车位分布的优化设计与评价

摘要

本文通过运用 T-检验，灰关联分析,蚁群算法、最优匹配 Kuhn-Munkras 等方法，采用熵权和粗糙集算法确定综合权重，建立了车位分布分配的加权模糊综合评价和优化模型，对住宅区地下车库车位分布综合评价相关问题进行了研究。

针对问题一，我们首先选取步行距离、行驶距离，时间成本、安全成本、转向成本、通行顺畅程度、空间利用效率等多个车位分布评价因素指标，然后采用灰关联分析遴选出最具代表性的八大因素，通过确定求权方法系数，综合熵权法和粗糙集算法确定综合权重，建立了加权模糊综合判定的系统评价模型，对车库车位分布分配合理性进行了研究。

针对问题二，我们采用问题一中建立的的综合加权模糊评价模型对附件一中的住宅区地下车库车位的分布合理性进行了评价分析并代入 MATLAB 编程运算，得出其模糊分布因数 $\mu = 72.056$ 远低于平均水平，说明附件一中车位分布分配合理性差，需要科学的方法优化提高。

针对问题三，我们对车位、出入口、电梯及匹配住户的楼层、户型位置进行了分析，将车位分配问题转化为产销平衡问题中的最大流最小费问题，利用图论相关知识建立了基于绒泡菌网络多入口多出口模型，采用蚁群算法和最佳匹配 KM 算法，对车位与住户的匹配关系做了深度划分和编程运算，并对 KM 最优模型进行求解得出最优车库车位设计方案。

本文的特色在于运用灰关联分析遴选指标并建立指标体系，采取熵权法和粗糙集确定综合权重建立模糊判断系统评价模型为以后研究最优车库车位分布分配规划设计问题提供了科学的理论依据。

关键词： 灰色关联度 粗糙集算法 模糊综合判断 蚁群算法 最优匹配 KM 算法

一、 问题重述

随着现代社会经济的快速发展，房地产成为国家经济发展中重要的经济增长点之一。而小区内汽车停车位的分布对于小区居民的上下班出行影响很大，题目要求我们根据题目背景进行如下分析研究：

问题一要求我们分析评判小区汽车停车位分布是否合理的几个关键指标，建立评判车位分布合理的数学模型。

问题二要求我们根据附件 1 是小区的汽车停车位分布方案，用问题 1 的模型评价此种车位分布的合理性。

问题三要求我们根据关于车位分布合理性的指标和评判模型，重新为附件 1 中的小区车位进行分配，给出最优分配方案。

二、 模型假设

1. 假设不考虑附件中未注明的地下车库立柱、遮挡物的相关信息。
2. 假设车库中每一个车位的几何坐标为其等效质点的几何中心坐标。
3. 假设综合权重系数为一定区间内的合理值，取$\alpha=0.5$。
4. 假设忽略车主主观因素影响，进行客观分析和评价。
5. 假设在灰关联分析和模糊评价过程中不考虑时间因素，进行静态分析。

三、 符号说明

符号	意义
X_{ij}	原始矩阵第i项第j个指标满意度原始评分
Y_{ij}	标准化处理后的评价决策矩阵元素
W_{ij}	熵权法和AHP综合权重
H_j	第j项指标的熵值
α	综合权重系数
P_{oi}	关联度系数修正常数

Q_{ij}	平均信息素流量
U	粗糙集论域
S	粗糙集决策表系统
d_error	车辆位移坐标点切线之相对距离误差
ξ	车位优化精度系数
ε	信息素影响因子
F	初始信息素量
η	分辨系数

四、问题分析

4.1 问题一分析

问题一要求建立一套系统的车位分布合理性评价模型，要解决此问题，首先要明确一套科学客观的测算方法，我们想到采用模糊综合判断对其合理性进行综合评价，采用灰关联算法遴选指标并建立指标体系，选取熵权法+粗糙集算法确定综合权重，对车库车位分布分配问题的合理性评价进行研究。

4.2 问题二的分析

问题二要求我们将附件一中的车位分布方案代入本文基于熵权法与粗糙集综合权重的模糊综合评价数学模型，使用 MATLAB 软件编程计算，得出了服务满意度评价指标体系中各指标的综合权重，并运用问题一中所建立的模糊综合判断模型对附件一中车库车位分布分配的合理性进行了系统评价。

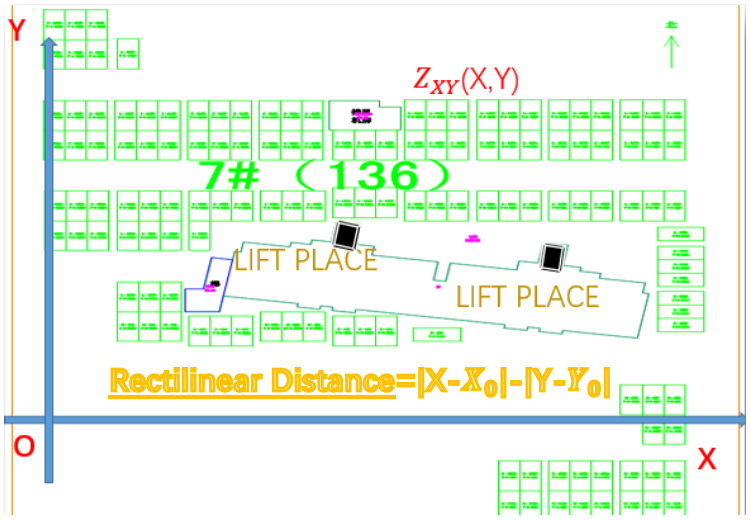
4.3 问题三分析

问题三中，要求综合考虑灰色关联度遴选后的 8 大因素，重新分配停车位，即将问题转换为产销平衡问题和停车位和住户的匹配问题。首先建立基于绒泡菌网络多入口多出口模型，采用蚁群算法对模型进行求解，将车位与电梯建立联系，再运用带权二分图的最优匹配 Kuhn-Munkras 算法匹配车位和住户，最终实现停车位和住户的最优匹配。

五、模型建立和解决

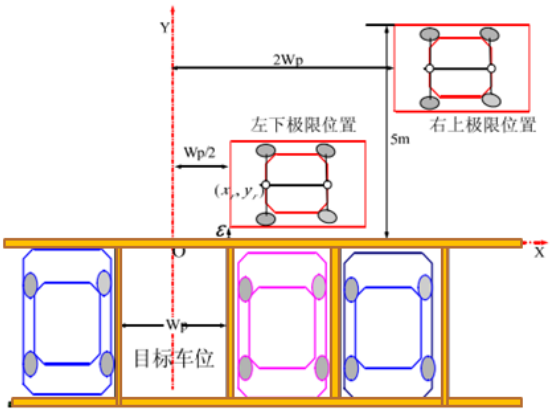
5.1 问题一的模型建立和解决

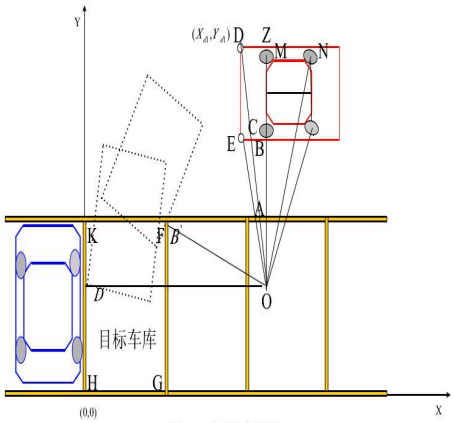
5.1.1. 坐标体系环境搭建



如图5-1所示，在分析7#楼地下车位分布图的空间排列和空间利用方式后，我们选择建立二维平面直角坐标系，从而便于对每一个车位以及每一个车位对应车辆驶入驶出过程中任意位置的表示以及拟合。

