# 2019 数维杯大学生数学建模竞赛论文

论文题目：火灾危害等级评估与救灾人员分配

队伍编号：20199138

# 火灾危害等级评估与救灾人员分配


## 摘　要

火的使用推动了人类的进步，然而一旦失控则将演化为火灾。火灾可能发生在任何地方，城市、森林、草原火灾都会对地区经济造成很大的影响，甚至威胁到人类的生命安全，良好的火灾等级评价方案及消防人员分配方案将大大降低火灾带来的危害。

对于问题一，针对不同火灾发生特点的多样性，本文首先对数据进行预处理，并通过火灾发生的经纬度获取其卫星地图及火灾所属行政区域信息。通过判断火灾发生位置的地形情况，将火灾分为农作物火灾、森林火灾、草原火灾及城市火灾四类。针对不同火灾的类型特点建立对应的评价模型，植被燃烧考虑横向扩散，城市建筑燃烧考虑纵向扩散，并采用回归拟合方法确定燃烧扩散时间与风力及救援难度的关系，从而建立火灾危害等级评价模型。最后，本文将问题给出的数据代入模型进行计算，评定火灾的危害等级，并筛选出 10 大火灾。

对于问题二，为了合理筛选出火灾重点防护位置，针对火灾的危害等级、频度及所属行政区域位置因素，本文按照地级市划分出 311 个火灾所属行政区域，对行政区域统计火灾危害及频度信息，在两个维度上使用 k-means 聚类，最终在 10 个结果类簇中选取危害等级及频度相对较高的 2 个类簇共 21 个行政区域作为重点防火位置。

对于问题三，为了合理安排火灾消防人员，针对问题二中重点防火位置的地理分布特点及火灾危害等级的影响，考虑到地区间消防人员的互相调配情况，本文建立多连通分支无向图模型用以表征重点防火区域间的关联关系，通过对经纬度信息聚类将重点防火位置映射到 3 个连通分支中，并对每个连通分支建立连通关系。使用距离表征连通分支中边的权值，并基于此建立救援调度模型。将危害等级与消防人员的分配关系定义为救援配比度，将优化救援配比度方差作为求解目标，使用蚁群算法求解得到 21 个行政区域消防人员数量的最优分配。



**关键词：火灾危害评估 救灾人员分配 聚类 连通图 蚁群算法**

# 一、问题的提出

## 1.1 问题的背景

随着人类经济社会的快速发展，全球气候受到人类社会活动的影响而频发极端气候现象，各类自然灾害的等级均有一定的上升趋势。其中，火灾作为最常见的灾害之一，常常由于气候环境及人为因素发生，往往对人类社会生产生活造成重大的财产及经济损失，甚至导致无辜人员的伤亡。因此，对于火灾而言，科学地评价火灾的危害等级并参考危害等级合理组织、安排、调度消防人员对人类生产生活有着重大价值及意义。

## 1.2 问题重述

众所周知“水火无情”，没有科学合理的火灾救援策略不仅会增加无辜人员伤亡，同时也会带来重大经济损失。2019 年清明节前后因北方大部分地区具有 较大等级的风，春季气候普遍干燥，使得众多地区发生了一定规模的火灾。2018 年美国的加州大火所造成的经济损失相当于烧掉了整个洛杉矶。为了合理量化评价火灾的危害等级，并参考危害等级安排救援策略，需要建立数学模型来解决这个问题。

一、选取合适的评价指标，对已知的 2033 个火灾预期危害等级建立评价模型，并筛选出 10 大火灾编号 。

二、结合问题一中所确定的火灾危害等级评价模型及具体放火位置，筛选出重点的放火位置作为预选的重点防护单位。

三、根据问题一及问题二得到的结果，将假设的 100000 名消防人员合理的分配到问题二中的重点位置中。

# 二、问题的分析

## 2.1 概论

这是一个评价规划问题，根据火灾初始面积、风力大小、预期蔓延速度、救援难度等已知指标及从经纬度挖掘得到的区域及火灾类型等隐藏指标研究火灾造成的危害及损失，判断已有记录的火灾中哪些是对人类经济社会造成重大损失的火灾。同时，需要根据评价得到的结果合理安排固定数量的消防员，这需要考虑不同位置区域的相关联程度等因素。问题的特点在于指标的可挖掘信息丰富；问题的难点在于选择合理的指标量化方法，对于火灾危害等级建立普适的评价及消防人员规划模型。

## 2.2 问题一

问题一要求对已知的 2033 个火灾进行危害等级评价，这需要从问题已给的指标中挖掘对评价具有一定价值的指标，建立合适的评价模型。由于问题已经给了一定量的数据，本文通过三个步骤解决该问题。首先对数据预处理筛除定位结果为空值的数据与经纬度定位在国外的数据；其次根据经纬度通过 Python 爬虫程序将 Google 地图[1]上的卫星地图块收集到本地，并根据收集到的地图块判断火灾发生的地形信息，从而对已给的火灾数据进行分类；最后对于不同的分类采用数学方法建立适当的评价模型，统一每个类型的评价系数使其整体具有普适性。

## 2.3 问题二

问题二要求结合问题一求得的每个火灾危害等级及其发生的位置信息，筛选出重点防火位置。

本文结合火灾的防护需要考虑到该地区火灾的频发程度及火灾的危害程度两个维度，对第一问求得的 311 个二级行政区域统计火灾发生数量及火灾危害等级，在两个维度上归一化后进行聚类，得到一系列的类簇用以综合表征该地区火灾防护的需求等级。

### 2.4 问题三

问题三要求将固定量的消防员合理分配到问题二筛选出的重点位置中。由于已经有了问题一的评价度量，同时能够从问题二的结果中获得这些位置的地理信息。本文对这些重点位置建立连通图模型，同时采用聚类方法确定连通分支，考虑在同一连通分支内的重点位置能够互相调配消防员参与救援。通过衡量火灾损失与消防员人数匹配程度求得最优的分配方案。

## 三、模型假设

1.假设相同评价体系下火灾的发生不受季节影响。
2.假设火灾预计蔓延速度不受地形因素影响。
3.假设初始着火面积为正圆形，并在各个方向上的扩散速度相同。

## 四、符号说明

| 符号 | 符号说明 |
|---|---|
| $l$ | 预计经济损失 |
| s | 燃烧总面积 |
| $s_0$ | 初始燃烧面积 |
| c | 单位面积价值 |
| $r$ | 燃烧半径 |
| $r_0$ | 初始燃烧半径 |
| $r_{ex}$ | 燃烧扩散半径 |
| $v$ | 预期蔓延速度 |
| t | 蔓延时间 |
| $h_{ex}$ | 蔓延高度 |
| $h_s$ | 楼层层高 |
| $a_h$ | 救援难度系数 |
| $p_w$ | 风力 |
| $c_{b_i}$ | 第 i 市建筑物价值 |
| $c_m$ | 平均原料价值 |
| $GDP_i$ | 第 i 市 GDP |
| $nor_{GDP_i}$ | 第 i 市归一化 GDP |
| $nor_{n_i}$ | 第 i 个二级行政区域归一化火灾发生次数 |
| $nor_{l_i}$ | 第 i 个二级行政区域归一化预计经济损失 |
| $n_{po}$ | 常驻消防员数量 |
| $n_{pd}$ | 可调配消防员数量 |
| $pr_h$ | 救援配比度 |

# 五、模型的建立与求解

## 5.1 问题一的模型建立与求解

### 5.1.1 问题一的分析

问题一需要对已知的火灾进行危害等级评价。为了建立合理、普适的火灾危害等级评价模型，本文首先对已知数据进行预处理，将不合理数据筛除；然后根据火灾发生的经纬度对火灾所属的行政区域进行划分，同时通过计算机程序处理的方法获取火灾地点卫星成像特征图，将火灾划分为 4 个主要类型；最后使用数学方法对这 4 个主要类型建立评价模型，并将已知数据代入模型计算筛选出 10 大火灾。问题求解流程如下图：

