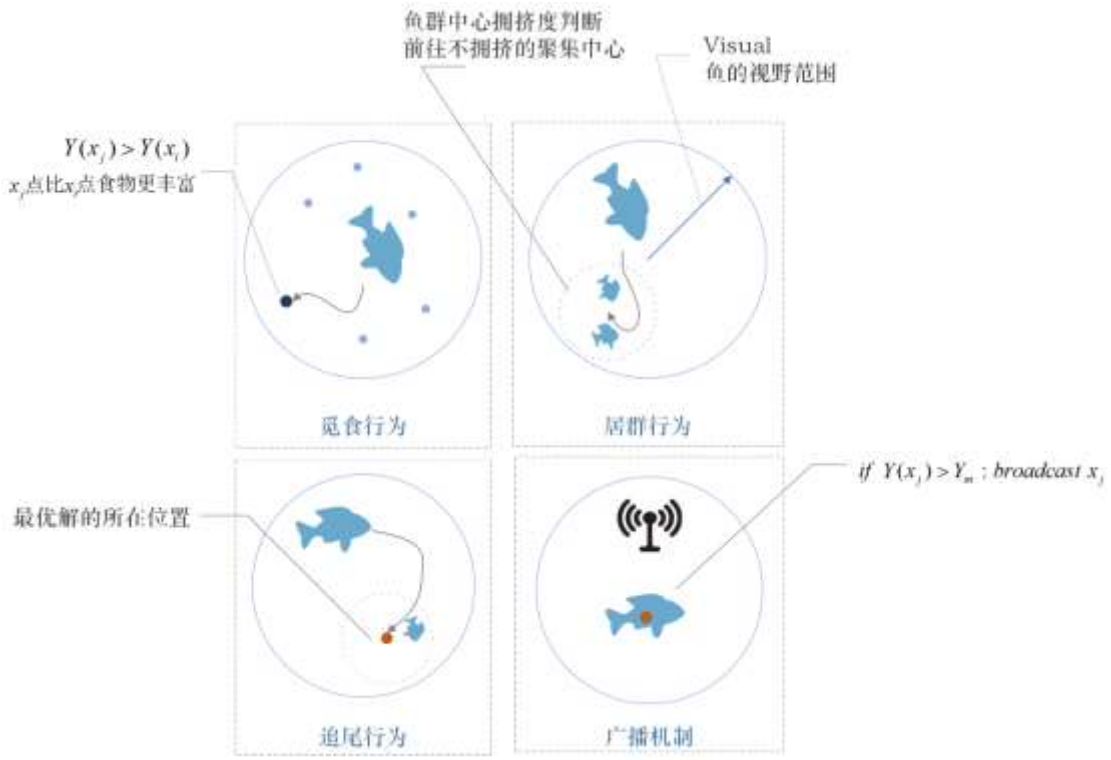


图 6-2 智能鱼群算法之中的鱼的几种模式

觅食行为.假设鱼当前的状态是 x_i，它会在所在点的邻域 $U(x_i, \text{Visual})$ 进行 try_num 次尝试，随机寻找点 $x_j \in U(x_i, \text{Visual})$，若满足 $Y(x_j) > Y(x_i)$，则证明点 x_j 处的食物更丰富.于是鱼就会向 x_j 的方向移动 *step* 距离.

聚群行为.假设在 $fish_i$ 的视野范围之中，有 n_f 条鱼存在. 那么，记探索中心的位置 $x_c : x_c = \sum_{i=1}^{n_f} \frac{x_i}{n_f}$.如果 $n_f / n < \delta$ 且满足 $Y(x_c) > Y(x_i)$,则证明在鱼群中心处的拥挤度满足条件且食物较为丰富.那么鱼群会向视野中的中心处移动 *step* 距离.

追尾行为. 鱼群的追尾行为表现为,探索其所在的邻域之内的最优解 $x_{\max}$,如果满足 $Y(x_{\max}) > Y(x_i)$ 以及 $n_f / n < \delta$;则鱼会向 $x_{\max}$ 处前进, 否则会执行觅食的行为.

广播机制. 通过这个机制来标记食物的最高浓度的位置. 如果鱼在经过比较之后, 发现自身所处位置的食物浓度高于”公告板”之中所标记的食物浓度, 就会用自身所处位置的信息 $(x_i, Y(x_i))$ 来代替原来的公告板之中的信息.