建立汽车运行轨迹的全过程分析的几何计算环境，分别根据入库、转弯、倒车、双向行车时同向异向的差异行驶计算阿克曼转角和构建各类因素分析的二维几何运算模型。





5.1.2 评价因素指标划分

5.1.2.1 评价因素集初步划分与解释

（1）平均步行距离

平均步行距离即业主从电梯或楼梯下至地下车库电梯口到车库自家车位出的实际距离，由于人活动的自由性，我们选择在搭建好的系统环境中用欧氏距离计算最终结果并进行所有住户的加权平均。

（2）平均行驶距离

平均行驶距离采用双向加权计算获得，即从自家车位开车行驶至左侧 C 出口的距离以及从外部驶入 C 出口到行驶到自家车位的距离，通过双向平均消去尽处停车场收费口的不确定性距离和时间成本因素。

（3）车库系统安全稳定性

车库在设计和安全上的合理性，保证行人和车辆同时在地下车库中的安全性。

（4）转向成本

分别计算不同位置不同住户所对应的车位在驶入驶出地下车库过程中的最小转向角度，最少转向次数以及转向综合成本。

（5）同行顺畅程度。

（6）综合时间成本

包括入库、出库、转向、等待等各个环节综合时间消耗的加权平均。

（7）楼层高度合理对应性

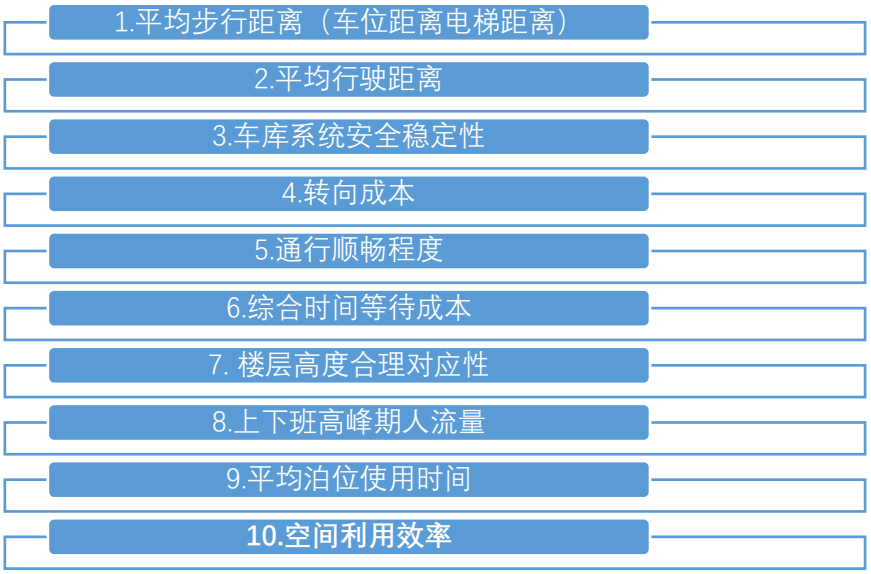
（8）上下班高峰期人流量

人流量的客观规律性分布在检验车库车位设计和分布分配上发挥重要作用，可以从实际情况检验车库车位合理性。

（9）平均泊位使用时间效率

（10）空间利用效率

为了对整个车库的车位分配合理性做出判断，我们划分出 9 个评判指标，但由于个别指标存在相关性差的缺点，本文首先采用灰色关联度对指标遴选，选择关联度高的指标建立指标体系



灰色关联分析法的基本思想是根据各比较数列集构成的曲线族与参考数列构成的曲线之间的几何相似程度来确定比较数列集与参考数列之间的关联度，比较数列构成的曲线与参考数列构成的曲线的几何形状越相似，其关联度越大。

5. 1. 2. 2 利用灰色关联分析进行综合评价的步骤

Step1：通过对原始数据设置进行对比分析和映射，得出相关车库车位距离和 9 大指标相关参数的相关指标列于备注中，如图

Step2：根据评价目的确定评价指标体系，收集数据并进行进行数据预处理，确定原始评价矩阵及参考数列。

$$X=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mj} & \cdots & x_{mn} \end{bmatrix}$$

根据指标目的及指标情况，设定参考列为

$$X_0=(x_{01},x_{02},...,x_{0j},...,x_{on})$$

Step3：对指标数据进行标准化处理，并记标准化处理后的数据序列为：

$$Y=\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1j} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2j} & \cdots & y_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{i1} & y_{i2} & \cdots & y_{ij} & \cdots & y_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{m1} & y_{m2} & \cdots & y_{mj} & \cdots & y_{mn} \end{bmatrix}$$

Step4：对标准化处理的数据序列，逐个计算每个被评价对象指标序列（比较序列）与参考数列对应元素的绝对差值，即：

$$\left|Y_I-Y_0\right|=\left|z_{ij}-z_{oj}\right|, i=1,2,...m,\ j=1,2,...n$$

Step5：确定最大与最小偏差差值

$$\min_{i=1}^{m}\left\{\min_{j=1}^{n}\left(\left|y_{ij}-y_{oj}\right|\right)\right\}$$

$$\max_{i=1}^{m}\left\{\max_{j=1}^{n}\left(\left|y_{ij}-y_{oj}\right|\right)\right\}$$

Step6：计算每个比较序列与参考数列对应元素的关联系数：

$$\frac{\min_{i=1}^{m}\left\{\min_{j=1}^{n}\left(\left|y_{ij}-y_{oj}\right|\right)\right\}+\eta*\max_{i=1}^{m}\left\{\max_{j=1}^{n}\left(\left|y_{ij}-y_{oj}\right|\right)\right\}}{\left|y_{ij}-y_{oj}\right|+\eta*\max_{i=1}^{m}\left\{\max_{j=1}^{n}\left(\left|y_{ij}-y_{oj}\right|\right)\right\}}$$

其中η为分辨系数，其值在区间（0,1）之间。η取值越小，关联系数间差异越大，

分辨能力越强，通常取 $\eta = 0.5$ 。关联系数 ξ_{ij} 是不超过 1 的正数，它反映第 i 个比较序列 X_i 与参考序列 X_0 在第 j 个属性上的关联程度。

Step7：计算关联度。对各评价对象（比较序列）分别计算其各个指标与参考序列对应元素的关联系数的均值，以反映各评价对象与参考序列的关联关系（即关联度），记为：

$$P = (p_{01}, p_{02}, \ldots, p_{oi}, \ldots, p_{om})^T, i = 1,2,\ldots,m$$

其中关联度的计算公式为：

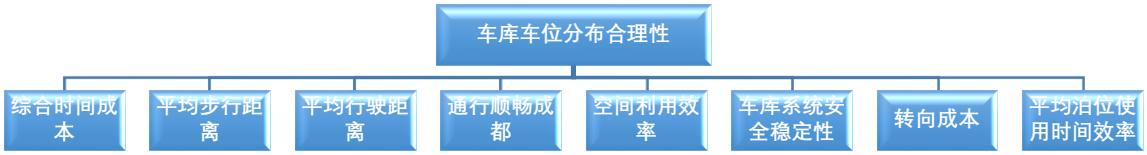
$$p_{0i} = \frac{1}{n}\sum_{j=1}^{n}\xi(j), i = 1,2,\ldots,n$$

Step8：依据各观察对象的关联度，得出综合评价结果。关联度越大，比较序列和参考序列的关系越密切，说明该指标能够更加有效反映车库车位分配设计的合理性，反之，关联度较小，说明两者的关系并没有达到能够有效反映车位分布合理性的综合评价的密切程度，故予以剔除。得到的最终关联度分析结果列表所示为：

