通过遗传算法对蚁群算法进行优化的0-1背包问题

## 摘要

背包问题(Knapsack problem)是一种组合优化的NP完全问题。问题可以描述为：给定一组物品，每种物品都有自己的重量和价格，在限定的总重量内，我们如何选择，才能使得物品的总价格最高。同时理论上加快了收敛。采用遗传算法的相关思想，对蚁群算法进行改进。收敛速度慢的原因在于信息素矩阵的更新变化速度比较慢，有可能的解都会根据距离以及蚂蚁路径而缓慢加强，所以针对可能的最好的解通过遗传的方式进行局部增强，并且由于遗传算法部分对信息素矩阵的更新，会有蚁群算法的行进过程中获得跳出局部最优解，丰富种群多样性的作用。

## 研究背景和意义

背包问题(Knapsack problem)是一种组合优化的NP完全问题。问题可以描述为：给定一组物品，每种物品都有自己的重量和价格，在限定的总重量内，我们如何选择，才能使得物品的总价格最高。推而广之，就是说在给定负载或承受能力以内，该如何选择任务使得得到的反馈最好。

背包问题出现在现实世界的各个领域的决策过程中，例如寻找削减原材料浪费最少的方法，选择投资和投资组合，选择用于资产支持证券化的资产，仓库的存储组合，城市规划中的电线路径规划，道路规划，公交路线规划等等。

背包问题的难点主要是背包的容量，这个问题中限制了背包的大小，想要在所有的组合中找到最优的那个解，这个问题的难易程度也跟不同的问题不一样，0-1背包和0-N背包的复杂程度也是不一样的。

## 问题建模与算法设计

## 3.1问题建模

背包问题：

设有一只背包，最多可容纳总重量为 W 的物品。 现有 n 件物品可供选择装入背包中，这 n 种物品的编号分别为 1,2, …,n。各物品的重量、价值分别为 w1,w2,…,wn 和 p1,p2,…,pn，每 种物品装入个数为 xj（xj ≥0 且为整数，j∈{1,2,…,n}），每件物 品均不能拆开。求如何选择装入的物品（各几件），可使总价值 z 达到最大。 （其中 W，w1,w2,…,wn 和 p1,p2,…,pn 均为整数），即求得：

通过遗传算法对蚁群算法进行优化的0-1背包问题：

(1)确定选择顺序：先确定蚂蚁数量m，并确定每个蚂蚁对物品的选择顺序，将n个物品标号1，2，3……,n后随机打乱顺序，蚂蚁将根据这个顺序对物品进行选择。

(2)确定决策变量：，确定是否需要装入第k件物品；，背包中已经装入的物品重量。

(3)建立状态转移方程：

(4)求解：对每个蚂蚁，根据状态转移方程计算各自的 ，并进行合并记录得到蚁群解ACO\_Table

(5)计算遗传：将所有蚂蚁的解进行对比，选择其中最好的几个解构成遗传中的原始解，并对这些解进行交叉变异，得到最终的遗传解GO\_Table

(6)更新信息素：通过ACO\_Table与GO\_Table更新信息素，

(7)判断循环中止：若当前循环迭代次数iter<itermax则跳转到第四步继续执行，否则输出ACO\_Table中的最好的解作为蚁群的整体解。

## 3.2算法设计

3.2.1算法设计的思路

背包问题是典型的np难问题，得到需比较所有的 ，这有 种可能，以比较为基础的求最大问题复杂度下界 ，故本问题 ，虽然说可以使用动态规划使得原本的指数次复杂度降低到，但由于w也是随着n的增长而增长的所以本质上也是n的一个函数，故而不算是一个多项式，背包问题找不到一个多项式时间内可以解决的方案。而不存在多项式时间算法，也就表示0/1背包问题不是NP问题。

根据蚁群算法的思想，蚂蚁找到最短路径要归功于信息素和环境，假设有两条路可从蚁窝通向食物，开始时两条路上的蚂蚁数量差不多：当蚂蚁到达终点之后会立即返回，距离短的路上的蚂蚁往返一次时间短，重复频率快，在单位时间里往返蚂蚁的数目就多，留下的信息素也多，会吸引更多蚂蚁过来，会留下更多信息素。而距离长的路正相反，因此越来越多的蚂蚁聚集到最短路径上来。但蚁群算法有以下明显的缺陷：

