# 优化算法工具箱总结探索

[优化目标 2](#_Toc1640274781)

[优化算法的主要类型 5](#_Toc1790142055)

[算法原理的简单介绍 7](#_Toc2075865743)

[粒子群算法基本原理 7](#_Toc1851876248)

[遗传算法基本原理 8](#_Toc1838040160)

[贝叶斯优化原理 9](#_Toc594159866)

[不同package的测试结果对比 10](#_Toc2142053187)

[时间和精度比较 10](#_Toc218008344)

[同一优化工具箱下不同算法的比较 11](#_Toc791028175)

[问题一的求解结果 12](#_Toc1667429758)

[组合优化问题的解决方案 13](#_Toc810682114)

[解法一：匈牙利算法 13](#_Toc1621621839)

[解法二：启发式算法 15](#_Toc207724574)

## 优化目标

**目标**：了解各种数值优化算法的原理， 总结优化算法的类型，探索现成的优化工具箱的使用，使其符合需求：能尽可能提高遗传进展的同时提高群体的亲缘多样性。

**优化目标**：在选择自交系和自交系组配生产DH系时考虑群体遗传多样性，通过优化算法平衡遗传进展和近交系数，在保持群体的遗传进展的情况下减慢群体遗传变异下降的速度。即在限制亲缘关系的同时最大化产量育种值。

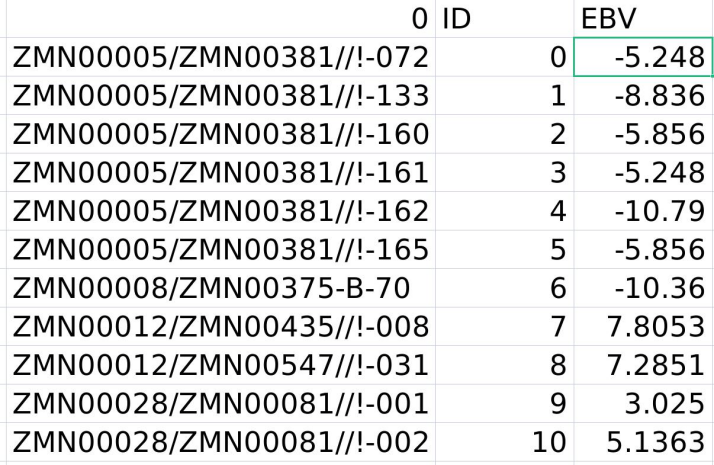
问题一（离散变量优化）： 假设EBV是一个的矩阵，每个元素代表一个样本（一个家系）的ebv值，REL是一个的对称矩阵，其中第（i,j）个元素代表第i和第j个样本（家系）之间的亲缘关系；ind是一个长度为m的列表，代表选取的m个样本的编号，我们的优化目标可以用以下式子进行表示：

（1）

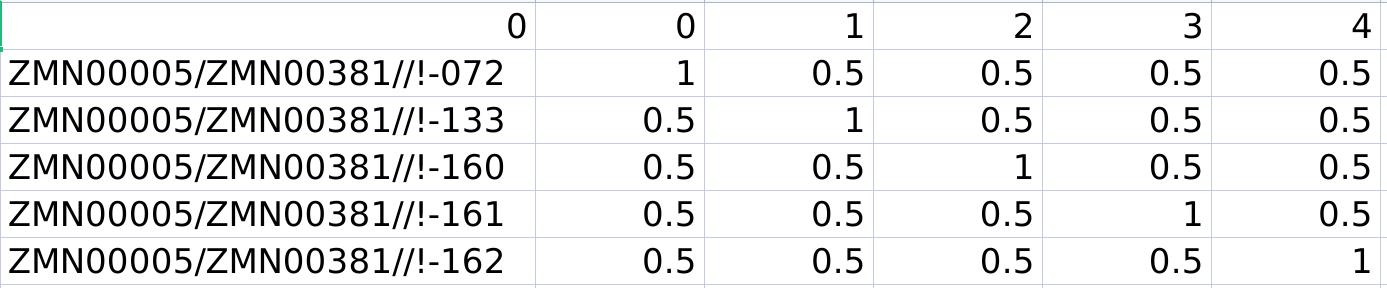
其中，表示从EBV矩阵中取出编号为ind的元素， 表示对取出EBV矩阵的所有值求平均，表示对亲缘关系的惩罚系数。

问题一的数据格式如下：

EBV文件：



REL文件：



优化算法需要给出的解是一个长度为m且元素不重复的列表，列表中的每个元素都是一个整数，取值范围为[1,N]。因此，该问题的解空间大小（解的所有可能情况数）为组合数。

问题二（组合优化）： 假设有个父本和个母本进行组合配对，需要选择m个组合（）,且某个特定的父本或者母本最多能重复使用n次，使得选出来的m个组合ebv值尽量大的同时亲缘关系尽可能的小。