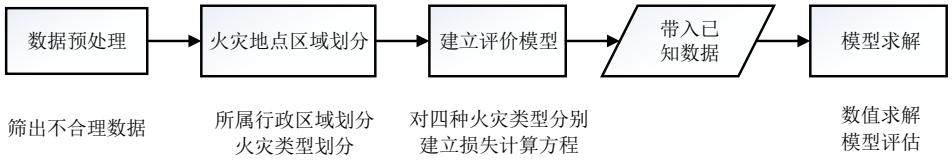


图 1 问题一求解流程图

### 5.1.2 问题一模型的建立

#### Step1 数据预处理

为了方便后续步骤的进行，提高模型精度，本文首先根据经纬度定位，通过使用 GeoPy[2]调用百度地图 API 对数据进行预处理（程序见附录一），筛除的数据满足如下规则：

1）通过经纬度无法查询到位置所在行政区域信息；

2）通过经纬度查询到的位置在中国边界范围之外。

经过预处理，删除数据 50 项，最终保留数据 1983 项。具体结果参见支撑材料：火灾数据.xlsx。

#### Step2 数据划分

为了方便问题二对火灾重点位置的筛选及问题三对 100000 名消防人员的划分，本文使用 GeoPy 查询经纬度信息获得火灾发生地点所属的行政区域，统计二级行政区域共 311 个。具体结果参见支撑材料：火灾数据.xlsx。

同时，为了便于建立合理模型，本文使用 Python 编写爬虫程序（程序见附录二）将 Google 地图卫星成像地图块爬取到本地，然后对地形地貌进行区分。主要的火灾点地形特征示例图如下：

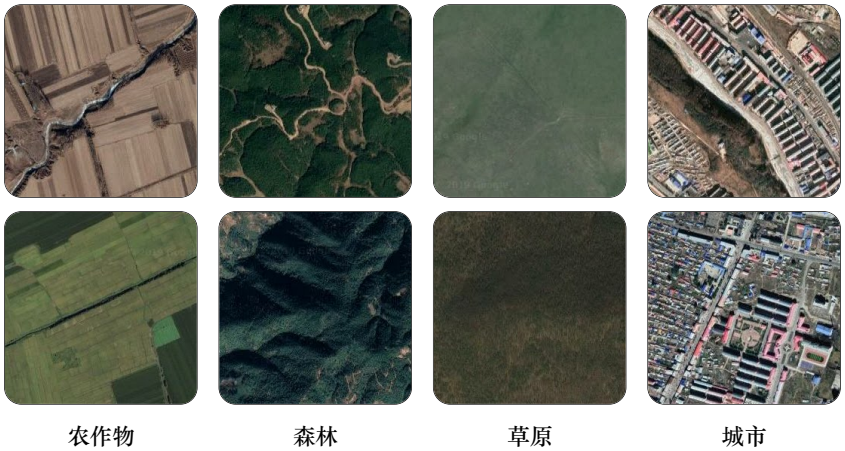


图 2 火灾点地形卫星成像图

通过卫星成像图，参考资料[3]中的分类，将预处理之后的火灾数据分为农作物火灾、森林火灾、草原火灾及城市火灾四种主要类型，并对数据进行标注。不同类型的数据量统计如下图：

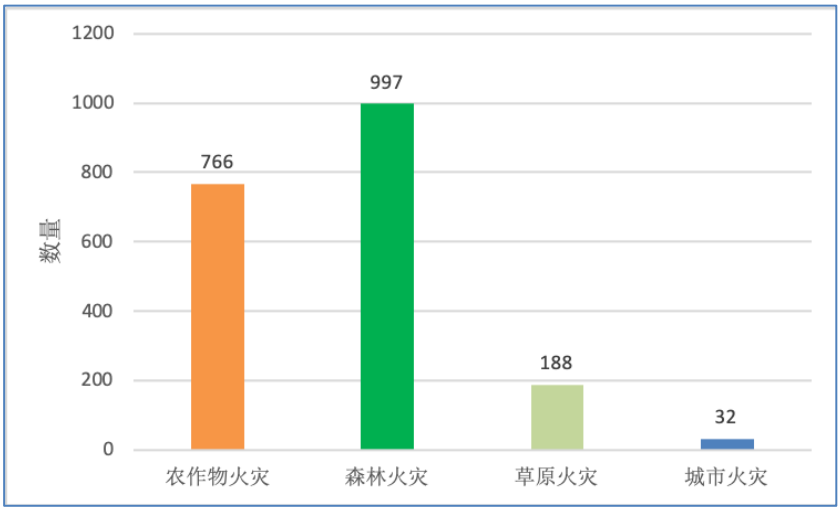


图 3 火灾类型统计图

## Step3 模型建立

通过 Step2 中划分出的农作物火灾、森林火灾、草原火灾及城市火灾的分类，本文对不同的类别分别建立模型。根据资料[4]，火灾事故常常根据直接财产损失或人员伤亡来定级与划分，见下表：

表 1 火灾等级判定

| 火灾等级 | 经济损失 | 伤亡人数 |
|---|---|---|
| 特别重大火灾 | 1 亿元以上 | 30 人以上 |
| 重大火灾 | 5000 万元以上 1 亿元以下 | 10 人以上 30 人以下 |
| 较大火灾 | 1000 万元以上 5000 万元以下 | 3 人以上 10 人以下 |
| 轻微火灾 | 1000 万元以下 | 3 人以下 |

由于已给的火灾数据条目中无法直接查证人员伤亡记录，且定级中财产损失与人员伤亡呈或相关，于是模型只考虑财产损失，并以财产损失作为危害等级评判标准。

考虑火灾发生后可燃物单位面积燃烧后的经济损失，其方程如下：

$$l = \mathrm{s} \times \mathrm{c} \tag{5.1}$$

其中单位面积的价值在 5.1.3 小节中求解得出，这里着重对燃烧总面积建立模型。

对于农作物火灾、森林火灾及草原火灾这三类火灾，由于可燃物类型均为植被，其火势蔓延的方式大致相同，由模型假设得到这三类火灾可燃物燃烧示意图如下：

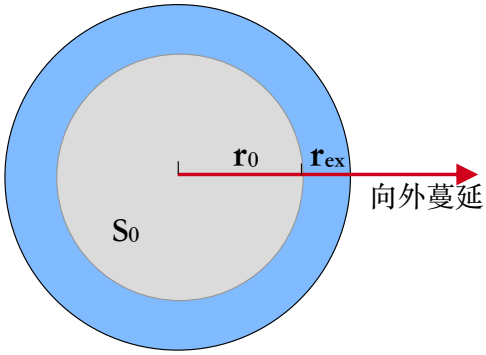


图 4 植被可燃物燃烧示意模型

因此燃烧总面积计算方程如下：

$$s = \pi \times r^2 \tag{5.2}$$

其中，燃烧半径可拆分为：

$$r = r_0 + r_{ex} \tag{5.3}$$

$$r_0 = \sqrt{\frac{s_0}{\pi}} \tag{5.4}$$

蔓延半径作为一维方向上的距离，表示为：

$$r_{ex} = v \times t \tag{5.5}$$

蔓延时间与救援难度系数及风力相关，表示为：

$$t = f_p(a_h, p_w) \tag{5.6}$$

综上，对于农作物火灾、森林火灾及草原火灾三类可燃物为植被的火灾，其损失计算方程为：

$$l = \pi \times \left(\sqrt{\frac{s_0}{\pi}} + v \times f_p(a_h, p_w)\right) \times c \tag{5.7}$$

