# 3.2 权重决策

## 3.1.1 多准则决策法

多准则决策法，Multiple Criteria Decision-Making（MCDM）也称之为 Multi-Criteria Decision Analysis（MCDA），Multi-attribute Decision-Making（MADM），或multi-object decision-making（MODM）（多目标决策法），是关于在同时考虑多个准则（criteria）/目标（objectives）下对备选方案（alternatives）进行排名（rank）或选择时做出决策。当准则越复杂， 备选方案增多时，确定哪个为最优方案的难度就越大。MCDM经过几十年的发展，各类算法涌现，已经成为解决现实问题的强大工具，可以帮助寻找最能满足要求众多相互冲突目标的决策[1]。MCDM被广泛应用于数学、经济学、管理学和工程领域等众多学科，服务于商业、非营利组织、政府、卫生、教育和个人决策等。MCDM（离散型，discrete）问题可以表述为一个矩阵，，式中为备选方案（feasible alternatives，actions，stimuli）；为决策准则（decision-making criterion）； pij ij

iVi  Viw*={*w*1​,*w*1​,…,*wn*​}的*确定[2]。

从上述对MCDM的描述及公式的表述，可以发现规划中的很多地理空间问题都可以转换为MCDM问题，例如其中的备选方案可以为城市的行政分区、子流域、样方、采样点、乃至栅格单元等；而决策准则为各类评价指标（体系），例如生态敏感性评价指标（压力层面的人口密度指数、建设用地占比、农业用地占比、污水排放率、废弃排放量、粉尘排放量、灾害易发度和地形坡度；状态层面的土地胁迫指数、生境质量指数、土地利用类型、水网密度指数、年降雨量、年总气温、植被覆盖度和经济发达指数；及响应层面的造林面积、生活污水处理率、工业SO2去除量和固体废物利用率等），采样熵值权重法和PCA权重法计算[3]；生态脆弱性评价指标（生态敏感性层面的景观破碎度、地形指数、土壤侵蚀量；生态恢复力层面的蔓延度、归一化植被指数、生物丰度等；生态压力度层面的人类活动强度和土地利用程度等），应用了熵权法计算[4]；景观感知决定因素（环境价值、文化价值、美学价值和个人经历或回忆等）[5]；热脆弱性指数（Heat Vulnerability Index，HVI）（涉及人口密度、小于5岁儿童人口比例、大于65，大于85的老人人口比例、大于65独居老人人口比例、小于5岁低于贫困线家庭的儿童人口比例、大于65岁低于贫困线家庭的老人人口比例、树冠覆盖率和早于1980年建成的建筑比例等），使用熵权法和TOPSIS（Technique for Order of Preference by Similarity to Ideal Solution ）算法计算[6]；生态系统服务评估和权衡（Integrated Valuation of Ecosystem Services and Tradeoffs，[InVEST](https://naturalcapitalproject.stanford.edu/software/invest)）①中的很多模型也可以从MCDM角度思考，例如生物多样指数（涉及到当前土地利用/土地覆盖类型、威胁影响距离、威胁地图、栖息地适宜性和栖息地对威胁的敏感性等）。

就MCMD问题，当前形成有多个[软件工具](https://en.wikipedia.org/wiki/Decision-making_software#Comparison_of_decision-making_software)②，Valdecy Pereira则将其集成到了Python库中，构建了[pyDecision](https://github.com/Valdecy/pyDecision)包③，包括40余种算法。对MCMD的解释，首先再现Soheil Boroushaki基于熵值权重法和改进的TOPSIS算法的MCDA计算[6]；并将其应用于城市环境传感器AoT污染气体浓度数据测量点污染度指数评估中；再引入解释pyDecision集成的各类算法。

# IPython extension to reload modules before executing user code.  
%load\_ext autoreload   
# Reload all modules (except those excluded by %aimport) every time before executing the Python code typed.  
%autoreload 2   
  
from usda import datasets as usda\_datasets  
from usda import weight as usda\_weight  
from usda import database as usda\_database  
from usda import utils as usda\_utils  
from usda import data\_visualization as usda\_vis  
from usda.database import postSQL2gpd,gpd2postSQL  
  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import pyDecision.algorithm as da  
  
np.set\_printoptions(linewidth=np.inf)

### 3.1.1.1 再现熵值权重法和改进的TOPSIS算法

#### 1) 熵值权重法

对于MCDM问题矩阵，包含一定数量的信息，可以用于探索决策准则之间对备选方案综合评价指数影响的程度，即计算决策准则权重。权重的主观方法是由决策者、规划者、利益相关者根据经验或调查配置权重值，而客观方法则可以根据矩阵决策准则中包含的信息量评估权重值。信息论（information theory）中香农的熵概念（Shannon’s entropy concept）已广泛应用于工程、信息科学和管理领域，用于衡量MCDM问题中包含的信息量，计算决策准则（例如各类指标）的信息量。

将Soheil Boroushaki构建的示范数据矩阵写入到USDA库中，可以通过usda\_datasets.load\_evaluation\_criteria\_raw\_values()方法调入，该数据用DataFrame格式表述，列为决策准则（criteria），包括坡度（slope）（度）、到水体距离（distance2water）（英里）、高程（elevation）（尺）和到人口中心的距离（distance2population）（英里）；行为备选方案（alternative），对应地理空间的位置区域，索引值为0、1、2和3。

evaluation\_criteria\_raw\_values\_df=usda\_datasets.load\_evaluation\_criteria\_raw\_values().data  
evaluation\_criteria\_raw\_values\_df

|  | **slope** | **distance2water** | **elevation** | **distance2population** |
| --- | --- | --- | --- | --- |
| **0** | 9 | 2.2 | 5700 | 1.2 |
| **1** | 20 | 3.2 | 3100 | 1.3 |
| **2** | 10 | 2.0 | 4900 | 1.2 |
| **3** | 16 | 3.5 | 3600 | 1.4 |

信息熵的计算公式为：，式中：，为原始决策准则值，为原始决策准则值占各自决策准则列之和的比例，定义df\_Pij()方法实现计算。由信息熵计算决策准则的权重值，公式为：，定义entropy\_weight()方法，实现权重值计算，输入矩阵为原始决策准则值，直接在该函数内部实现。

df\_Pij=usda\_weight.df\_Pij(evaluation\_criteria\_raw\_values\_df)  
df\_Pij

|  | **slope** | **distance2water** | **elevation** | **distance2population** |
| --- | --- | --- | --- | --- |
| **0** | 0.163636 | 0.201835 | 0.329480 | 0.235294 |
| **1** | 0.363636 | 0.293578 | 0.179191 | 0.254902 |
| **2** | 0.181818 | 0.183486 | 0.283237 | 0.235294 |
| **3** | 0.290909 | 0.321101 | 0.208092 | 0.274510 |

Ej列为信息熵，Wj列为权重值。其中slope具有相对最小的信息熵，表明slope决策准则值在备选方案（地理空间区域）中分布相对不均匀，权重值相对较高；而对于distance2population决策准则值在备选方案中分布相对均匀，权重值则相对较低。

entropy\_weight\_df=usda\_weight.entropy\_weight(evaluation\_criteria\_raw\_values\_df)  
entropy\_weight\_df

|  | **Ej** | **1-Ej** | **Wj** |
| --- | --- | --- | --- |
| **slope** | 0.961706 | 0.038294 | 0.477279 |
| **distance2water** | 0.980097 | 0.019903 | 0.248060 |
| **elevation** | 0.979473 | 0.020527 | 0.255832 |
| **distance2population** | 0.998489 | 0.001511 | 0.018829 |

#### 2) TOPSIS（理想解法）

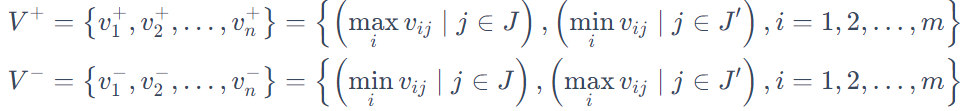
决策规则（decision rule）可以定义为通过规范化具有权重的决策准则，计算每个备选方案的综合性指标，进行排序，选择最佳方案的过程。MCDM的决策规则包含有加权线性组合（weighted linear combination，WLC）、加权求和/布尔叠加法（weighted summation/Boolean overlay methods）、层次分析法（Analytic Hierarchy Process，AHP），理想/参考点法（ideal/reference point methods）和排序法（outranking methods）等众多算法[7,8]。TOPSIS (Technique for Order Performance by Similarity to Ideal Solution)算法的基本原理是折衷的解决方案必须同时靠近正理想解（positive ideal solution，PIS）而远离负理想解（ negative ideal solution，NIS）。正理想解表示一个假设的备选方案，其由每个决策准则的最理想的规范化属性值组成；而负理想解表示的假设备选方案由每个决策准则的最差性能的规范化属性值组成。TOPSIS的基本假设是决策准则（评估标注）值的单调递增或递减，因此识别一组决策准则值的最大值和最小值，可以很容易确定每个准则的正理想和负理想解。TOPSIS 结合到正理想解和负理想解的距离来衡量每个备选位置与正理想解的相对接近程度，然后，备选方案将根据这一综合衡量标准进行排名[9]。

首先通过将决策矩阵中的每个决策准则值标准化为无量纲的来构建归一化决策矩阵，以便比较不同的属性，其公式为： ，定义df\_standardized\_evaluation()方法实现。

standardized\_evaluation\_criteria\_df=usda\_weight.df\_standardized\_evaluation(evaluation\_criteria\_raw\_values\_df)  
standardized\_evaluation\_criteria\_df

|  | **slope** | **distance2water** | **elevation** | **distance2population** |
| --- | --- | --- | --- | --- |
| **0** | 0.311086 | 0.393045 | 0.641016 | 0.469596 |
| **1** | 0.691301 | 0.571702 | 0.348623 | 0.508729 |
| **2** | 0.345651 | 0.357314 | 0.551049 | 0.469596 |
| **3** | 0.553041 | 0.625299 | 0.404852 | 0.547862 |

然后，确定正理想解（）和负理想解（V- (NIS)



JJ’与成本、最小化或反向评估标准相关联。定义PIS\_NIS()方法实现，对于每个决策准则方向的确定（值越大越好，约趋向正理想解，还是正好相反；负理想解同，通常与正理想解相反），通过传入pis参数确定，该参数的形式设计为一个字典，以决策准则（列名）为键，值为1或者0，为1时表明正理想解越大越好；为0时表明正理想解越小越好。负理想解与正理想解相反（由程序内部判断）。例如，对于slope决策准则为值越低越好，因此正理想解为0.311为最小值，而负理想解为最大值0.691；对于distance2population决策准则值则越大越好，因此正理想解为0.547，而负理想解为0.469。

pis\_nis=usda\_weight.PIS\_NIS(standardized\_evaluation\_criteria\_df,pis={'slope':0,'distance2water':0,'elevation':0,'distance2population':1})   
pis\_nis

|  | **slope** | **distance2water** | **elevation** | **distance2population** |
| --- | --- | --- | --- | --- |
| **V+** | 0.311086 | 0.357314 | 0.348623 | 0.547862 |
| **V-** | 0.691301 | 0.625299 | 0.641016 | 0.469596 |

进而，计算各备选方案到正负理想解的距离（和），公式为：，式中， w\_jp$本次实验配置值为2，为欧氏距离度量。

最后，计算各备选方案的排序指标值（即综合评价指数），为到正理想解的相对接近度（ relative closeness），公式为： 。

定义closeness\_pis\_nis()方法实现TOPSIS算法，包括决策矩阵标准化、确定正负理想解及距离计算和综合评价指数计算过程。计算结果显示，备选方案2为最优解，是最适合的选择区域。

separation\_measure\_df=usda\_weight.closeness\_pis\_nis(standardized\_evaluation\_criteria\_df,pis={'slope':0,'distance2water':0,'elevation':0,'distance2population':1},Wj=entropy\_weight\_df['Wj'])  
separation\_measure\_df

|  | **Si+** | **Si-** | **Ci+** | **Rank** |
| --- | --- | --- | --- | --- |
| **0** | 0.075341 | 0.190395 | 0.716481 | 2.0 |
| **1** | 0.189103 | 0.075979 | 0.286626 | 4.0 |
| **2** | 0.054371 | 0.179345 | 0.767362 | 1.0 |
| **3** | 0.134021 | 0.089482 | 0.400361 | 3.0 |

### 3.1.1.2 污染气体浓度数据测量点污染浓度综合指数评估

使用“2.7.1 城市环境传感器AoT（SAGE）数据预处理” 一章中处理后的数据，就污染气体（CO, H2S, NO2, O3, SO2, oxidizing\_gases和reducing\_gases）浓度值评估不同测量点位污染的综合指数，最优解的测量点位应该是各类污染气体浓度值相对最低的区域。

参数管理使用AttrDict()方法（具体查看“Cityscapes数据集——参数管理”一节），将该方法写入到USDA库中，方便调用。

\_\_C=usda\_utils.AttrDict()   
args=\_\_C  
  
\_\_C.db=usda\_utils.AttrDict()   
\_\_C.db.UN='postgres'  
\_\_C.db.PW='123456'  
\_\_C.db.DB='AoT\_20220831'  
\_\_C.db.GC='geometry'   
\_\_C.db.db\_info=dict(geom\_col=args.db.GC,myusername=args.db.UN,mypassword=args.db.PW,mydatabase=args.db.DB)

gas\_concentration\_gdf=postSQL2gpd(table\_name="concentration",\*\*args.db.db\_info)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
The data has been read from PostgreSQL database. The table name is concentration.

gas\_concentration\_gdf.head(3)

|  | **timestamp** | **node\_id** | **subsystem** | **sensor** | **parameter** | **value\_hrf\_sum** | **value\_hrf\_count** | **value\_hrf\_average** | **value\_hrf\_min** | **value\_hrf\_max** | **project\_id** | **vsn** | **address** | **lat** | **lon** | **description** | **start\_timestamp** | **end\_timestamp** | **geometry** | **ts** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2020/01/01 00:30:00 | 001e0610ee43 | chemsense | co | concentration | 6.383 | 77 | 0.083 | 0.001 | 0.281 | AoT\_Chicago | 08B | UChicago Chicago IL | 41.788608 | -87.598713 | AoT Chicago (S) [C] {UChicago} | 2018/02/23 00:00:00 | None | POINT (450252.141 4626479.257) | 2020-01-01 00:30:00 |
| **1** | 2020/01/01 00:30:00 | 001e061183f5 | chemsense | co | concentration | 7.711 | 53 | 0.145 | 0.009 | 0.457 | AoT\_Chicago | 0BF | 111th & Michigan Ave Chicago IL | 41.692703 | -87.621020 | AoT Chicago (S) [CP] | 2019/06/24 00:00:00 | None | POINT (448321.777 4615844.593) | 2020-01-01 00:30:00 |
| **2** | 2020/01/01 00:30:00 | 001e061183f5 | chemsense | h2s | concentration | 223.790 | 143 | 1.565 | 1.244 | 1.929 | AoT\_Chicago | 0BF | 111th & Michigan Ave Chicago IL | 41.692703 | -87.621020 | AoT Chicago (S) [CP] | 2019/06/24 00:00:00 | None | POINT (448321.777 4615844.593) | 2020-01-01 00:30:00 |

计算读取的gas\_concentration\_gdf时间序列数据为各个测量点位各类污染气体的年均值，以各类气体污染浓度为决策准则，以测量点位（node\_id）为备选方案。

gas\_grouped=gas\_concentration\_gdf.groupby(['node\_id','sensor']).agg({'value\_hrf\_average':'mean','geometry':'first'})  
gas\_grouuped\_val=gas\_grouped['value\_hrf\_average'].unstack(level=1)  
gas\_grouuped\_val.dropna(axis=0,how='any',inplace=True)  
  
gas\_grouped\_geometry=gas\_grouped['geometry']  
gas\_grouped\_geometry\_series=gas\_grouped\_geometry[~gas\_grouped\_geometry.index.get\_level\_values(0).duplicated(keep='last')].droplevel(level=1)  
  
gas\_concentration\_grouuped\_gdf=gas\_grouuped\_val.join(gas\_grouped\_geometry\_series).set\_geometry('geometry',crs=gas\_concentration\_gdf.crs)  
print(gas\_concentration\_grouuped\_gdf.shape)  
gas\_concentration\_grouuped\_gdf.head(3)

(17, 8)

|  | **co** | **h2s** | **no2** | **o3** | **oxidizing\_gases** | **reducing\_gases** | **so2** | **geometry** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **node\_id** |  |  |  |  |  |  |  |  |
| **001e0610ee43** | 0.100156 | 0.016942 | 0.005921 | 0.002911 | 0.100419 | 0.023512 | 0.155692 | POINT (450252.141 4626479.257) |
| **001e06113107** | 1.381645 | 0.042800 | 0.005717 | 0.002976 | 0.715194 | 0.088307 | 0.304424 | POINT (440722.179 4622392.015) |
| **001e06117b41** | 0.191377 | 0.349793 | 0.014280 | 0.004547 | 1.306005 | 0.419687 | 0.547236 | POINT (448297.079 4626171.925) |

读取芝加哥城的行政区划数据，方便测量单位分布位置的查看。

chicago\_community\_areas=postSQL2gpd(table\_name='chicago\_community\_areas',\*\*args.db.db\_info)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
The data has been read from PostgreSQL database. The table name is chicago\_community\_areas.