假设有一个的目标矩阵obj，obj第个元素表示选择第i个父本和第j个母本时的值：

（2）

其中和分别表示父本和母本的EBV矩阵。

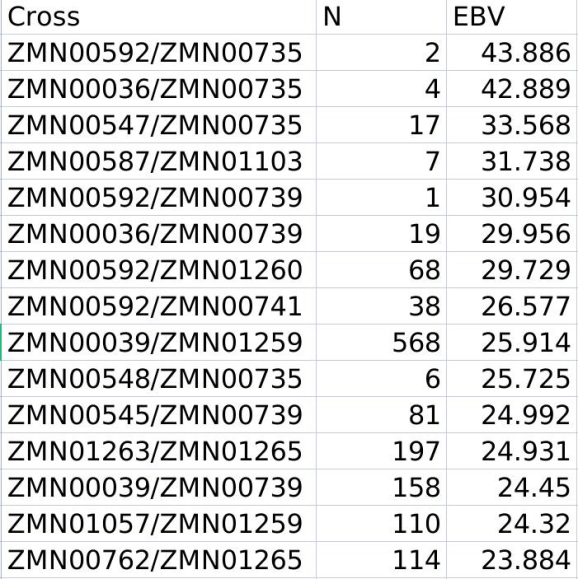
问题三（群体优化）：假设EBV是一个的矩阵，每个元素代表一个家系的ebv值，REL是一个的矩阵，其中第（i,j）个元素代表第i个家系和第j个家系之间的亲缘关系；同时，group是一个的矩阵，表示每个家系具有多少个DH个体（可以看作每个组中有多少个个体,用表示）。

假设表示第i个优化变量，表示第i个组取多少个个体（每个家系选出多少个DH个体）；其取值为，其中；表示第i组中的个体数量。

目标函数与问题一相似，但有如下约束条件：

（3）

问题三的数据格式(REL文件与问题一类似)：



**探索工具**

python pygad（遗传算法）

python pyswarms (粒子群算法)

python geatpy （遗传算法）

python scikit-opt （遗传算法、粒子群算法、模拟退火算法、鱼群算法、免疫优化算法等）

python optuna (贝叶斯优化)

python bayes\_opt (贝叶斯优化)

matlab optimization toolbox（遗传算法、粒子群算法、模拟退火算法、代理优化）

c++ galib （遗传算法）

c++/python ortools (一些组合优化问题)

python scipy (一些组合优化问题)

## 优化算法的主要类型

优化问题根据不同的角度可以具有多种分类。

1，根据优化问题的变量的类型可分为：

continuous optimization：即变量都是连续型

integer optimization：即变量是整数型

mixed integer optimization：即变量既有整数型也有连续型

2，根据优化问题的目标函数和约束条件分类：

convex optimization：目标函数是凸函数和约束条件产生的可行域是凸集（注意这里是 minimization problem）

non-convex optimization：目标函数是非凸函数或者约束条件产生的可行域是非凸集

constrained optimization：优化问题有约束条件

unconstrained optimization：优化问题没有约束条件。有些情况下也认为如果优化问题里只有 variable range 作为约束条件时也是无约束优化问题。

linear optimization：目标函数和约束条件的函数都是线性。

1. 根据需要优化的目标函数数量，可以分为单目标优化和多目标优化。

传统上，解决优化算法有以下三种比较成熟的解法：



其中，又以启发式算法的应用较为广泛，因为它速度快、很容易扩展到大规模问题，且可以适应各种问题类型。因此，本报告主要将注意力放在测试各种**启发式算法**的工具箱中。

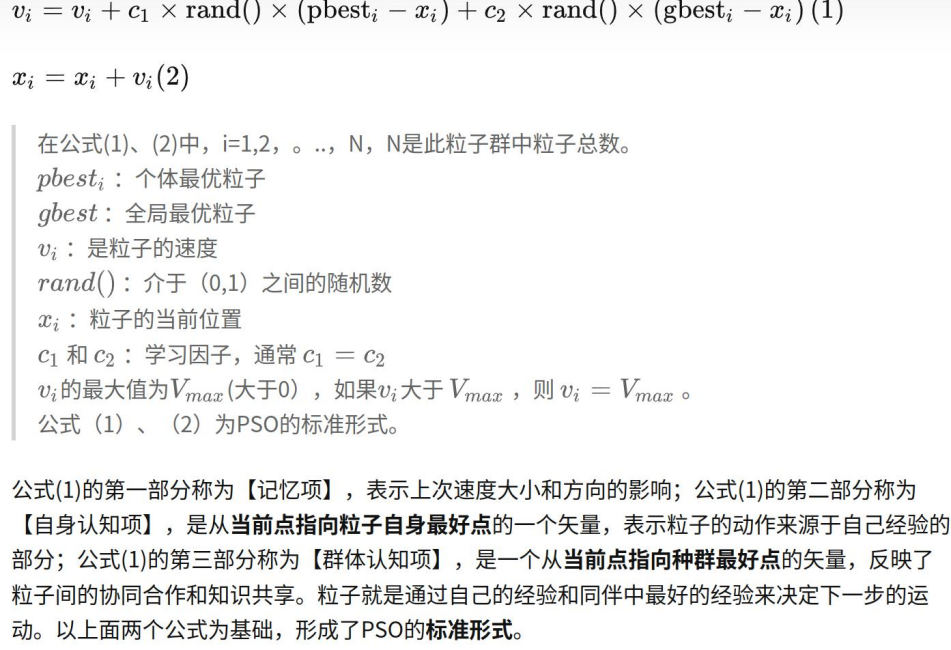
近年来，出现了许多利用机器学习解决优化问题的方法，包括利用强化学习、图神经网络等方法，不过这些方法仍然处于探索阶段，不够成熟，因此暂时可以不考虑。