另，由于城市火灾常常是厂房、仓库、居民楼等的建筑物类型的火灾，考虑其典型性影响与模型的简化，只考虑高度方向上的蔓延及影响，其燃烧示意图如下：

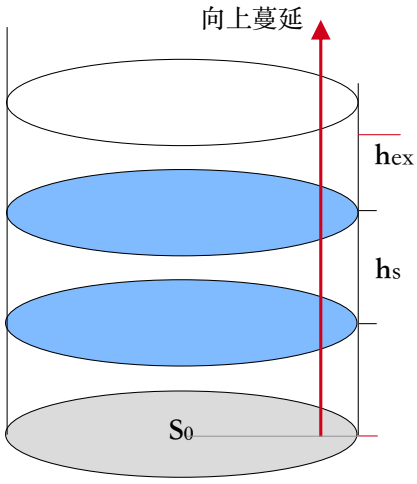


图 5 建筑物燃烧示意模型

对于城市火灾分类下的评估模型，考虑建筑物每层的面积相同，燃烧总面积计算方程如下：

$$s = s_0 \times \left\lceil \frac{h_{ex}}{h_s} \right\rceil \tag{5.8}$$

其中蔓延高度同植被燃烧的蔓延半径，为一维方向上的距离，表示为：

$$h_{ex} = v \times t \tag{5.9}$$

同（5.6）蔓延时间与救援难度及风力相关，表示为：

$$t = f_b(a_h, p_w) \tag{5.10}$$

综上，对于城市火灾一类建筑物类型的火灾，其损失计算方程为：

$$l = s_0 \times \left\lceil \frac{v \times f_b(a_h, p_w)}{h_s} \right\rceil \times c \tag{5.11}$$

### 5.1.3 问题一模型的求解

根据上一小节 Step3 分类建立的模型，需要求解蔓延时间函数及农作物、森林、草原及城市建筑物四种燃烧影响物的单位面积价值。确定评价模型之后，需要对问题给出的数据进行评估，筛选出 10 大火灾。本文采用回归拟合方式得到蔓延时间方程，并查阅资料确定了不同火灾分类下影响的单位面积价值，最终确定危害等级评价模型。

#### Step1 蔓延时间求解

火灾蔓延时间影响了火灾向外扩散的面积与过火面积，为定量分析火灾最终的过火面积，需要确定火灾的蔓延时间。火灾的蔓延时间与风力成正相关，风力越大，火灾燃烧时的空气越充足，热量扩散越快，使火势快速向周围扩散。根据实际情况，当火灾发生时，相关部门会根据火灾大小及时派遣救援部队到场救援，救援部队的水扑灭、建立隔离带等行为一定程度抑制火灾的蔓延，但是

受实际救援环境影响，火灾的救援难度存在不同的等级，当火灾的救援难度较小时，救援人员能够比较容易的抑制火势的增长，减少火灾的蔓延时间；随着火灾救援难度的增长，救援人员对火灾扩散的抑制作用不断减小，当救援难度过大时，救援人员对火灾扩散的抑制作用相对较小，火灾持续向周边可燃物蔓延，相应的火灾蔓延时间较长。结合文献及实际情况，本文通过对火灾的风力及救援难度进行二次曲线型的建模。

为了便于统一风力及火灾救援难度系数对火灾蔓延时间的影响，本文对风力值及救援难度系数进行了归一化处理。最终火灾的蔓延时间可以表示为：

$$t = \alpha \times {p_w}^2 + \beta \times {a_h}^2 + \gamma \tag{5.12}$$

其中的系数通过对实际数据进行拟合确定。拟合结果如下所示：

表 2 火灾蔓延时间系数取值

| 系数 | $\alpha$ | $\beta$ | $\gamma$ |
|---|---|---|---|
| 取值 | 0.128 | 0.131 | 0.243 |

为定量评定火灾的过火面积，需要对火灾蔓延速度进行定量的评定，由于火灾蔓延速度受风力、风速、地形坡度、温度、可燃物类型等因素的影响，火灾蔓延速度较慢时，火灾单位时间向周围蔓延的距离较短，火灾的蔓延速度越快，火灾单位时间向周围蔓延的距离越长。本文对火灾蔓延速度的五个等级进行二次曲线型的建模，结合实际情况，得出火灾蔓延速度的取值如下：

表 3 火灾蔓延速度取值

| 预期蔓延速度 | 蔓延速度取值/(m • $min^{-1}$ ) |
|---|---|
| 缓慢 | 200 |
| 慢 | 400 |
| 适中 | 800 |
| 快 | 1000 |
| 特快 | 1500 |

### Step2 确定单位面积价值

根据文献[5]，农作物价值包括农产品生产价值、社会保障价值、气体调节价值、水土保持功能价值、环境净化价值，同时还具有水资源消耗、化肥农药等带来的环境污染等负向价值，总结来说农作物提供的价值为 1.89606 元/$m^2$。

根据文献[6]，森林价值包括林木价值、林下产品价值、气候调节价值、水源涵养价值、土壤保持价值、空气净化价值等，文章对不同省份地区分类，得到森林提供的价值如下表：

表 4 各省森林价值表

| 省份 | 广西 | 广东 | 云南 | 江西 | 福建 | 四川 | 湖南 | 浙江 | 贵州 | 湖北 |
|---|---|---|---|---|---|---|---|---|---|---|
| 单位面积价值（$10^4$ 元/$hm^2$） | 766.7 | 888.0 | 412.6 | 741.0 | 862.1 | 389.9 | 597.9 | 915.9 | 462.0 | 469.8 |

| 省份 | 内蒙古 | 黑龙江 | 陕西 | 安徽 | 西藏 | 重庆 | 辽宁 | 吉林 | 河北 | 河南 |
|---|---|---|---|---|---|---|---|---|---|---|
| 单位面积价值（$10^4$元/hm$^2$） | 140.1 | 156.6 | 324.1 | 651.0 | 207.5 | 753.2 | 400.3 | 207.1 | 384.4 | 307.6 |

| 省份 | 海南 | 山东 | 山西 | 新疆 | 甘肃 | 江苏 | 北京 | 青海 | 宁夏 | 上海 | 天津 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 单位面积价值（$10^4$元/hm$^2$） | 954.1 | 468.7 | 283.8 | 295.5 | 141.3 | 213.3 | 429.5 | 116.5 | 114.2 | 328.9 | 183.1 |

由于已经划分好省份地区，本文也已在上一小节 Step2 中划分好行政区域，在这里不对全国森林价值求解平均值，模型最终将按照省级行政区计算不同森林火灾对应的森林价值。

根据文献[7]，草原价值包括食物和原材料生产价值、旅游价值、有机物生产价值、气体调节价值、水土保持价值、水源涵养价值、生物多样性价值、生物控制价值，总结来说草原提供的价值为15.84595 元/m$^2$。

根据文献[8]，城市建筑物价值主要由其房地产成本价值决定，并且与该城市的人均 GDP 相关。建筑房地产成本主要由钢筋、砼、门窗、承包商管理费等费用组成，其建筑房地产成本平均为 1914 元/m$^2$。本文通过数据库[9]获得城市火灾所在城市的人均 GDP，为合理刻画城市人均 GDP 对城市建筑价值的影响，对城市人均 GDP 进行归一化处理。最终的城市建筑物价值可以由以下方程得出：

$$c_{b_i} = c_m \times (0.5 + nor_{GDP_i}) \tag{5.13}$$

$$nor_{GDP_i} = \frac{GDP_i}{\max(GDP_0, GDP_1, \ldots, GDP_m)} \tag{5.14}$$

综上，得到燃烧物为植被的农作物火灾、森林火灾及草原火灾评价模型为：

$$l_p = \pi \times \left(\sqrt{\frac{s_0}{\pi}} + v \times 0.128{p_w}^2 + 0.131{a_h}^2 + 0.243\right) \times c \tag{5.14}$$