关联度	指标序号	对应指标
0.8524	Q6	综合时间成本
0.8376	Q1	平均步行距离
0.8331	Q2	平均行驶距离
0.8273	Q5	通行顺畅程度
0.8156	Q10	空间利用效率
0.8119	Q3	车库系统安全稳定性
0.8109	Q4	转向成本
0.8068	Q9	平均泊位使用时间效率
0.7794	Q7	楼层高度合理对应性
0.7665	Q8	上下班高峰期人流量

通过灰关联分析和相关运算处理，我们选取低于关联度高于 0.800 的八个有高度关联性和合理性说服力的指标，将楼层高度的合理对应性和上下班高峰期人流量

因素剔除，从而遴选分析构成了综合评价系统的指标体系。



5.1.3 熵权法+粗糙集算法综合指标权重的确定

5.1.3.1 熵权法确定指标权重

熵权法相对于其他主观赋值方法，精度较高，可观性更强并且可以用于任何需要确定权重的过程，是一种集高度客观性和适应性为一体的权重确定方法。

Step1 建立原始评价矩阵，根据数据预处理结果建立如下原始评价矩阵

其中个待评项目，n 个评价指标，其中 y_{ij} 为第 j 个指标下第 i 个项目的评价值。

$$Y=\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1j} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2j} & \cdots & y_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{i1} & y_{i2} & \cdots & y_{ij} & \cdots & y_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{m1} & y_{m2} & \cdots & y_{mj} & \cdots & y_{mn} \end{bmatrix}$$

Step2 计算第 j 项指标的熵值 H_j

$$H_j=-k\sum_{i=1}^{n} f_{ij}\ln f_{ij},(j=1,2,...,m)$$

其中，$k=\frac{1}{\ln n}$，$f_{ij}=\frac{y_{ij}}{\sum_{i=1}^{n} y_{ij}}$，当$f_{ij}=0$时，$f_{ij}\ln f_{ij}=0$

Step3 计算第 j 项指标的熵权（权重） ω_j

$$\omega_j=\frac{1-H_j}{m-\sum_{j=1}^{m} H_j},(j=1,2,...,m)$$

$$0\le\omega_j\le 1,\text{且}\sum_{j=1}^{n}\omega_j=1$$

5.1.3.2 粗糙集算法确定指标权重

在车库车位优化方案的选择中，我们需要对距离、时间等几项关联度高的因素影响因素的权重进行计算，但是我们并不能获取足够的数据，也无法获得专家的打分，而粗糙集的最大优势在于其无需提供除问题所需处理数据之外的任何先验信息，所以我们决定采用在以粗糙集理论确定属性权重的方法，以此计算出的权重能一定程度上反映客观情况以及在客观上得到这些因素对车位分配优化过程中影响的大小。然后通过灵敏度检验最终得出能在一定程度上反映客观情况权重。

（1）基于粗糙集理论的购房影响因素权重计算模型

设有决策表系统$S=(U,A,V,F)$，且U是非空有限集合，称为论域，$A=C\cap D$，C,D，分别为条件属性与决策属性，$C\cap D=\varnothing$，V为属性的值域集，$V=U_{a\in A}V_a$，V_a是属性a的值域。$f:U\times A\to V$是一个信息函数，对于$\forall u\in U,x\in A$，存在$f(x,a)=V_a$。对任一子集$P\subseteq A$，定义U上的二元关系IND(P)如下:

$$IND(P)=\{(x,y)\in U\times U\mid \forall a\in p, f(x,a)=f(y,a)\}$$

IND(P)是U上的等价关系。用$U/IND(P)$来表示U的一个划分，对于$\forall x\in U$，它的P等价类定义为：

$$[X]_p=\{y\in U:(x,y)\in IND(P)\}$$

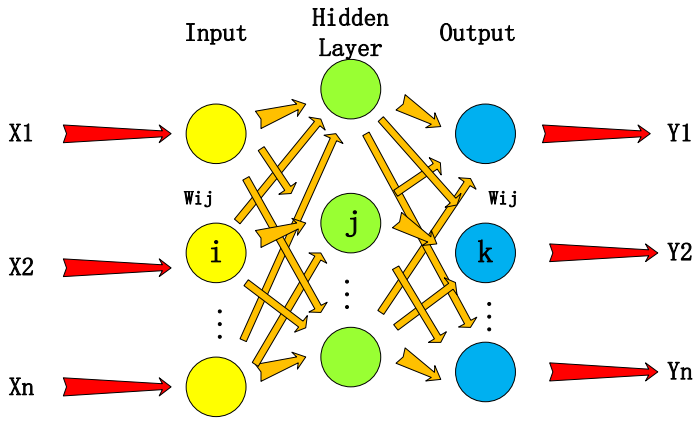
通常记为U/P。

（2）设$X\subseteq U$,R是U上的等价关系，当X为R的某些等价类的并时，称X是R上可定义的，否则称X是R上不可定义的，R可定义集称为R精确集，否则称为R粗糙集。粗糙集可用两个精确集，即粗糙集的下近似和上近似来表示，包含在X中的最大可定义集称为X的R下近似，包含X的最小可定义集称为X的R上近似，分别定义为：

$$\underline{R}X=\{x\in U\mid [x]_R\subseteq X\}$$

$$\overline{R}X=\{x\in U\mid [x]_R\cap X\neq\varnothing\}$$

集合$POS_R(X)=\underline{R}X$，称为X的R正域。



设有决策系统 $S=(U,C\cap D,V,f)$，决策属性 D 对条件属性 C 的依赖度定义为：

$$k=\gamma_c(D)=|POS_c(D)|/|U|$$

依赖度 $\gamma_c(D)$ 表示在条件属性 C 下能确切划入决策在 U/D 的对象占论域中总对象数的比率。

在决策系统 $S=(U,C\cap D,V,f)$ 中，$a_i\in C$ 的属性重要性定义为：

$$\sigma(C,D)(a_i)=[\gamma_c(D)-\gamma_{c-\{a_i\}}(D)]/\gamma_c(D)=sig(a_i)$$

则条件属性 a_i 的权重定义为：

$$W(a_i)=sig(a_i)/\sum sig(a_i)$$

$$U/C=\{U_1,\{U_2,U_3,U_4,U_5\},U_6,U_7,U_8,U_9,\{U_{10},U_{11},U_{12}\},U_{13},U_{14}\}$$

$$U/D=\{\{U_1,U_3,U_4,U_6,U_9\},\{U_2\},\{U_5,U_{11}\},\{U_7,U_{14}\},\{U_8\},\{U_{10}\},\{U_{12},U_{13}\}$$

$$U/(C-\{a\})=\{U_1,\{U_2,U_3,U_4,U_5\},U_6,U_7,U_8,\{U_9,U_{10},U_{11},U_{12},U_{13}\},U_{14}\}$$

$$U/(C-\{b\})=\{U_1,\{U_2,U_3,U_4,U_5\},U_6,U_7,U_8,\{U_9,U_{13}\},\{U_{10},U_{11},U_{12}\},U_{14}\}$$

$$U/(C-\{c\})=\{U_1,\{U_2,U_3,U_4,U_5,U_7\},U_6,U_8,\{U_9,U_{13}\},\{U_{10},U_{11},U_{12}\},U_{14}\}$$

$$U/(C-\{d\})=\{U_1,U_2,U_3,U_4,U_5,U_6\},\{U_7,U_8\},U_9,\{U_{10},U_{11},U_{12}\},\{U_{13},U_{14}\}\}$$

$$U/(C-\{f\})=\{U_1\},\{U_2,U_3,U_4,U_5\},U_6,U_7,U_8,U_9,\{U_{10},U_{11},U_{12}\},U_{13},U_{14}\}$$

从公式可以看出，去掉某一属性 a 得出的 $sig(a)$ 越大，说明属性 a 的重要度越高，其权重也越大。直接将各属性的重要度归一化作为权重，在一定程度上能反映属性的重要关系。