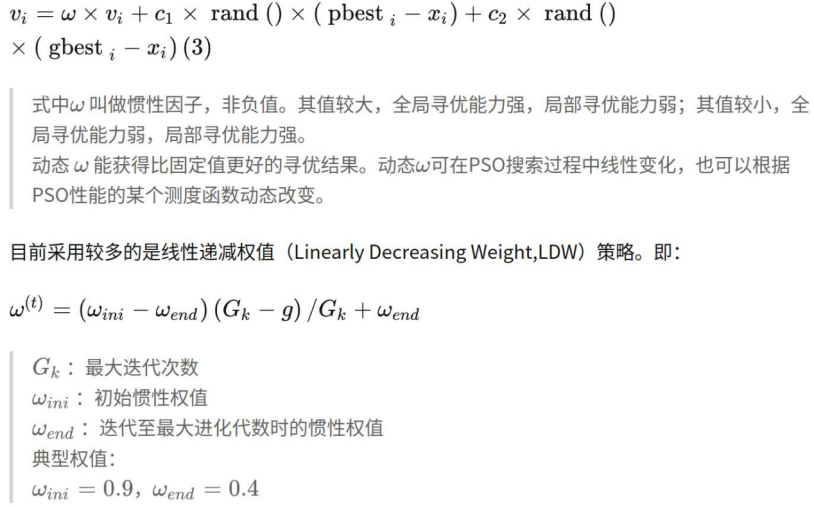
## 算法原理的简单介绍

### 粒子群算法基本原理

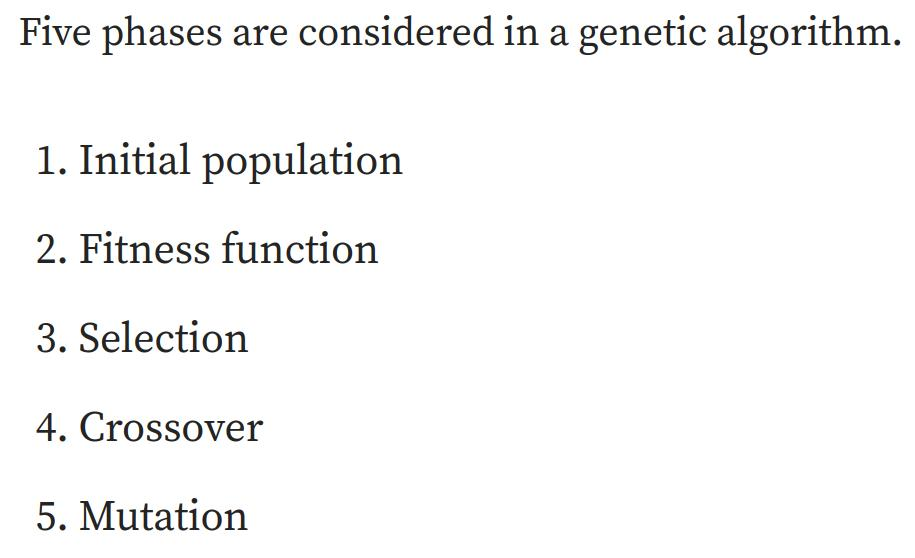
粒子群算法通过设计一种无质量的粒子来模拟鸟群中的鸟，粒子仅具有两个属性：速度和位置，速度代表移动的快慢，位置代表移动的方向。每个粒子在搜索空间中单独的搜寻最优解，记为个体当前极值，并将个体极值与整个粒子群里的其他粒子共享，整个粒子群的当前全局最优解为最优的那个个体极值。