城市火灾评价模型为：

$$l_c = s_0 \times \left\lceil \frac{v \times 0.128{p_w}^2 + 0.131{a_h}^2 + 0.243}{h_s} \right\rceil \times c_m(0.5 + nor_{GDP_i}) \tag{5.15}$$

### Step3 火灾评估

通过以上的数据处理及模型建立与求解，编写 python 程序（见附录三）将每个火灾数据条目代入模型计算，得到所有已知火灾条目的危害等级，按照经纬度映射到全国各地区如下图：

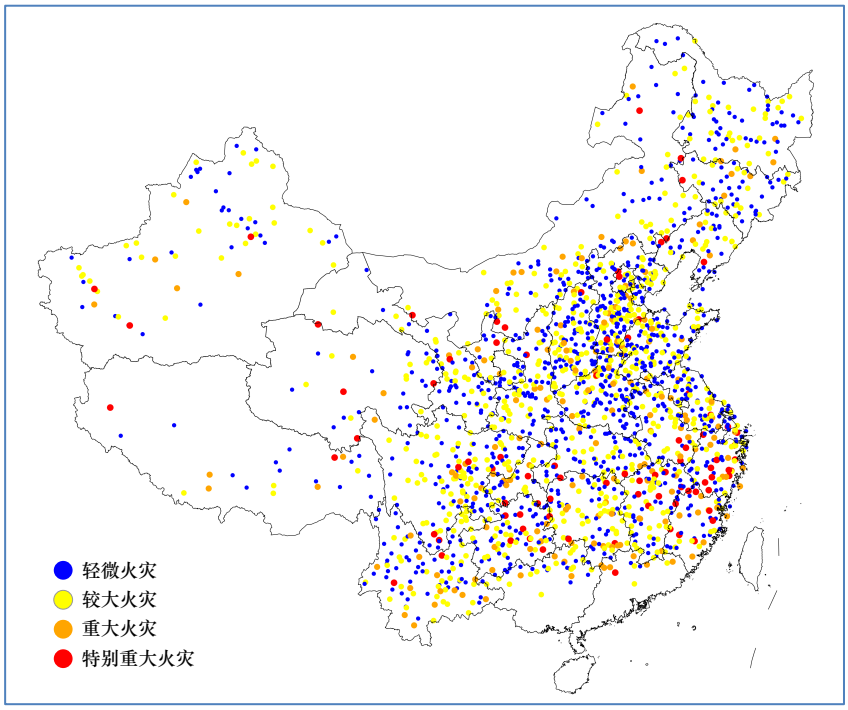


图 6 火灾危害等级映射

其中，轻微火灾 1031 起，较大火灾 681 起，重大火灾 183 起，特别重大火灾 88 起，具体比例见下图：

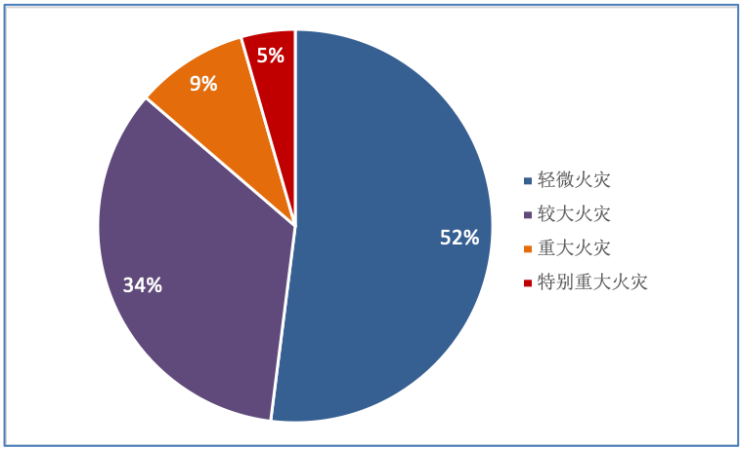


图 7 火灾比例图

危害最大的 10 个火灾如下表：

表 5 十大火灾危害表

| 编号 | 火灾类型 | 行政区域 | 初始面积（平方米） | 风力 | 预期蔓延速度 | 救援难度 | 预计经济损失（万元） |
|---|---|---|---|---|---|---|---|
| 880 | 草原 | 西藏-阿里-革吉 | 15077 | 7 | 特快 | 10 | 46760.466 |
| 1860 | 森林 | 浙江-宁波-宁海 | 5983 | 6 | 特快 | 10 | 43913.485 |
| 1960 | 森林 | 福建-龙岩-连城 | 11155 | 6 | 特快 | 10 | 40357.624 |
| 1969 | 森林 | 福建-泉州-安溪 | 398 | 5 | 特快 | 10 | 32318.182 |
| 200 | 草原 | 甘肃-兰州-永登 | 11382 | 5 | 特快 | 10 | 31902.775 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1445 | 森林 | 重庆-酉阳 | 14259 | 7 | 特快 | 8 | 30229.293 |
| 1908 | 森林 | 江西-吉安-峡江 | 1876 | 5 | 特快 | 10 | 29085.889 |
| 1730 | 城市 | 江苏-无锡-江阴 | 19265 | 7 | 特快 | 7 | 28891.254 |
| 1465 | 森林 | 湖南-岳阳 | 7789 | 6 | 特快 | 10 | 28794.590 |
| 1942 | 森林 | 福建-三明-宁化 | 8047 | 7 | 特快 | 7 | 27312.402 |

## 5.2 问题二的模型建立与求解

### 5.2.1 问题二的分析

问题二要求结合问题一中得到的火灾危害等级表及火灾发生的位置，筛选出重点的防火位置。考虑到合理、充分应用已有的信息，本文结合问题一计算得到的火灾危害等级（预计经济损失）及行政区域划分结果，统计每个行政区域所有火灾的危害等级及发生次数，归一化后使用 k-means 聚类，得到每个行政区域位置的火灾防护等级。问题求解流程如下图：