鱼群算法整体的寻优算法如下所示.

AFSA Algorithm:

Initialize: 种群数量 **N**, 鱼初始位置矩阵 **X**, 视野半径 ***Visual***, 步长 ***step***, 拥挤度因子 ***δ***, 鱼重复的次数 ***try_num***

CALC: 初始鱼群的适应值函数 $Y(x_i)$,

布告牌赋予最优的位置值 $mark \leftarrow \max(Y(x_i))$

For epoch in Iteration:

for each $fish_i$ *choose between {Pray(), Swarm(),Follow(),Bulletin }*

X. Update () 更新鱼群在 t+1 时刻所在的位置

$mark \leftarrow \max(Y(x_i))$ 更新公告板之中的最新数据

End For

编写基于线性运算库 Numpy 的 Python 程序，其主体框架如下图所示：

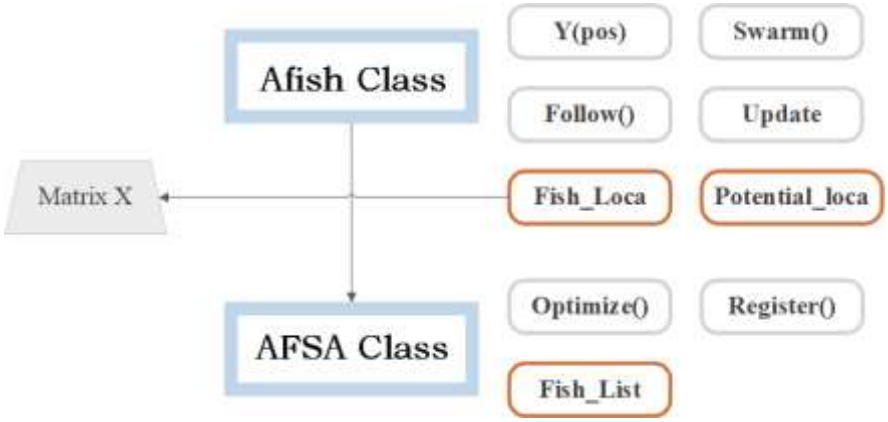


图 6-3 程序主体结构示意图

在矩阵 **X** 之中，登记有所有鱼群在某一个时刻的经纬度信息，同时，AFSA 对象可以视作为 Afish 对象的全体存在。

在“鱼”的类之中包含了以下的几个主要方法和属性。

Follow() : 寻找附近的最优解鱼，如果满足拥挤度要求则向其移动

Swarm() : 在附近寻找鱼，如果不拥挤则前往附近的鱼群中心

Pray(): 循环 try_num 次，随机寻找，看附近是否有更好的地点

Update(): 根据 Follow()和 Swarm()方法得到的潜在位置，决定下一个地点,进行更新

Potential_Loca : 登记 Follow()方法和 Swarm()方法得到的较优解