粒子群中的所有粒子根据个体自身当前极值和整个粒子群共享的当前全局最优解来调整自己的速度和位置。

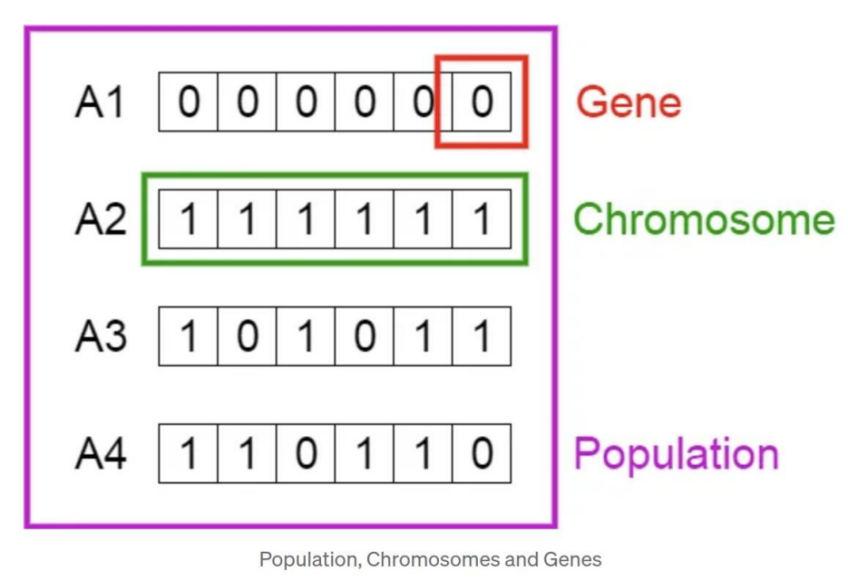




### 遗传算法基本原理



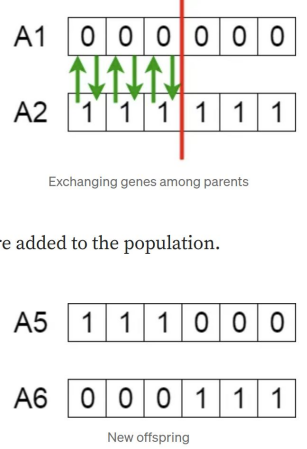
其中第一步对应算法的初始化，第二步对应算法的目标函数（需要最大化适应度函数值），第三四五步对应新解的产生和最优解的选择。



在遗传算法中，群体对应候选解的集合，染色体对应每个群体中的候选解，基因对应每个解的维度。

两种产生新解的方式

crossover:



mutation:



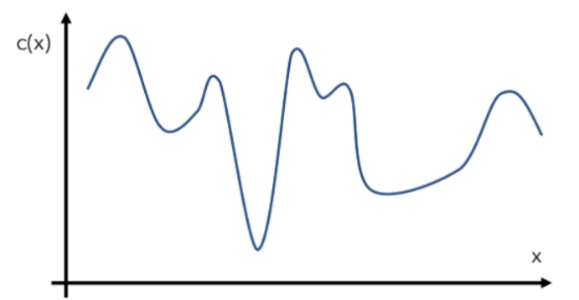
### 贝叶斯优化原理

贝叶斯优化一般用于对机器学习模型的超参数进行调参，不太会用于解决常见的优化问题。

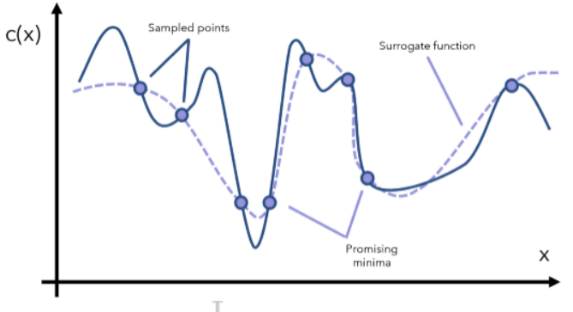
假设有一个未知的目标函数f(x)，其计算成本很高，它不一定是分析表达式，而且你不知道它的导数。但需要找到它的全局最小值。

贝叶斯优化可通过一种名为“代理优化（surrogate optimization）”的方法解决这一问题，而代理函数是指目标函数的一种近似。代理函数可基于采样得到的数据点而构建。我们可以根据代理函数来识别哪些点是有潜力的最小值。然后我们在这些有潜力的区域执行更多采样（哪些点比较有潜力可根据采样函数确定），采样更多的数据点，然后据此更新代理函数。在每一次迭代中，我们都要继续观察当前的代理函数，通过采样对相关区域有更多了解，然后更新函数。

假设真实目标函数如下图：



通过利用采样函数不断地采样，可构建代理函数逼近真实函数：



利用近似出来的代理函数，寻找最小点。

## 不同package的测试结果对比

matalb可以实现遗传算法、粒子群算法、模拟退化算法以及贝叶斯优化算法；geatpy主要实现遗传算法及其变种差分进化算法； scikit-opt可以实现7种启发式算法：差分进化算法、遗传算法、粒子群算法、模拟退火算法、蚁群算法、鱼群算法、免疫优化算法；pyswarms可以实现粒子群算法，optuna主要实现贝叶斯优化算法，ortools主要实现组合优化问题。

