基于"XXX"的低碳建筑研究

摘要

关键词

1 问题重述

1.1 问题背景

“双碳”即碳达峰与碳中和的简称，我国力争 2030 年前实现碳达峰，2060 年前实现碳中和。“双碳”战略倡导绿色、环保、低碳的生活方式。我国加快降低碳排放步伐，大力推进绿色低碳科技创新，以提高产业和经济的全球竞争力。低碳建筑是指在建筑材料与设备制造、施工建造和建筑物使用的整个生命周期内，减少化石能源的使用，提高能效，降低二氧化碳排放量。

1.2 目标任务

问题一：计算给定建筑通过空调调节温度的年碳排放量。

问题二：建立综合评价模型，找出易于量化的指标，评估居住建筑整个生命周期的碳排放。

问题三：基于问题二，考虑建筑生命周期三个阶段的碳排放问题，对江苏省 13 个地级市的居住建筑进行评价，验证模型的有效性。

问题四：建立碳排放预测模型，基于江苏省建筑全过程碳排放的历史数据，对 2023 年江苏省建筑全过程的碳排放量进行预测。

问题五：结合前面的讨论给出江苏省建筑碳减排的政策建议。

2 问题分析

2.1 问题一

问题一要求计算通过空题调节温度产生的年碳排放量。我们需先求出空调制热和制冷的热量，借此通过 COP 和 EER 求出空调消耗的电量，最后转换成碳排放。其中 COP 和 EER 的定义分别为

$$COP = \frac{Q_{heat}}{W}, \qquad EER = \frac{Q_{cold}}{W} \tag{2.1}$$

Q_{heat}/Q_{cold} 指的是单位时间内的制热/制冷量，单位为 J，公式中 W 指的是单位为时间内空调消耗的功率，单位为 W

首先计算出建筑物各个月的能量需求量。设室内温度要维持的温度为 t_{in}，室外温度为 t_{out}, 当月该地区平均温度为 t_{ave} ，方便起见，我们规定

$$t_{out} = t_{ave}$$

$$t_{in} = \begin{cases} 18°C & t_{out} < 18°C \\ t_{out} & t_{out} \in [18°C, 26°C] \\ 26°C & t_{out} > 26°C \end{cases}$$

我们使用热传导方程计算用来需要制热/制冷的热量，其形式为

$$\Phi = \frac{\lambda \cdot A \cdot |\Delta T|}{\delta} \tag{2.2}$$

其中 Φ 表示传热速率，λ 为导热系数，A 为传热面积，ΔT 是室内外温度差，即 $t_{in} - t_{out}$，δ 表示材料厚度。

将建筑分成墙、门窗、房顶、地面四个部分，分别计算并累加即可得到需要制热/制冷的热量，设为 Q_{make}，由 COP 和 EER 的定义可得到需电量 Q_{elec} 和热量 Q_{make} 的转化关系

$$Q_{elec} = \begin{cases} \frac{Q_{make}}{EER} & \Delta t < 0 \\ 0 & \Delta t = 0 \\ \frac{Q_{make}}{COP} & \Delta t > 0 \end{cases} \tag{2.3}$$

最后根据需电量与碳排放的换算关系 $m = Q_{elec} \cdot 0.28$ 求出每月碳排放后累加，即得到年度碳排放量。

3 模型假设

4 符号说明

5 模型的建立与求解

5.1 问题一的模型建立与求解

5.2 问题二的模型建立与求解

5.2.1 建立层次结构模型

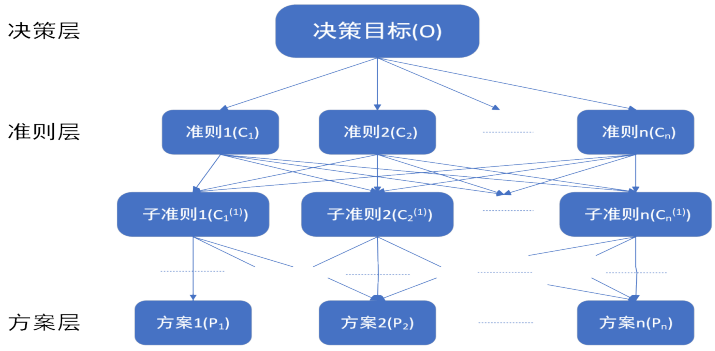

图 1: 层次分析法框架

准则层中准则因素之间相互独立。
我们选择的准则因素有：生活使用能耗、地区差异、周边产业、建造与拆除能耗、生产运输。

5.2.2 构建成对比较矩阵及归一化

1. 构建比较矩阵
   构造比较矩阵是通过比较同一层次上的各因素对上-层相关因素的影响作用. 而不是把所有因素放在一起比较，即将同一层的各因素进行两两对比。设某层有 n 个因素，$x = \{x_1, x_2 \dots x_n\}$ 要比较它们对上一层某一准则 (或目标) 的影响程度，确定在该层中相对于某一准则所

