# 基于班级条件权重表的试探性学生分班算法

**——算法思路详解**

**作者：王金塑**

## 目录

[基于班级条件权重表的试探性学生分班算法 1](#_Toc16714)

[1学生分班业务分析 3](#_Toc23734)

[1.1需求分析 3](#_Toc32423)

[1.2难点分析 3](#_Toc18532)

[2分班问题的解决思路 4](#_Toc30701)

[3基于班级条件权重表的试探性学生分班算法 5](#_Toc12484)

[3.1算法步骤 5](#_Toc9949)

[3.2算法详解 6](#_Toc15850)

[3.2.1权重表数据结构 6](#_Toc15145)

[3.2.2权重计算规则 8](#_Toc12522)

[3.2.3更新权重表 9](#_Toc10520)

[3.3算法解释 9](#_Toc3693)

[3.3.1建立权重表以及计算权重的原因 9](#_Toc9795)

[3.3.2区别对待第一个未分班的学生以及跳过分班选择不唯一的学生的原因 9](#_Toc13036)

[3.4算法复杂度分析 10](#_Toc19809)

[3.4.1时间复杂度 10](#_Toc17890)

[3.4.2空间复杂度 10](#_Toc10547)

[3.5算法扩展性 11](#_Toc27506)

[3.5.1规则权重系数的扩展性 11](#_Toc10256)

[3.5.2算法对聚合型分班规则的扩展性 12](#_Toc20563)

# 1学生分班业务分析

## 1.1需求分析

学生分班问题是每所高校在迎新过程中必须面对的难题。

学生分班的业务意义在于需要根据不同学校的学生特点，通过将具有某种或某几种共同特点的学生进行打散分班，使得每个班级的学生群体更加平均，避免出现同属性学生扎堆的情况。

学校大多采取下列常见的分班策略中的一种或几种：

1. **尽量将同名的学生分到不同的班级**（避免同班同名）；
2. **尽量使得每个班级的男女比例接近总的男女比例**（避免男女比例失衡）；
3. **尽量将来自同一生源地的学生分到不同的班级**（避免同源地学生扎堆）；
4. **尽量使得民族相同的学生分到不同班级**（避免少数民族扎堆）；
5. **尽量使得每个班的学生的高考分数都尽量分散**（使学生的学习能力具有层次性）；

在现实中，学生分班的策略需要根据学校的学生类型特点来进行针对性的选择：

民族类的大学，会比其他类型的学校更需要根据学生的民族属性来打散分班；

师范类的大学，会比其他类型的学校更需要根据学生的性别属性来打散分班；

绝大部分学校都需要尽可能地在分班时打散同名、同生源地的学生；

因此不同的分班策略对于不同的学校来说有着不同的优先级，并且学校往往需要选择多种策略来进行学生的打散分班。

## 1.2难点分析

实现上述分班需求的难点如下：

1. **分班的打散规则可由学校自行选择并指定优先级；**
2. **不同学校的学生特点各不相同，必须保证用一套算法在不同的学生数据、不同的分班 打散规则下都能做到分班结果接近最优；**
3. **学校新生数据量可能达到1W+，因此对分班算法的性能有要求；**

# 2分班问题的解决思路

如果某一个专业的男女比例是3比2，假设按照性别打散的规则进行分班，最优的分班结果是什么？当然是这个专业下的每个班级的男女比例都是3比2最好，假设按照相同生源地打散的规则进行分班，最优的分班结果是什么？当然是每个班的学生都来自五湖四海最好。

那么如果对该专业先按照性别打散、再按照生源地打散，那么最优的分班结果应该是什么？自然是在保证每个班级男女比例大致在3比2这个比例的同时，使得每个班的学生都尽量来自五湖四海，尽可能避免出现某个班的学生全是来自同一个省份的情况。

这仅仅是同时采取两个打散规则，看起来就已经很麻烦了，如果再加上同姓名打散、同民族打散、高考分数打散等等条件，情况就变得更糟了，似乎难以找到一个很好的解决方案。

其实用分治的算法思想来看待这个问题就显得简单直白了，如果能够在为每一个学生进行分班的时候，都能保证给该学生分的班级是当前打散规则下最适合他的，那么按照这样的规则，每个学生都能进入到最适合他的班级，那么自然学生分班的结果就是最优的。