下面的对比试验中，若package能实现多种算法，则主要使用遗传算法与其他package进行对比（遗传算法的效果最好）。

### 时间和精度比较

为了比较不同package的性能（包含效率和精度），利用问题一在的情况下进行测试。下表会展示当优化变量的个数为10、50、150和200时算法的精度（算法给出的优化值/真实最优值），同时也会给出当优化变量个数为150时算法运行的时间。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **toolbox** | **time** | **10** | **50** | **150** | **200** |
| matlab | 30.62 | 23.91/25.55 | 22.32/23.05 | 20.11/21.2 | 19.59/20.58 |
| geatpy | 2334 | 24.63/25.55 | 22.14/23.05 | 20.44/21.2 | 19.34/20.58 |
| scikit-opt | 27.48 | 24.25/25.55 | 21.86/23.05 | 18.67/21.2 | 16.27/20.58 |
| pygad | 46 | 11.72/25.5 | 4.27/23.05 | 5.13/21.2 | 4.51/20.58 |
| pyswarms | 18.81 | 18.1/25.5 | 4.19/23.05 | 2.52/21.2 | 2.13/20.58 |
| optuna | 精度不足 | | | | |
| galib | 使用复杂 | | | | |
| ortools | 无法解决问题一的类型 | | | | |
| optuna+scikit-opt | 4549 | 25.55/25.55 | 22.83/23.05 | 4.97/21.2 | 3.33/20.58 |

从上表可以看出，matlab和geatpy两个工具箱的优化精度是最高的，但是geatpy的运算时间约为matlab工具箱的近80倍，时间花销比较大；而基于sklearn的优化工具箱scikit-opt的精度虽然在优化变量个数比较多时略逊色于matlab和geatpy，但是依然远高于其他工具箱。optuna工具箱主要用于机器学习算法的调参优化，优化变量个数比较多时同样不适用。为此，我使用optuna工具箱对scikit-opt的三个超参数进行探究，这三个超参数是：size\_pop（种群数量）、max\_iter（最大迭代次数）和prob\_mut（杂交率）。在优化变量个数分别为10和50时，确实有助于提升优化精度。

### 同一优化工具箱下不同算法的比较

不同的工具箱往往具有多种优化算法可以提供选择，下面三个表对三个工具箱的不同算法的性能进行了比较。

matlab optimization toolbox

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| algorithm | 100 | | 150 | |
| accuracy | time | accuracy | time |
| genetic algorithm | 20.71/21.9 | 15.37 | 20.11/21.2 | 30.62 |
| particleswarm | 20.53/21.9 | 21.22 | 19.74/21.2 | 14.49 |
| simulated annealing | 2.56/21.9 | 6.15 | 1.09/21.2 | 19.55 |

geatpy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| algorithm | 100 | | 150 | |
| accuracy | time | accuracy | time |
| genetic algorithm | 21/21.9 | 775.11 | 20.44/21.2 | 2334 |
| differential evolution | 3.51/21.9 | 64 | 1.76/21.2 | 10.4 |