占的比重。上述比较是两两因素之间进行的比较，比较时常取 1 9 尺度。

尺度	含义
1	第 i 个因素与第 j 个因素影响相同
3	第 i 个因素与第 j 个因素影响稍强
5	第 i 个因素与第 j 个因素影响较强
7	第 i 个因素与第 j 个因素影响明显强
9	第 i 个因素与第 j 个因素影响极端强
2, 4, 6, 8	两相邻判断的中间值

用 a_{ij} 表示第 i 个因素相对于第 j 个因素的比较结果，则

$$a_{ij} = \frac{1}{a_{ji}}$$

$$A = (a_{ij})_{n\times n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \tag{5.1}$$

A 则称为成对比较矩阵。

2. 归一化

对各城市的数据进行归一化处理

数据 城市 / 指标	苏州	南京	南通	无锡	常州
直接	48	51.5	27	57.2	29
间接	126.3	134.3	118.7	97.8	71.6
运营	4.8	4.9	3.5	2.7	2.8
归一化比例	0.226	0.242	0.127	0.269	0.126
	0.230	0.245	0.216	0.178	0.130
	0.257	0.262	0.187	0.144	0.150

5.2.3 层次单排序及一致性检验

1. 层次单排序

和积法：取判断矩阵 n 个列向量归一化后的算术平均值，近似作为权重，即

$$W_i = \frac{1}{n}\sum_{j=1}^{n}\frac{(\prod_{j=1}^{n} a_{ij})^{\frac{1}{n}}}{\sum_{k=1}^{n}(\prod_{j=1}^{n} a_{kj})^{\frac{1}{n}}}(i = 1, 2, \cdots, n) \tag{5.2}$$

求根法（几何平均法）：将比较矩阵的各列（或行）向量求几何平均后归一化，可近似作权重，即

$$W_i = \sum_{j=1}^{n}\frac{(\prod_{j=1}^{n} a_{ij})_{\frac{1}{n}}}{\sum_{k=1}^{n}(\prod_{j=1}^{n})_{\frac{1}{n}}}(i = 1, 2, \cdots, n) \tag{5.3}$$

在我们的模型中综合采用了两种方法得到权重，降低了单一方法带来的不确定性，使数据结果更加可靠。

2. 一致性检验

通常情况下，由实际得到的判断矩阵不一定是一致的，即不一定满足传递性和一致性实际中，也不必要求一致性绝对成立，但要求大体上是一致的，即不一致的程度应在容许的范围内主要考查以下指标：

(a) 一致性指标 CI

$$CI = \frac{\lambda_{\max} - n}{n-1} \tag{5.4}$$

(b) 平均随机一致性指标 RI

为衡量 CI 的大小，引入随机一致性指标 RI：

$$RI = \frac{CI_1 + CI_2 + \cdots + CI_n}{n} \tag{5.5}$$

其中，随机一致性指标 RI 和判断矩阵的阶数有关，一般情况下，矩阵阶数越大，则出现一致性随机偏离的可能性也越大，对于阶数小于 9，其对应关系如图：

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

(c) 检验系数 CR

考虑到一致性的偏离有可能是由于随机原因造成的，因此在检验判断矩阵是否具有满意的一致性时，还需将 CI 和 RI 进行比较，得出检验系数 CR，公式如下：

$$CR = \frac{CI}{RI} \tag{5.6}$$

一般地，如果 $CR \leq 0.1$，则认为该判断矩阵通过一致性检验，$A_{\max}$ 对应的特征向量 W 可以作为排序的权重向量，此时

$$\lambda_{\max} \approx \sum_{i=1}^{n} \frac{(A \cdot W)_i}{nw_i} = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{j=1}^{n} a_{ij} w_j}{w_i} \tag{5.7}$$