5.1.4 加权模糊综合判定

综合评判是对多种属性的事物，或者说其总体优劣受多种因素影响的事物，做出一个能合理地综合这些属性或因素的总体评判。例如，教学质量的评估就是一个多因素、多指标的复杂的评估过程，不能单纯地用好与坏来区分。而模糊逻辑是通过使用模糊集合来工作的，是一种精确解决不精确不完全信息的方法，其最大特点就是用它可以比较自然地处理人类思维的主动性和模糊性。因此对这些诸多因素进行综合，才能做出合理的评价，在多数情况下，评判涉及模糊因素，用模糊数学的方法进行评判是一条可行的也是一条较好的途径。

针对本文中的问题，首先需要制作出小区地下车库车位分布的原始表格，然后得出统计结果

5.1.4.1 确定因素集 F 和评定（语）集 E

因素集 F 即评价项目或指标的集合，一般有 $F=\{f_i\}$，$i=1,2,\ldots,n$。故本问题中的因素集为：$F=\{f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8,f_9,f_{10}\}$。评定集或评语集 E 即评价等级的集合，一般有 $E=\{e_j\}$，$j=1,2,\ldots,m$。故在本问题中的评定集为：$E=\{e_1,e_2,e_3,e_4\}=$｛好，较好，一般，差｝。

5.1.4.2 统计、确定单因素评价隶属度向量，并形成隶属度矩阵 R

隶属度是模糊综合评判中最基本和最重要的概念。所谓隶属度 r_{ij}，是指多个评价主体对某个评价对象在 f_i 方面作出 e_j 评定的可能性大小（可能性程度）。

隶属度向量：

$$R_i=(r_{i1},r_{i2},\ldots,r_{im},),i=1,2,\ldots,n,\ \sum_{j=1}^{m}r_{ij}=1$$

隶属度矩阵：

$R=(R_1,R_2,\ldots,R_n)^T=r_{ij}$。

在该问题中，n=10，m=4，根据数据统计分析求得隶属度矩阵。

5.1.4.3 确定权重向量W_F等

W_F为评价项目或指标的权重或权系数向量。本问题中，

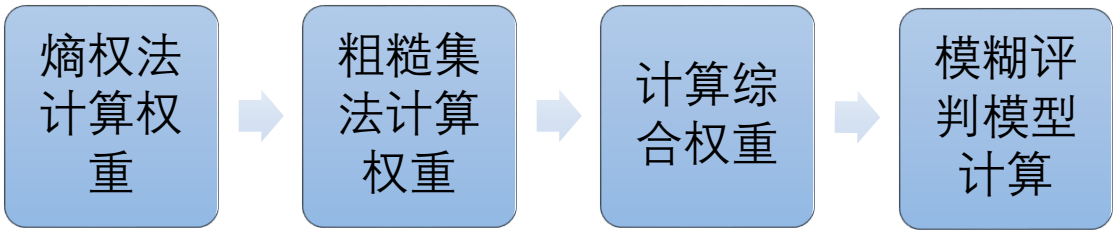
另外，还有评定（语）集的数值化结果（标准满意度向量），$W_E^{'}$或权重W_E（$W_E^{'}$归一化的结果）。

本问题我们使用W_F和$W_E^{'}$是“总分法”。

5.1.4.4 按某种运算法则，计算综合评定向量（综合隶属度向量）S 及综合评价 值（综合得分μ）

通常$S=W_F R$，$\mu=W_E^{'} S^T$。

5.2 问题二的模型建立和解决



根据附件 1 中地下车库车位分布以及问题一中建立的综合权重模糊综合评价模型，对该小区的汽车停车位分布方案评价此种车位分布的合理性。

5.2.1. 权重运算

5.2.1.1 熵权法计算各项评价指标权重

$$\omega_j=\frac{1-H_j}{m-\sum_{j=1}^{m}H_j},(j=1,2,...,m) \qquad 0\le\omega_j\le 1,\text{且}\sum_{j=1}^{n}\omega_j=1$$

熵权法权重计算结果：

指标	1	2	3	4	5	6	7	8
Weights	0.0364	0.0297	0.0283	0.0544	0.0815	0.0399	0.0374	0.0414

表 5.2.1　熵权法权重分析结果

5.2.1.2 粗糙集算法计算各项评价指标权重

粗糙集算法确定权重的结果：

指标	1	2	3	4	5	6	7	8
Weights	0.0358	0.0289	0.0286	0.0548	0.0811	0.0401	0.0378	0.0410

表 5.2.2　粗糙集分析结果

5.2.1.3 计算综合权重

把粗糙集算法和熵权法的结果相结合，得到综合考虑主客观因素的指标权重向量，即

$$\omega_j = \alpha\omega_j^{'} + (1-\alpha)\omega_j^{''} \quad (0 \le \alpha \le 1)$$

指标	1	2	3	4	5	6	7	8
Weights	0.0361	0.0293	0.0286	0.0548	0.0811	0.0401	0.0378	0.041

表 5.2.3　综合权重分析结果

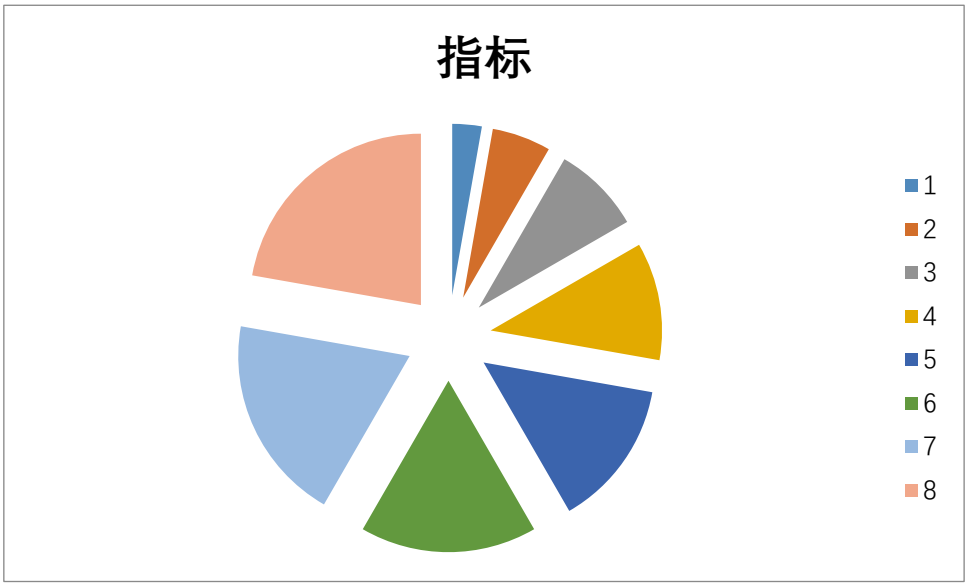


图 5.2.4　综合权重分析结果

5.2.1.4 模糊综合判定得到评价结果

采用问题一中建立的的综合加权模糊评价模型对附件一中的住宅区地下车库车位的分布合理性进行了评价分析并代入 MATLAB 编程运算，得出其模糊分布因数 $\mu = 72.056$ 远低于平均水平，说明附件一中车位分布分配合理性差，需要科学的方法优化提高。