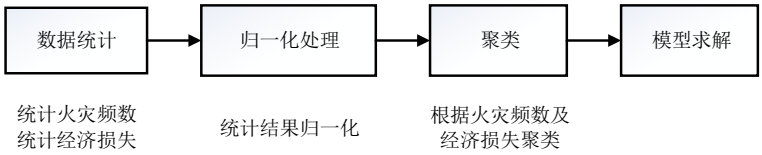


图 8 问题二求解流程图

### 5.2.2 问题二模型的建立

本文重点考虑区域位置火灾频发程度及火灾危害等级两个维度，对问题一已有结果进行统计，具体统计方案如下：

1）对二级行政区域，统计已知火灾发生次数；

2）对二级行政区域，对已知的火灾危害等级，即对预计经济损失求和。

为了便于之后聚类计算，对发生次数进行归一化处理：

$$nor_{n_i} = \frac{n_i}{\max(n_0, n_1, \ldots, n_m)} \tag{5.16}$$

对经济损失进行归一化处理：

$$nor_{l_i} = \frac{l_i}{\max(l_0, l_1, \ldots, l_m)} \tag{5.17}$$

### 5.2.3 问题二模型的求解

通过编写 python 程序（见附录四）将上一小节处理好的数据进行聚类，输出的聚类结果如下图：

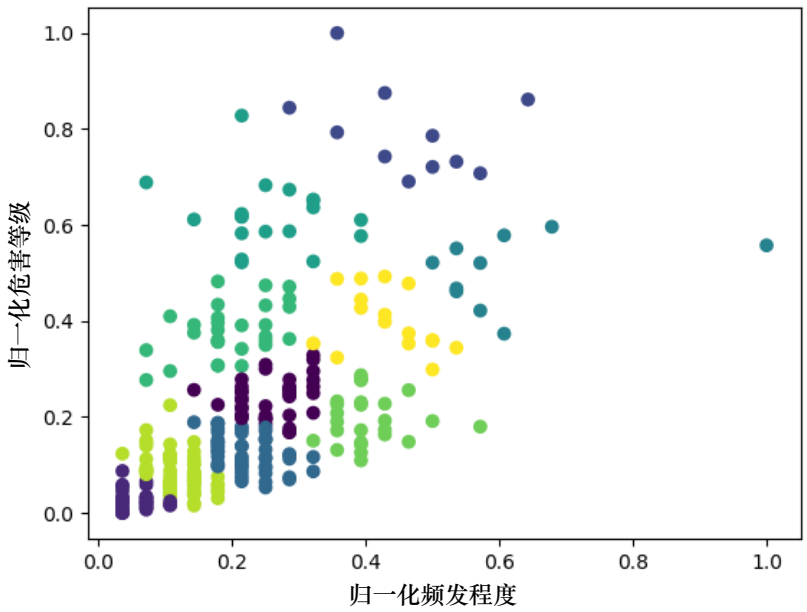


图 9 聚类结果图

其中，取聚类离原点最远的 2 类为重点防火位置，属于这些类簇的行政区域共 21 个，具体见下表：

表 6 重点防火位置表

| 行政区域 | 火灾频发程度（次） | 火灾危害等级<br>（预计经济损失-万元） |
|---|---|---|
| 河北省-保定市 | 28 | 38457 |
| 内蒙古自治区-呼伦贝尔市 | 19 | 41129 |
| 内蒙古自治区-鄂尔多斯市 | 18 | 59407 |
| 山西省-临汾市 | 17 | 25762 |
| 河北省-承德市 | 17 | 39892 |
| 四川省-甘孜藏族自治州 | 16 | 29084 |
| 江西省-赣州市 | 16 | 35922 |
| 江西省-上饶市 | 16 | 48812 |
| 内蒙古自治区-锡林郭勒盟 | 15 | 31838 |
| 内蒙古自治区-乌兰察布市 | 15 | 32262 |
| 山西省-吕梁市 | 15 | 38019 |
| 北京市 | 15 | 50473 |
| 四川省-凉山彝族自治州 | 14 | 35976 |
| 贵州省-黔东南苗族侗族自治州 | 14 | 49723 |
| 江西省-吉安市 | 14 | 54204 |
| 四川省-成都市 | 13 | 47624 |
| 重庆市-渝西地区 | 12 | 51219 |
| 湖南省-郴州市 | 12 | 60345 |
| 浙江省-宁波市 | 10 | 68959 |
| 江西省-抚州市 | 10 | 54688 |
| 重庆市-渝东南地区 | 8 | 58229 |

将重点防火位置映射到地图上得到直观的全国重点防火位置分布如下图：

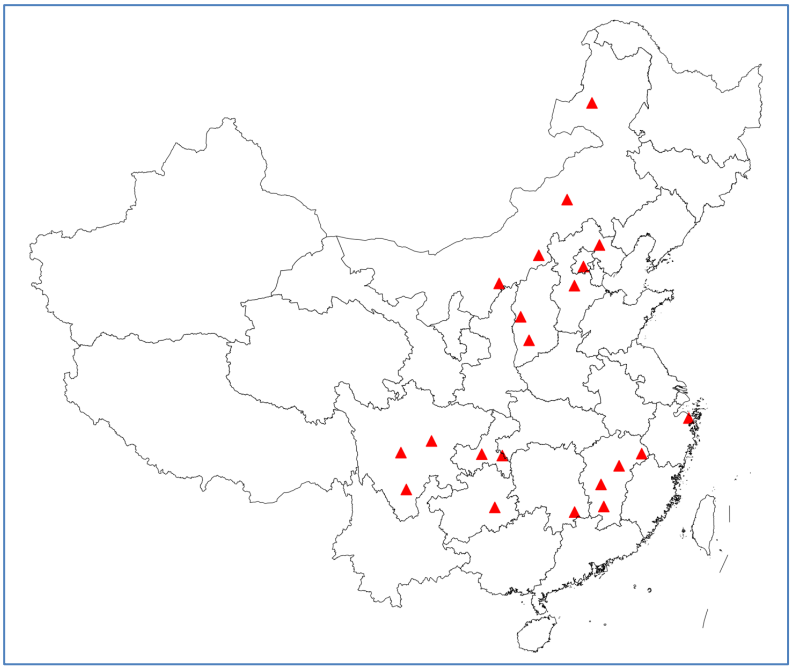

图 10 全国重点防火位置分布

## 5.3 问题三的模型建立与求解

### 5.3.1 问题三的分析

问题三要求将 100000 名消防人员合理分配到问题二中所筛选出的重点位置中。本文考虑不同的重点位置在火灾发生时候可能存在消防人员调配支援的现象，首先对问题二中得到的重点位置在地理坐标上聚类，同一类簇中对重点位置间建立连通图模型，以火灾损失与消防员配比作为优化目标，采用蚁群算法求解得到消防人员最优分配。问题的解决流程如下图：

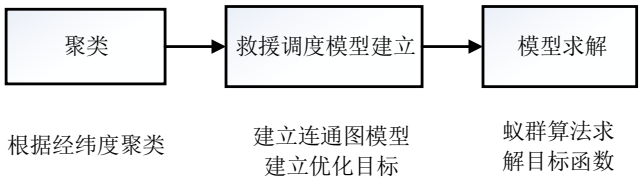


图 11 问题三求解流程图

### 5.3.2 问题三模型的建立

#### Step1 聚类

考虑到地理位置相关性不大的地区在火灾发生时候不互相调配消防员援助救援，本文首先对重点防护位置的经纬度信息聚类，得到的不同类簇间互相不调配消防人员，同一类簇中的重点区域可依据距离调配消防人员。使用 python 编写代码（见附录四）实现对于重点位置的经纬度信息的聚类，其结果如下图：

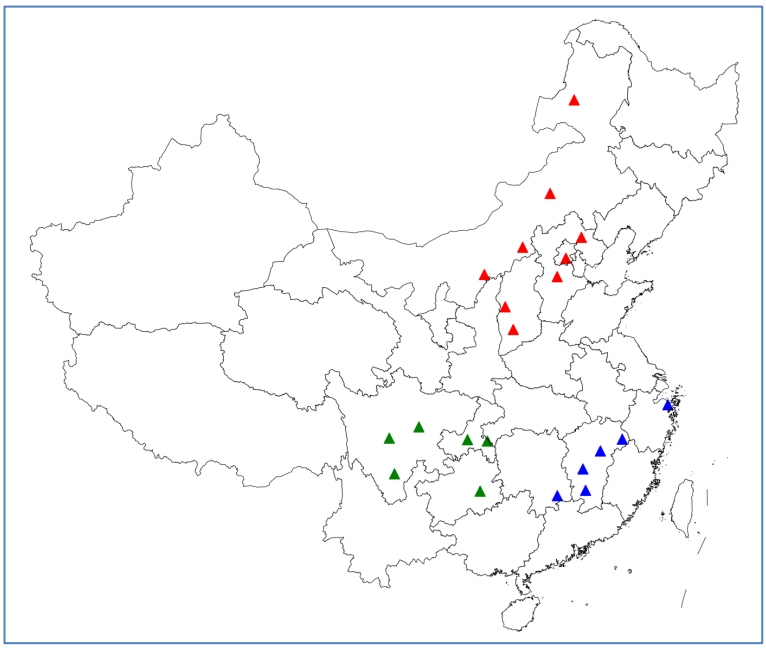

图 12 重点防护位置聚类图

### Step2 救援调度模型建立

对于消防员的分配，本文建立多连通分支无向图模型。其中，顶点为重点防护位置，连通分支为可互相调配消防员的区域单位。需要分配的消防员总人数满足：

$$sum_p = \sum\sum_{G_{i=0}}^{N} n_{po} \tag{5.18}$$

对于单个连通分支 $G_i$，$G_i$ 中两个顶点有边相连表示为两个重点位置相邻，重大火灾发生时候可直接调配消防人员支援，边权值表示为两个重点位置之间的距离。该模型的示意如下图：