1. 收敛速度慢。蚁群算法中信息素初值相同，选择下一个节点时倾向于随机选择。虽然随机选择能探索更大的任务空间，有助于找到潜在的全局最优解，但是需要较长时间才能发挥正反馈的作用，导致算法初期收敛速度较慢。
2. 局部最优问题。蚁群算法具有正反馈的特点，初始时刻环境中的信息素完全相同，蚂蚁几乎按随机方式完成解的构建，这些解必然会存在优劣之分。在信息素更新时，蚁群算法在较优解经过的路径上留下更多的信息激素，而更多的信息激素又吸引了更多的蚂蚁，这个正反馈的过程迅速地扩大初始的差异，引导整个系统向最优解的方向进化。虽然正反馈使算法具有较好的收敛速度，但是如果算法开始得到的较优解为次优解， 那么正反馈会使次优解很快占据优势，使算法陷入局部最优，且难以跳出局部最优。
3. 优化能力问题。蚁群算法中参数众多并且具有一定的关联性，虽然蚁群算法在很多领域都有广泛应用，但是参数选择更多是依赖经验和试错，不恰当的初始参数会减弱算法的寻优能力
4. 种群多样性与收敛速度的矛盾。种群多样性对应于候选解在问题空间的分布。个体分布越均匀，种群多样性就越好，得到全局最优解的概率就越大，但是寻优时间就越长；个体分布越集中，种群多样性就越差，不利于发挥算法的探索能力。正反馈加快了蚁群算法的收敛速度，却使算法较早地集中于部分候选解，因此正反馈降低了种群的多样性，也不利于提高算法的全局寻优能力。

每一代的蚂蚁寻找路径都会根据信息素的浓度来进行，所以为了改进蚁群算法的以上缺陷，采用遗传算法的相关思想，对蚁群算法进行改进。收敛速度慢的原因在于信息素矩阵的更新变化速度比较慢，有可能的解都会根据距离以及蚂蚁路径而缓慢加强，所以针对可能的最好的解通过遗传的方式进行局部增强，但这样有可能会过早的陷入局部最优解中，故而人为的提高了变异的概率，加强了原有蚁群算法的探索能力，同时理论上加快了收敛。并且由于遗传算法部分对信息素矩阵的更新，会有蚁群算法的行进过程中获得跳出局部最优解，丰富种群多样性的作用，克服了以上问题。

3.2.2算法伪代码

**procedure** AS

**for** each edge

set initial pheromone value

**end for**

**while** not stop

**for** each ant k

**for** i=1 to n

choose next item j with the probability given.

**end for**

**end for**

compute the value C­k­­ of the item list constructed by the kth ant.

choose best two item list to fill the genetic matrix

update the genetic matrix by chosen item list.

**for** each edge

update the pheromone value

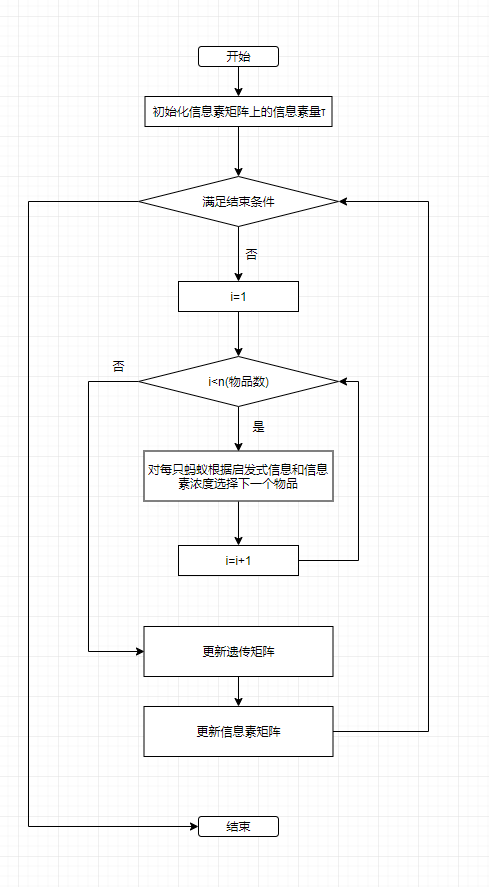
**end for**

**end while**

print result

**end procedure**

3.2.3算法流程图

%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%E5%BC%80%E5%A7%8B%22%20style%3D%22rounded%3D1%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22340%22%20y%3D%2270%22%20width%3D%2280%22%20height%3D%2220%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