在“AFSA”类之中包含了下列的主要的方法：

Optimize() : 根据最大迭代次数 epoch,进行迭代

Register() : 将 fish_list 之中，所有“鱼”对象的最新位置更新至矩阵 **X** 之中

改变程序运行的不同次数 epoch,得到了鱼群分布图如下所示：

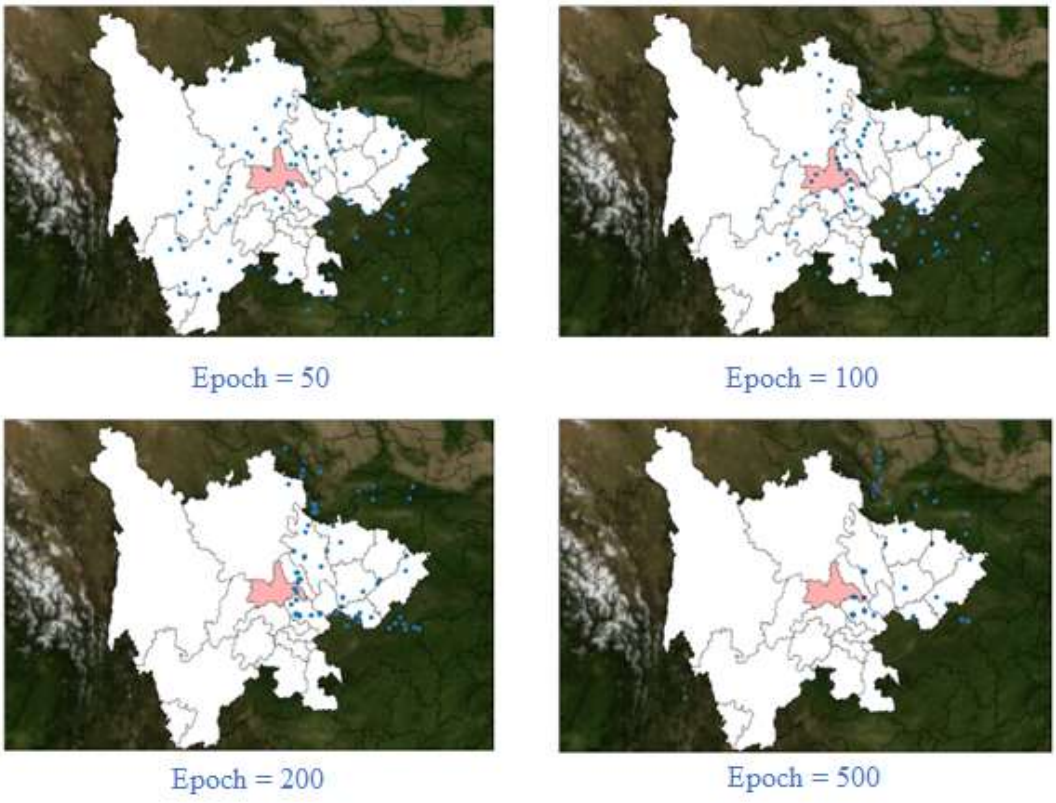


图 6-4 运行次数 epoch 对于鱼群分布的影响

同时，随着运行次数 epoch 的增长，距离函数 Y(x)会先增长然后之后趋于稳定。鱼群散点图之中，鱼群的分布也会收敛于几个点的邻域之内。位置目标函数随 epoch 的变化如下图所示。

最终定位得到的结果为：在 30.13141ºN, 104.624239ºE 处，飞行距离和达到最小值。从网上查阅资料可得，在四川省资阳市雁江区境内。

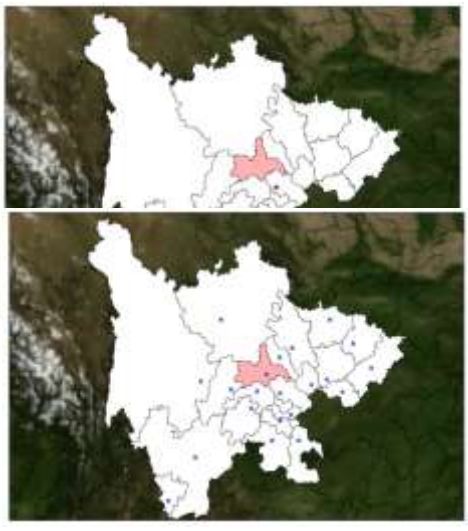


图 6-5 最终结果示意图

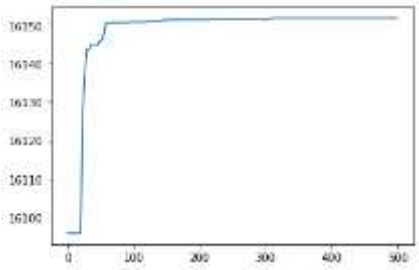


图 6-6 优化函数值随 epoch 变化

图 6-7 Google Map 定位最终结果

6.3 背包问题的求解

如图 6-8 ，Matlab 程序建立的思路如下图所示：

图 6-8 Lingo 编程思路

得到最终的结果为：最大的总价值为 4471。

被选中的医疗物资的索引在附件 1 之中给出。

七、最终结果的验证

在 Matlab 之中，使用传统的欧氏几何距离，求解两点之间的飞行距离。得到：最小距离的坐标为：104.52^0E, 30.1532^0N, 与鱼群算法优化的 Haversine 距离较为接近。其绘出的结果如图 7-1 所示：