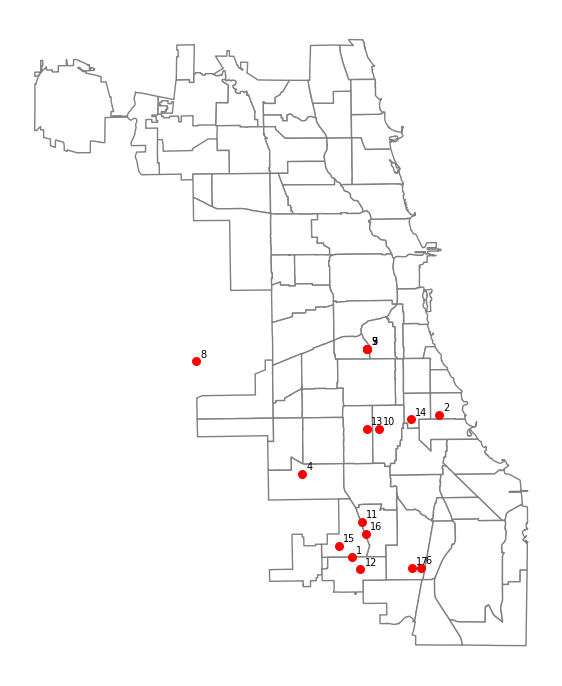
应用上述定义的熵值权重法和TOPSIS算法，计算污染气体浓度下的MCDM问题。从计算结果来看，测量点位001e0611856d为最优方案，各类气体污染浓度相对较小。

entropy\_weight\_gas\_grouuped\_val=usda\_weight.entropy\_weight(gas\_grouuped\_val)  
standardized\_evaluation\_criteria\_gas\_grouuped\_val=usda\_weight.df\_standardized\_evaluation(gas\_grouuped\_val)  
pis={col:0 for col in gas\_grouuped\_val.columns}  
separation\_measure\_gas\_grouuped=usda\_weight.closeness\_pis\_nis(standardized\_evaluation\_criteria\_gas\_grouuped\_val,pis=pis,Wj=entropy\_weight\_gas\_grouuped\_val['Wj'])  
separation\_measure\_gas\_grouuped['node\_id']=gas\_grouuped\_val.index  
separation\_measure\_gas\_grouuped

|  | **Si+** | **Si-** | **Ci+** | **Rank** | **node\_id** |
| --- | --- | --- | --- | --- | --- |
| **0** | 0.003842 | 0.426265 | 0.991068 | 2.0 | 001e0610ee43 |
| **1** | 0.009027 | 0.420979 | 0.979008 | 4.0 | 001e06113107 |
| **2** | 0.025925 | 0.415602 | 0.941282 | 14.0 | 001e06117b41 |
| **3** | 0.016292 | 0.415278 | 0.962251 | 11.0 | 001e06117b44 |
| **4** | 0.008117 | 0.421103 | 0.981090 | 3.0 | 001e0611804d |
| **5** | 0.020310 | 0.416891 | 0.953546 | 13.0 | 001e06118182 |
| **6** | 0.012700 | 0.424827 | 0.970974 | 8.0 | 001e06118295 |
| **7** | 0.012010 | 0.420971 | 0.972261 | 7.0 | 001e061182a2 |
| **8** | 0.016872 | 0.421351 | 0.961500 | 12.0 | 001e061182a7 |
| **9** | 0.009138 | 0.424579 | 0.978931 | 5.0 | 001e061183bf |
| **10** | 0.038781 | 0.407858 | 0.913172 | 15.0 | 001e061183f3 |
| **11** | 0.316724 | 0.272641 | 0.462601 | 17.0 | 001e061183f5 |
| **12** | 0.014791 | 0.424615 | 0.966340 | 9.0 | 001e06118433 |
| **13** | 0.296861 | 0.279669 | 0.485090 | 16.0 | 001e061184a3 |
| **14** | 0.014759 | 0.422157 | 0.966221 | 10.0 | 001e06118501 |
| **15** | 0.011269 | 0.422414 | 0.974015 | 6.0 | 001e0611850f |
| **16** | 0.002697 | 0.425934 | 0.993708 | 1.0 | 001e0611856d |

打印测量点位，并标识污染浓度综合指数（距理想解的排序）。

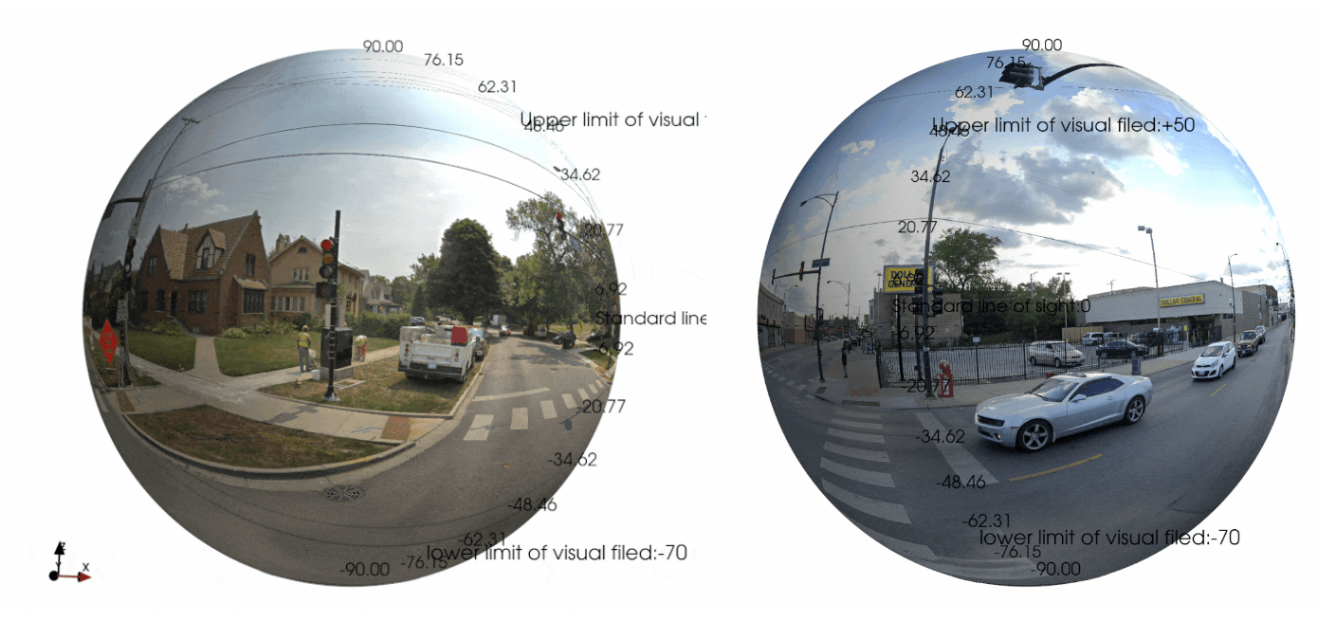
fig, ax=plt.subplots(figsize=(7,15))  
chicago\_community\_areas.plot(ax=ax,color='white', edgecolor='gray')  
gas\_concentration\_grouuped\_gdf.plot(ax=ax,markersize=30,legend=True,legend\_kwds={'loc': 'lower left'},c='r')  
  
for x, y, label in zip(gas\_concentration\_grouuped\_gdf.geometry.x, gas\_concentration\_grouuped\_gdf.geometry.y,separation\_measure\_gas\_grouuped.Rank.astype(int)):  
 ax.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset points",fontsize=7)  
  
ax.axis('off')  
plt.show()



为了查看第1排名（最优方案）和最后排名测量点的场地环境情况，可以下载街道全景图（Street View panorama），一种方式是从[Google Developers Console\_Street View Static API](https://console.cloud.google.com/apis/dashboard?pli=1&project=gsv-streetscape)④ 支付下载；或者从[istreetview](https://istreetview.com/)⑤通过节点坐标获取全景图索引（PanoID），然后用[Street View Download 360](https://svd360.istreetview.com/)⑥辅助工具下载。将下载的等量矩形投影图转换为球面全景图查看，使用[mayavi](https://docs.enthought.com/mayavi/mayavi/)⑦库定义sphere\_panorama\_label()方法实现。

在进一步的分析中，可以结合土地利用、土地覆盖数据，全景图语义分割后视域指标（绿视率等）、分析与测量点位污染浓度综合指数的关系。

pano\_1\_fn='../data/panorama\_AoT/001e0611856d.jpg'  
# pano\_17\_fn='../data/panorama\_AoT/001e061183f5.jpg'  
usda\_vis.sphere\_panorama\_label(pano\_17\_fn)



## 3.1.2 MCDM算法集成

Valdecy Pereira在构建pyDecisionPython工具包时，对MCDM众多算法均给出了[Colab Demo](https://github.com/Valdecy/pyDecision)⑧代码操作案例，方便快速的学习和应用对应算法于实际的实验研究中；同时给出了算法的出处（研究论文），方便查看和理解具体算法的计算过程。下表引入了各类算法的名称，参考论文及输入条件等信息，方便查阅：

| 序号 | 算法名 | 参考论文 | 输入条件 | 备注 |
| --- | --- | --- | --- | --- |
| 1 | AHP（Analytic Hierarchy Process），层次分析法 | How to make a decision: The Analytic Hierarchy Process[10] | 判断矩阵 |  |
| 2 | F-AHP （Fuzzy analytic hierarchy process） ，模糊层次分析法 | A fuzzy AHP approach for supplier selection problem: a case study in a gearmotor company [11] | 判断矩阵（模糊数） |  |
| 3 | ARAS （Additive Ratio Assessment ） | A new additive ratio assessment (ARAS) method in multicriteria decision‐making[12] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 4 | Borda | Mémoire sur les élections au scrutin \*（投票选举简介）[13] | 1.准则类型2.判断矩阵 |  |
| 5 | BWM（Best-worst multi-criteria decision-making method），最优最劣法 | Best-worst multi-criteria decision-making method[2] | 1.最佳准则比较向量2.最差准则比较向量 |  |
| 6 | CODAS （Combinative distance-based assessment），基于组合距离评估 | A new combinative distance-based assessment (codas) method for multi-criteria decision-making[14] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 7 | COPRAS（COmplex PRoportional ASsessment），复杂比例评估 | A complex proportional assessment method for group decision making in an interval-valued intuitionistic fuzzy environment[15] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 8 | CRITIC（CRiteria Importance Through Intercriteria Correlation），基于准则间相关性权重法 | Determining objective weights in multiple criteria problems: the critic method [16] | 1.准则类型2.决策矩阵 |  |
| 9 | DEMATEL（Decision making trial and evaluation laboratory），决策实验室法 | DEMATEL Technique: A Systematic Review of the State-of-the-Art Literature on Methodologies and Applications[17] | 判断（关系）矩阵 |  |
| 10 | Fuzzy DEMATEL （Fuzzy-based Decision making trial and evaluation laboratory）, 模糊决策实验室法 | Developing global managers’ competencies using the fuzzy DEMATEL method[18] | 判断（关系）矩阵（模糊数） |  |
| 11 | EDAS （Distance from Average Solution） | Multi-Criteria Inventory Classification Using a New Method of Evaluation Based on Distance from Average Solution (EDAS)[19] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 12 | Fuzzy EDAS（Fuzzy-based distance from average solution ） | Fuzzy-EDAS (Evaluation Based on Distance from Average Solution) for Material Selection Problems[20] | 1.准则权重（模糊数）2.准则类型3.决策矩阵（模糊数） |  |
| 13 | ELECTRE I | Classement et choix en presence de points de vue multiples[21] | 1. 满意度（一致）阈值$ $ 3.准则权重4.决策矩阵 |  |
| 14 | ELECTRE I\_s | The Electre Methodology [22] | 1. 向量系数2.准则权重 3.决策矩阵 |  |
| 15 | ELECTRE I\_v | ’’ | 1. 满意度（一致）阈值2. 不满意度（不一致）阈值3.不满意度比较向量 4.准则权重5.决策矩阵 |  |
| 16 | ELECTRE II | ’’ | 1. 满意度（一致）阈值$c-,c,c+ d-,d+$ 3.准则权重4.决策矩阵 |  |
| 17 | ELECTRE III | Un algorithme de classemeuts fonde sur une representation floue des preferences en presence de cr1teres multiples[23] | 1. 向量系数2.准则权重 3.决策矩阵 |  |
| 18 | ELECTRE IV | The Electre Methodology [22] | 1. 向量系数 2.决策矩阵 |  |
| 19 | ELECTRE Tri-B | Aide multicritère à la décision dans le cadre de la problématique du tri Concepts, méthodes et applications [24] | 1. 向量系数2.准则权重 3.决策矩阵 |  |
| 20 | GRA （Grey Relation Analysis)），灰色关联度分析 | Control problems of grey systems[25] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 21 | IDOCRIW （Integrated Determination of Objective CRIteria Weights） | New Methods and Applications in Multiple Attribute Decision Making (MADM) [26] | 1.准则类型2.决策矩阵 |  |
| 22 | MABAC （Multi-Attributive Border Approximation area Comparison）， 多属性边界近似区域比较 | The selection of transport and handling resources in logistics centers using Multi-Attributive Border Approximation area Comparison (MABAC)[27] | 1.准则类型2.决策矩阵 |  |
| 23 | MAUT （Multi-attribute utility theory），多属性效用理论 | Multi-attribute utility theory: models and assessment procedures[28] | 1.准则权重2.准则类型3. 效用参数4.决策矩阵 |  |
| 24 | MOORA | The MOORA method and its application to privatization in a transition economy[29] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 25 | MOOSRA （ Multi-objective optimisation on the basis of simple ratio analysis），基于简单比率分析的多目标优化 | Green cutting fluid selection using moosra method[30] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 26 | MULTIMOORA （Multi (Full Multiplicative Form) Multi-Objective Optimiza- tion by Ratio analysis） | Project management by multimooraas an instrument f or transition economies[31] | 1.准则类型2.决策矩阵 |  |
| 27 | PROMETHEE I | Promethee methods[32] | 1. 向量系数2.准则权重 3. 计算类型（）3.决策矩阵 |  |
| 28 | PROMETHEE II | ’’ | 1. 向量系数2.准则权重 3. 计算类型（）3.决策矩阵 |  |
| 29 | PROMETHEE III | PROMETHEE I-II-III Methods[33] | 1. 向量系数2.准则权重 3. 计算类型（）3.决策矩阵 |  |
| 30 | PROMETHEE IV | Promethee methods[32] | 1. 向量系数2.准则权重 3. 计算类型（）3.决策矩阵 |  |
| 31 | PROMETHEE V | ’’ | 1. 向量系数2.准则权重 3. 计算类型（）4.约束1（所选条件的最大数量）5.约束2（准则成本（cost）和预算（budget））6.约束3（选定条件的禁止集（forbidden））7.决策矩阵 |  |
| 32 | PROMETHEE VI | ’’ | 1. 向量系数2.准则权重 (区间： 权重低值，权重高值)3. 计算类型（）3.决策矩阵 |  |
| 33 | PROMETHEE Gaia | ’’ | 1. 向量系数2.准则权重 3. 计算类型（）3.决策矩阵 |  |
| 34 | SAW（Simple Additive Weighting ） | Simple Additive Weighting (SAW) method in Determining Beneficiaries of Foundation Benefits[34] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 35 | SMART（Simple Multi-Attribute Rating Technique） | Decision Aids for Selection Problems[35] | 1.得分（4（最差）-10（最好））2.决策阈值（低值向量和高值向量）3.准则类型4.决策矩阵 |  |
| 36 | TOPSIS（Technique for Order Performance by Similarity to Ideal Solution），理想解法 | A Reconciliation Among Discrete Compromise Solutions[36] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 37 | Fuzzy TOPSIS （Fuzzy-based Technique for Order Performance by Similarity to Ideal Solution）模糊理想解法 | Fuzzy TOPSIS: A General View[37] | 1.准则权重（模糊数）2.准则类型3.决策矩阵（模糊数） |  |
| 38 | VIKOR （ VIseKriterijumska Optimizacija I Kompromisno Resenje ） | VIseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR)[38] | 1.准则权重2.准则类型3.决策矩阵 |  |
| 39 | Fuzzy VIKOR | Fuzzy VIKOR with an application to water resources planning[39] | 1.准则权重（模糊数）2.准则类型3.决策矩阵（模糊数） |  |
| 40 | WINGS | Weighted Inﬂuence Non-linear Gauge System (WINGS) – an analysis method for the systems of interrelated components[40] | 判断矩阵（Scale: 0 (No Influence), 1 (Low Influence), 2 (Medium Influence), 3 (High Influence), 4 (Very High Influence)） |  |
| 41 | WSM, WPM, WASPAS（Weighted Aggregated Sum Product Assessment） | Weighted Aggregated Sum Product Assessment (WASPAS)[41] | 1.准则权重2.准则类型3.决策矩阵 |  |