scikit-opt

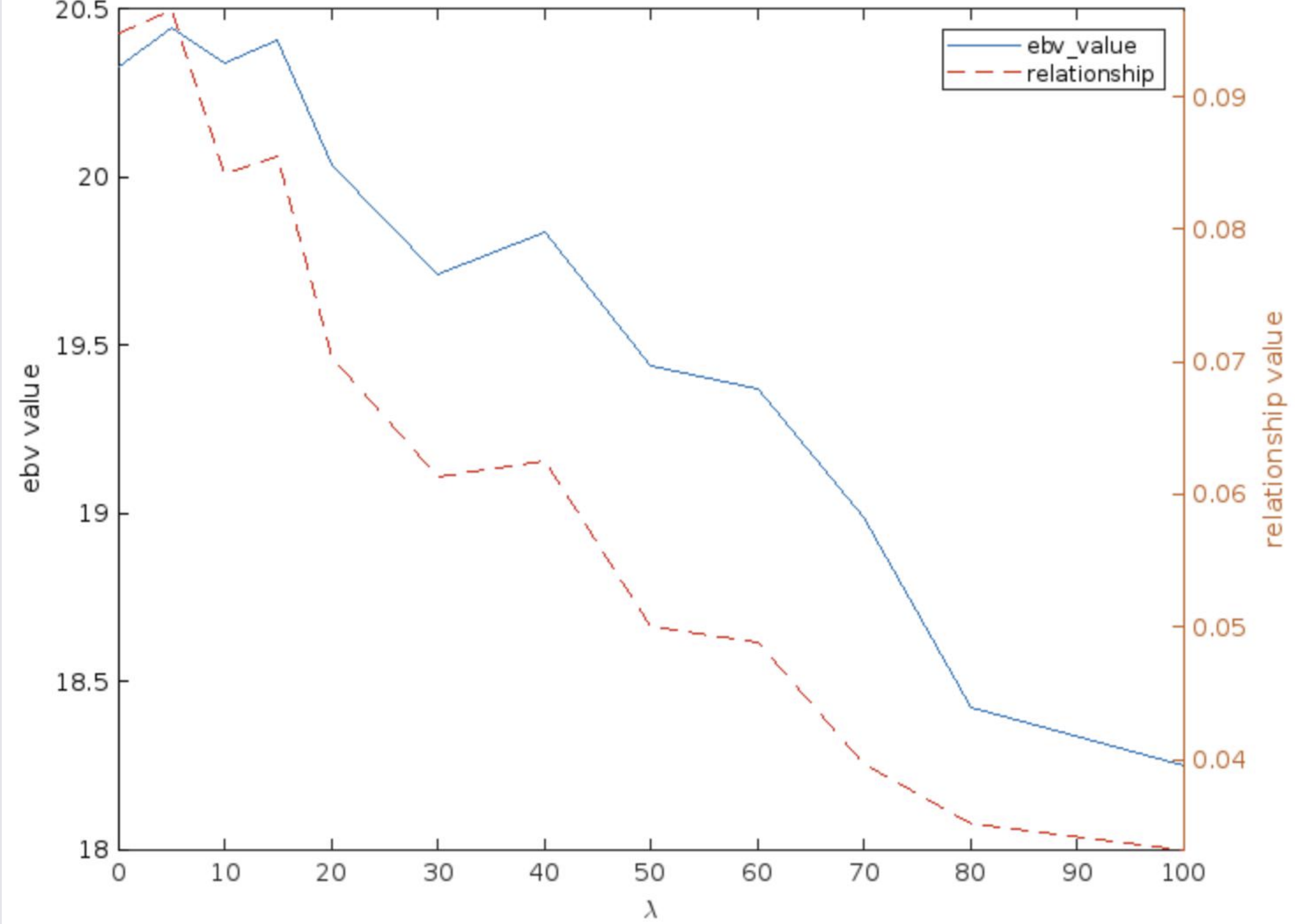
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| algorithm | 100 | | 150 | |
| accuracy | time | accuracy | time |
| genetic algorithm | 20.11/21.9 | 20.12 | 18.67/21.2 | 27.48 |
| particleswarm | fail/21.9 | 1.8 | fail/21.2 | 1.78 |
| simulated annealing | 2.55/21.9 | 0.69 | 1.08/21.2 | 0.86 |
| differential evolution | 8.07/21.9 | 12.93 | 5.7/21.2 | 16.33 |

从上面三个表可以看出，对优化变量为离散变量的优化问题，遗传算法的精度都是最优的。可能的原因有：

1. 遗传算法的编码方式天然适应于优化变量为离散变量的优化问题
2. 遗传算法在优化的过程中会同时提供大量样本以供候选，算法找到全局最优解的可能性更大。

## 问题一的求解结果

在样本个数为4763，优化变量个数为150且取0，10，20，30，40，50，60，70，80，90和100的情况下，遗传算法优化得到的EBV平均值和亲缘关系平均值如下图所示：

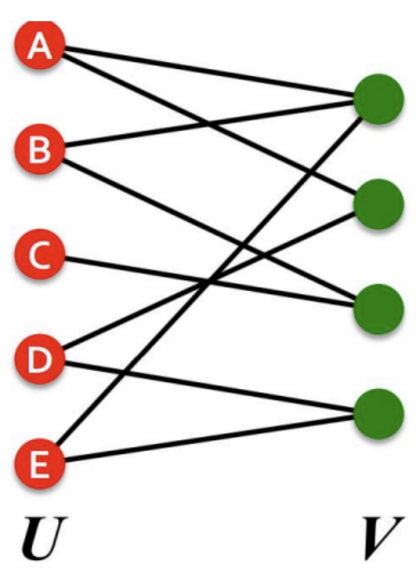


两条曲线随着的变化趋势基本符合生物学的先验。需要注意的是，遗传算法有一定的误差，比如在取0时算法没有找到最优值。但在解空间如此庞大的情况下上述结果还是比较接近最优值。