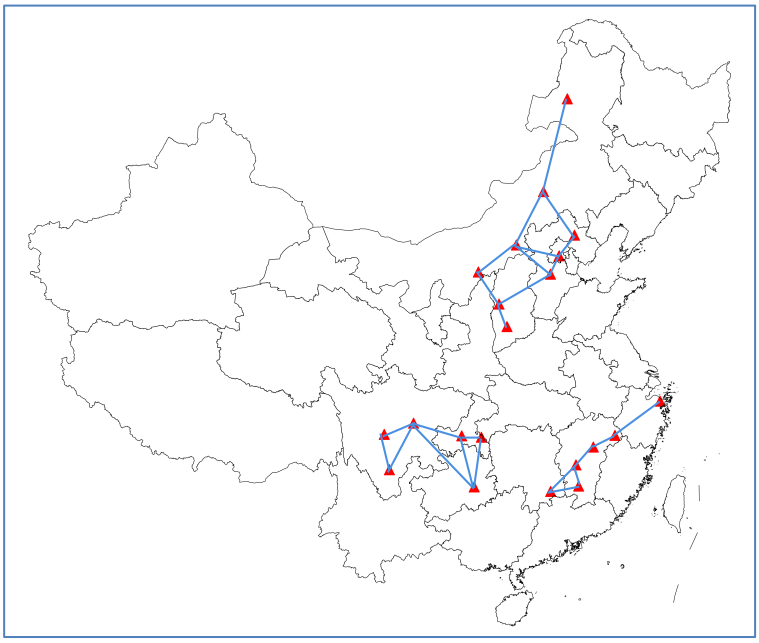

图 13 模型示意图

本文将每个重点位置的火灾危害等级（预计经济损失）与救援人数的比值定义为救援配比度，即该重点位置每个消防员匹配救援的预计经济损失量，表示为：

$$pr_h = \frac{l}{n_p} \tag{5.19}$$

其中，因为各个连通分支中能够按照位置距离相互调配消防员，于是救援人数满足：

$$n_p = n_{po} + n_{pd} \tag{5.20}$$

因为存在距离关系的影响，可调配人数表示为：

$$n_{pd} = \sum_i f(n_{po_i}, w) \tag{5.21}$$

至此，消防员最合理的分配可以表示为求解该图模型中所有顶点的救援匹配度的方差最小值：

$$\mathrm{D} = \sum_i (pr_{h_i} - E) \tag{5.22}$$

其中，E 是救援匹配度平均值，D 是救援匹配度方差。

### 5.3.3 问题三模型的求解

根据上一小节建立的模型，危害等级（预计经济损失）可以使用问题一的评价模型直接计算得出，因此模型的重点在于可调配人数与连通点常驻人数及边权值（距离）的关系，同时需要求得救援匹配度方差最小的解。本文使用蚁群算法求解最优分配方案。

#### Step1 可调配人数方程

由于城市之间距离决定了到达火灾现场救援的速度，本文假设可调配人数与边权值成反比，得到计算公式如下：

$$n_{pd_i} = \sum_{k=1}^{m} \frac{n_{po_k}}{distance(k,i)}，\ k\text{为与 i 相邻的城市} \tag{5.23}$$

#### Step2 求解最优分配方案

对于算法的初始状态，本文先将所有消防人员均匀分配到各个位置，并加上一些噪音值来保证每个位置初始状态的不一致性，每条边累计的信息素量决定了下一次这条边被选中的概率，初始化这些边的信息素都为 0，同时按迭代轮数均匀分配蚂蚁到各个位置，步长定为 5。

每轮迭代所有蚂蚁都进行移动操作，每次有向的移动都将带来该条有向边的人员调度，比如蚂蚁从城市 A 移动到城市 B，会将城市 A 中的消防人员个数减少一个步长的人数（即 5 人），同时为城市 B 增加一个步长的人数。蚂蚁移动的方向根据边上的信息素计算的概率得到，具体概率公式如下：

$$P_{ij}^{k}(t)=\begin{cases}\frac{\tau_{ij}^{\alpha}(t)\cdot\eta_{ij}^{\beta}(t)}{\sum_{s\neq prev}\tau_{is}^{\alpha}(t)\cdot\eta_{is}^{\beta}(t)}, j\neq prev\\ 0, other\end{cases} \quad (5.24)$$

其中，$\alpha$ 作为信息素启发式因子反映了信息素对蚂蚁路径选择的作用；$\beta$作为期望启发式因子反映了信息素在蚂蚁路径选择时被重视的程度；$\tau_{ij}$为城市 i 到城市 j 的路径上的信息素的量，$\eta_{ij}$为启发函数，表示为：

$$\eta_{ij}(\mathrm{t})=\frac{1}{d_{ij}},\ d_{ij}\text{为城市 i 和 j 间的距离} \quad (5.25)$$

每次蚂蚁移动之后，都会更新此条边上的信息素浓度，其变化公式如下：

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t+1) \quad (5.26)$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t+1) \quad (5.27)$$

其中，$\Delta\tau_{ij}^{k}(t+1)$表示经过一次循环后蚂蚁 k 在它走过的路上的信息素增量，ρ为信息素挥发因子，满足$\rho\in[0,1)$。

信息素的增量由经此轮迭代后救援匹配度方差的变化值来决定，若方差变大则信息素变化值为方差差值取反，反之则取正，目标是使得系统方差收敛至一个最小值，得到一个保证消防人员均匀分配的方案。

本文将以上过程使用 python 代码进行编程实现（见附录五），设定基本参数后迭代 10000 轮，在经过 2402 轮迭代后，救援匹配度方差收敛，得到结果如下：

表 7 消防人员分配结果

| 经度 | 纬度 | 行政区域 | 所分配的消防员人数（人） |
|---|---|---|---|
| 115.47 | 38.87 | 河北省-保定市 | 4012 |
| 119.77 | 49.22 | 内蒙古自治区-呼伦贝尔市 | 4697 |
| 109.80 | 39.62 | 内蒙古自治区-鄂尔多斯市 | 6617 |
| 111.52 | 36.08 | 山西省-临汾市 | 2447 |
| 117.93 | 40.97 | 河北省-保定市 | 4272 |
| 101.97 | 30.05 | 四川省-甘孜藏族自治州 | 2832 |
| 114.93 | 25.83 | 江西省-赣州市 | 3192 |
| 117.97 | 28.45 | 江西省-上饶市 | 4977 |
| 116.07 | 43.95 | 内蒙古自治区-锡林郭勒盟 | 3332 |
| 113.12 | 40.98 | 内蒙古自治区-乌兰察布市 | 3182 |
| 111.13 | 37.52 | 山西省-吕梁市 | 3862 |
| 116.40 | 39.90 | 北京市 | 5542 |
| 102.27 | 27.90 | 四川省-凉山彝族自治州 | 3782 |
| 107.97 | 26.58 | 贵州省-黔东南苗族侗族自治州 | 5332 |
| 114.98 | 27.12 | 江西省-吉安市 | 5442 |
| 104.07 | 30.67 | 四川省-成都市 | 5112 |
| 107.40 | 29.72 | 重庆市-渝西地区 | 5267 |

| | | | |
|---|---|---|---|
| 113.02 | 25.78 | 湖南省-郴州市 | 6367 |
| 121.55 | 29.88 | 浙江省-宁波市 | 7947 |
| 116.35 | 28.00 | 江西省-抚州市 | 5541 |
| 108.77 | 29.53 | 重庆市-渝东南地区 | 6246 |