于是通过分治，可以将一个看上去很笼统而且复杂的求整体学生分班最优问题，划分成了求单个学生的分班最优问题。

那么如何保证在为单个学生分班的时候，能挑选出最适合该学生的班级？

举例：

假设按照民族打散分班，并且现在有3个班级

假设在分班过程中，这三个班级内的已有学生的民族信息如下：

班级1：4个汉族，3个苗族；

班级2：5个汉族，1个苗族，1个藏族；

班级3：1个土族、3个回族、1个蒙古族，2个苗族；

接下来看看给下一个学生进行分班时，学生民族不同时的最优班级选择：

1. 学生民族：汉族 分班最优选择：班级3；原因：班级3没有汉族；
2. 学生民族：苗族 分班最优选择：班级2；原因：班级2苗族人数最少；
3. 学生民族：藏族 分班最优选择：班级1或班级3；原因：班级1、3都没有藏族；

是什么支撑我们在上述例子中给不同民族的学生进行分班决策的？

答案是每个班级中与当前学生民族相同的学生数量。

如果将班级中与当前学生民族相同的学生数量称作班级对于当前学生的民族权重(weight)，并且将上述决策过程抽象化会是这样的：

对于某R值为V的学生（分班规则：按照R打散）

1. 遍历所有班级
2. 计算班级对于当前学生的R权重：weight = 班级中R值等于V的学生总数
3. 记录该班级的R权重
4. 选出R权重最小的班级(一个或多个)；

另外从上述的过程中可以思考得出一个结论，第i (0<i<n,n为待分班学生总数) 个学生的分班结果，可能会对第i+1个学生的分班决策产生影响，因为当第i个学生进入某个班级后，对于第i+1个学生来说，计算第i个学生进入的班级的权重时，就要加上第i个学生的相应值，因此计算的结果就和第i个学生进入该班级前有所变化，这就很有可能导致适合第i+1个学生进入的班级发生变化。

# 3基于班级条件权重表的试探性学生分班算法

基于上述的一系列分析，可以得知，一个可以达到分班结果最优的算法，一定要能保证为每个学生分班时都能从当前所有班级中选择出最适合该学生的班级。因此基于上述分析中提到的班级属性权重的概念，本文设计了“基于班级条件权重表的试探性学生分班算法”。