评价结果 / 评价等级 / 评价项目及权重	好（100）	较好（85）	4 一般（70）	较差（55）
1. 步行距离（0.0361）	0.16	0.30	0.46	0.08
2. 行驶距离(0.0293)	0.04	0.20	0.36	0.40
3 . 综合时间成本(0.0286)	0.20	0.30	0.46	0.04
4. 通行顺畅程度(0.0548)	0.00	0.20	0.26	0.54
5. 空间利用效率(0.0811)	0.08	0.18	0.50	0.20
6. 转向成本(0.0401)	0.00	0.10	0.34	0.56
7. 车库系统安全稳定性(0.14)	0.00	0.08	0.74	0.18
8. 平均泊位使用时间效率(0.03)	0.06	0.14	0.76	0.02
综合隶属度	0.0772	0.1864	0.4062	0.3302
综合得分	72.0562			

表 5.2.4　模糊评价结果

5.3 问题三的模型建立与求解

5.3.1 车位分配问题结构模型

问题五中涉及的地下车位分配问题可以类比于产销平衡运输问题，将车位，电梯，住户分别类比于制造商，配送中心，消费者，建立如下结构模型：

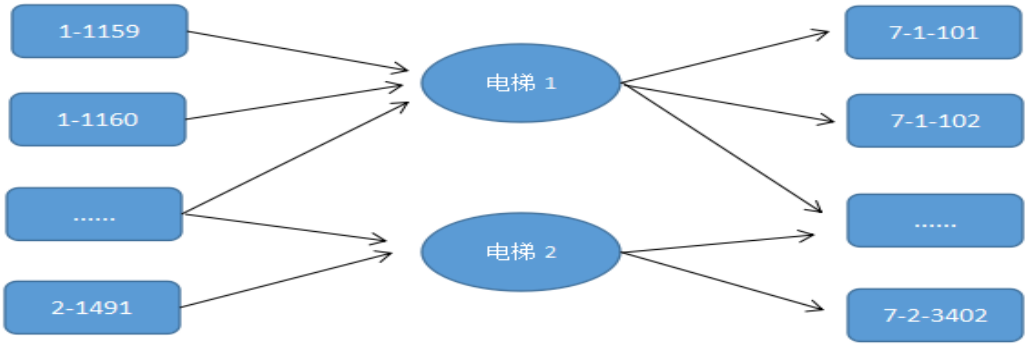


图 5.3.1　车位分配的三层结构

将问题转化成了产销平衡问题中物流配送网络的优化，即通过使用线性规划和人工智能算法等寻找最优路径。

5.3.2 基于绒泡菌网络多入口多出口模型的蚁群算法

基于绒泡菌网络多入口多出口模型的蚁群算法（PN_{MM}-ACO）的主要思想，来自于模型 PN_{SS} 在运算过程中体现的“重点管道重点培养”特性。在混合算法中，网络管道中流通的是信息素，并且“重点管道”流通的信息素较多，而非“重点管道”流通的信息素较少。在 PN_{MM}-ACO 运算过程中，除了考虑蚂蚁自身释放的信息素量外，仍需综合考虑流通的信息素量，以此更新整个网络中的信息素量。但是，如果仅用 PN_{SS} 模型中的两个点分别作为入口和出口，并不能完全解释整个管道网络中的信息素流通情况，所以将模型改进成 PN 多入口多出口的模型（PN_{MM}）。

（1）绒泡菌网络的多入口多出口模型（PN_{MM}）

如图 3.1 所示，设网络中管道条数为 M。t 时刻，将网络中每一条管道两端节点分别作为入口点和出口点，初始注入的信息素量定义为 $I_0:=I_0/M$ 然后利用公

式（3.1）求得管道中平均信息素流量Q_{ij} ;，并将Q_{ij}代入公式（2.13）后继续运算 t+1 时刻各管道的传导性，如此重复上述操作直至各管道的传导性不再变化为止。

$$\overrightarrow{Q_{ij}} = \frac{1}{M}\sum_{k=1}^{M} | Q_{ij}^{(k)} |$$

图 5.3.2(a)是初始网络，经过 PN 多入口多出口模型运算之后得到图 5-2(b)的网络结构。发现一些管道变得粗壮，一些管道变得非常细小（一些非常细小的管道在图 5.3.2(b)中没有显示出来，认为已经消失），这些剩下的管道，可仍称其为“重点管道”。

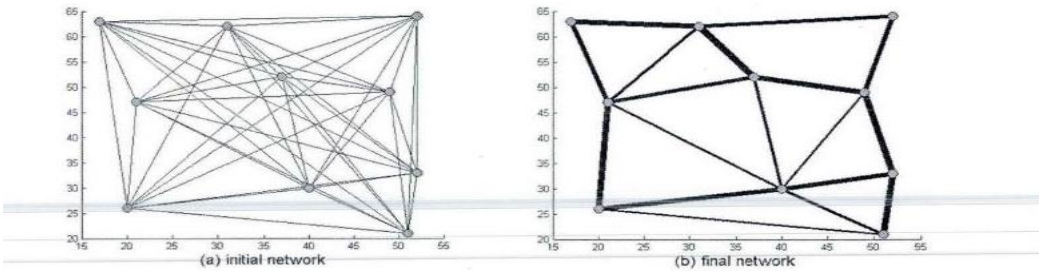


图 5.3.2　PN 多入口多出口模型

（2）基于信息素更新策略的蚁群算法（PN_{MM}-ACO）

在（PN_{MM}-ACO）算法中，为综合考虑管道中流通的信息素对于蚁群算法中信息素矩阵的影响，在 ACO 算法更新信息素矩阵时，将额外添加由绒泡菌网络多入口多出口模型运算获得的网络中流通的信息素量，这种策略称为绒泡菌网络多入口多出口信息素更新策略（以下简称 PN_{MM} 策略）。故将基本 AS、ACS 和 MMAS 算法中信息素更新优化为公式（3.2）、(3.3)和（3.4），式中ε表示管道流通信息素量整个算法中信息素总量的影响因子；$Q_{ij}(t)$表示 t 时刻管道五 Eij 流通的信息素量，且 $I_0=F/M$。公式(3.5)中 tempsteps 表示算法运算过程中绒泡菌网络流通信息素影响总步数。

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{k=1}^{m}\frac{F}{S_k}+\varepsilon\frac{Q_{ij}(t)}{I_0}$$

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\rho(\frac{F}{S_{best}(t)}+\varepsilon\frac{Q_{ij}(t)}{I_0})$$

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\frac{F}{S_{best}(t)}+\varepsilon\frac{Q_{ij}(t)}{I_0}$$

$$\varepsilon=1-\frac{1}{1+\lambda^{tempsteps/2-t}}$$

(3) 基于 PN_{MM} 的蚁群算法求解

如图所示,将应用策略的算法统称为绒泡菌网络多入口多出口模型的蚁群算法（。算法的具体描述如算法所示。

算法:绒泡菌网络多入口多出口模型的蚁群算法(PN_{MM}-ACS)

Step1：初始化信息素矩阵以及管道传导性矩阵为全 1 矩阵；迭代计数器 N:=0;

Step2：将 m 只蚂奴放置在各车位和房间，计算概率，选择下一个未经过的电梯，直至返回起点；

Step3：记录所有蚂蚁走过的路程长度，记录其中最优解 S_{min};

Step4：计算此时刻管道信息素流通量，计算下一时刻的管道传导性；

Step5：使用公式 (3.2)或 (3.3)或 (3.4)更新网络中信息素量，且 N=N+1;

Step6：如果 N<totalsteps(totalsteps 表示总迭代步数)，跳转 Step2;

Step7：输出最优解 S_{min};