否则就不具有满意一致性，需要调整对比较矩阵。

5.2.4 计算组合权重及得分

得到最大特征值对应的特征向量

$$T = [t_1 \quad t_2 \quad \cdots \quad t_n]$$

计算得到权重向量

$$W = [w_1 \quad w_2 \quad \cdots \quad w_n]$$

$$w_i = \frac{t_i}{\sum_{i=1}^{n} t_i}$$

设指标评分矩阵为 P，那么最后的得分矩阵 S 为：

$$S = P \times W$$

6 结果检验与误差分析

7 模型评价

8 模型推广与改进

9 参考文献

10 附录

附录 A 问题一

Listing 1: **Question1**

```
# -*- coding:utf-8 -*-
"""
Project: MathematicsModeling-2023.5
Written by: Evan Wong
DATE: 2023/5/9
TIME: 17:09
"""

import matplotlib.pyplot as plt

monthdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
temperatures = [-1, 2, 6, 12, 22, 28, 31, 32, 26, 23, 15, 2]

COP = 3.5
EER = 2.7

length = 4
width = 3
height = 3

roof_thickness = 0.3
roof_thermal = 0.2
roof_square = length * width
windowAndDoor_thickness = 0.1
windowAndDoor_thermal = 1.6
windowAndDoor_square = 5
wall_thickness = 0.3
wall_thermal = 0.3
wall_square = (length + width) * height * 2 - windowAndDoor_square
ground_thickness = 1.0
ground_thermal = 0.25
ground_square = length * width


def get_delta_t(tout):
    if tout < 18:
```

```
37          return 18 - tout
38      elif tout > 26:
39          return 26 - tout
40      else:
41          return 0
42
43
44  def get_qmake(k, a, t, d):
45      return (k * a * t) / d
46
47
48  def get_day_qmake(dt):
49      if dt == 0:
50          return 0.0
51
52      day_qmake = \
53          get_qmake(wall_thermal, wall_square, dt, wall_thickness) + \
54          get_qmake(windowAndDoor_thermal, windowAndDoor_square, dt,
                windowAndDoor_thickness) + \
55          get_qmake(roof_thermal, roof_square, dt, roof_thickness) + \
56          get_qmake(ground_thermal, ground_square, dt, ground_thickness)
57      return day_qmake * 86400
58
59
60  def get_month_qmake(month):
61      return get_day_qmake( abs(get_delta_t(temperatures[month]))) * monthdays[month]
62
63
64  def get_qelec(qmake, dt):
65      if dt < 0:
66          return qmake / EER
67      elif dt == 0:
68          return 0
69      else:
70          return qmake / COP
71
72
73  def get_carbon_emission(qelec):
74      return (qelec / 3600000) * 0.28
75
76
77  carbonEmissions = []
```

```
for i in  range(0, 12):
    qelec = get_qelec(get_month_qmake(i), get_delta_t(temperatures[i]))
    carbonEmissions.append(get_carbon_emission(qelec))

print("Sum of carbon emissions: " +  str( sum(carbonEmissions)) + "kg")

plt.figure(figsize=(10, 6))
plt.bar( range(1,  len(carbonEmissions) + 1), carbonEmissions, fc='g')
plt.title("Carbon Emissions")
plt.xlabel("Month")
plt.ylabel("kg")

plt.show()
```

附录 B　　问题二

Listing 2: **Question2**

```
import numpy as np
import matplotlib.pyplot as plt


cities = ["SuZhou", "NanJing", "NanTong", "WuXi", "ChangZhou"]

discriminantMatrix = np.array([[1, 1/5, 1/5], [5, 1, 1/2], [5, 2, 1]])
eigens = np.linalg.eig(discriminantMatrix)
maxEigenvalue = np. max(eigens[0])
rowAndColumn = np.argwhere(eigens[0] == maxEigenvalue)
maxEigenvector = eigens[1][::-1, rowAndColumn[0]]

RI_LIST = [0, 0.001, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49, 1.52, 1.54, 1.56,
    1.58, 1.59]
dim = discriminantMatrix.shape[0]
RI = RI_LIST[dim]
CI = (maxEigenvalue - dim) / (dim - 1)
CR = CI / RI
print("CR: " +  str(CR))
print("Passed consistency test") if CR < 0.1 else print("Failed to passed consistency
    test")

Scores = np.array([[48.0, 126.3, 4.8],
```

```
                    [51.5, 134.3, 4.9],
                    [27, 118.7, 3.5],
                    [57.2, 97.8, 2.7],
                    [29, 71.6, 2.8]])

sums = np. sum(Scores, axis=0)
weightedScores = Scores / sums.reshape(1, -1)

weight = maxEigenvector /  sum(maxEigenvector)

finalScores = np.dot(weightedScores, weight)
for i in  range( len(finalScores)):
    print('City {:}, Scores {:}'. format(cities[i], finalScores[i, 0].real))

plt.figure(figsize=(10, 6))
x = [1, 2, 3, 4, 5]
x_label = cities
plt.bar(x, finalScores[:, 0].real, fc='g')
plt.title("Scores given by evaluation model")
plt.xticks(x, x_label)
plt.xlabel("City")
plt.ylabel("Evaluation score")

plt.show()
```

附录 C　　问题三

附录 D　　问题四

附录 E　　问题五