为理解相关算法的具体过程，再现 AHP、F-AHP、 ARAS、 BWM、 DEMATEL、IDOCRIW和ELECTRE-I等7种算法。

### 3.1.2.1 再现 AHP 算法

再现Thomas L. Saaty在论文中给出的第一个案例（购房选择问题），阐释AHP算法。为了方便应用 AHP 算法，定义AHP类可以方便构建判断矩阵，计算权重矩阵和一致性指数，及总的得分。

对于这个购房选择问题，包括8个决策准则，由变量criteria给出；3个备选方案，由变量alternatives给出。首先是要构建两类两两比较矩阵（判断矩阵），一个是决策准则对于分析目标的重要性排序，为决策准则之间的两两比较；另一个是对于每一个决策准则，例如对于size\_of\_house而言，3个备选方案比较这一size\_of\_house这一决策准则，哪个备选方案的房子相对会大一些。为了方便和规范比较方式和比较值，作者给出了度量标度，如表：

| 标度（Intensity of importance on an absolute scale） | 定义（Definition） | 解释（Explanation） |
| --- | --- | --- |
| 1 | Equal importance（同样重要性） | Two activities contribute equally to the objective |
| 3 | Moderate importance of one over another（一个因素比另一个因素稍微重要） | Experience and judgment strongly favor one activity over another |
| 5 | Essential or strong importance（一个因素比另一个因素明显重要） | Experience and judgement strongly favor one activity over another |
| 7 | Very strong importance（一个因素比另一个因素强烈重要） | An activity is strongly favored and its dominance demonstrated in practice |
| 9 | Extreme importance（一个因素比另一个因素极端重要） | The evidence favoring one activity over another is of tile highest possible order of affirmation |
| 2，4，6，8 | Intermediate values between the two adjacent judgments（上述两相邻判断的中值） | When compromise is needed |
| 倒数（Reciprocals） |  | If activity i has one of the above numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i （如果因素 i 与因素 j 相比分配了上述数字之一，则 j 与 i 相比为倒数），例如$a\_{ij}=3 a\_{ji}=1/3 $ |
| 有理数（Rationals ） | Ratios arising from the scale | If consistency were to be forced by obtaining n numerical values to span the matrix |

为了方便判断矩阵的输入，定义了criteria\_pairs\_lst()和alternative\_pairs()方法生成criteria\_pairs\_lst和alternative\_pairs\_lst类属性，可以查看对于定义的类AHP两两比较输入的顺序。输入时为字符串形式输入，是为了应用fractions库提供的Fraction方法显示为分数形式，而不是浮点数。因为判断矩阵为正互反矩阵（reciprocal matrix），如，有，因此倒数部分可以书写代码自动生成。

criteria=['size\_of\_house','location\_to\_bus\_line','neighborhood','age\_of\_house','yard\_space','modern\_facilities','general\_condition','financing\_available']  
alternatives=['A','B','C']   
  
choosing\_best\_house=usda\_weight.AHP(criteria,alternatives)

choosing\_best\_house.criteria\_pairs\_lst

[('size\_of\_house', 'location\_to\_bus\_line'),  
 ('size\_of\_house', 'neighborhood'),  
 ('size\_of\_house', 'age\_of\_house'),  
 ('size\_of\_house', 'yard\_space'),  
 ('size\_of\_house', 'modern\_facilities'),  
 ('size\_of\_house', 'general\_condition'),  
 ('size\_of\_house', 'financing\_available'),  
 ('location\_to\_bus\_line', 'neighborhood'),  
 ('location\_to\_bus\_line', 'age\_of\_house'),  
 ('location\_to\_bus\_line', 'yard\_space'),  
 ('location\_to\_bus\_line', 'modern\_facilities'),  
 ('location\_to\_bus\_line', 'general\_condition'),  
 ('location\_to\_bus\_line', 'financing\_available'),  
 ('neighborhood', 'age\_of\_house'),  
 ('neighborhood', 'yard\_space'),  
 ('neighborhood', 'modern\_facilities'),  
 ('neighborhood', 'general\_condition'),  
 ('neighborhood', 'financing\_available'),  
 ('age\_of\_house', 'yard\_space'),  
 ('age\_of\_house', 'modern\_facilities'),  
 ('age\_of\_house', 'general\_condition'),  
 ('age\_of\_house', 'financing\_available'),  
 ('yard\_space', 'modern\_facilities'),  
 ('yard\_space', 'general\_condition'),  
 ('yard\_space', 'financing\_available'),  
 ('modern\_facilities', 'general\_condition'),  
 ('modern\_facilities', 'financing\_available'),  
 ('general\_condition', 'financing\_available')]

按照上述criteria\_pairs\_lst对顺序确定两两比较的标度，通过pairwise\_comparison\_matrix()方法构建决策准则自身之间的比较矩阵，其中参数alternatives\_criteria为确定生成两类判断矩阵的哪类。

criteria\_pairwise\_vals=['5','3','7','6','6','1/3','1/4','1/3','5','3','3','1/5','1/7','6','3','4','6','1/5','1/3','1/4','1/7','1/8','1/2','1/5','1/6','1/5','1/6','1/2']  
criteria\_global\_pairwise\_comparison\_matrix=choosing\_best\_house.pairwise\_comparison\_matrix(criteria\_pairwise\_vals,alternatives\_criteria='A')  
criteria\_global\_pairwise\_comparison\_matrix

|  | **size\_of\_house** | **location\_to\_bus\_line** | **neighborhood** | **age\_of\_house** | **yard\_space** | **modern\_facilities** | **general\_condition** | **financing\_available** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **size\_of\_house** | 1 | 5 | 3 | 7 | 6 | 6 | 1/3 | 1/4 |
| **location\_to\_bus\_line** | 1/5 | 1 | 1/3 | 5 | 3 | 3 | 1/5 | 1/7 |
| **neighborhood** | 1/3 | 3 | 1 | 6 | 3 | 4 | 6 | 1/5 |
| **age\_of\_house** | 1/7 | 1/5 | 1/6 | 1 | 1/3 | 1/4 | 1/7 | 1/8 |
| **yard\_space** | 1/6 | 1/3 | 1/3 | 3 | 1 | 1/2 | 1/5 | 1/6 |
| **modern\_facilities** | 1/6 | 1/3 | 1/4 | 4 | 2 | 1 | 1/5 | 1/6 |
| **general\_condition** | 3 | 5 | 1/6 | 7 | 5 | 5 | 1 | 1/2 |
| **financing\_available** | 4 | 7 | 5 | 8 | 6 | 6 | 2 | 1 |

假设一个判断矩阵为：，其中，且，但产生了矛盾，因此提出对于一个正互反矩阵如果没有上述矛盾则称之为一致矩阵（consistent matrix），一致矩阵是正互反矩阵的特例。对于一致矩阵有，且矩阵各行（各列）之间成倍数关系，例如对于上述假设矩阵*aAB*​×*aBC*​=*aAC*，即。在构造判断矩阵时，难免会出现类似上述的矛盾，因此需要检验构造的判断矩阵和一致矩阵是否有太大差别，只要该矩阵向一致性矩阵靠拢，这个差距参数一致性比例（Consistency Ratio，CR）不高于10%，则可以接受这个权重估计。对于CR公式为： CIRI

RIRI[0, 0, 0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49, 1.51, 1.48, 1.56, 1.57, 1.59]λmax*nnr(A)=1，并当且仅当λ*max=n ，因此定义CI为 (λmax*​−*n*)/(*n*−1)*n的差距就越大。

权重值的计算包括算术平均法求权重和几何平均法求权重，有weight()方法的wd参数判断。假设判断矩阵为，算术平均法求权重的公式为：，即判断矩阵按列归一化（每个元素除以其所在列的和）后，按行求和，并除以矩阵的阶；几何平均法求权重公式为：，即将矩阵按行相乘得到一个新的列向量后开次方，并进行归一化得到权重值。

迁移pyDecision库中ahp方法为USDA库类AHP中的函数，直接调用该方法计算权重和CI值及CR值，返回值仅包含CR值。计算结果如下，CR值约为0.163，作者给出的结果为0.169，实际超出了0.1的阈值界限。

gp\_weight,gp\_cr=choosing\_best\_house.weight(criteria\_global\_pairwise\_comparison\_matrix)  
print(gp\_weight,'\n',gp\_cr)

[0.1743749238483884 0.06608449083968755 0.17015215186835408 0.01889929722237991 0.0352504763592837 0.04445701659354435 0.1784887930700104 0.3122928501983515]   
 0.16395000832304404

同样的方法批量构建每一决策准则不同备选方案的判断矩阵，并计算权重值和CI值。

choosing\_best\_house.alternative\_pairs\_lst

[('A', 'B'), ('A', 'C'), ('B', 'C')]

local\_pairwise\_vals\_dict=dict(size\_of\_house=['6','8','4'],   
 location\_to\_bus\_line=['7','1/5','1/8'],   
 neighborhood=['8','6','1/4'],   
 age\_of\_house=['1','1','1'],   
 yard\_space=['5','4','1/3'],   
 modern\_facilities=['8','6','1/5'],   
 general\_condition=['1/2','1/2','1'],   
 financing\_available=['1/7','1/5','3'])  
local\_pairwise\_matrix\_dict={}  
for k,v in local\_pairwise\_vals\_dict.items():  
 local\_pairwise\_matrix\_dict[k]=choosing\_best\_house.pairwise\_comparison\_matrix(v,alternatives\_criteria='C')  
local\_pairwise\_matrix\_dict

{'size\_of\_house': A B C  
 A 1 6 8  
 B 1/6 1 4  
 C 1/8 1/4 1,  
 'location\_to\_bus\_line': A B C  
 A 1 7 1/5  
 B 1/7 1 1/8  
 C 5 8 1,  
 'neighborhood': A B C  
 A 1 8 6  
 B 1/8 1 1/4  
 C 1/6 4 1,  
 'age\_of\_house': A B C  
 A 1 1 1  
 B 1 1 1  
 C 1 1 1,  
 'yard\_space': A B C  
 A 1 5 4  
 B 1/5 1 1/3  
 C 1/4 3 1,  
 'modern\_facilities': A B C  
 A 1 8 6  
 B 1/8 1 1/5  
 C 1/6 5 1,  
 'general\_condition': A B C  
 A 1 1/2 1/2  
 B 2 1 1  
 C 2 1 1,  
 'financing\_available': A B C  
 A 1 1/7 1/5  
 B 7 1 3  
 C 5 1/3 1}

local\_weight\_rc\_dict={}  
for k,v in local\_pairwise\_matrix\_dict.items():  
 local\_weight\_rc\_dict[k]={k:v for k,v in zip(['weight','rc'],choosing\_best\_house.weight(v))}  
   
local\_weight\_rc\_dict

{'size\_of\_house': {'weight': array([0.7390547902227547, 0.19155186674652747, 0.06939334303071788], dtype=object),  
 'rc': 0.12062025335167682},  
 'location\_to\_bus\_line': {'weight': array([0.25041136463361124, 0.06003181219833261, 0.6895568231680563], dtype=object),  
 'rc': 0.2220676402473212},  
 'neighborhood': {'weight': array([0.7390547902227547, 0.06939334303071788, 0.19155186674652747], dtype=object),  
 'rc': 0.12062025335167682},  
 'age\_of\_house': {'weight': array([0.3333333333333333, 0.3333333333333333, 0.3333333333333333], dtype=object),  
 'rc': 0.0},  
 'yard\_space': {'weight': array([0.6650702426564495, 0.10384738186462324, 0.2310823754789272], dtype=object),  
 'rc': 0.07495638655172662},  
 'modern\_facilities': {'weight': array([0.7263184843830004, 0.0653268475849121, 0.20835466803208738], dtype=object),  
 'rc': 0.17731739051985576},  
 'general\_condition': {'weight': array([0.20000000000000004, 0.4000000000000001, 0.4000000000000001], dtype=object),  
 'rc': 0.0},  
 'financing\_available': {'weight': array([0.07377210603017055, 0.6433888691953209, 0.2828390247745087], dtype=object),  
 'rc': 0.056475713035283204}}

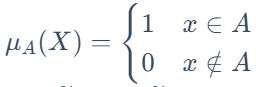
最后建立weight\_matrix()方法汇总决策准则之间和各准则对备选方案之间的权重值，并计算最后基于决策准则的备选方案得分global\_priority。从计算结果得知，备选方案A得分最高，为购房选择问题的最优答案。

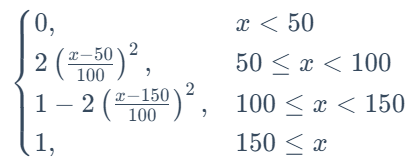
weight\_A=choosing\_best\_house.weight\_matrix(gp\_weight,local\_weight\_rc\_dict)  
weight\_A

|  | **size\_of\_house** | **location\_to\_bus\_line** | **neighborhood** | **age\_of\_house** | **yard\_space** | **modern\_facilities** | **general\_condition** | **financing\_available** | **global\_priority** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Wj** | 0.174375 | 0.066084 | 0.170152 | 0.018899 | 0.03525 | 0.044457 | 0.178489 | 0.312293 | NaN |
| **A** | 0.739055 | 0.250411 | 0.739055 | 0.333333 | 0.66507 | 0.726318 | 0.2 | 0.073772 | 0.391943 |
| **B** | 0.191552 | 0.060032 | 0.069393 | 0.333333 | 0.103847 | 0.065327 | 0.4 | 0.643389 | 0.334362 |
| **C** | 0.069393 | 0.689557 | 0.191552 | 0.333333 | 0.231082 | 0.208355 | 0.4 | 0.282839 | 0.273695 |