## 3.1算法步骤

1. **获取用户在页面上配置的打散规则序列**

（

规则在序列中的顺序即为优先级，

如序列：{1.姓名2.生源地3.民族}

则该序列代表优先按照姓名打散，其次按照生源地打散，最好按照民族打散

）

**2.遍历学校未分班的每个专业：**

**(1)** 获取专业下所有未分班的学生列表；

**(2)** 获取专业下的班级列表，并为每个班级建立权重表；

**(3)** 遍历该专业的未分班学生列表：

**A.**计算每个班对于该学生的权重

**B.**如果当前学生是**当前学生列表中第一个未分班的学生**，则将学生分入对于该学生来说权重最小且人数未分满的班级（有多个班级满足时取第一个），并根据该学生的属性更新班级的权重表

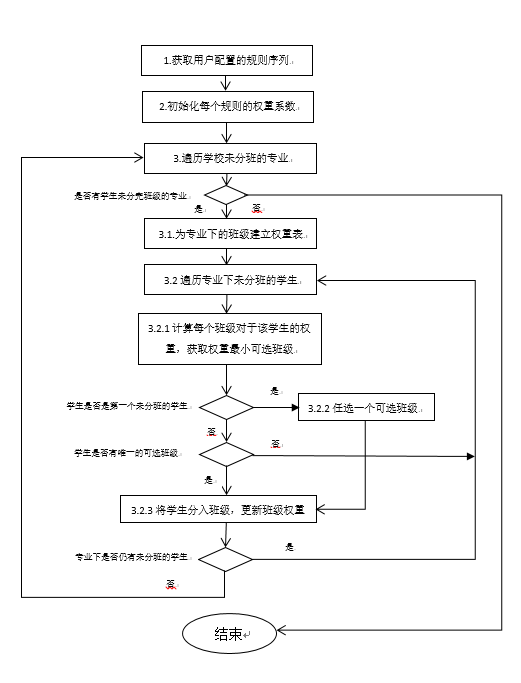
将当前学生标记为已分班；

**C.**如果当前学生不是**当前学生列表中第一个未分班的学生**，则如果有唯一的权重最小的班级，直接分班，更新所分班级的权重表，并且标记该学生为已分班，否则如果有多个权重最小的班级，则跳过该学生，等待下一轮分班；

**(4)** 判断是否所有学生都被标记为已分班，如果仍有未分班的学生，继续执行第(3)步；

**3.分班完成，进行入库操作。**

**算法流程示意图：**



## 3.2算法详解

### **3.2.1权重表数据结构**

对于规则序列：{R1,R2,R3...,Rn}，

有班级权重表：

{

R1:

{

V1:C1,

V2:C2,

V3:C3,

...

Vm:Cm

},

R2:

{

V1:C1,

V2:C2,

V3:C3,

...

Vk:Ck

},

R3:

{

V1:C1,

V2:C2,

V3:C3,

...

Vq:Cq

},

...

Rn:

{

V1:C1,

V2:C2,

V3:C3,

...

Vp:Cp

}

}

其中V代表属性R的具体值，C代表班级中属性值为V的学生数量。

举例：

若规则序列为：{1.姓名2.生源地3.民族}

则班级的初始权重表为：

{

姓名:{},

生源地:{},

民族:{}

}

假设在分班的过程中，某个班级的学生数据如下：

1. “张三、上海、汉族”；

2.“李四、贵州、汉族”；

3.“王五、贵州、苗族”；

则该班级此时的权重表为：

{

姓名:

{

张三：1，

李四：1，

王五：1

}，

生源地:

{

上海：1，

贵州：2

}，

民族:

{

汉族：2，

苗族：1

}

}

### **3.2.2权重计算规则**

对于规则序列：R\_List={R1,R2,R3...,Rn}，

当前学生的每个规则对应的属性序列{V1，V2，V3...Vn}，

则班级(权重表名称为weightMap)对于当前学生的权重计算公式为：

weight := 0;

For(i=0;i<n;++i)

{

weight += weightMap.get(Ri).get(Vi)\*Ri的权重系数;

}

其中权重系数是为了体现规则的优先级的，权重系数有不同的计算策略，如：取规则在规则序列中的反序: n-(i+1)+1，或者取规则在规则序列中的反序\*2: [n-(i+1)+1]\*2，又或者取2的规则反序次方: 2^[n-(i+1)+1]等等，只要保证使得序列越靠前的规则的权重系数越大即可。

举例：

若规则序列为：{1.姓名2.生源地3.民族}

为属性为“张三，贵州，汉族”的学生分班，则每个班级对于该学生的权重计算公式如下(权重系数取规则的反序)：

weight = weightMap.get(“姓名”).get(“张三”)\*3

+weightMap.get(“生源地”).get(“贵州”)\*2

+weightMap.get(“民族”).get(“汉族”)\*1;

### **3.2.3更新权重表**

如属性为“张三，贵州，汉族”的学生将使得其进入的班级的权重表中的“姓名”权重中“张三”的权重值、“生源地”权重中“贵州”的权重值、“民族”权重中“汉族”的权重值分别加1。

## 3.3算法解释

算法的核心思路有**4**点：

1. **为每个班级建立一个可以反映出在某个时刻班级中各种属性学生数量的权重表**；
2. **为每个学生分班时，都去计算所有人数未分满的班级的相对当前学生的权重值，获取适合当前学生进入的可分配班级或班级列表**；
3. **如果是第一个未分班的学生，则有适合的班级就直接分班**；
4. **如果不是第一个未分班的学生，则只当有唯一适合该学生的班级时才分班，否则跳过该学生，先为之后的学生进行分班**。