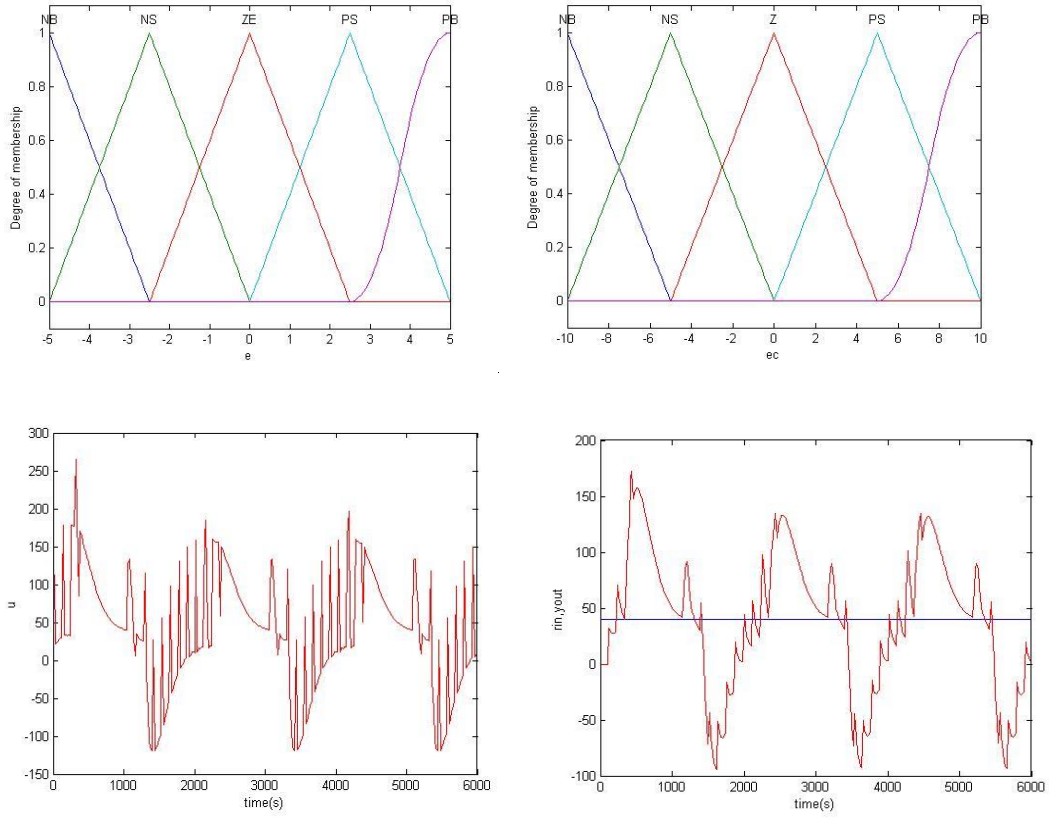


图 5.3.3 蚁群算法拟合图

(4) 参数确定

为分析 PNMM-ACO 算法对车位分配问题求解结果收敛性能和目标结果的影响，本文以 30 个车位的人工数据集为研究对象通过大量仿真实验研究，探讨 PNMM-ACO 算法中两个重要参数 :信息素影响因子ε和初始信息素总量 F 对算法的影响，从而选取较优的ε和 F 值进行算法调优。实验中选取 PNMM-ACO 中的 PNMM-ACS 算法和 ACO 算法中 ACS 算法进行ε和 F 的参数分析。实验中其他参数α、β、ρ、q_0。tempsteps、totaltimes 和重复运算次数的基础值分别设置为 1、2、0.7、0.9、300、300 和 50。

(1) 信息素影响因子ε

在 PN_{MM}-ACS 算法中，影响因子ε的值由公式(9)中λ的值所决定，讨论ε的取值范围则需讨论的λ取值范围。在 PN_{MM}-ACS 算法初期，管道中流通的信息素影响

力较强，随着迭代步数的增加，管道中的信息素影响力将逐渐下降，图 3.5 描述了ε与λ的关系。其中，当$\lambda=1$时，ε恒等于 0.5；而$\lambda\geq1.1$ 当时，影响因子ε的曲线在步数区间[145,155]上，将出现瞬间下降的现象，不利于算法收敛。

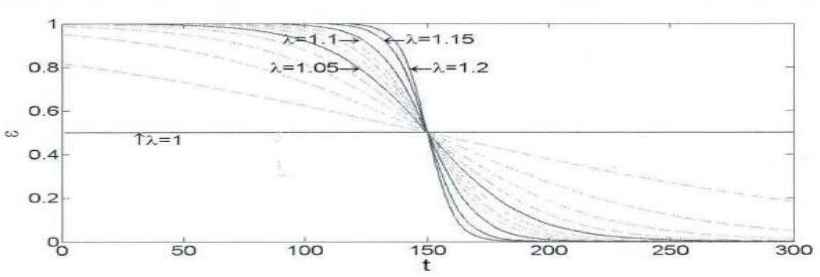


图 5-3　ε与λ的关系图

（其中虚线表示λ分别取值 1.01、1.02、1.03、1.04、1.06、1.07、1.08 和 1.09 时ε曲线）

另 F=100，则 30 个车位的类产销平衡问题在不同λ取值下$S_{average}$曲线图，如图 3.5 所示。图 3.6 中，当$\lambda\in[1.03,1.09]$，发现其结果较稳定，尤其当$\lambda=1.05$时能取得最优$S_{average}$值，而$\lambda\geq1.1$后，$S_{average}$值参差不齐，波动较大。因此，PNMM-ACS 算法中λ取值范围在[1.03,1.09]间最佳。

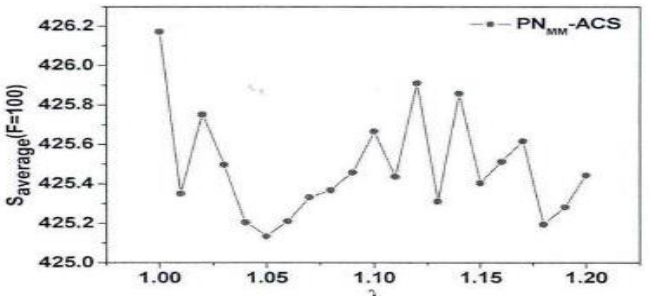


图 5-4

(2)初始信息素量 F

由于在 PN_{MM}-ACS 算法中，管道中流动的信息素量 I_0由蚂蚁携带的初始信息素 F 总量决定，因此在 PN_{MM}-ACS 算法中值 F 的选取尤为重要。当 F 值偏大时，管道中的信息量影响力将下降，容易出现过早收敛的现象；当 F 值偏小时，将会弱化蚂奴搜索最优路径的力度。

另$\lambda=1.05$，图 3.7 显示了在求解 30 个车位的类产销平衡问题时，不同 F 值下的$S_{average}$曲线图。图 3.7 (a)、(b) 分别对应$F\in[1,1000]$和$F\in[1,100]$时$S_{average}$曲线图。图 3.7 中，当$F\geq 200$时，$S_{average}$值不断上升，无法得到最优解。而$F\in[1,100]$间，曲线变化缓慢，算法收敛效果较好，且 F=20 时取得最优$S_{average}$值。

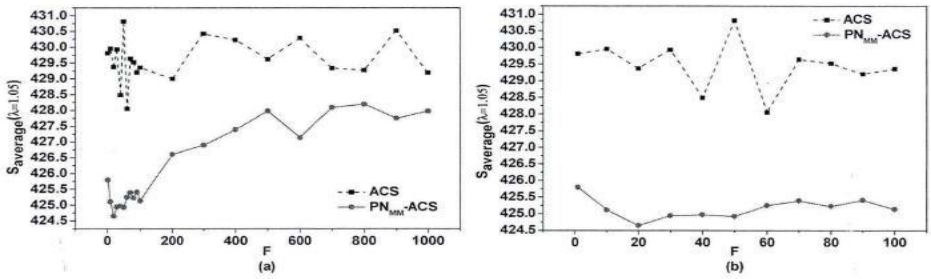


图 5-5 $\lambda=1.05$时不同 F 值下$S_{average}$曲线

5.3.3 带权二分图的最优匹配 Kuhn-Munkras 算法匹配车位和住户

5.3.3.1 最优匹配

G 是加权完全二分图，V(G)的二分图划分为 X，Y；

X={x1,...,xn},Y={y1,y2,...yn},w(xiyi)>=0 是住户 xi 使用 yi 车位时的效益，求权最大的完备匹配，这种完备匹配称为最佳匹配。