# 六、模型的评价

## 6.1 模型的优点

模型合理，考虑全面。本文考虑到不同火灾类型的特征，通过参考卫星地图的方式进行了分类，并在模型建立时候考虑到了不同火灾类型的影响，分别建立了火灾评估模型进行表征求解。并且在救援人员分配模型建立时考虑到地理位置因素驱动的两点间救援人员调度问题，符合实际情况。

方法得当，结果直观。本文在考虑可燃物燃烧蔓延时采用了物理模型，在筛选火灾重点防护位置时采用了聚类方法，在建立救援人员分配模型时采用了连通图模型，在求解连通图模型时采用了蚁群算法。同时在展示每个问题步骤的求解结果时尽量使用图表形式展示，使其直观可见。

本文对火灾类型进行细分并对不同类型建立不同的火灾蔓延模型，通过蔓延模型建立统一的评价体系。同时在建立救援人员分配模型时考虑到现实火灾场景下应有消防人员的调配，建立的连通图模型与现实的契合度高。

本文在考虑受灾区域单位经济价值时，忽略对模型影响较小而又难以获得的因子，简化了模型的复杂度，并提高了求解模型的效率。

## 6.2 模型的缺点

在建立模型的同时，忽略了一些复杂的情况，如火灾造成的影响分为完全破坏面积以及过火面积，其中过火面积的部分也分为不同的过火程度，最终造成的经济损失也是不同的。倘若使用更为复杂的非线性危害评估模型，那么模型求解所需的数据量相对庞大，且正确性难以保证。故本文采用回归拟合的方法对火灾蔓延时间进行评估，尽可能的简单化考虑实际情况，以得到相对客观可靠且具有普适性的模型。

# 七、模型的推广与改进

## 7.1 模型推广

本文所设计的火灾危害等级评估模型和救援人员分配模型，原理简单且易于理解。对于真实场景下不同环境的火灾危害评估，考虑城市环境与植被环境的差异性，改变其扩散的方向即可求出对应结果；考虑植被环境下草原、森林、农田的差异性，只需改变单位面积经济价值即可推算出不同情况下的危害等级。同时，救援人员分配模型可以广泛适用于各种资源分配场景的求解，如网络资源分配及仓储物流分配等，只需修改蚁群算法中信息素更新策略及初始参数设定即可。

## 7.2 模型改进

本文的危害等级评估模型可以对于不同环境情况有较好的模拟，但真实情况下的火灾，其蔓延速度及蔓延时间还受可燃物载量、地形海拔等因素的影响。对于不同的环境进一步采用更为具体的元胞自动机进行模拟，可以将不同环境下的可燃物载量等因素纳入考虑，达到更加接近现实模拟的效果。同时在现实场景下，不同地区的道路情况差异很大，这也会带来一定交通上的延误，若能根据聚类情况进一步描述城市间距离，则能得到更为准确的结果。

# 八、参考文献

[1] Google 地图，http://www.google.cn/maps/，2019.6.16.
[2] GeoPy，https://geopy.readthedocs.io/en/stable/，2019.6.16
[3] 维基百科：分类-各类火灾，
https://zh.wikipedia.org/wiki/Category:%E5%90%84%E9%A1%9E%E7%81%AB%E7%81%BD，
2019.6.16
[4] 火灾统计管理规定，https://wenku.baidu.com/view/b744fc7bb4daa58da0114abe.html，2019.6.16.
[5] 孙新章. 中国农田生态系统的服务功能及其经济价值[J]. 中国人口·资源与环境, 2007, No.98(04):59-64.
[6] 中国森林生态资产价值评估[J]. 生态学报, 2017(12).
[7] 蔡学彩. 锡林郭勒草原生态系统服务价值评价[D]. 中国科学院研究生院（植物研究所）, 2005.
[8] 建筑估算价格表，https://wenku.baidu.com/view/9767654ffbd6195f312b3169a45177232f60e4b5.html，
2019.6.16.
[9] CEIC 数据库，https://www.ceicdata.com/zh-hans，2019.6.16

# 九、附录

## 附录一 GeoPy 爬虫

```
# -*- coding: utf-8 -*-
    import xlrd
    import xlwt
    import json
    import requests
    import numpy as np
    from geopy.geocoders import Baidu
    geocoder = Baidu(
               api_key='***',
               security_key='***',
               timeout=200
           )

    workbook_1 = xlrd.open_workbook(r'b.xlsx')

    sheet_1 = workbook_1.sheet_by_index(0)
    col_1 = sheet_1.col_values(1)
    col_2 = sheet_1.col_values(2)
    longitudes = []
    latitudes = []
    for i in range(1, len(sheet_1.col_values(0))):
        longitudes.append(col_1[i])
        latitudes.append(col_2[i])

    locations = []
    for i in range(len(latitudes)):
        locations.append(str(latitudes[i])+','+str(longitudes[i]))

    index = 0
    for i in locations:
        location= geocoder.reverse(i)
        print(location.raw)

print(location.raw['addressComponent']['province']+","+location.raw['addressCompo
nent']['city']+","+location.raw['addressComponent']['district'])
```

## 附录二 卫星地图爬虫

```
import xlrd
    import requests
    import os
```

```
    import math
    def deg2num(lat_deg, lon_deg, zoom):
        lat_rad = math.radians(lat_deg)
        n = 2.0 ** zoom
        xtile = int((lon_deg + 180.0) / 360.0 * n)
        ytile = int((1.0 - math.log(math.tan(lat_rad) + (1 / math.cos(lat_rad))) /
math.pi) / 2.0 * n)
        return (xtile, ytile)

    workbook_1 = xlrd.open_workbook(r'b-deal.xlsx')
    sheet_1 = workbook_1.sheet_by_index(0)
    col_1 = sheet_1.col_values(1)
    col_2 = sheet_1.col_values(2)
    col_3 = sheet_1.col_values(0)
    longitudes = []
    latitudes = []
    index = []

    for i in range(1, len(sheet_1.col_values(0))):
        longitudes.append(col_1[i])
        latitudes.append(col_2[i])
        index.append(col_3[i])

    for i in range(len(index)):
        x,y = deg2num(latitudes[i], longitudes[i], 15)
        URL = "http://www.google.cn/maps/vt?lyrs=s@815&gl=cn&x=" + str(x) + "&y=" +
str(y) + "&z=15"
        r = requests.get(URL)
        with open('./map/img'+str(i)+'.png','wb') as f:
            f.write(r.content)
```