### **3.3.1建立权重表以及计算权重的原因**

第1、2点使得班级对于每个学生在进行分班时，其内部的学生情况都是可见，可计算的，这样就可以进行班级和学生匹配与否的判断；

### **3.3.2区别对待第一个未分班的学生以及跳过分班选择不唯一的学生的原因**

第3点保证每遍历一轮学生列表，都至少能给列表中第1个未分班的学生进行分班，可以保证分班轮次最多N轮（N为学生总数）就会结束；

第4点是本算法名称中试探性三字的体现，这是为了尽量使得每个学生都进入最适合该学生的班级，即通过班级权重计算后，有且仅有一个合适的班级。

之前在分析分班问题时，提出了一个结论:每个学生进入某个班级后，都会对后续的学生分班决策产生不可预知的影响，

根据这个结论，当为某个学生进行分班时，计算出适合这个学生的班级有多个，导致无法直接确定学生该进入哪个班级时，先略过这一类的学生，并且去试探后续学生的是否能够计算出唯一的权重最小班级进行直接分班，如果有的话，代表在这一轮分班轮次中，有部分班级的情况更适合某些学生。

由于每一轮的跳过了可选班级不唯一的学生，这就保证了每一轮分班都只为具有最优班级选择的学生分班。这就有可能导致在下一轮分班时，上一轮那些可选班级不唯一的学生，由于受到上一轮完成分班的学生的影响，这些之前无法做出唯一选择的学生可以在本轮分班中计算出唯一的班级进行分班。

这样一来，可以使得在为当前学生分班时，能够进行更全局范围的考虑。如果对每个学生都是随意选择一个可选班级分班，就忽略了本次分班的结果可能会对之后的学生的分班决策造成的影响，从而难以接近最优的分班结果。

## 3.4算法复杂度分析

### 3.4.1时间复杂度

对于某个专业来说，分班的复杂度为：

假设专业下未分班的人数为n，班级个数为m；

考虑最糟糕的情况，即每个学生都无法计算出唯一的班级进行分班，则意味着每一轮分 班只能给第一个学生进行分班，所以共需要进行n轮分班才能将n个学生分班完毕，并 且为每个学生分班时，都需要计算所有班级对于该学生的权重，则时间复杂度的计算公 式为：

时间复杂度 = n\*m + (n-1)\*m + (n-2)\*m + ... + 2\*m + 1\*m

= m\*(n + n-1 + n-2 + ... + 1)

= m\*[n\*(n+1)/2]

= m\*O(n2)

假设共有k个专业：

总时间复杂度 = m1\*O(n12) + m2\*O(n22) + ... + mk\*O(nk2)

= (m1+m2+...+mk) \* O([n1+n2+...+n3]2)

= M\*O(N2)

其中M代表班级总数，N代表学生总数

由于一般情况下学校一个年级的班级总数小于500，学生总数小于10000

即 500 << 10000\*10000 = 100000000

因此最终得到的时间复杂度为：O(N2)

该算法的时间复杂度和待分班学生的总数成平方关系。

### 3.4.2空间复杂度

由于每次只取一个专业的班级和学生进行分班，所以空间复杂度仅考虑单个专业即可。

假设专业下未分班的人数为N，班级个数为M，班级人数分别为n1,n2,…,nm，分班规则个数为K；

1.需要一个大小为N的列表存储学生列表： O(N)

2.需要一个大小为M的列表存储班级列表： O(M)

3.需要一个大小为K的列表存储规则列表： O(K)

4.每个班都需要建立权重表，权重表分为两层，第一层存储每个规则，第二层存储每个规则的各种属性在班级中学生人数，考虑最糟情况，即班级中的每个学生对于每个规则的属性值都不同： O(K\*n1)+ O(K\*n2)+…+ O(K\*nm)

= O(K\*(n1+n2+…+nm))

= O(KN)

总空间复杂度 = O(N) + O(M) + O(K) + O(KN)

= O(N+M+K+KN)

= O(M+K+(K+1)N)

由于一般情况下专业中的学生数大约在500以下，班级个数在10以下，分班规则个数一般在10以下

即 M + K = 10+10 << (K+1)N = 11\*500 = 5500

因此最终得到的时间复杂度为：O(KN) K为分班规则个数，N为专业下的学生数量

该算法的空间复杂度为配置的分班规则个数与待分班学生的总数乘积。

## 3.5算法扩展性

### 3.5.1规则权重系数的扩展性

上文在**3.2.2权重计算规则**一节中介绍了权重系数的概念，权重系数体现了规则在规则序列中的优先级，上文介绍说权重系数的计算规则可以有多种，只要能使得规则序列中越靠前的规则的系数越大而越靠后的规则的系数越小即可。