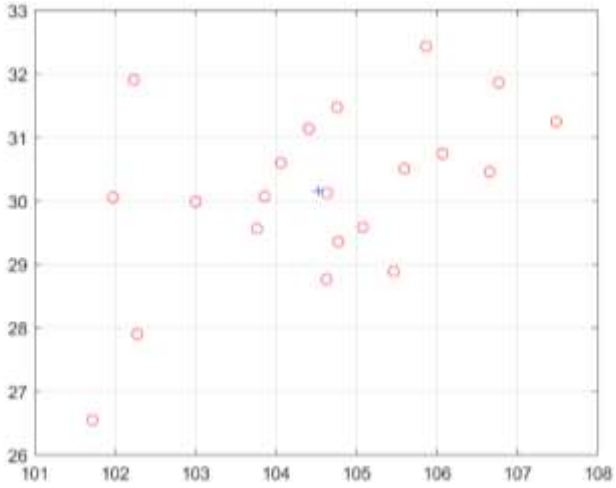


图 7-1 Matlab 结果验证

可以认为之前得到的结果具有较高的可靠性。

八、模型的评价

使用鱼群算法，具有的好处是：

具有较快的收敛速度和计算精度；

较好的抗收敛能力；

在后续的模型之中，可以对于鱼群的位置进行限制，从而得到更多范围之内的解。

九、参考文献

[1] [1]王功琪. 球面上两点间距离的求法[J]. 河南科技, 2013(05):189-190.
[2] [1]樊东卫, 何勃亮, 李长华, 韩军, 许允飞, 崔辰州. 球面距离计算方法及精度比较[J]. 天文研究与技术, 2019, 16(01):69-76. DOI:10.14005/j.cnki.issn1672-7673.20180523.001.
[3] [1]杨鹏. 人工鱼群算法的应用研究[D]. 西北师范大学, 2017.

plot_boundary.py 实现数据的可视化

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun  4 22:46:31 2022
@author: Bealliant
"""
import numpy as np
hospitals = np.loadtxt("cities.txt")
def plot_scatters(x,*args):

    '''

    input: -> numpy matrix, [[Longitude,Latitude], ...]

    function plot_boundary will draw the scatter plot on the map.

    '''
    import warnings

    from mpl_toolkits.basemap import Basemap
    import matplotlib.pyplot as plt
    from matplotlib.patches import Polygon


    warnings.filterwarnings('ignore')


    fig = plt.figure(figsize=(16, 9))

    ax = plt.gca()   # 获取当前绘图区的坐标系
    bmap = Basemap(llcrnrlon=97,llcrnrlat=25,urcrnrlon=110,urcrnrlat=36,
                projection='lcc',lat_1=33,lat_2=45,lon_0=120)

    # bmap.etopo()
    bmap.bluemarble()

    # 画省
    shp_info =
bmap.readshapefile('Basemap_Package\gadm36_CHN_1','states',drawbounds=False)


    for info, shp in zip(bmap.states_info, bmap.states):
        proid = info['NAME_1'] # NAME_1 代表各省的拼音

        if 'Sichuan' in proid:
            poly = Polygon(shp,facecolor='w',edgecolor='b', lw=0.2)
            ax.add_patch(poly)
    # 画市
    bmap.readshapefile(shapefile='Basemap_Package\gadm36_CHN_2',
                   name='citys',
                   drawbounds=True) # 市

    for info, shp in zip(bmap.citys_info, bmap.citys):
```

```
        proid = info['NAME_2']  # NAME_2 代表各市的拼音


        if "Chengdu" in proid:
            poly = Polygon(shp,
                           facecolor='r',alpha=0.3,
                           lw=3)
            ax.add_patch(poly)

    x1 =
    y1 = []
    x=[]
    y=[]

    try:
        for k,v in x:
            x1.append(k)
            y1.append(v)
    except TypeError:
        k,v = x
        x1.append(k)
        y1.append(v)

    longi1,lati1=bmap(x1,y1)
    bmap.scatter(longi1,lati1,marker='o',facecolor=
args)#c=range(len(wechatPos)),cmap=plt.cm.viridis

    for k,v in hospitals:
        x.append(k)
        y.append(v)

    longi,lati = bmap(x,y)
    bmap.scatter(longi,lati,marker='x',facecolor=
'b')#c=range(len(wechatPos)),cmap=plt.cm.viridis
    bmap.drawcoastlines()
    bmap.drawcountries()