## 附录三 问题一模型计算

```
# -*- coding: utf-8 -*-
    import math
    import xlrd
    v_f = {"缓慢" : 200, "慢" : 400, "适中" : 800, "快" : 1000, "特快" : 1500}
    v_c = {"缓慢" : 1, "慢" : 2, "适中" : 3, "快" : 4, "特快" : 5}

    def get_s(s0, vm, vf, rf, p0):
        t = 0.243 + 0.128 * vf ** 2 / 8 + 0.131 * rf ** 2 / 10
        delta_r = v_f[vm] * t
        return math.pi * (math.sqrt(s0 / math.pi) + delta_r) ** 2 * p0
```

```
    def get_city(s0, vm, vf, rf, p0, p1, p2):
        t = 0.243 + 0.128 * vf ** 2 / 8 + 0.131 * rf ** 2 / 10
        delta_r = v_c[vm] * t
        return s0 * p0 + delta_r / 3 * s0 * p1 + delta_r / 3 * math.sqrt(s0 / math.pi)
* math.pi * 2 * p2

    workbook_1 = xlrd.open_workbook(r'b-dealp.xlsx')
    sheet_1 = workbook_1.sheet_by_index(0)
    s0 = sheet_1.col_values(3)[1:]
    wind = sheet_1.col_values(4)[1:]
    pre_v = sheet_1.col_values(5)[1:]
    diff = sheet_1.col_values(6)[1:]
    prov = sheet_1.col_values(7)[1:]
    fire_t = sheet_1.col_values(8)[1:]
    data_input = []
    for i in range(len(s0)):
        tmp = prov[i].split(',')[0]
        data_input.append([int(s0[i]),   int(wind[i]),   pre_v[i],   int(diff[i]),
int(fire_t[i]), tmp])

    forest_value = {}
    f = open('forest','r')
    for i in f.readlines():
        i = i.split(' ')
        forest_value[i[0]] = float(i[1])

    if __name__ == '__main__':
        countera = 0
        counterb = 0
        counterc = 0
        counterd = 0
        result = []
        for data in data_input:
            s0, vf, vm, rf, tp, pv = data
            vf = vf + 1

            if tp == 4:
                ss = get_city(s0, vm, vf, rf, 5000, 1914, 1000)
            elif tp == 3:
                ss = get_s(s0, vm, vf, rf, 15.8)
            elif tp == 2:
                ss = get_s(s0, vm, vf, rf, forest_value[pv])
            elif tp == 1:
                ss = get_s(s0, vm, vf, rf, 1.8)
```

```
            if ss < 10000000:
                countera = countera + 1
            if ss >= 10000000 and ss < 50000000:
                counterb = counterb + 1
            if ss >= 50000000 and ss < 100000000:
                counterc = counterc + 1
            if ss >= 100000000:
                counterd = counterd + 1
            if ss < 10000000:
                print("轻微火灾")
            elif ss < 50000000:
                print("较大火灾")
            elif ss < 100000000:
                print("重大火灾")
            else:
                print("特别重大火灾")
            print(int(ss / 10000))
            result.append([s0, vf, vm, rf, tp, pv, ss / 10000])
        result.sort(key = lambda enum: enum[6], reverse = True)
        for i in result[0:10]:
            print(i)
        print("轻微火灾 : ", countera)
        print("较大火灾 : ", counterb)
        print("重大火灾 : ", counterc)
        print("特别重大火灾 : ", counterd)
```

## 附录四 k-means 聚类

```
# -*- coding: utf-8 -*-
    import math
    import pandas as pd
    import matplotlib.pyplot as plt
    from sklearn.cluster import KMeans

    if __name__ == '__main__':
        f = open('counter2.txt', 'r')
        cites = []
        c_f = []
        values = []
        maxf = 0
        maxv = 0
        for i in f.readlines():
            tmp = i.strip().split(',')
            cites.append(tmp[0])
```

```
            c_f.append(int(tmp[1]))
            values.append(float(tmp[2]))
            if (int(tmp[1]) > maxf):
                maxf = int(tmp[1])
            if (float(tmp[2]) > maxv):
                maxv = float(tmp[2])
        for i in range(len(c_f)):
            c_f[i] = c_f[i] / maxf
        for i in range(len(values)):
            values[i] = values[i] / maxv
        df = pd.DataFrame({'frequency' : c_f, 'value' : values})
        X = df.ix[:,['frequency', 'value']]
        clf_KMeans = KMeans(n_clusters=10)
        counter = [0] * 10
        cluster = clf_KMeans.fit_predict(X)
        index = 0
        f.close()
        f = open('counter2.txt', 'r')
        ff = open('counter3.txt', 'w')
        for i in f.readlines():
            ff.write(i.strip() + ',' + str(cluster[index]) + '\n')
            counter[cluster[index]] = counter[cluster[index]] + 1
            index = index + 1
        plt.figure()
        plt.scatter(X['frequency'],X['value'],c=cluster)
        plt.xlabel(u'归一化频发程度')
        plt.ylabel(u'归一化火灾等级')
        plt.show()
```

## 附录五 蚁群算法

```
# -*- coding: utf-8 -*-
from math import radians, cos, sin, asin, sqrt

    def haversine(lon1, lat1, lon2, lat2):
        lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
        c = 2 * asin(sqrt(a))
        r = 6371
        return c * r * 1000

    def variance(data):
        sum1 = 0
```

```
        sum2 = 0
        for i in range(len(data)):
            sum1 += data[i]
            sum2 += data[i] ** 2
        mean = sum1 / len(data)
        var = sum2 / len(data) - mean ** 2
        return var

    city_neighbor = {
        "河北省": ("山西省", "内蒙古省", "辽宁省"),
        "内蒙古省": ("黑龙江省", "吉林省", "辽宁省", "河北省", "山西省", "陕西省", "宁夏省",
"甘肃省"),
        "山西省": ("河南省", "内蒙古省", "河北省", "陕西省"),
        "四川省": ("重庆市", "云南省", "陕西省", "甘肃省", "贵州省", "西藏省", "青海省"),
        "江西省": ("浙江省", "福建省", "广东省", "湖南省", "湖北省", "安徽省"),
        "北京市": ("天津市", "河北省"),
        "贵州省": ("四川省", "重庆市", "云南省", "湖南省", "广西省"),
        "重庆市": ("四川省", "湖北市", "陕西省", "贵州省"),
        "湖南省": ("江西省", "湖北市", "重庆市", "贵州省", "广东省", "广西省"),
        "浙江省": ("江西省", "福建市", "上海市", "安徽省", "江苏省"),
    }

    latitudes = []
    longitudes = []
    provinces = []
    values = []
    fs = [0] * 21
    ps = [0] * 21
    ns = [4761] * 21
    for i in range(19):
        ns[i] += 1
    f = open('counter4.txt', 'r')
    for i in f.readlines():
        tmp = i.strip().split(',')
        provinces.append(tmp[-1])
        latitudes.append(float(tmp[-2]))
        longitudes.append(float(tmp[-3]))
        values.append(int(tmp[2]))

    graph = []
    for i in range(len(provinces)):
        dis = []
        for j in range(len(provinces)):
            if    (provinces[i]    ==    provinces[j])    or    (provinces[j]    in
```

```
city_neighbor[provinces[i]]):

dis.append(int(haversine(longitudes[i],latitudes[i],longitudes[j],latitudes[j])  /
1000))
          else:
             dis.append(-1)
       graph.append(dis)
   rounds = 10000
   step = 5
   while(rounds > 0):
       for i in range(len(graph)):
          ps[i] = 0
          for j in range(len(graph[i])):
             if graph[i][j] > 0:
                ps[i] = ps[i] + int(ns[j] / (graph[i][j] / 10))
       for i in range(len(fs)):
          fs[i] = values[i] / (ps[i] + ns[i])
       print("Round ", rounds, " : ", variance(fs))
       maxn = fs.index(max(fs))
       minx = fs.index(min(fs))
       ns[maxn] += step
       ns[minx] -= step
       rounds -= 1
   f.close()
   f = open('counter4.txt', 'r')
   ff = open('counter5.txt', 'w')
   index = 0
   for i in f.readlines():
       ff.write(i.strip() + ',' + str(ns[index]) + '\n')
       index += 1
```