### 3.1.2.2 再现 F-AHP 算法

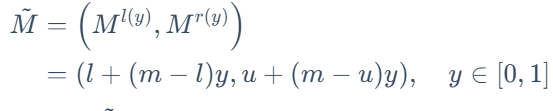
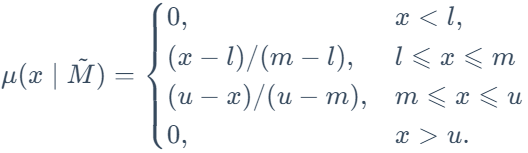
再现Ayhan, M. B.在论文中给出的案例（最佳供应商选择问题），阐释F-AHP算法。为了方便应用 F-AHP 算法，定义F\_AHP类可以方便构建判断矩阵，计算权重矩阵和一致性指数，及总的得分。

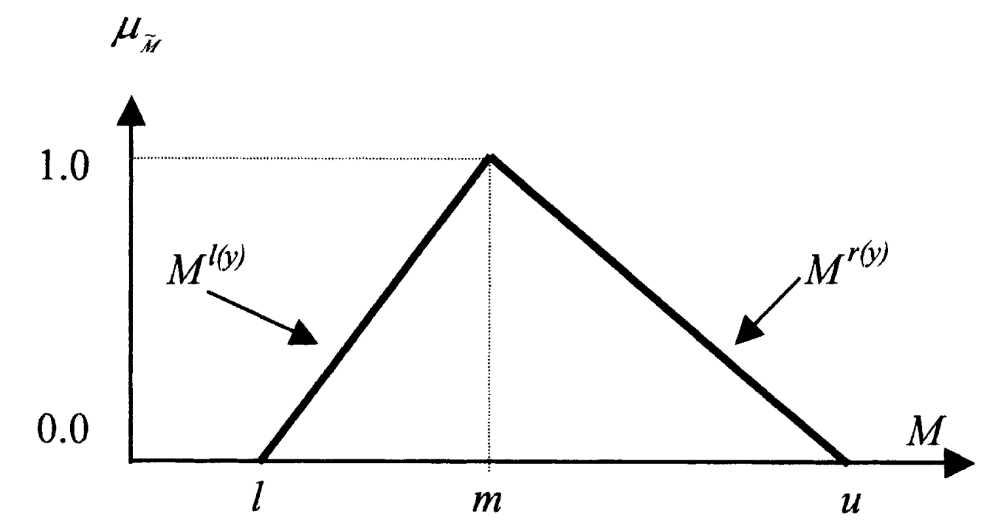
对于这个最佳供应商选择问题，包括5个决策准则，由变量criteria给出；3个备选方案，由变量alternatives给出。F-AHP 方法与 AHP 最大的差异是判断矩阵的构建使用的是三角模糊数（triangular fuzzy number，TFN）。对 TFN 的理解可以从对论域、模糊集、隶属函数等概念的理解引入。论域是论题所包括同类事物的非空总和，是将讨论的对象限制在一定范围内，用表示。模糊集合是用来表达模糊性概念的集合，又称为模糊集（或模糊子集）。对于普通集合，元素 要么属于集合，要么不属于，表达为：  Ux*μ*(*μ*∈[0,1])  x ∈ x  F(U)U *μA*​(*x*):*U*→[0,1] *μA*​(*x*)x(x∈ A) *μA*​(*x*) * Ux* UF(U)

u\_A(x)= 

*μA*(*x*)

UA是不健康空气对应的模糊集。

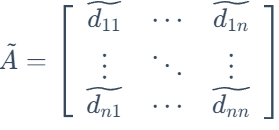
对于任意一个模糊集，都对应一个隶属函数，确定隶属函数通常参考概率论中的分布函数，称为模糊分布函数，例如正态分布、梯形分布、K次抛物线分布、柯西（Cauchy）分布和S型函数（sigmoid function）分布等。这些函数论域为实数，带有参数，值域为[0,1]。在AHP中，两两比较判断矩阵通常没有考虑人的判断模糊性，仅为隶属于给定的标度，因此引入模糊数，使用TFN形式的三值判断（三角模糊数），最低可能值（smallest possible value，记作）、最可能值（most promising value，记作）和最高可能值（largest possible value ，记作）。当时，为一个非模糊数。每个 TFN 在其左侧和右侧都有线性表示，因此其隶属函数可以定义为[43]：l(y)r(y)，如图[43]表示：



从上图可知，当时，完全属于lulu 。

判断矩阵中为了考虑人的模糊性，使用TFN，作者同样给出了度量标度，如表：

| Saaty scale（标度） | Definition （定义） | Fuzzy Triangular Scale （TFN） |
| --- | --- | --- |
| 1 | Equally important (Eq. Imp.) （同等重要） | (1, 1, 1) |
| 3 | Weakly important (W. Imp.) （稍微重要） | (2, 3, 4) |
| 5 | Fairly important (F. Imp.) （重要） | (4, 5, 6) |
| 7 | Strongly important (S. Imp.) （明显重要） | (6, 7, 8) |
| 9 | Absolutely important (A. Imp.) （非常重要） | (9, 9, 9) |
| 2 | The intermittent values between two adjacent scales （中间重要性） | (1, 2, 3) |
| 4 | ” | (3, 4, 5) |
| 5 | ” | (5, 6, 7) |
| 6 | ” | (7, 8, 9) |

第1步，定义决策准则之间的判断矩阵。基于TFN的判断矩阵可以表达为：，式中，    。定义pairwise\_comparison\_TFN\_matrix()方法实现基于TFN判断矩阵的构建。

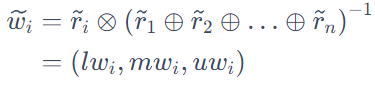
criteria=['quality','origin','cost','delivery','after\_sales']  
alternatives=['A1','A2','A3']   
  
best\_supplier=usda\_weight.F\_AHP(criteria,alternatives)

best\_supplier.criteria\_pairs\_lst

[('quality', 'origin'),  
 ('quality', 'cost'),  
 ('quality', 'delivery'),  
 ('quality', 'after\_sales'),  
 ('origin', 'cost'),  
 ('origin', 'delivery'),  
 ('origin', 'after\_sales'),  
 ('cost', 'delivery'),  
 ('cost', 'after\_sales'),  
 ('delivery', 'after\_sales')]

criteria\_pairwise\_TFN=[('1','1','1'),('4','5','6'),('6','7','8'),('4','5','6'),('4','5','6'),('6','7','8'),('6','7','8'),('1/4','1/3','1/2'),('2','3','4'),('1/6','1/5','1/4')]  
criteria\_pairwise\_comparison\_TFN\_matrix=best\_supplier.pairwise\_comparison\_TFN\_matrix(criteria\_pairwise\_TFN,alternatives\_criteria='C')  
criteria\_pairwise\_comparison\_TFN\_matrix

|  | **quality** | **origin** | **cost** | **delivery** | **after\_sales** |
| --- | --- | --- | --- | --- | --- |
| **quality** | (1, 1, 1) | (1, 1, 1) | (4, 5, 6) | (6, 7, 8) | (4, 5, 6) |
| **origin** | (1, 1, 1) | (1, 1, 1) | (4, 5, 6) | (6, 7, 8) | (6, 7, 8) |
| **cost** | (1/6, 1/5, 1/4) | (1/6, 1/5, 1/4) | (1, 1, 1) | (1/4, 1/3, 1/2) | (2, 3, 4) |
| **delivery** | (1/8, 1/7, 1/6) | (1/8, 1/7, 1/6) | (2, 3, 4) | (1, 1, 1) | (1/6, 1/5, 1/4) |
| **after\_sales** | (1/6, 1/5, 1/4) | (1/8, 1/7, 1/6) | (1/4, 1/3, 1/2) | (4, 5, 6) | (1, 1, 1) |

第2步，计算决策准则之间的权重和一致性比例CR值。采用几何平均法求权重，首先计算每个决策准则模糊判断矩阵的几何平均值，公式为：；进而计算每个的向量和及其-1次幂，并递增排序（reverse）（TFN形式）；计算第个决策准则的模糊权重 

。

对于F-AHP算法直接迁移pyDecision库中fuzzy\_ahp\_method方法为USDA库类F\_AHP中的函数进行计算。

fuzzy\_weights, defuzzified\_weights, normalized\_weights, cr=best\_supplier.weight(criteria\_pairwise\_comparison\_TFN\_matrix)  
print(f\_w,'\n', d\_w,'\n', n\_w,'\n' ,cr)

[(0.30387157227617195, 0.3849442719555282, 0.4827648944504069), (0.3295401422029697, 0.41174038735956564, 0.5113560858232844), (0.051852455454028286, 0.07197841021214509, 0.10262169573734545), (0.04261628425875972, 0.05680430962243532, 0.07596205129037176), (0.05623252420717982, 0.07453262085032564, 0.10262169573734545)]   
 [0.3905269128940357, 0.4175455384619399, 0.07548418713450628, 0.0584608817238556, 0.07779561359828364]   
 [0.3829396778153188, 0.40943338011437996, 0.07401766522882969, 0.057325092005137, 0.07628418483633463]   
 0.25672257239611473

第3步，同样的方式计算每一决策准则不同备选方案的模糊判断矩阵，并计算权重值和CI值。因为作者仅给出了quality决策准则模糊矩阵，其它准则不同于该文实验参数，因此算法再现结果不同于原文实验。

best\_supplier.alternative\_pairs\_lst

[('A1', 'A2'), ('A1', 'A3'), ('A2', 'A3')]

alternatives\_vals\_dict=dict(quality=[('1/6','1/5','1/4'),('1/9','1/9','1/9'),('1/4','1/3','1/2')],  
 origin=[('2','3','4'),('6','7','8'),('4','5','6')],  
 cost=[('4','5','6'),('1/6','1/5','1/4'),('1/4','1/3','1/2')],  
 delivery=[('9','9','9'),('6','7','8'),('1/8','1/7','1/7')],  
 after\_sales=[('1/8','1/7','1/6'),('1/4','1/3','1/2'),('2','3','4')])  
alternatives\_matrix\_dict={}  
for k,v in alternatives\_vals\_dict.items():  
 alternatives\_matrix\_dict[k]=best\_supplier.pairwise\_comparison\_TFN\_matrix(v,alternatives\_criteria='A')  
alternatives\_matrix\_dict

{'quality': A1 A2 A3  
 A1 (1, 1, 1) (1/6, 1/5, 1/4) (1/9, 1/9, 1/9)  
 A2 (4, 5, 6) (1, 1, 1) (1/4, 1/3, 1/2)  
 A3 (9, 9, 9) (2, 3, 4) (1, 1, 1),  
 'origin': A1 A2 A3  
 A1 (1, 1, 1) (2, 3, 4) (6, 7, 8)  
 A2 (1/4, 1/3, 1/2) (1, 1, 1) (4, 5, 6)  
 A3 (1/8, 1/7, 1/6) (1/6, 1/5, 1/4) (1, 1, 1),  
 'cost': A1 A2 A3  
 A1 (1, 1, 1) (4, 5, 6) (1/6, 1/5, 1/4)  
 A2 (1/6, 1/5, 1/4) (1, 1, 1) (1/4, 1/3, 1/2)  
 A3 (4, 5, 6) (2, 3, 4) (1, 1, 1),  
 'delivery': A1 A2 A3  
 A1 (1, 1, 1) (9, 9, 9) (6, 7, 8)  
 A2 (1/9, 1/9, 1/9) (1, 1, 1) (1/8, 1/7, 1/7)  
 A3 (1/8, 1/7, 1/6) (7, 7, 8) (1, 1, 1),  
 'after\_sales': A1 A2 A3  
 A1 (1, 1, 1) (1/8, 1/7, 1/6) (1/4, 1/3, 1/2)  
 A2 (6, 7, 8) (1, 1, 1) (2, 3, 4)  
 A3 (2, 3, 4) (1/4, 1/3, 1/2) (1, 1, 1)}

alternatives\_weight\_cr\_dict={}  
for k,v in alternatives\_matrix\_dict.items():  
 alternatives\_weight\_cr\_dict[k]={k:v for k,v in zip(['fuzzy\_weights', 'defuzzified\_weights', 'normalized\_weights', 'cr'],best\_supplier.weight(v))}  
   
alternatives\_weight\_cr\_dict

{'quality': {'fuzzy\_weights': [(0.052420300418135744,  
 0.06294120510225566,  
 0.07794836695752445),  
 (0.1981363198158667, 0.2654333418834977, 0.3712059591227326),  
 (0.5192640550376543, 0.6716254530142467, 0.8498494966569238)],  
 'defuzzified\_weights': [0.06443662415930529,  
 0.27825854027403235,  
 0.6802463349029416],  
 'normalized\_weights': [0.06299150459837054,  
 0.2720180386215403,  
 0.6649904567800892],  
 'cr': 0.038802175909399925},  
 'origin': {'fuzzy\_weights': [(0.46123125532773124,  
 0.6491180046313251,  
 0.8906502333108046),  
 (0.20146130719005298, 0.27895456548641834, 0.4046047199899532),  
 (0.05543421498406391, 0.07192742988225646, 0.09725683059164322)],  
 'defuzzified\_weights': [0.6669998310899535,  
 0.2950068642221415,  
 0.07487282515265453],  
 'normalized\_weights': [0.643276116391026,  
 0.28451411991425357,  
 0.07220976369472036],  
 'cr': 0.07101595764337375},  
 'cost': {'fuzzy\_weights': [(0.19287686201378226,  
 0.258284994374495,  
 0.355472493169515),  
 (0.07654323341525081, 0.10472943388074786, 0.15526691289199787),  
 (0.44157778201234055, 0.636985571744757, 0.8957345536057669)],  
 'defuzzified\_weights': [0.2688781165192641,  
 0.11217986006266552,  
 0.6580993024542882],  
 'normalized\_weights': [0.25874631486837024,  
 0.10795272508383758,  
 0.6333009600477921],  
 'cr': 0.4768968318277456},  
 'delivery': {'fuzzy\_weights': [(0.6857178581616268,  
 0.7607597376820608,  
 0.8359451389785031),  
 (0.04360838881170998, 0.04804930980478052, 0.050499463168466624),  
 (0.17352027382598298, 0.19119095251315868, 0.22116336550036308)],  
 'defuzzified\_weights': [0.7608075782740635,  
 0.04738572059498571,  
 0.19529153061316826],  
 'normalized\_weights': [0.7581655007845295,  
 0.0472211628943469,  
 0.1946133363211237],  
 'cr': 0.2922872691708499},  
 'after\_sales': {'fuzzy\_weights': [(0.0646575791757096,  
 0.08794620881905604,  
 0.1285391955362206),  
 (0.4699624744114258, 0.6694168694489877, 0.9342848767797738),  
 (0.1629268900774419, 0.24263692173195622, 0.3707711990595482)],  
 'defuzzified\_weights': [0.09371432784366207,  
 0.6912214068800623,  
 0.25877833695631547],  
 'normalized\_weights': [0.08978927312229538,  
 0.6622708514099278,  
 0.24793987546777696],  
 'cr': 0.030424894517038763}}

第4步，计算最后基于决策准则的备选方案得分。

local\_w\_stack=np.stack([v['normalized\_weights'] for v in alternatives\_weight\_cr\_dict.values()],axis=-1)  
weight\_A=best\_supplier.weight\_matrix(normalized\_weights,local\_w\_stack)  
weight\_A

|  | **quality** | **origin** | **cost** | **delivery** | **after\_sales** | **global\_priority** |
| --- | --- | --- | --- | --- | --- | --- |
| **Wj** | 0.382940 | 0.409433 | 0.074018 | 0.057325 | 0.076284 | NaN |
| **A1** | 0.062992 | 0.643276 | 0.258746 | 0.758166 | 0.089789 | 0.356964 |
| **A2** | 0.272018 | 0.284514 | 0.107953 | 0.047221 | 0.662271 | 0.281874 |
| **A3** | 0.664990 | 0.072210 | 0.633301 | 0.194613 | 0.247940 | 0.361162 |