    # plt.savefig('fig_province.png', dpi=100, bbox_inches='tight')
    plt.show()


x = np.array([[104.61379529 , 30.12607128]])
plot_scatters(x,'r')
```

AFSA.py 鱼群算法的实现

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun  5 20:55:30 2022
@author: Bealliant
"""
import numpy as np
import numpy.random as rd
import sys
from plot_boundary import plot_scatters
import math
hospitals = np.loadtxt("cities.txt")
R = 6371
class AFish():

    def __init__(self, fish_num=100, try_num = 5):
        '''
            Initialization:
```

```
            *    wdir : load the .txt file to get the positions of the cities
            *    center : Allow Users to Designate the initial point of searching

             We can know that the range of both the longitude and the magnitude is
            Approximately 6. The whole area is devided into 10K parts (100*100).
            So the scale of one grid is 0.08 * 0.08.
            Let's say one fish can move half size of one grid. And can see twice
            the size of a grid.

        *    visual_distance = 10km
        *    visual = 0.10 degree
        *
        *    step = 5

         '''

         self.longi = rd.random([1])*8 + 100
         self.magni = rd.random([1])*8 + 26
         self.fish_loca = np.concatenate((self.longi,self.magni))

         self.potential_loca_swarm = self.fish_loca
         self.potential_loca_follow = self.fish_loca
         self.potential_loca = self.fish_loca

         self.fish_num = fish_num

         self.step = 0.04
         self.visual_dist = 10        #判断距离的 visual 是采用已有的经纬度定位的方法
         self.visual = 0.30
         self.theta = 0.8
         self.try_num = try_num

         self.bulletin = np.zeros((1,2))

    def hav(self,theta):
         '''
             Haversine Formula
             $hav(\theta) = sin{^2}(\frac{\theta}{2})$
         '''
         theta = theta  * np.pi/180
         return np.sin(theta/2)**2

    def cos(self,theta):
         # input : theta -> degrees
         # output : cos(theta)
         return np.cos(theta*np.pi/180)

    def sin(self,theta):
         # input : theta -> degrees
         # output : sin(theta)
         return np.sin(theta*np.pi /180)


    def distance_Calc(self,center,cities = hospitals):
         '''
         d = 2r*asin(hav(lamda1-lambda2)+cos(lamda1)cos(lamda2)hav(\delta \phai))
         Attention : 为了方便鱼群算法的执行，将距离的变量变成了负号，追求“越大越好”
         '''
         try:
             return 2*R *np.arcsin(np.sqrt(self.hav(cities[:,1]-center[1]) +\
                                     self.cos(center[1])*self.cos(cities[:,1])*self
.hav(cities[:,0]-center[0])))
         except IndexError:
             return 2*R *np.arcsin(np.sqrt(self.hav(cities[:,1]-center[0,1]) +\
                                     self.cos(center[0,1])*self.cos(cities[:,1])*se
lf.hav(cities[:,0]-center[0,0])))

#        return self.R *2*np.arcsin(self.diff_hav(self.cities[:,:1], center) +\
```

```
    def Y(self,center):
        '''
            The Object Function.
            Return the Tolerance Score\Food Abundance of the point named CENTER
        '''
        return self.distance_Calc(center).sum()*-1 + 20000

    def Pray(self):
        '随机寻找目标函数较大的解'
        for _ in range(self.try_num):
            pin = self.fish_loca + rd.random([1,2]) * self.visual
            if self.Y(pin) > self.Y(self.fish_loca):
                self.potential_loca = self.fish_loca + self.step * (pin -
self.fish_loca) / self.visual
                self.fish_loca = self.potential_loca
                break

    def update(self):
        '''
            To Execute the comparison between potential coordinates and Updating.
            将这个鱼的 potential 坐标更新到汇总的矩阵 X 之中'
        '''
        self.Swarm()
        self.Follow()

        if self.Y(self.potential_loca_follow) > self.Y(self.fish_loca) and\
            self.Y(self.potential_loca_follow) >
self.Y(self.potential_loca_swarm):
                self.fish_loca = self.potential_loca_follow

        elif self.Y(self.potential_loca_swarm) > self.Y(self.fish_loca):
            self.fish_loca = self.potential_loca_swarm

        else:
            self.Pray()

        self.potential_loca = self.fish_loca
        self.potential_loca_follow = self.fish_loca
        self.potential_loca_swarm = self.fish_loca


    def find(self, position):
        '''