映射 l:V(G)->R，满足：任意 x∈X,任意 y∈Y,成立 l(x)+l(y)>=w(xy),则称 l(v)是二分图 G 的可行顶标；令 El={xy|xy∈E(G),l(x)+l(y)=w(xy)},称以 El 为边集的 G 之生成子图为相等子图，记为 Gl 可行顶标是存在的，例如 l(x)=max w(xy),x∈X;l(y)=0，y∈Y。

5.3.3.2 Kuhn-Munkras 算法

(0) 选定初始的可行顶标 l,确定 Gl,在 Gl 中选取一个匹配 M。

(1) X 中顶皆被 M 许配，止，M 即为最佳匹配；否则，取 Gl 中未被 M 许配的顶 u,令 S={u},T 为空。

(2) 若 N(S)真包含 T，转(3)；若 N(S)=T，取

al=min(l(x)+l(y)-w(xy)}(x∈S,y∈T),

l(v)-al,v∈S;l(v)= l(v)+al,v∈T;

l(v),其它则 l=l,Gl=Gl。

(3) 选 N(S)-T 中一顶 y,若 y 已被 M 许配，且 yz∈M,则 S=S∪{z},T=T∪{y},转(2)；否则，取 Gl 中一个 M 的可增广轨 P(u,y),令 M=M⊙E(P)，转(1)。

通过带入模型计算的出车位和住户的最终匹配方式和原始匹配方式分别为：

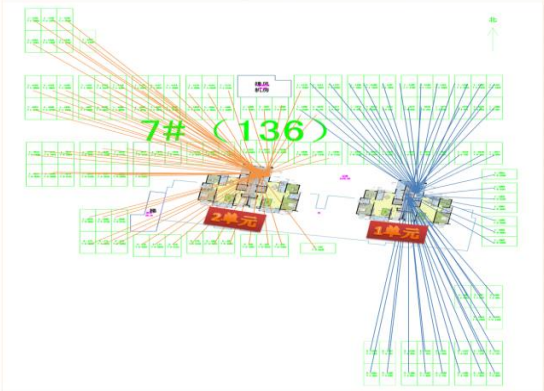


图 5-6-1 最终匹配

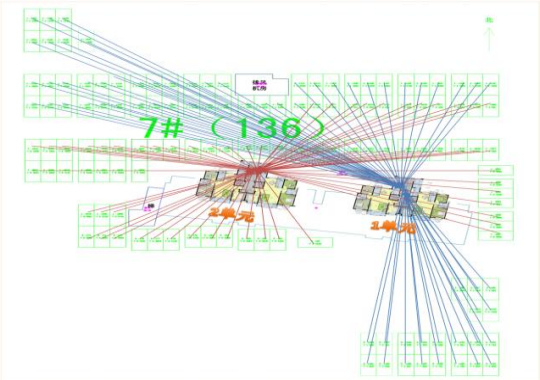


图 5-6-2 原始匹配

六、模型评价及改进

6.1 模型评价

本文最大的特点是综合运用熵权法、粗糙集理论和模糊综合评价模型，粗糙集的最大优势在于其无需提供除问题所需处理数据之外的任何先验信息，所以我们决定采用在以粗糙集理论确定属性权重的方法，以此计算出的权重能一定程度上反映客观情况。并且最大限度的减少了赋值的主观性，对问题作出了定性分析和定量计算。

其次，在第一问中运用了平面几何与解析几何分析，确定了各因素对车库车位分布的合理性的影响以及计算方法，然后采用灰色关联度分析评价排序确定在关联度更高更具代表性的评价指标，建立小区地下车库评价指标体系，用思路清

晰简单的方法去解决复杂的问题，简明易懂，并且基于严密的数学推导，求解过程严瑾，结果可信度高，说服力强。最后在第三问中综合先后运用蚁群算法和带权二分图的最优匹配 Kuhn-Munkras 算法，确保分析结果的可靠性。

6.2 模型改进和推广

此文中尚需改进的地方，考虑到数据分布较为均衡，我们忽略了具体车位的边界因素并在坐标运算中简化为了质点，此方法可能产生一些误差，在进行灰色关联度分析时，灰关联系数结果差距不大，因为存在系统误差，直接分析讨论最优和次优的指标和策略有些不当。

本文所建立的车位分布分配模型使用方便，充分结合了多种数据处理，分配权重和综合评价方法的优点，为设计规划人员提供了良好的方案，可以得出准确的综合评价结果,且评价结果客观,合理,对于其他一些指标评价问题同样适用,具有较广的适用范围和极大的使用价值。

七、参考文献

[1] 陈子良，居住小区地下车库设计要点控制研究，2012.8。

[2] 孙明宝，地下车库设计策略， 2012.8。

[3] 运筹学.北京：清华大学出版社。2012.8

[4] 李梅，改进的 K 均值算法在中文文本聚类中的研究，2010.6。

[5] 胡运康，景荣春，理论力学，北京：高等教育出版社，2006.5。

[6] 姚明洁，居住小区地下车库设计研究， 2009.8。

[12] 曹秀英，基于粗集理论的属性权重确定方法， 2002,10。

八、附录

8.1 数据预处理

同趋势化归一化

```
function [ std_attrValues ] = vecStd( attrValues )
%UNTITLED5 Summary of this function goes here
%     Detailed explanation goes here
if iscolumn(attrValues)~=1    % 如果不是指标数据列向量，则提示错误
      warndlg('输入数据必需为列向量，数据标准化处理失败，返回空值！','失败！');
      std_attrValues=[];          % 调用错误时返回空值
else
      m=length(attrValues);              % 列向量长度
      std_attrValues=ones(m,1);        % 初始化标准化属性值，与属性值一样是个 m×1 的列向
量
      sqAttr=attrValues.^2;              % 原数据每个值的平方
      sqSumByAttr=sum(sqAttr,1);       % 列向量的平方和
      sqrtByAttr=sqrt(sqSumByAttr); % 对平方和开根号
      if sqSumByAttr==0
            warndlg('指标值全为 0，不能采用向量归一化法进行标准化处理，返回空值，请选
择其他方法！','失败！');
            std_attrValues=[];     % 调用错误时返回空值
      else
            for i=1:m
                  std_attrValues(i,1)=attrValues(i,1)/sqrtByAttr;    % 计算标准化属性值
            end
            disp('数据标准化处理成功！');
      end
end
End
```

8.2 灰色关联度分析

```
clc,clear
load x.txt     %把原始数据存放在纯文本文件 x.txt 中
for i=1:22
      x(i,:)=x(i,:)/x(i,1);     %标准化数据
end
data=x;
n=size(data,1);
ck=data(1,:);m1=size(ck,1);
bj=data(2:n,:);m2=size(bj,1);
for i=1:m1
      for j=1:m2
         t(j,:)=bj(j,:)-ck(i,:);
```

```
    end
    jc1=min(min(abs(t')));jc2=max(max(abs(t')));
    rho=0.5;
    ksi=(jc1+rho*jc2)./(abs(t)+rho*jc2);
    rt=sum(ksi')/size(ksi,2);
    r(i,:)=rt;
end
r
```