### 3.1.2.3 再现 ARAS 算法

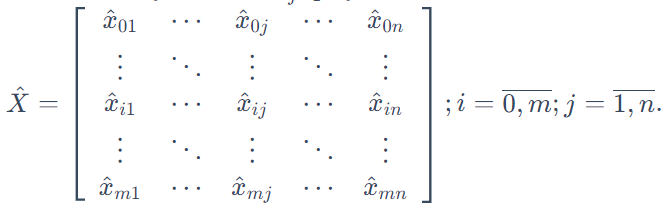
再现Zavadskas, E. K.等人在论文中给出的案例（办公室微气候评估），阐释 ARAS 算法。直接迁移pyDecision库提供的aras\_method方法。

ARAS 算法基本同于一般基于权重计算得分的方法，其中决策准则权重值文中为专家打分获得；类似TOPSIS算法，考虑到决策准则的优化方向，即测量值是越大越好，还是越小越好；决策矩阵测量值根据决策准则优化方向计算方式有所调整。具体流程如下：

第1步，构建决策矩阵（decision-making matrix，DMM），公式为： ，式中，为备选方案索引，为决策准则索引，为第个决策准则值下第个备选方案的测量值。为第个决策准则的优化值，如果不知优化值，则可以通过下述方式确定，。

第2步，为了比较不同单位（unit）、不同尺度(dimentions)的决策准则测量值，需要标准化为无量纲（dimensionless）的决策矩阵。优选值为最大值（maxima）的标准化公式为：；优选值为最小值（minima）的标准化公式为：。标准化后的决策矩阵可以表示为：。

第3步，定义归一化加权决策矩阵。决策准则的权重满足，且 w\_jj

。

第4步，确定最优解。对每一备选方案对应的决策准则归一化加权值求和，即，为备选方案得分。备选方案的效用度（degree of the alternative utility），可以通过计算，式中，为理想情况下得分值最高备选方案的值，通过与最高值比较确定效用度。

将办公室微气候评估数据写入到USDA数据集模块datasets下方便调用。读取该数据，包括分析所用的测量值决策矩阵、决策准则权重值和优化方向列表。

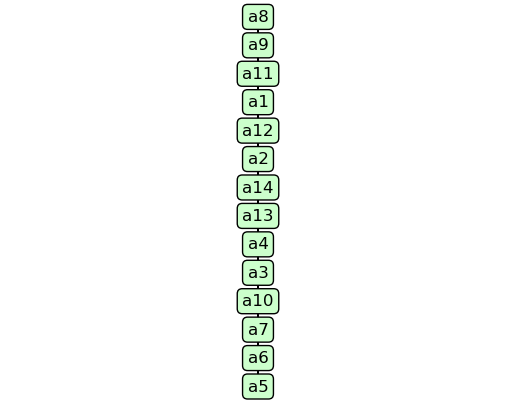
microclimate\_in\_office\_rooms=usda\_datasets.load\_microclimate\_in\_office\_rooms()  
microclimate\_in\_office\_rooms

{'data': amount \_of\_air\_per\_head relative\_air\_humidity air\_temperature \  
 1 7.6 46.0 18.0   
 2 5.5 32.0 21.0   
 3 5.3 32.0 21.0   
 4 5.7 37.0 19.0   
 5 4.2 38.0 19.0   
 6 4.4 38.0 19.0   
 7 3.9 42.0 16.0   
 8 7.9 44.0 20.0   
 9 8.1 44.0 20.0   
 10 4.5 46.0 18.0   
 11 5.7 48.0 20.0   
 12 5.2 48.0 20.0   
 13 7.1 49.0 19.0   
 14 6.9 50.0 16.0   
   
 illumination\_during\_work\_hours rate\_of\_air\_flow dew\_point   
 1 390.0 0.10 11.0   
 2 360.0 0.05 11.0   
 3 290.0 0.05 11.0   
 4 270.0 0.05 9.0   
 5 240.0 0.10 8.0   
 6 260.0 0.10 8.0   
 7 270.0 0.10 5.0   
 8 400.0 0.05 6.0   
 9 380.0 0.05 6.0   
 10 320.0 0.10 7.0   
 11 320.0 0.05 11.0   
 12 310.0 0.05 11.0   
 13 280.0 0.10 12.0   
 14 250.0 0.05 10.0 ,  
 'name': 'microclimate\_in\_office\_rooms.pickle',  
 'measurement\_units': ['m3/h', '%', '°C', 'lx', 'm/s', '°C'],  
 'optimisation\_direction': ['max', 'max', 'max', 'max', 'min', 'min'],  
 'weight\_of\_criteria': [0.21, 0.16, 0.26, 0.17, 0.12, 0.08],  
 'optimal\_value': [15, 50, 24.5, 400, 0.05, 5]}

从计算结果可以得知索引值为8的备选方案为最高得分，并可以配置graph参数打印排序图表。

rank=usda\_weight.aras\_method(microclimate\_in\_office\_rooms.data, microclimate\_in\_office\_rooms.weight\_of\_criteria, microclimate\_in\_office\_rooms.optimisation\_direction, graph=True)

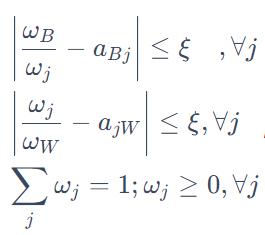
a1: 0.821  
a2: 0.8  
a3: 0.764  
a4: 0.771  
a5: 0.666  
a6: 0.68  
a7: 0.688  
a8: 0.949  
a9: 0.945  
a10: 0.731  
a11: 0.825  
a12: 0.807  
a13: 0.776  
a14: 0.796

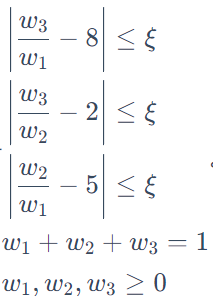


rank

array([[ 8. , 0.94850216],  
 [ 9. , 0.94548367],  
 [11. , 0.82549096],  
 [ 1. , 0.82104672],  
 [12. , 0.80717434],  
 [ 2. , 0.80024248],  
 [14. , 0.79601395],  
 [13. , 0.77558685],  
 [ 4. , 0.77056324],  
 [ 3. , 0.76446671],  
 [10. , 0.73067136],  
 [ 7. , 0.68789147],  
 [ 6. , 0.67985115],  
 [ 5. , 0.66562774]])

### 3.1.2.4 再现 BWM 算法（基于 GWO 元启发式算法）

AHP和F-AHP需要给出判断矩阵，并通过参考CR值，不断调整修正判断矩阵的标度使其趋于一致矩阵，即在一致性检验过程中一致性比率CR值应满足要求。这个调整过程相对繁琐，也不容易使得每项决策准则之间和对于各个准则备选方案之间的判断矩阵修正到满足要求。而 BWM 方法并不需要对决策准则两两比较构建判断矩阵，而类似TOPSIS算法考虑最佳（Best）（正理想值） 和最差（Worst）（负理想值），将各个决策准则与最佳和最差的准则比较作为输入值。以作者给出的第1个案例解释 BWM 算法，该案例表述为公司要选择一种将其产品运输到市场的一种运输方式，涉及3个决策准则，为负载的灵活性C1 C2 C3 C3 =(8,2,1)jC1 =(1,5,8)jwj



*(*w*1∗​,*w*2∗​,…,*wn*∗​)*ξ*∗。*求解的方法使用 GWO（Grey Wolf Optimizer） 优化算法，为一种元启发式算法（meta-heuristic）[45]。

GWO 优化算法求解 BWM时，依据上述求解公式， 对于给定的c，可得 wj △wwj △w △w为最小值对应的权重值更新，即为最优解。对应的代码为：

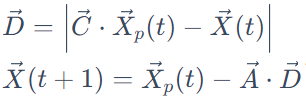
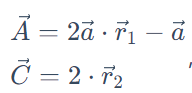
def target\_function(variables):  
 eps = [float('+inf')]  
 for pair in pairs\_b:  
 i, j = pair  
 diff = abs(variables[i] - variables[j]\*mic[j])  
 if ( i != j):  
 eps.append(diff)  
 for pair in pairs\_w:  
 i, j = pair  
 diff = abs(variables[i] - variables[j]\*lic[j])  
 if ( i != j):  
 eps.append(diff)  
 if ( np.sum(variables) == 0):  
 eps = float('+inf')  
 else:  
 eps = max(eps[1:])  
 return eps

对于，对应diff = abs(variables[i] - variables[j]\*mic[j])；对于，对应diff = abs(variables[i] - variables[j]\*lic[j])。

依据 GWO 的等级制度思想，从每行为一组权重向量，初始化数量由参数size确定的权重集群（矩阵）（灰狼群）中，依据△w  
αω

wδ，对应代码行为：

def update\_pack(position, alpha, beta, delta):  
 updated\_position = np.copy(position)  
 for i in range(0, position.shape[0]):  
 if (updated\_position[i,-1] < alpha[0,-1]):  
 alpha[0,:] = np.copy(updated\_position[i,:])  
 if (updated\_position[i,-1] > alpha[0,-1] and updated\_position[i,-1] < beta[0,-1]):  
 beta[0,:] = np.copy(updated\_position[i,:])  
 if (updated\_position[i,-1] > alpha[0,-1] and updated\_position[i,-1] > beta[0,-1] and updated\_position[i,-1] < delta[0,-1]):  
 delta[0,:] = np.copy(updated\_position[i,:])  
 return alpha, beta, delta

逐行逐列（各行各个决策准则权重值）更新权重集群时，是同时考虑、和δtα*、*βδ

为[0,1]的随机数。对于狩猎行为，论文中定义的公式为： ，是分别计算、和δ

*△w*值。对应的代码行为：

def update\_position(position, alpha, beta, delta, a\_linear\_component = 2, min\_values = [-5,-5], max\_values = [5,5], target\_function = target\_function):  
 updated\_position = np.copy(position)  
 for i in range(0, updated\_position.shape[0]):  
 for j in range (0, len(min\_values)):   
 r1\_alpha = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 r2\_alpha = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 a\_alpha = 2\*a\_linear\_component\*r1\_alpha - a\_linear\_component  
 c\_alpha = 2\*r2\_alpha   
 distance\_alpha = abs(c\_alpha\*alpha[0,j] - position[i,j])   
 x1 = alpha[0,j] - a\_alpha\*distance\_alpha   
 r1\_beta = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 r2\_beta = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 a\_beta = 2\*a\_linear\_component\*r1\_beta - a\_linear\_component  
 c\_beta = 2\*r2\_beta   
 distance\_beta = abs(c\_beta\*beta[0,j] - position[i,j])   
 x2 = beta[0,j] - a\_beta\*distance\_beta   
 r1\_delta = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 r2\_delta = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 a\_delta = 2\*a\_linear\_component\*r1\_delta - a\_linear\_component  
 c\_delta = 2\*r2\_delta   
 distance\_delta = abs(c\_delta\*delta[0,j] - position[i,j])   
 x3 = delta[0,j] - a\_delta\*distance\_delta   
 updated\_position[i,j] = np.clip(((x1 + x2 + x3)/3),min\_values[j],max\_values[j])   
 updated\_position[i,-1] = target\_function(updated\_position[i,0:updated\_position.shape[1]-1])  
 return updated\_position

BWM 给出了一致性比例计算的方法。对于一致矩阵有，实际给出的判断向量往往产生非一致性问题，即，当且仅当时，误差达到最大，由，考虑最大误差发生时有：，即，。求解不同的$a\_{BW} {1,2, ,9} $（max ），其表为：

|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Consistency index（CI） （max ） | 0.00 | 0.44 | 1.00 | 1.63 | 2.30 | 3.00 | 3.73 | 4.47 | 5.23 |

一致性比率（CR）则为：

应用基于 GWO 优化算法求解 BWM的方法，再现论文案例1，其计算结果为0.100、0.250和0.649，对应论文中的结果0.0714、0.3387和0.5899，重要性排序保持一致。

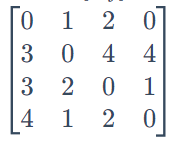
mic=np.array([8,2,1]) # Most Important Criteria  
lic=np.array([1,5,8]) # Least Important Criteria  
  
weights=usda\_weight.bw\_method(mic, lic, size=50, iterations=200,verbose=50)  
print(weights)

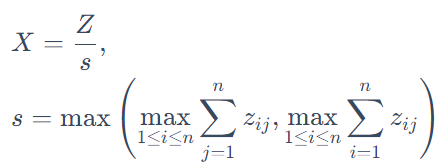
Iteration = 0 f(x) = inf  
Iteration = 50 f(x) = 0.015477062414798246  
Iteration = 100 f(x) = 0.015112102494579303  
Iteration = 150 f(x) = 0.015112102494579303  
Iteration = 200 f(x) = 0.015020926002282144  
[0.10000609 0.24996459 0.65002932]

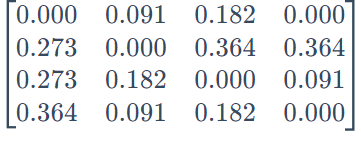
### 3.1.2.5 再现 DEMATEL 算法

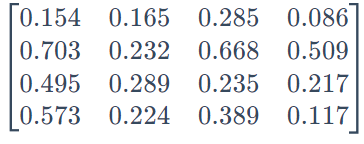
Dematel 是一种运用图论和矩阵工具解释问题的系统分析方法，通过评估系统中各个因子（factors）（决策准则）之间的相互依赖关系，可以计算每个因子对其它因子的影响度和被影响度，从而计算每个因子的原因度和中心度，确定因子间的因果关系和每个因子在系统中的地位。Dematel 有许多不同的变体，例如根据使用方法可归为有5类： 经典型（classical-dematel），模糊性（fuzzy-dematel）， grey-dematel, ANP-dematel（analytical network process，ANP）和其它类型。pyDecision库提供的dematel\_method 方法为经典型，将其迁移至USDA库中调用。

第1步，该方法首先需要构建因子关系（影响，判断）矩阵（direct-influence matrix），评估系统中因子之间的关系。假设决策组中有个名专家参与构建关系矩阵，通过标度（0，无影响（no influence）；1，低影响（low influence）；2，中等影响（medium influence）；3，高等影响（high influence）；4，极度影响（very high influence）），指出因子对的影响程度。第位专家构建的关系矩阵表示为：** EkFiFj

**, i, j=1,2, , n。例如`pyDecision`库案例提供的关系矩阵为：，对应因子g1、g2、g3和g4。

第2步，（最大值）归一化关系矩阵为 

**Zs。

第3步，计算综合关系矩阵   
T=X+X2+X3+…+Xh=X(1−X)−1,h⟶∞I。

第4步，生成影响力关系图（influential relation map，IRM）。综合关系矩阵的行之和为，列之和为，对应公式为：，式中，为矩阵第行之和，表示因子对其它因子的直接和间接影响的总和，为影响度（D）值，对应实验数据结果为，可知因子g2对其它因子直接和间接影响总和最大，而因子g1对其它因子影响最小；为矩阵第列之和，表示因子受其它因子直接和间接影响的总和，为被影响度（C）值，对应实验数据结果为，可知因子g1受到其它因子的影响最大，而g2受到其它因子的影响最小。

另，影响度和被影响度之和为，代表该因子在系统中的核心作用程度，为中心度（Prominence）；同样，代表该因子对系统贡献的净效应，为原因度（Relation）。如果为正，则因子对其它因子有净影响，可以归为原因组（cause group）；如果为负，则因子整体上受到其它因素的影响，应归入效应组（effect group）。实验数据对应的中心度和原因度分别为，和\begin{bmatrix} -1.236 & 1.202 & -0.341 & 0.375 \end{bmatrix}