            return a numpy array that find the surrounding points of A within a
range of self.visual_dist
            The original Point A is EXCLUDED.
        '''

        dist = self.distance_Calc(center = position, cities = X)
        #nearby_list = np.where(dist<=self.visual_dist and dist > 0.1)
        #return nearby_list

        return_list = []

        for i,ele in enumerate(dist):
            if ele <self.visual_dist and ele > 0.001 :
                return_list.append(i)

        return return_list

    def Swarm(self):
        '执行这个鱼的个体的聚群行为'
        nearby_list = self.find(self.fish_loca)
        if len(nearby_list) == 0:
            self.potential_loca_swarm = self.fish_loca
            return

        center = np.mean(X[nearby_list], axis = 0)
        if len(nearby_list)/self.fish_num <= self.theta:
```

```
            # 拥挤度满足要求
            if self.Y(center)>=self.Y(self.potential_loca) :
                delta_Coordi = center - self.fish_loca
                delta_Coordi = delta_Coordi /np.linalg.norm(delta_Coordi)
                self.potential_loca_swarm = self.potential_loca + self.step *
delta_Coordi

    def Follow(self):
        '执行这个鱼的个体的追尾行为'
        nearby_list = self.find(self.fish_loca)
        nearby_score_list = []
        try:
            for i,fish in enumerate(nearby_list):
                nearby_score_list.append(self.Y(X[fish]))

            nearby_max_score = max(nearby_score_list)
            nearby_max_score_num = nearby_score_list.index(nearby_max_score)

            nearby_max_score_index = nearby_list[nearby_max_score_num]

            '''
             From the Point Xi we find the Xj in its nearby region
             when Y reach its high.
            Again we should use find to Investigate
            if the Congestion Index of the Point Xj satisfies
           the request.
            '''

            nearby_list_j = self.find(X[nearby_max_score_index,:])

            if self.Y(X[nearby_max_score_index,:]) > self.Y(self.fish_loca) and\
            len(nearby_list_j)/self.fish_num < self.theta :
                    delta_Coordi = X[nearby_max_score_index,:] - self.fish_loca
                    delta_Coordi = delta_Coordi/np.linalg.norm(delta_Coordi)
                    self.potential_loca_follow = delta_Coordi * self.step

        except ValueError:
            self.potential_loca_follow = self.fish_loca


    def Bulletin(self):
        pass


X = np.empty([500,2])
'''
    The 500*2 Matrix X stores the positional coordinates of all fish identities.
'''

class AFSA(AFish):
    '''
                Hypder-Parameter Settings:

                *    epoch : Times of Iteration, Defaulat is 1000
                *    fish_num : number of fishes in an ant group.
                *    fish_loca : randomized locations of each fish in the group.
    '''
    def __init__(self, fish_num = 100, epoch = 100):
        self.epoch = epoch
        self.fish_num = fish_num
        self.fish_list = []
        self.maxpos = np.empty([1,2])
        self.maxscore = -5000

        for _ in range(fish_num):
            temp_fish = AFish()
            self.fish_list.append(temp_fish)
```

```
    def register(self):
        for i,temp_fish in enumerate(self.fish_list):
            if self.Y(temp_fish.fish_loca) > self.maxscore :
                self.maxscore = self.Y(temp_fish.fish_loca)
                self.maxpos = temp_fish.fish_loca

            X[i,:] = temp_fish.fish_loca
    def optimize(self):
        list_m = []
        self.register()
        for i in range(self.epoch):
            print(i,'    ',self.maxpos)
            for afish in self.fish_list:
                afish.update()
            self.register()
            list_m.append(self.maxscore)

        return list_m


k = AFSA()
list_out = k.optimize()
plot_scatters(hospitals,'r')
plot_scatters(k.maxpos,'b')
```

Distance.m 实现医疗基地到城市之间距离和的计算

```
function y = Distance(p,a)
    X = load("cities.txt");
    X = X - p;
    y = sum(sum(X.^2));
End
```

main.m 实现最小距离点的求解以及地图的可视化

```
X = load("cities.txt");
pos = fmincon('Distance',[104,30],[],[]);
plot(X(:,1),X(:,2),'ro');
grid on;
hold on;
plot(pos(1),pos(2),'b+');
```

int_lin_prog.m 实现背包问题的求解

```
x = load("prob2.txt");
price = x(:,2);
volume = x(:,1);

select = intlinprog(-1 *
price,[1:100],volume',3000,[],[],zeros(100,1),ones(100,1))
```