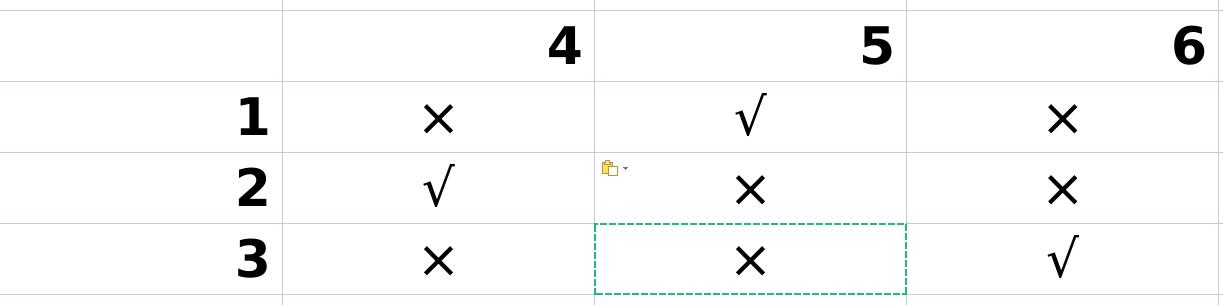
## 组合优化问题的解决方案

### 解法一：匈牙利算法

在优化问题二中，如果父本和母本均只能使用一次，则该问题实际上可以抽象成二分图的最优匹配问题：

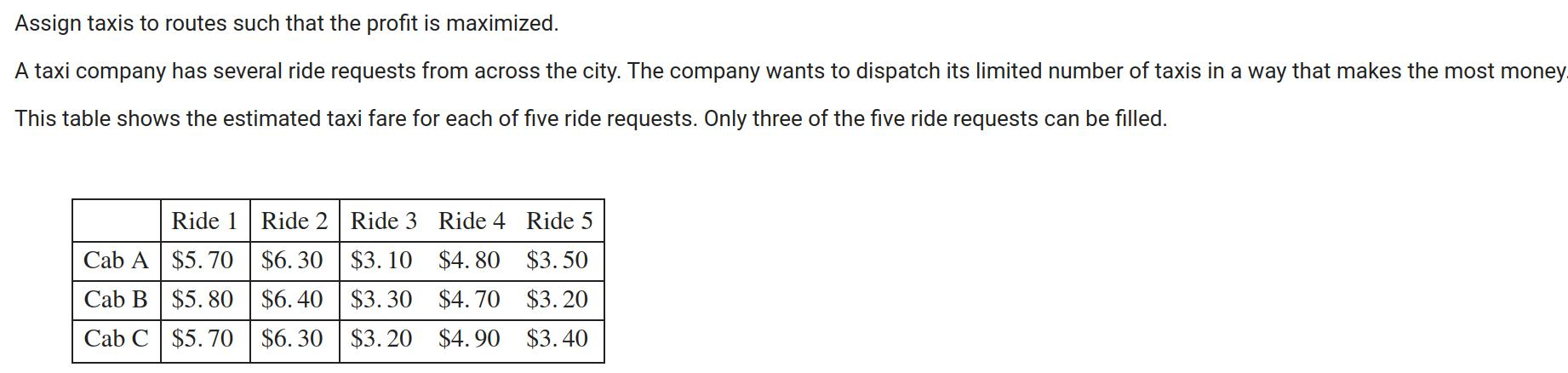


以一个简单的例子进行解释。根据式子（2）我们可以利用EBV值和亲缘关系矩阵构造一个成本矩阵（目标矩阵）。假设父本的编号是1，2，3，母本的编号是4，5，6，我们需要找到三对最优的组配方式。为了方便理解，可以画如下的示意图：