。可以通过映射的数据集创建IRM（如下图），为决策指定提供有价值的见解。四象限图中：

第1象限，（Prominence：高，Relation：高），具有最重要的因子；

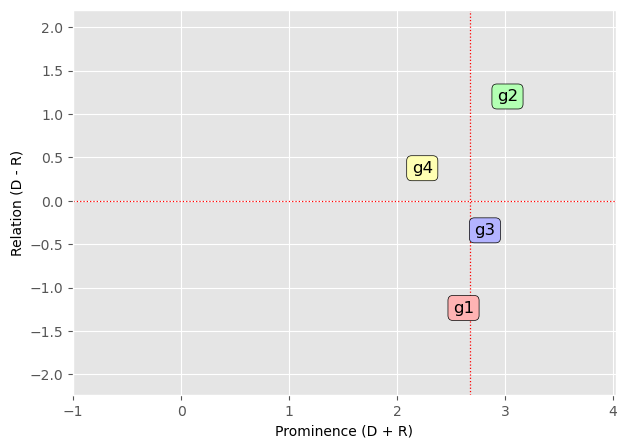
第2象限，（Prominence：低，Relation：高），具有可以通过其它因子改进的重要因子；

第3象限，（Prominence：低，Relation：低），具有不重要的因子；

第4象限，（Prominence：高，Relation：低），具有无法通过其它因子改进的重要因子。

dataset = np.array([  
 # g1 g2 g3 g4  
 [ 0, 1, 2, 0 ], #g1  
 [ 3, 0, 4, 4 ], #g2  
 [ 3, 2, 0, 1 ], #g3  
 [ 4, 1, 2, 0 ] #g4  
 ])  
  
D\_plus\_R, D\_minus\_R, weights=usda\_weight.dematel\_method(dataset, size\_x=7, size\_y=5)  
print(D\_plus\_R, D\_minus\_R, weights)

QUADRANT I has the Most Important Criteria (Prominence: High, Relation: High)  
QUADRANT II has Important Criteira that can be Improved by Other Criteria (Prominence: Low, Relation: High)  
QUADRANT III has Criteria that are not Important (Prominence: Low, Relation: Low)  
QUADRANT IV has Important Criteria that cannot be Improved by Other Criteria (Prominence: High, Relation: Low)  
  
g1: Quadrant III  
g2: Quadrant I  
g3: Quadrant IV  
g4: Quadrant II



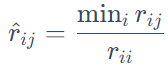
[2.61441523 3.02130553 2.81287398 2.23245694] [-1.2356301 1.20172257 -0.34106981 0.37497733] [0.24477133 0.28286592 0.26335178 0.20901097]

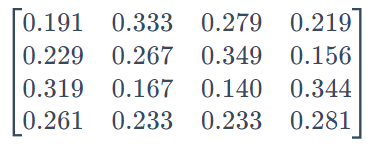
### 3.1.2.6 再现 IDOCRIW 算法（基于 GA 元启发式算法）

Zavadskas 和 Podvezko 提出的 IDOCRIW 算法受益于信息熵（Entropy）和 准则影响损失（Criterion Impact LOSs，CILOS）方法。通过结合这两种方法确定相对影响损失及属性权重。IDOCRIW 算法的输入条件为决策矩阵和决策准则方向（越大越好还是越小越好）。作者给出的案例数据为：，为一个组织打算在、、和等4个地块中选择一个建造一座行政大楼。公司主管部分拟根据专家意见，对成本（C1 C2 C3 C4）等属性进行审查和优先排序。

第1步，标准化决策矩阵，公式为：，式中，表示第个决策准则下第个备选方案的标准化值。实验数据结果为：。

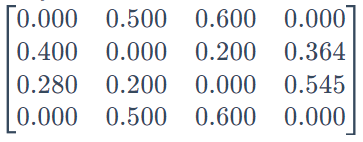
第2步，熵值权重法，公式为：，实验数据决策准则的信息熵权重结果为，。

第3步，统一决策准则方向为最大方向（maximized）并再次标准化（该公式同前），公式为：;i,j∈{1,…,n}​



第4步，构建方阵（Square Matrix），是依次提取决策矩阵每一决策准则列最大值所在的行形成一个新的矩阵，即对原决策矩阵的行按列的最大值排序，从而新的矩阵对角线为各列的最大值，其公式为：，式中， jkiaij=aki jajj=aj。

第5步，计算相对影响损失矩阵（Relative Impact Loss Matrix），公式为：pijj

。

第6步，权重系统矩阵（ Weight System Matrix），将相对损失矩阵对角线的值替换为各列值和取反，公式为：。实验数据计算结果为： 。

第7步，计算决策准则影响损失权重（Criterion Impact Loss Weight，），根据公式，计算权重值，实验数据对应的方程为：，求解上述方程即可获得权重。

第8步，使用遗传算法（Genetic Algorithm， GA）求解第7步的方程。

由第7步方程构建目标函数，对应的代码为：

def target\_function(variable = [0]\*WP.shape[1]):  
 variable = [variable[i]/sum(variable) for i in range(0, len(variable))]  
 WP\_s = np.copy(WP)  
 for i in range(0, WP.shape[0]):  
 for j in range(0, WP.shape[1]):  
 WP\_s[i, j] = WP\_s[i, j]\*variable[j]  
 total = abs(WP\_s.sum(axis = 1))   
 total = sum(total)   
 return total

其中，WP变量为权重系统矩阵，variable为每次迭代的决策准则权重向量，代入第7步公式，满足各行值总和趋于0，即total（值越小越好）。

遗传算法为元启发式算法，受启发于自然进化理论，通过模拟自然选择的过程，从种群（population）选择最适合的个体（individuals）进行繁殖产生后代（offspring ）。产生的后代继承了父代的特征，并遗传至下一代。如果父代有较好的适应度（fitness score），后代就会比父代有更好的生存机会。上述过程不断迭代，直至进化到最适合的一代。整个过程可以分为5步，初始化种群（对应initial\_population函数），适应函数（对应fitness\_function函数），选择和交叉（对应breeding函数），变异（对应mutation函数）。

第8-1步，初始化种群（Initial Population）

种群是包含多个个体的集合，本例中个体即为决策准则权重值，为一个包含4个值的向量，那么初始化种群则是随机产生了多个权重向量的集合。个体的特征是一组称为基因（Genes）的参数，对应着各个决策准则，而个体组成的向量就形成了一个染色体（Chromosome），即决策准则权重值。初始化种群对应的代码为：

# Function: Initialize Variables  
def initial\_population(population\_size = 5, min\_values = [-5,-5], max\_values = [5,5], target\_function = target\_function):  
 population = np.zeros((population\_size, len(min\_values) + 1))  
 for i in range(0, population\_size):  
 for j in range(0, len(min\_values)):  
 population[i,j] = random.uniform(min\_values[j], max\_values[j])   
 population[i,-1] = target\_function(population[i,0:population.shape[1]-1])  
 return population

代码初始化的种群population个人均增加了一位，用于存储目标函数target\_function的返回值total（）。初始化的权重值位于给定的区间[0,1]内。

第8-2步，定义适应度函数（Fitness Function）

适应度函数通过给每一个体适应度值，确定个体与其它个体竞争的能力，从而更容易被选择进行繁殖。本例个体适应度为包含两个值的向量，公式为：，式中，c为适应度第1个值（索引为0），SS\_{min}SSSS 的影响，当适应度第1个值越大，第2个值也越大，且如果其父代第2个值越大，其自身（子代）第2个值也越大。 对应的代码行为：

# Function: Fitness  
def fitness\_function(population):   
 fitness = np.zeros((population.shape[0], 2))  
 for i in range(0, fitness.shape[0]):  
 fitness[i,0] = 1/(1+ population[i,-1] + abs(population[:,-1].min()))  
 fit\_sum = fitness[:,0].sum()  
 fitness[0,1] = fitness[0,0]  
 for i in range(1, fitness.shape[0]):  
 fitness[i,1] = (fitness[i,0] + fitness[i-1,1])  
 for i in range(0, fitness.shape[0]):  
 fitness[i,1] = fitness[i,1]/fit\_sum  
 return fitness

第8-3步，繁殖（选择（Selection）与交叉（Crossover））

* 选择，是选择最适合的个体传递基因给下一代，需要选择两个个体（父母）为和，选择前可以将前一迭代适应度最大的前行替换对应种群的前行。选择的过程是产生随机数，循环适应度矩阵，只要满足，即为选择的适合的父代。代码为：

# Function: Selection  
def roulette\_wheel(fitness):   
 ix = 0  
 random = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 for i in range(0, fitness.shape[0]):  
 if (random <= fitness[i, 1]):  
 ix = i  
 break  
 return ix

* 交叉，为遗传算法中最重要的部分，子代是通过父代和交换基因产生的， 公式为：，式中，为相对父代个体基因，为第个个体的第个基因，|P|rrb*μ*为交叉系数。代码为：

# Function: Offspring  
def breeding(population, fitness, min\_values = [-5,-5], max\_values = [5,5], m u = 1, elite = 0, target\_function = target\_function):  
 offspring = np.copy(population)  
 b\_offspring = 0  
 if (elite > 0):  
 preserve = np.copy(population[population[:,-1].argsort()])  
 for i in range(0, elite):  
 for j in range(0, offspring.shape[1]):  
 offspring[i,j] = preserve[i,j]  
 for i in range (elite, offspring.shape[0]):  
 parent\_1, parent\_2 = roulette\_wheel(fitness), roulette\_wheel(fitness)  
 while parent\_1 == parent\_2:  
 parent\_2 = random.sample(range(0, len(population) - 1), 1)[0]  
 for j in range(0, offspring.shape[1] - 1):  
 rand = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 rand\_b = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)   
 if (rand <= 0.5):  
 b\_offspring = 2\*(rand\_b)  
 b\_offspring = b\_offspring\*\*(1/(mu + 1))  
 elif (rand > 0.5):   
 b\_offspring = 1/(2\*(1 - rand\_b))  
 b\_offspring = b\_offspring\*\*(1/(mu + 1))   
 offspring[i,j] = np.clip(((1 + b\_offspring)\*population[parent\_1, j] + (1 - b\_offspring)\*population[parent\_2, j])/2, min\_values[j], max\_values[j])   
 if(i < population.shape[0] - 1):   
 offspring[i+1,j] = np.clip(((1 - b\_offspring)\*population[parent\_1, j] + (1 + b\_offspring)\*population[parent\_2, j])/2, min\_values[j], max\_values[j])   
 offspring[i,-1] = target\_function(offspring[i,0:offspring.shape[1]-1])   
 return offspring

第8-4步，变异（Mutation）

某些新生成的后代，基因可以发生随机概率较低的突变，以维持种群的多样性防止过早收敛。公式为：，式中， Pijijjrr\_ddm为变异值。对应代码为：

# Function: Mutation  
def mutation(offspring, mutation\_rate = 0.1, eta = 1, min\_values = [-5,-5], max\_values = [5,5], target\_function = target\_function):  
 d\_mutation = 0   
 for i in range (0, offspring.shape[0]):  
 for j in range(0, offspring.shape[1] - 1):  
 probability = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 if (probability < mutation\_rate):  
 rand = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)  
 rand\_d = int.from\_bytes(os.urandom(8), byteorder = "big") / ((1 << 64) - 1)   
 if (rand <= 0.5):  
 d\_mutation = 2\*(rand\_d)  
 d\_mutation = d\_mutation\*\*(1/(eta + 1)) - 1  
 elif (rand > 0.5):   
 d\_mutation = 2\*(1 - rand\_d)  
 d\_mutation = 1 - d\_mutation\*\*(1/(eta + 1))   
 offspring[i,j] = np.clip((offspring[i,j] + d\_mutation), min\_values[j], max\_values[j])  
 offspring[i,-1] = target\_function(offspring[i,0:offspring.shape[1]-1])   
 return offspring

第8-5步，迭代

遗传算法通过迭代选择、交叉、变异，更新适应度，使种群个体（决策准则权重）趋于最优解，对应代码为：

# GA Function  
def genetic\_algorithm(population\_size = 5, mutation\_rate = 0.1, elite = 0, min\_values = [-5,-5], max\_values = [5,5], eta = 1, mu = 1, generations = 50, target\_function = target\_function, verbose = True):   
 count = 0  
 population = initial\_population(population\_size = population\_size, min\_values = min\_values, max\_values = max\_values, target\_function = target\_function)  
 fitness = fitness\_function(population)   
 elite\_ind = np.copy(population[population[:,-1].argsort()][0,:])  
 while (count <= generations):   
 if (verbose == True):  
 print('Generation = ', count, ' f(x) = ', elite\_ind[-1])   
 offspring = breeding(population, fitness, min\_values = min\_values, max\_values = max\_values, mu = mu, elite = elite, target\_function = target\_function)   
 population = mutation(offspring, mutation\_rate = mutation\_rate, eta = eta, min\_values = min\_values, max\_values = max\_values, target\_function = target\_function)  
 fitness = fitness\_function(population)  
 value = np.copy(population[population[:,-1].argsort()][0,:])  
 if(elite\_ind[-1] > value[-1]):  
 elite\_ind = np.copy(value)   
 count = count + 1   
 return elite\_ind

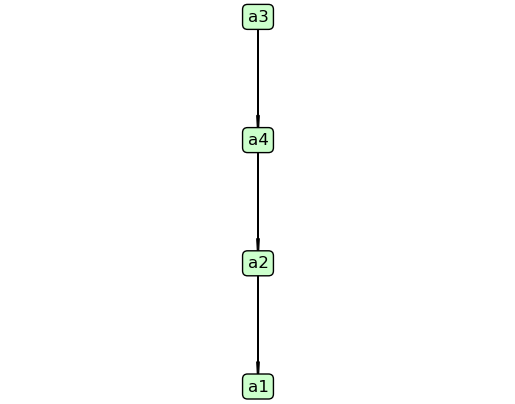
由遗传算法求得第7步方程解为。

第9步，聚合权重（Aggregate Weight）

将熵值权重（）和CILOS权重（）聚合得决策准则权重向量，其公式为：。最终求得权重值为： 。

criterion\_type=['min','max','min','max']  
dataset=np.array([[3,100,10,7],  
 [2.5,80,8,5],  
 [1.8,50,20,11],  
 [2.2,70,12,9]])  
  
rank=usda\_weight.idocriw\_method(dataset, criterion\_type, size = 20, gen = 12000, graph = True,verbose=1000)

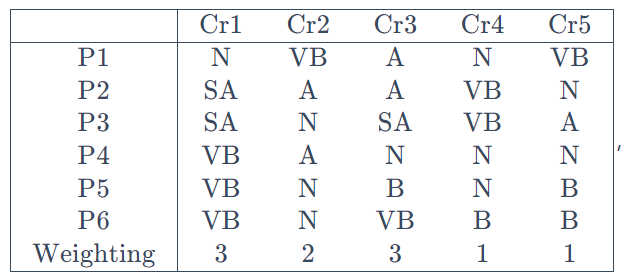
Generation = 0 f(x) = 0.25174145471050513  
Generation = 1000 f(x) = 0.005088300343106242  
Generation = 2000 f(x) = 0.005088300343106242  
Generation = 3000 f(x) = 0.002048160299766136  
Generation = 4000 f(x) = 0.002048160299766136  
Generation = 5000 f(x) = 0.0017442771772882698  
Generation = 6000 f(x) = 0.0017442771772882698  
Generation = 7000 f(x) = 0.0017442771772882698  
Generation = 8000 f(x) = 0.0017442771772882698  
Generation = 9000 f(x) = 0.0017442771772882698  
Generation = 10000 f(x) = 0.0017442771772882698  
Generation = 11000 f(x) = 0.0017442771772882698  
Generation = 12000 f(x) = 0.0017442771772882698  
solution:[0.33330718 0.21991215 0.19594158 0.25083909]  
a1: 0.1652  
a2: 0.1884  
a3: 0.3547  
a4: 0.2917