然而权重系数的计算规则，其实是受到在现实中的概念影响的。

举例：

假如现在有8个条件，其中第1个条件为按民族打散，第8个条件为按生源地打散。

假设为一个“汉族、上海”的学生进行分班，如果此时有以下两个班级

第1个班级中有1个学生，学生为汉族；

第2个班级中有8个学生，学生都来自上海；

下面考虑使用不同的权重系数时，计算这两个班级对于学生的权重有何不同：

1. 取规则在规则序列中的反序: n-(i+1)+1：
   1. 民族规则的权重系数为8 –(0+1)+1 = 8
   2. 生源地规则的权重系数为 8-(7+1)+1 = 1

班级1的权重：w1 = 8\*1 + 1\*0 = 8

班级2的权重：w2= 8\*0 + 1\* = 8

结果表示在这种权重系数的计算规则下，对于“汉族、上海”的学生来说，1个汉族的学生的权重顶的上8个上海的学生的权重，也就是说如果有一个有1个汉族学生，而另一个班有7个上海学生时，基于民族、生源地打散的规则，就会把当前这个“汉族、上海”的学生分到有7个上海学生的班级中去，而不是分到有1个汉族学生的班级，由此体现民族规则的高优先级，生源地规则的低优先级。

1. 取规则在规则序列中的反序的两倍: [n-(i+1)+1]\*2：
   1. 民族规则的权重系数为[8 –(0+1)+1]\*2 = 16
   2. 生源地规则的权重系数为 [8-(7+1)+1]\*2 = 2

班级1的权重：w1 = 16\*1 + 1\*0 = 16

班级2的权重：w2= 16\*0 + 2\*8 = 16

结果表示在这种权重系数的计算规则下，对于“汉族、上海”的学生来说，1个汉族的学生的权重顶的上16个上海的学生的权重，也就是说如果有一个有1个汉族学生，而另一个班有15个上海学生时，基于民族、生源地打散的规则，就会把当前这个“汉族、上海”的学生分到有15个上海学生的班级中去，而不是分到有1个汉族学生的班级，由此体现民族规则的高优先级，生源地规则的低优先级。

(3)取2的反序次方: 2^[n-(i+1)+1]：

* 1. 民族规则的权重系数为2^[8 –(0+1)+1] = 2^8 = 256
  2. 生源地规则的权重系数为 2^[8-(7+1)+1] = 2^1 = 2

班级1的权重：w1 = 256\*1 + 1\*0 = 256

班级2的权重：w2= 256\*0 + 1\*256 = 256

结果表示在这种权重系数的计算规则下，对于“汉族、上海”的学生来说，2个汉族的学生的权重顶的上256个上海的学生的权重，也就是说如果有一个有2个汉族学生，而另一个班有255个上海学生时，基于民族、生源地打散的规则，就会把当前这个“汉族、上海”的学生分到有255个上海学生（假设允许这么多学生在一个班）的班级中去，而不是分到有2个汉族学生的班级，由此体现民族规则的高优先级，生源地规则的低优先级。

基于上面的描述，可以得出一个结论，由于权重系数的计算规则不同，使得规则优先级的体现程度也发生变化，如果权重系数间的差越大，则规则优先级的体现越明显。

因此本算法应该支持根据不同学校的不同特点学生数据，选择不同的权重系数计算规则，至于什么样的权重系数计算规则更适合学校，则需要根据整体的学生情况，以及学校指定的分班规则来综合考量了。

### 3.5.2算法对聚合型分班规则的扩展性

上文中，讲述的都是如何基于规则将学生打散的分班问题，其实有的学校可能有一种特殊的需求，即按规则聚合学生。聚合意为将某一类学生聚拢到一个班，比如将所有藏族学生聚拢在一个班级中（仅是一个例子，不一定存在这样的需求）。

那么如何使得本文算法支持聚合呢？其实很简单，只需要将聚合型规则的权重系数取负值即可。这是因为在本文算法中，认为权重越小的班级更适合学生分班，那么如果当一个班级的某种类型的学生越多时，这个班按照聚拢规则去计算权重时，因为权重系数时负数，所以计算出来的权重就越小。