8.3 熵权法和粗糙集

```
function [ weights ] = EntropyWeight( R )
% 熵权法求指标权重,R 为输入矩阵,返回权重向量 weights
[rows,cols]=size(R);    % 输入矩阵的大小,rows 为对象个数，cols 为指标个数
k=1/log(rows);          % 求 k

f=zeros(rows,cols);     % 初始化 fij
sumBycols=sum(R,1);     % 输入矩阵的每一列之和(结果为一个 1*cols 的行向量)
% 计算 fij
for i=1:rows
    for j=1:cols
        f(i,j)=R(i,j)./sumBycols(1,j);
    end
end
lnfij=zeros(rows,cols);  % 初始化 lnfij
% 计算 lnfij
for i=1:rows
    for j=1:cols
        if f(i,j)==0
            lnfij(i,j)=0;
        else
            lnfij(i,j)=log(f(i,j));
        end
    end
end

Hj=-k*(sum(f.*lnfij,1)); % 计算熵值 Hj
weights=(1-Hj)/(cols-sum(Hj));
end


testdata=[0.3214  0.3182  0.3212  0.3171  0.3106  0.3189  0.3169  0.316  0.3223
    0.3277  0.3202  0.3314  0.3214  0.3051  0.3297
0.3202  0.3206  0.3154  0.3143  0.3121  0.315  0.3173  0.3169  0.3185  0.3223
    0.3124  0.3138  0.3309  0.3137  0.3199
```

```
0.3278  0.3244  0.307   0.3061  0.3229  0.3102  0.3204  0.3222  0.3219  0.32
    0.3287  0.3089  0.3177  0.3208  0.3356
0.2998  0.3031  0.325   0.3331  0.2903  0.3364  0.2953  0.2938  0.2891  0.2855
    0.2808  0.2752  0.2949  0.3075  0.2831
0.3239  0.3255  0.3065  0.3048  0.3399  0.301   0.3292  0.3269  0.3262  0.3226
    0.338   0.3141  0.2917  0.3317  0.3093
0.3106  0.316   0.3219  0.3256  0.3089  0.3211  0.3149  0.3137  0.3134  0.3162
    0.3109  0.3064  0.3086  0.3098  0.3128
0.3098  0.3031  0.3264  0.3299  0.311   0.3225  0.3079  0.3075  0.3083  0.309
    0.3037  0.3385  0.3061  0.3067  0.3228
0.3154  0.3232  0.3055  0.2997  0.3311  0.31 0.3221  0.3234  0.3199  0.3171
    0.3279  0.3197  0.3144  0.3206  0.3149
0.3249  0.3163  0.3119  0.3114  0.3137  0.3118  0.3201  0.3191  0.3229  0.3207
    0.322   0.3369  0.3539  0.3266  0.3185
0.3074  0.3109  0.3205  0.3186  0.3191  0.314   0.317   0.3216  0.3181  0.3192
    0.3141  0.3125  0.3182  0.3186  0.313]
weights = EntropyWeight( testdata )
```

8.4 模糊综合判定

```
p;  ColAverage(i)=ColAverage(i)/m;

  %使用隶属函数，做预处理对每一列进行排序

  SortedMatrix=a;

  for j=1:n

  for i=1:m

  for k=i:m

  if SortedMatrix(i,j)>SortedMatrix(k,j)

  tmp=SortedMatrix(i,j);

  SortedMatrix(i,j)=SortedMatrix(k,j);

  SortedMatrix(k,j)=tmp;

  end

  end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

%计算单因素隶属度

c=SortedMatrix;

for j = 1 : n

for i = 1 : m

for k = 1 : m

if a(i, j) == c(k, j)

if k == 1

if b(j) < c(k, j)

LSDResult(j, i) = 1;

end

if b(j) >= c(k, j) & b(j) < c(k + 1, j)

LSDResult(j, i) = ((c(k + 1, j) - b(j)) / (c(k + 1, j) - c(k, j)));

end

if b(j) >= c(k + 1, j)

LSDResult(j, i) = 0;

end

end

if k > 1 & k < m
```

```
if b(j) < c(k - 1, j)
LSDResult(j, i) = 0;
end
if b(j) >= c(k - 1, j) & b(j) < c(k, j)
LSDResult(j, i) = ((b(j) - c(k - 1, j)) / (c(k, j) - c(k - 1, j)));
end
if b(j) >= c(k, j) & b(j) < c(k + 1, j)
LSDResult(j, i) = ((c(k + 1, j) - b(j)) / (c(k + 1, j) - c(k, j)));
end
if b(j) >= c(k + 1, j)
LSDResult(j, i) = 0;
end
end
if k == m
if b(j) < c(k - 1, j)
LSDResult(j, i) = 0;
end
%权重乘以单因素隶属度得到最终结果
R=LSDResult;
E=EResult;
FuzzyEvalution=E*R;
```

```
TheResultMoHu=[TheResultMoHu;FuzzyEvalution];

end

TheResult
```

8.3 KM 算法（C 语言）

```
#include <stdio.h>
#include <string.h>
#define M 310
#define inf 0x3f3f3f3f

int n,nx,ny;
int link[M],lx[M],ly[M],slack[M];       //lx,ly 为顶标，nx,ny 分别为 x 点集 y 点集的个数
int visx[M],visy[M],w[M][M];

int DFS(int x)
{
    visx[x] = 1;
    for (int y = 1;y <= ny;y ++)
    {
        if (visy[y])
            continue;
        int t = lx[x] + ly[y] - w[x][y];
        if (t == 0)             //
        {
            visy[y] = 1;
            if (link[y] == -1||DFS(link[y]))
            {
                link[y] = x;
                return 1;
            }
        }
        else if (slack[y] > t)    //不在相等子图中 slack  取最小的
            slack[y] = t;
    }
    return 0;
}
int KM()
{
    int i,j;
    memset (link,-1,sizeof(link));
    memset (ly,0,sizeof(ly));
    for (i = 1;i <= nx;i ++)                    //lx 初始化为与它关联边中最大的
```

```
        for (j = 1,lx[i] = -inf;j <= ny;j ++)
            if (w[i][j] > lx[i])
                lx[i] = w[i][j];

    for (int x = 1;x <= nx;x ++)
    {
        for (i = 1;i <= ny;i ++)
            slack[i] = inf;
        while (1)
        {
            memset (visx,0,sizeof(visx));
            memset (visy,0,sizeof(visy));
            if  (DFS(x))        //若成功（找到了增广轨），则该点增广完成，进入下一个点的增广
                break;  //若失败（没有找到增广轨），则需要改变一些点的标号，使得图中可行边的数量增加。
                        //方法为：将所有在增广轨中（就是在增广过程中遍历到）的 X 方点的标号全部减去一个常数 d，
                        //所有在增广轨中的 Y 方点的标号全部加上一个常数 d
            int d = inf;
            for (i = 1;i <= ny;i ++)
                if (!visy[i]&&d > slack[i])
                    d = slack[i];
            for (i = 1;i <= nx;i ++)
                if (visx[i])
                    lx[i] -= d;
            for (i = 1;i <= ny;i ++)   //修改顶标后，要把所有不在交错树中的 Y 顶点的 slack 值都减去 d
                if (visy[i])
                    ly[i] += d;
                else
                    slack[i] -= d;
        }
    }
    int res = 0;
    for (i = 1;i <= ny;i ++)
        if (link[i] > -1)
            res += w[link[i]][i];
    return res;
}
int main ()
{
    int i,j;
    while (scanf ("%d",&n)!=EOF)
```

```
    {
        nx = ny = n;
      //   memset (w,0,sizeof(w));
        for (i = 1;i <= n;i ++)
            for (j = 1;j <= n;j ++)
                scanf ("%d",&w[i][j]);
        int ans = KM();
        printf ("%d\n",ans);
    }
    return 0;
}
```