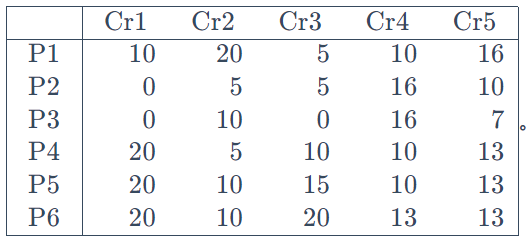


### 3.1.2.7 再现 ELECTRE-I 算法

1966年 Bernard Roy 在罗马发表了一篇论文，开发了一个实用的决策系统，现在称之为 ELECTRE。该方法根据备选方案在一组通用决策准则上的得分，对所有备选方案可能配对之间的决策准则关系进行系统分析。结果是衡量每个备选方案优于所有其它备选方案的程度。即若某一备选方案具有多数的决策准则优于其它备选方案且没有任何准则低于不可接受的阈值程度，则该方案优于其它备选方案。该方法需要构建排名靠前的关系，生成满意度/一致性（concordance）和不满意度/不一致（discordance）矩阵（指数）（包含对每个决策准则的相对重要性，即权重的考虑）。 ELECTRE 有6个主要版本，为I、II、III、IV、Tri 和 IS。这里以 ELECTRE-I 为例，再现该算法。

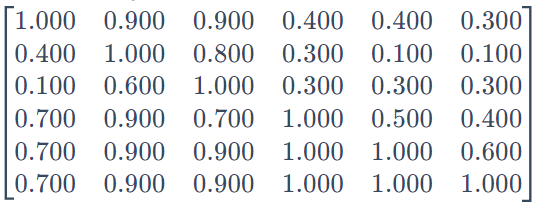
再现 ELECTRE-I 算法的实验数据选择*ELECTRE AND DECISION SUPPORT*[22]提供的案例数据，该案例拟议的高速公路项目有6个备选方案，包含5个环境相关的决策准则，为噪音污染影响（）、社区和土地隔离影响（）、空气污染影响（）、土地利用影响（）和休闲影响（）。决策准则的权重为：；定性等级评估决策矩阵：

，

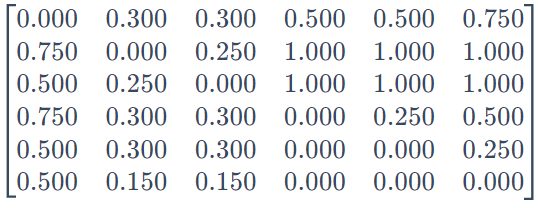
其中，评估等级解释为VB（Very Beneficial，非常有益）、B（Beneficial，有益）、N（Neutral，中性）、A（Adverse，不利）和SA（Severely Adverse，严重不利）。将定性等级评估转换为数值，按照如下评分系统：对于，；对于，，因此决策矩阵为：。

第1步，计算满意度矩阵，公式为：，为决策准则的权重，为两两备选方案，和gj​(b)j

z

。

第2步，计算不满意度矩阵。对于的值位于[0.1]区间，测量备选方案优于备选方案的强度，然而，对的任何优势强度都可以被不满意度矩阵消弱或抵消，其公式为：，为计算两两备选方案（）比较下，各个对应决策准则值的差值（为比较项减当前项）的最大值除以δ

。

第3步，阈值（threshold），ELECTRE-I 的排名优势关系（outranking relationship）是通过比较具有指定限制（阈值）的满意度矩阵和不满意度矩阵来建立的。为满意度阈值，最大值为1；为不满意度阈值，最小值为0，因此有：，式中，表示备选方案排名高于备选方案，当满意度指数（矩阵）大于等于阈值，且不满意度指数小于等于阈值。

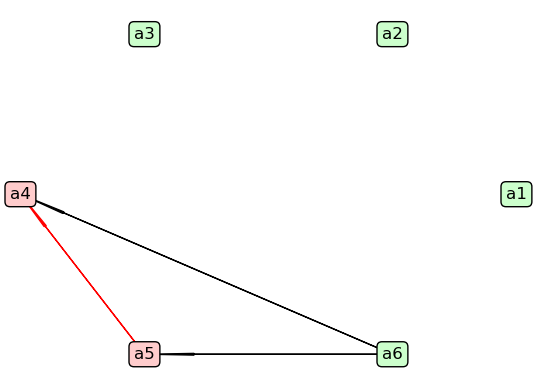
对于排名优势关系（结构）的比较有如下表述公式：

配置不同的阈值，当，时，排名靠前的核心（kernel）备选方案由1、2、3和6组成（绿色标识），核心之外的为4和5备选方案。且可知排名图中，，但之间及与其它方案之间则不可比较；

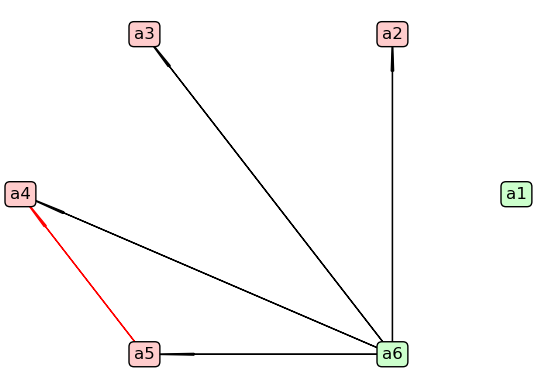
当，时，排名靠前的核心备选方案由1和6组成，除了和不可比之外，均高于其它备选方案；

当，时，排名靠前的核心备选方案仅有。

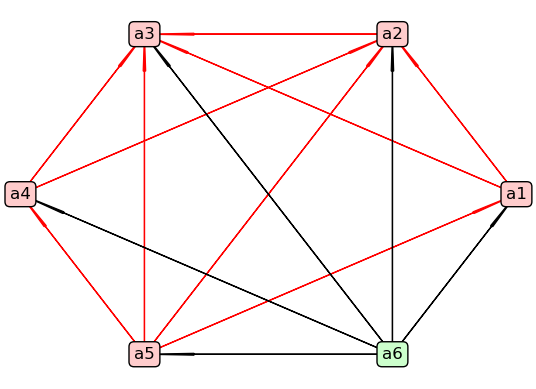
W\_proposed\_motorway=[3,2,3,1,1]  
proposed\_motorway=np.array([  
 [ 10, 20, 5, 10, 16],  
 [ 0, 5, 5, 16, 10 ],  
 [ 0, 10, 0, 16, 7 ],  
 [ 20, 5, 10, 10, 13 ],  
 [20, 10, 15, 10, 13 ],  
 [ 20, 10, 20, 13, 13 ]  
 ])  
  
c\_hat=1  
d\_hat=0   
concordance, discordance, dominance, kernel, dominated=usda\_weight.electre\_i(proposed\_motorway, W = W\_proposed\_motorway, remove\_cycles = True, c\_hat = c\_hat, d\_hat = d\_hat, graph = True)



c\_hat=0.9  
d\_hat=0.15   
concordance, discordance, dominance, kernel, dominated=usda\_weight.electre\_i(proposed\_motorway, W = W\_proposed\_motorway, remove\_cycles = True, c\_hat = c\_hat, d\_hat = d\_hat, graph = True)



c\_hat=0.7  
d\_hat=0.5   
concordance, discordance, dominance, kernel, dominated=usda\_weight.electre\_i(proposed\_motorway, W = W\_proposed\_motorway, remove\_cycles = True, c\_hat = c\_hat, d\_hat = d\_hat, graph = True)



## 3.1.3 LST（降温）与绿地和建设用地矛盾的匹配评估

LST 地表温度数据和 NLCD 美国土地覆盖类型数据集，及对上述数据的处理参看“3.5 标记距离”一节，下述实验直接读取已经处理的数据。城市是人类聚居地，需要满足住房、交通、卫生、公用事业、经济活动和娱乐休闲等需求，又需要改善城市居住的环境，拥有更多的绿地、水体，同时平衡城市热岛效应，那么上述的需求就存在了矛盾，好的城市环境需要更多的绿地，而居住和经济活动等需要建设开发土地，那么对建成区的评估则是基于具有矛盾的多因子（决策准则）的权衡。该实验选择的因子包括 LST 地表温度（temp）、绿地栅格频数(green\_freq)、水体栅格频数(water\_freq)(含开放水体和湿地)、开发用地栅格频数(developed\_freq)和耕地栅格频数(cultivated\_freq)等5个决策准则。评估的对象（备选方案）为基于 LST 栅格精度提取的样方组，总共 2785 个样方。样方单元的大小为1000m。决策准则类型对应temp、green\_freq、water\_freq、developed\_freq和cultivated\_freq，为’min’、‘max’、‘max’、‘max’和’min’。对应决策准则的权重值为0.2、0.3、0.1、0.3和0.1（权重和为1）。权重的配置上将绿地和建设用地全部配置为0.3，二者部分轻重，同时减小水体和耕地的权重比例，LST 保持在一个平均的值上。MCDM 选择 WASPAS 算法。

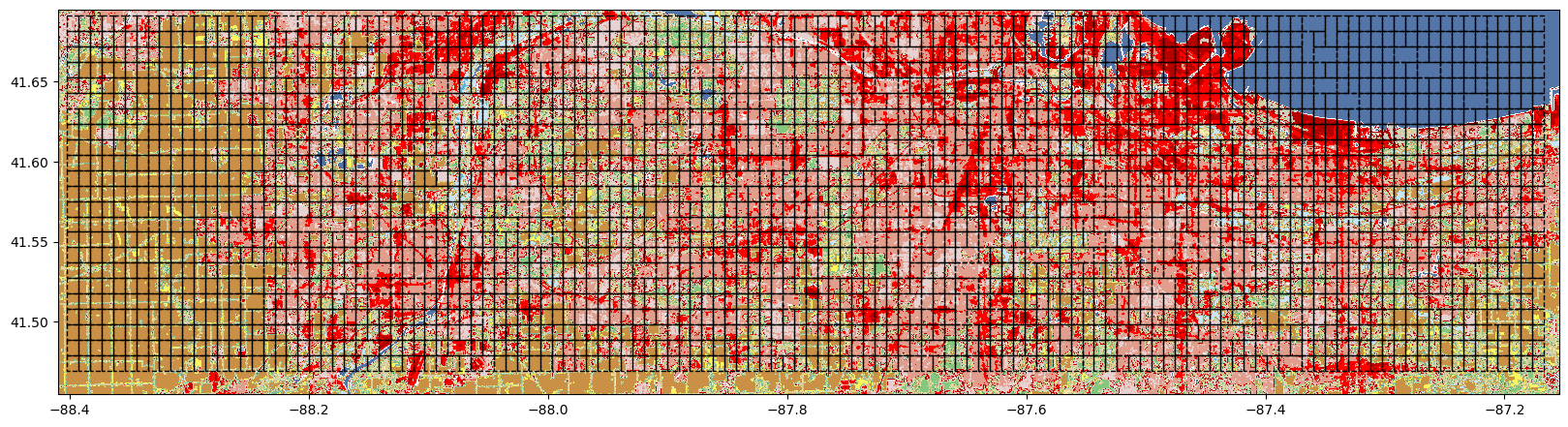
# IPython extension to reload modules before executing user code.  
%load\_ext autoreload   
# Reload all modules (except those excluded by %aimport) every time before executing the Python code typed.  
%autoreload 2   
  
from usda import geodata\_process as usda\_geodataProcess  
from usda import weight as usda\_weight  
  
import rasterio as rio  
import matplotlib.pyplot as plt  
from rasterio.plot import show  
import matplotlib  
import geopandas as gpd  
import pandas as pd

The autoreload extension is already loaded. To reload it, use:  
 %reload\_ext autoreload

### 3.1.3.1 数据预处理

土地覆盖合并类型由LU\_merge\_dict变量确定，在栅格单元统计分类频数时，按照提取的4大类，绿地、水体、开发用地和耕地计算。

pt\_leftBottom=[-88.41018,41.45427]  
pt\_rightTop=[-87.15433,41.69524]   
  
nlcd\_class\_idNcolor={  
 '0':[0,"#64b3d5"],  
 'Open\_Water':[11,"#5475a8"],  
 'Perennial\_IceSnow':[12,"#ffffff"],  
 'Developed\_Open\_Space':[21,"#e8d1d1"],  
 'Developed\_Low\_Intensity':[22,"#e29e8c"],  
 'Developed\_Medium\_Intensity':[23,"#ff0000"],  
 'Developed\_High\_Intensity':[24,"#b50000"],  
 'Barren\_Land\_RockSandClay) ':[31,"#d2cdc0"],  
 'Deciduous\_Forest':[41,'#85c77e'],  
 'Evergreen\_Forest':[42,"#38814e"],  
 'Mixed Fores':[43,"#d4e7b0"],  
 'Dwarf\_Scrub':[51,"#af963c"],  
 'Shrub\_Scrub':[52,"#dcca8f"],  
 'Grassland\_Herbaceous':[71,"#fde9aa"],  
 'Sedge\_Herbaceous':[72,"#d1d182"],  
 'Lichens':[73,"#a3cc51"],  
 'Moss':[74,"#82ba9e"],  
 'Pasture\_Hay':[81,"#fbf65d"],  
 'Cultivated\_Crops':[82,"#ca9146"],  
 'Woody\_Wetlands':[90,"#c8e6f8"],  
 'Emergent\_Herbaceous\_Wetlands':[95,"#64b3d5"],   
 }  
  
nlcd\_class\_color={v[0]:v[1] for v in nlcd\_class\_idNcolor.values()}  
cmap\_LC, norm=matplotlib.colors.from\_levels\_and\_colors(list(nlcd\_class\_color.keys()),list(nlcd\_class\_color.values()),extend='max')  
cmap\_LC  
  
chicago\_nlcd\_2019\_lc\_reproj\_fn=r'E:\data\NLCD\_landcover\_2019\_release\_all\_files\_20210604\clipped\chicago\_nlcd\_2019\_lc\_4326.tif'  
with rio.open(chicago\_nlcd\_2019\_lc\_reproj\_fn) as src:   
 epsg\_chicago=src.crs  
 transform\_chicago=src.transform  
   
chicago\_pt\_leftBottom\_pj=usda\_geodataProcess.pt\_coordi\_transform(4326,epsg\_chicago,pt\_leftBottom)  
chicago\_pt\_rightTop\_pj=usda\_geodataProcess.pt\_coordi\_transform(4326,epsg\_chicago,pt\_rightTop)   
  
Chicago\_LC\_reproj,Chicago\_reproj\_transform,Chicago\_reproj\_ras\_meta=usda\_geodataProcess.rio\_read\_subset(chicago\_nlcd\_2019\_lc\_reproj\_fn,[chicago\_pt\_leftBottom\_pj,chicago\_pt\_rightTop\_pj])   
  