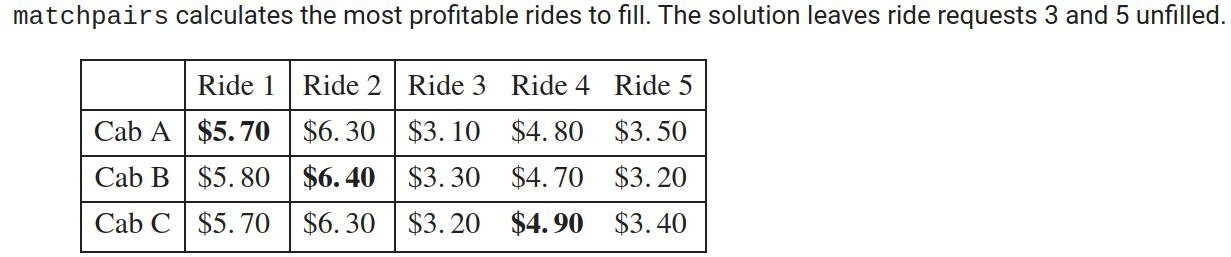


在上图中，我们选择的三对最优组合是（1，5）、（2，4）和（3，6）。

由于每个父本和每个母本分别只能使用一次，则相当于每一行和每一列分别只能有一个样本被选中，这与如下的匈牙利匹配问题很相似，可以直接套用匈牙利算法求解：

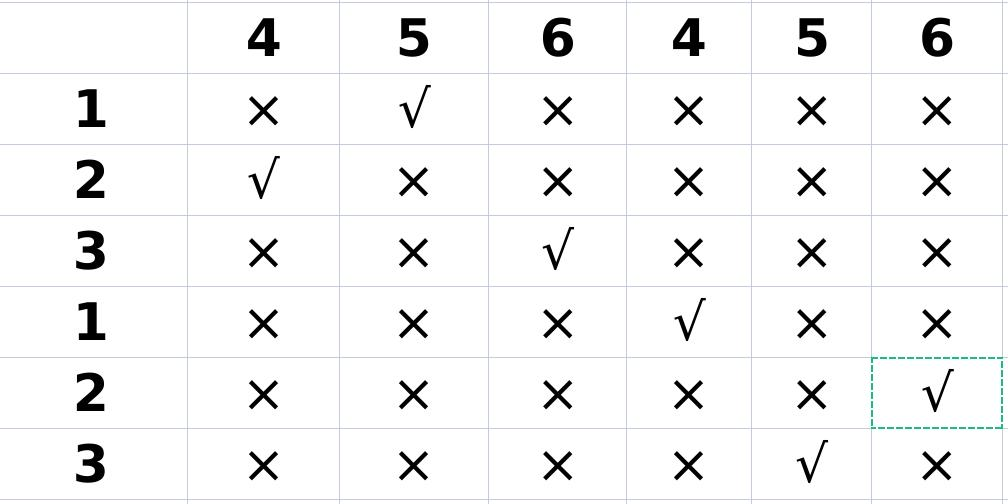


最终的结果如下：



如果每个父本和母本必须使用两次或者三次或者任意次数，依然能套用匈牙利算法进行求解。

以每个父本和母本必须使用两次且需要选择6对最优组合为例，有如下示意图：



每一行和每一列依然分别只能有一个样本被选中，选择的最优组合分别为：（1，5）、（2，4）、（3，6）、（1，4）、（2，6）和（3，5）。

但是上面这个例子中，如果是每个父本和母本最多能使用2次，且要选择5对最优组合，则无法套用匈牙利算法。

### 解法二：启发式算法

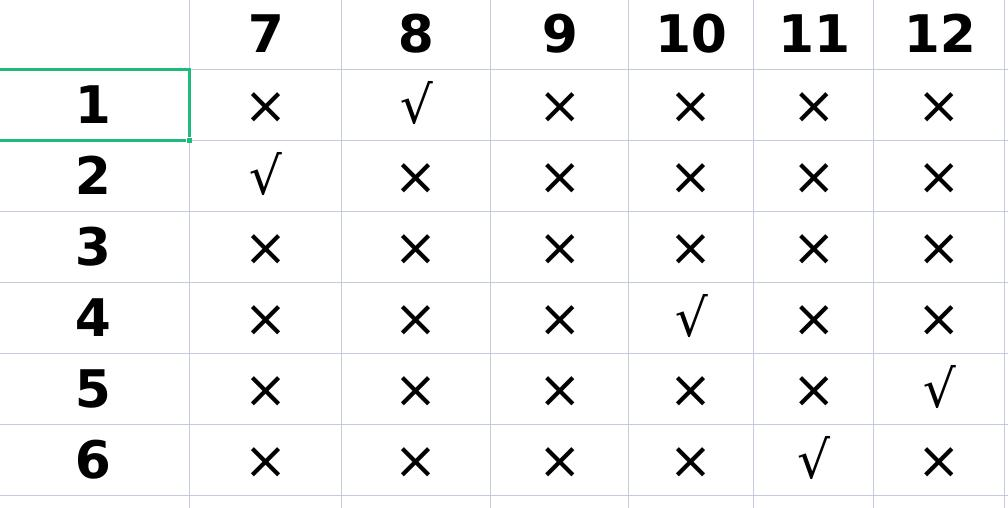
另一个解决方法是继续套用遗传算法，但需要对解进行一定的设计。

还是假设有三个父本和三个母本进行配对，且父本和母本均只能使用一次。需要明确的是，优化算法给出的解需要分成两个部分：编号为1到3的解表示选择的父本编号，编号为4到6的解表示选择的母本编号。如下的例子：x=[2,3,1,6,5,4]，表示算法认为的最优组合是(2,6),(3,5)和(1,4)。因此，解的第一部分的解空间是3个不重复的整数，范围是[1,3]；第二部分的解空间是3个不重复的整数，范围是[4,6]。

回到问题二的一般化设定， 假设有个父本和个母本进行组合配对，需要选择m个组合（）,且某个特定的父本或者母本最多能重复使用1次，则解空间的大小是。

下面关注另一个问题： 假设有三个父本和三个母本进行配对，且父本和母本均最多能使用2次，要选择5对最优组合。

假设父本的原始编号为1，2，3；母本的原始编号为7，8，9。而编号4，5，6分别表示样本1，2，3的“复制品”，编号10，11，12分别表示样本7，8，9的“复制品”。优化算法需要给出的解同样分成两个部分，第一部分表示选择的父本编号，是一个长度为5的、元素没有重复的列表，范围为[1,6]；类似的，第二部分表示选择的母本编号，是一个长度为5的、元素没有重复的列表，范围为[7,12]。示意图如下：



从图中可以看出，被选中的最优组合为（1，8），（2，7），（4，10），（5，12）和（6，11）。实质的选择是（1，8），（2，7），（1，7），（2，9）和（3，8），即1、2、7、8被使用了2次，3和9只被选择了一次。

### 总结

通过上面的两种方案，基本上可以把问题类型二（父本和母本组配优化）解决，且优化效果较好。