LST\_shp\_fn=r'E:\data\LST\LST\_Chicago\LST\_Chicago.shp'   
LST\_gdf=gpd.read\_file(LST\_shp\_fn)  
  
f, ax=plt.subplots(figsize=(20,20))  
show(Chicago\_LC\_reproj,ax=ax,transform=Chicago\_reproj\_transform,cmap=cmap\_LC,norm=norm)  
LST\_gdf.plot(color='none',edgecolor='k',linewidth=1,ax=ax,linestyle='--')  
plt.show()



lc\_stats=usda\_geodataProcess.zonal\_stats\_raster(chicago\_nlcd\_2019\_lc\_reproj\_fn,LST\_gdf,band=1,stats=['majority'],add\_stats=['frequency'])

lc\_stats.head(3)

|  | **temp** | **geometry** | **majority** | **index** | **frequency\_21** | **frequency\_22** | **frequency\_23** | **frequency\_81** | **frequency\_24** | **frequency\_41** | **frequency\_82** | **frequency\_71** | **frequency\_42** | **frequency\_43** | **frequency\_90** | **frequency\_11** | **frequency\_31** | **frequency\_95** | **frequency\_52** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 15166 | POLYGON ((-88.40252 41.69121, -88.39287 41.691... | 21.0 | 0 | 318.0 | 303.0 | 122.0 | 21.0 | 19.0 | 1.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1** | 15191 | POLYGON ((-88.39287 41.69121, -88.38322 41.691... | 22.0 | 0 | 162.0 | 163.0 | 54.0 | 159.0 | 11.0 | 135.0 | 44.0 | 30.0 | 15.0 | 6.0 | 3.0 | 1.0 | 1.0 | NaN | NaN |
| **2** | 15245 | POLYGON ((-88.38322 41.69121, -88.37356 41.691... | 22.0 | 0 | 117.0 | 218.0 | 208.0 | 1.0 | 70.0 | 52.0 | 24.0 | 79.0 | NaN | NaN | 4.0 | 1.0 | 1.0 | 9.0 | NaN |

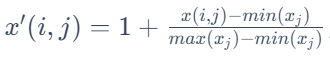
import warnings  
warnings.filterwarnings('ignore')  
  
LU\_merge\_dict=dict(green\_lst=[21,41,42,43,51,52,71,72,73,74],  
 water\_lst=[11,12,90,95],  
 developed\_lst=[22,23,24],  
 cultivated\_lst=[81,82])  
  
def LC\_fre\_merge\_func(row,args):  
 cols=[i for i in row.index if i.split('\_')[0]=='frequency']  
 LU\_merged\_cols={}  
 for k,v in args.items():  
 lu\_cols=[i for i in cols if int(i.split('\_')[1]) in v]  
 freq\_sum=row[lu\_cols].sum()  
 LU\_merged\_cols[k]=freq\_sum  
 return pd.Series(LU\_merged\_cols.values())   
   
lc\_stats\_copy=lc\_stats.copy(deep=True)  
reclassify=['green\_freq','water\_freq','developed\_freq','cultivated\_freq']  
lc\_stats\_copy[reclassify]=lc\_stats\_copy.apply(LC\_fre\_merge\_func,args=(LU\_merge\_dict,),axis=1)  
criteria\_df=lc\_stats\_copy[['temp']+reclassify]+1  
criteria\_df.head(3)

|  | **temp** | **green\_freq** | **wet\_lands\_freq** | **developed\_freq** | **cultivated\_freq** |
| --- | --- | --- | --- | --- | --- |
| **0** | 15167 | 320.0 | 1.0 | 445.0 | 22.0 |
| **1** | 15192 | 349.0 | 5.0 | 229.0 | 204.0 |
| **2** | 15246 | 249.0 | 15.0 | 497.0 | 26.0 |

### 3.1.3.2 WASPAS 算法评估样方单元综合评价指数

WASPAS 为聚合加权的和积估计（ Weighted aggregated sum product assessment），具体的方法为：

第1步，构建决策矩阵，为上述数据预处理后criteria\_df变量结果；

第2步，标准化决策矩阵，对于决策类型为max的决策准则使用公式：；对于决策类型为min的使用公式： ；

第3步，第个备选方案的第1个相对权重（重要性）类似 WSM 技术，有公式：jiiij个决策准则权重值幂的乘积；

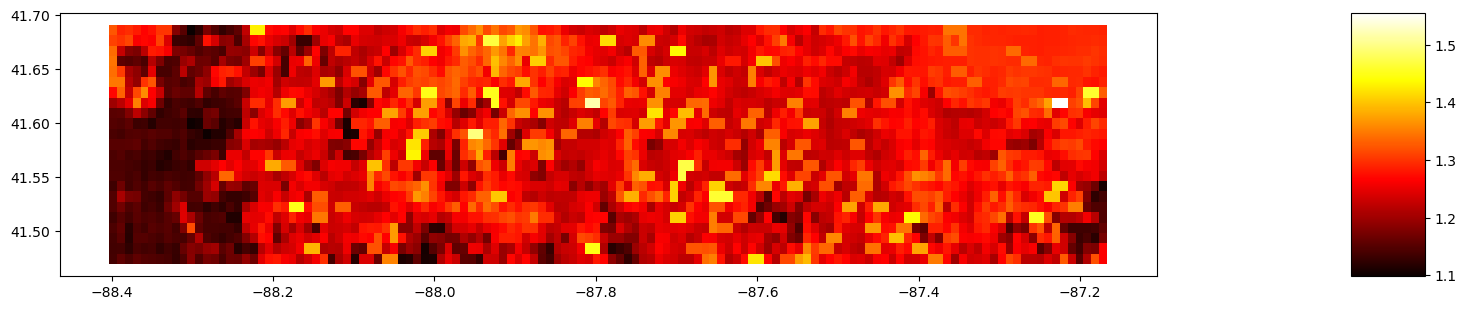
第4步，计算第个备选方案综合相对权重，公式为：，式中 为组合参数，位于[0,1]之间，如果值为0，则为WPM方法；如果值为1，则为WSM方法。 用于弥补 MCDM 在排名精度上的问题。

迁移pyDecision库的waspas\_method方法至USDA库，计算结果如下。

weights=[0.2,0.3,0.1,0.3,0.1]  
criterion\_type=['min','max','max','max','min']  
lambda\_value=0.5  
wsm, wpm, waspas=usda\_weight.waspas\_method(criteria\_df.to\_numpy(), criterion\_type, weights, lambda\_value)

LST\_gdf['waspas']=waspas

from mpl\_toolkits.axes\_grid1 import make\_axes\_locatable  
fig, ax=plt.subplots(1, 1,figsize=(20,10),)  
divider=make\_axes\_locatable(ax)  
cax=divider.append\_axes("right", size="5%", pad=0.1)  
LST\_gdf.plot(column='waspas',legend=True,cax=cax, cmap='hot',ax=ax);



注释（Notes）：

① InVEST，（<https://naturalcapitalproject.stanford.edu/software/invest>）。

② 多准则决策法相关软件，（<https://en.wikipedia.org/wiki/Decision-making_software#Comparison_of_decision-making_software>）。

③ pyDecision（Python包），用于多准则决策法计算，（<https://github.com/Valdecy/pyDecision>）。

④ Google Developers Console\_Street View Static API，（<https://console.cloud.google.com/apis/dashboard?pli=1&project=gsv-streetscape>）。

⑤ istreetview，（<https://istreetview.com>）。

⑥ Street View Download 360，（<https://svd360.istreetview.com>）。

⑦ mayavi，（<https://docs.enthought.com/mayavi/mayavi>）。

⑧ pyDecision工具包的示例文件，（<https://github.com/Valdecy/pyDecision>）。

参考文献（References）:

[1] Gal, T., Stewart, T.J. and Hanne, T. (1999) Multicriteria decision making: Advances in MCDM models, algorithms, theory, and applications. Boston, MA: Kluwer academic.

[2] Rezaei, J. Best-worst multi-criteria decision-making method. Omega (United Kingdom) 53, 49–57 (2015).

[3] 黎家琦,武雪玲,唐诗怡.压力-状态-响应模型下的生态敏感性分析算法[J].测绘科学,2020,45(11):75-83+106.DOI:10.16251/j.cnki.1009-2307.2020.11.012.

[4] 张金茜,李红瑛,曹二佳,巩杰.多尺度流域生态脆弱性评价及其空间关联性——以甘肃白龙江流域为例[J].应用生态学报,2018,29(09):2897-2906.DOI:10.13287/j.1001-9332.201809.008.

[5] Solecka, I., Rinne, T., Caracciolo Martins, R., Kytta, M. & Albert, C. Important places in landscape – investigating the determinants of perceived landscape value in the suburban area of Wrocław, Poland. Landsc Urban Plan 218, 104289 (2022).

[6] Boroushaki, S. Entropy-Based Weights for MultiCriteria Spatial Decision-Making. Yearbook of the Association of Pacific Coast Geographers 79, 168–187 (2017).

[7] Malczewski, J. GIS-based multicriteria decision analysis: A survey of the literature. International Journal of Geographical Information Science 20, 703–726 (2006).

[8] Malczewski, J. & Rinner, C. Advances in Geographic Information Science Multicriteria Decision Analysis in Geographic Information Science. http://www.springer.com/series/7712.

[9] Hwang, C.-L., and K. Yoon. 1981. Multiple Attribute Decision Making. Methods and Applications A State-of-the-Art Survey. Berlin: Springer-Verlag.

[10] Saaty, T. L. & Katz, J. M. How to make a decision: The Analytic Hierarchy Process. European Journal of Operational Research vol. 48 (1990).

[11] Ayhan, M. B. A Fuzzy Ahp Approach For Supplier Selection Problem: A Case Study In A Gearmotor Company. International Journal of Managing Value and Supply Chains 4, 11–23 (2013).

[12] Zavadskas, E. K. & Turskis, Z. A new additive ratio assessment (ARAS) method in multicriteria decision-making. Technological and Economic Development of Economy 16, 159–172 (2010).

[13] Par M. SUR LES ÉLECTIONS AU SCRUTIN \*. Des sciences (1781).

[14] Keshavarz-Ghorabaee, M., Zavadskas, E., Turskis, Z., & Antucheviciene, J. (2016). A new combinative distance-based assessment (CODAS) method for multi-criteria decision-making. Economic Computation and Economic Cybernetics Studies and Research / Academy of Economic Studies, 50, 25–44.

[15] Razavi Hajiagha, S. H., Hashemi, S. S. & Zavadskas, E. K. A complex proportional assessment method for group decision making in an interval-valued intuitionistic fuzzy environment. Technological and Economic Development of Economy 19, 22–37 (2013).

[16] Diakoulaki,’:, D., Mavrotas, G. & Papayannakis, L. DETERMINING OBJECTIVE WEIGHTS IN MULTIPLE CRITERIA PROBLEMS: THE CRITIC METHOD. Computers Ops Res vol. 22 (1995).

[17] Si, S. L., You, X. Y., Liu, H. C. & Zhang, P. DEMATEL Technique: A Systematic Review of the State-of-the-Art Literature on Methodologies and Applications. Mathematical Problems in Engineering vol. 2018 Preprint at https://doi.org/10.1155/2018/3696457 (2018).

[18] Wu, W.-W., & Lee, Y.-T. (2007). Developing global managers’ competencies using the fuzzy DEMATEL method. Expert Systems with Applications, 32(2), 499–507. doi:10.1016/j.eswa.2005.12.005

[19] Mehdi Keshavarz Ghorabaee, Edmundas Kazimieras Zavadskas, Laya Olfat, Zenonas Turskis, Multi-Criteria Inventory Classification Using a New Method of Evaluation Based on Distance from Average Solution (EDAS), Informatica 26(2015), no. 3, 435-451, DOI 10.15388/Informatica.2015.57

[20] Zindani, D., Maity, S. R., & Bhowmik, S. (2019). Fuzzy-EDAS (Evaluation Based on Distance from Average Solution) for Material Selection Problems. In R. G. Narayanan, S. N. Joshi, & U. S. Dixit (Eds.), Advances in Computational Methods in Manufacturing (pp. 755–771). Singapore: Springer Singapore.

[21] par B. ROY.Classement et choix en presence de points de vue multiples. 8, 57-75 (1968)

[22] Rogers, M., Bruen, M., & Maystre, L.-Y. (2000). The Electre Methodology. In ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure Investment (pp. 45–85). doi:10.1007/978-1-4757-5057-7\_3

[23] B.BOY. Un algorithme de classemeuts fonde sur une representation floue des preferences en presence de cr1teres multiples.1978.

[24] Wei Yu. Aide multicritère à la décision dans le cadre de la problématique du tri Concepts, méthodes et applications.Thèse pour (’obtention du titre de Docteur en Méthodes Scientifiques de Gestion). 1992.

[25] Ju-Long, D. (1982). Control problems of grey systems. Systems & Control Letters, 1(5), 288–294. doi:10.1016/S0167-6911(82)80025-X

[26] Alinezhad, A., & Khalili, J. (2019). IDOCRIW Method. In New Methods and Applications in Multiple Attribute Decision Making (MADM) (pp. 133–141). doi:10.1007/978-3-030-15009-9\_19

[27] Pamučar, D. & Ćirović, G. The selection of transport and handling resources in logistics centers using Multi-Attributive Border Approximation area Comparison (MABAC). Expert Syst Appl 42, 3016–3028 (2015).

[28] Detlof V. Winterfeldt, Gregory W.Fischer. Multi-attribute utility theory: models and assessment procedures. Office of Naval Research Advanced Research Projects Agency, 1973.11.

[29] Karel, W., Brauers, W., & Zavadskas, E. (01 2006). The MOORA method and its application to privatization in a transition economy. Control and Cybernetics, 35.

[30] Ray, A. GREEN CUTTING FLUID SELECTION USING MOOSRA METHOD. IJRET: International Journal of Research in Engineering and Technology http://www.ijret.org.

[31] Brauers, W. K. M. & Zavadskas, E. K. Project management by multimoora as an instrument for transition economies. Technological and Economic Development of Economy 16, 5–24 (2010).

[32] Brans, J.-P., & Mareschal, B. (2005). Promethee Methods. In Multiple Criteria Decision Analysis: State of the Art Surveys (pp. 163–186). doi:10.1007/0-387-23081-5\_5

[33] Alinezhad, A., & Khalili, J. (2019). PROMETHEE I-II-III Methods. In New Methods and Applications in Multiple Attribute Decision Making (MADM) (pp. 29–39). doi:10.1007/978-3-030-15009-9\_5

[34] Panjaitan, M. I. Login : Jurnal Teknologi Komputer Simple Additive Weighting (SAW) method in Determining Beneficiaries of Foundation Benefits. Jl. Kol. Yos Sudarso No.45 AB 13, 19–25 (2019).

[35] Olson, D. L. (1996). Smart. In Decision Aids for Selection Problems (pp. 34–48). doi:10.1007/978-1-4612-3982-6\_4

[36] Yoon, K. (1987). A Reconciliation Among Discrete Compromise Solutions. Journal of the Operational Research Society, 38(3), 277–286. doi:10.1057/jors.1987.44

[36] Nădăban, S., Dzitac, S., & Dzitac, I. (2016). Fuzzy TOPSIS: A General View. Procedia Computer Science, 91, 823–831. doi:10.1016/j.procs.2016.07.088

[38] Thakkar, J. J. (2021). VIseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR). In Multi-Criteria Decision Making (pp. 129–138). doi:10.1007/978-981-33-4745-8\_8

[39] Opricovic, S. (2011). Fuzzy VIKOR with an application to water resources planning. Expert Systems with Applications, 38(10), 12983–12990. doi:10.1016/j.eswa.2011.04.097

[40] Michnik, J. (2013). Weighted Influence Non-linear Gauge System (WINGS) – An analysis method for the systems of interrelated components. European Journal of Operational Research, 228(3), 536–544. doi:10.1016/j.ejor.2013.02.007

[41] Thakkar, J. J. (2021). Weighted Aggregated Sum Product Assessment (WASPAS). In Multi-Criteria Decision Making (pp. 253–279). doi:10.1007/978-981-33-4745-8\_15

[42] Donegan, H. A. & Dodd, F. J. A NOTE ON SAATY’S RANDOM INDEXES. Mathl. Comput. Modelling vol. 15 (1991).

[43] Kahraman, C., Cebeci, U. & Ruan, D. Multi-attribute comparison of catering service companies using fuzzy AHP: The case of Turkey. Int J Prod Econ 87, 171–184 (2004).

[44] Chou, S. W. & Chang, Y. C. The implementation factors that influence the ERP (enterprise resource planning) benefits. Decis Support Syst 46, 149–157 (2008).

[45] Mirjalili, S., Mirjalili, S. M. & Lewis, A. Grey Wolf Optimizer. Advances in Engineering Software 69, 46–61 (2014).