3.2.4算法算例

|  |  |  |  |
| --- | --- | --- | --- |
| **物品id** | **重量** | **价值** | **重量价值比** |
| 1 | 80 | 220 | 2.75 |
| 2 | 82 | 208 | 2.536585366 |
| 3 | 85 | 198 | 2.329411765 |
| 4 | 70 | 192 | 2.742857143 |
| 5 | 72 | 180 | 2.5 |
| 6 | 70 | 180 | 2.571428571 |
| 7 | 66 | 165 | 2.5 |
| 8 | 50 | 162 | 3.24 |
| 9 | 55 | 160 | 2.909090909 |
| 10 | 25 | 158 | 6.32 |
| 11 | 50 | 155 | 3.1 |
| 12 | 55 | 130 | 2.363636364 |
| 13 | 40 | 125 | 3.125 |
| 14 | 48 | 122 | 2.541666667 |
| 15 | 50 | 120 | 2.4 |
| 16 | 32 | 118 | 3.6875 |
| 17 | 22 | 115 | 5.227272727 |
| 18 | 60 | 110 | 1.833333333 |
| 19 | 30 | 105 | 3.5 |
| 20 | 32 | 101 | 3.15625 |
| 21 | 40 | 100 | 2.5 |
| 22 | 38 | 100 | 2.631578947 |
| 23 | 35 | 98 | 2.8 |
| 24 | 32 | 96 | 3 |
| 25 | 25 | 95 | 3.8 |
| 26 | 28 | 90 | 3.214285714 |
| 27 | 30 | 88 | 2.933333333 |
| 28 | 22 | 82 | 3.727272727 |
| 29 | 25 | 80 | 3.2 |
| 30 | 30 | 77 | 2.566666667 |
| 31 | 45 | 75 | 1.666666667 |
| 32 | 30 | 73 | 2.433333333 |
| 33 | 60 | 72 | 1.2 |
| 34 | 50 | 70 | 1.4 |
| 35 | 20 | 69 | 3.45 |
| 36 | 65 | 66 | 1.015384615 |
| 37 | 20 | 65 | 3.25 |
| 38 | 25 | 63 | 2.52 |
| 39 | 30 | 60 | 2 |
| 40 | 10 | 58 | 5.8 |
| 41 | 20 | 56 | 2.8 |
| 42 | 25 | 50 | 2 |
| 43 | 15 | 30 | 2 |
| 44 | 10 | 20 | 2 |
| 45 | 10 | 158 | 15.8 |
| 46 | 10 | 10 | 1 |
| 47 | 4 | 8 | 2 |
| 48 | 4 | 5 | 1.25 |
| 49 | 2 | 3 | 1.5 |
| 50 | 1 | 1 | 1 |

3.2.5算法复杂度分析

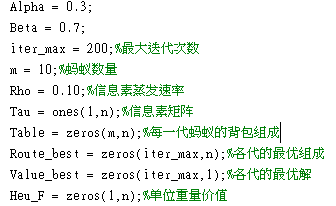
该算法主体是在T的迭代循环中进行蚂蚁次数m的物品数量n次的计算物品总价值的平均为n/2次的循环，故而时间复杂度为O(T\*m\*n\*n/2)

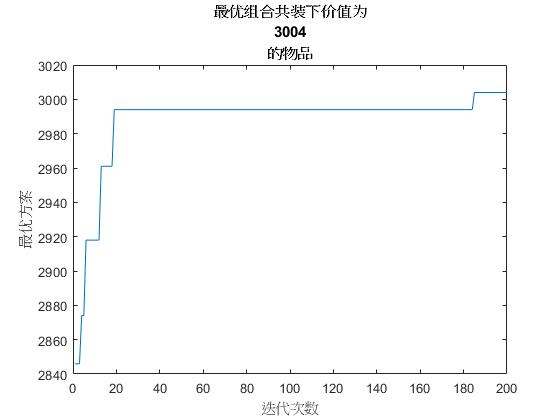
## 仿真验证

#### 4.1参数验证

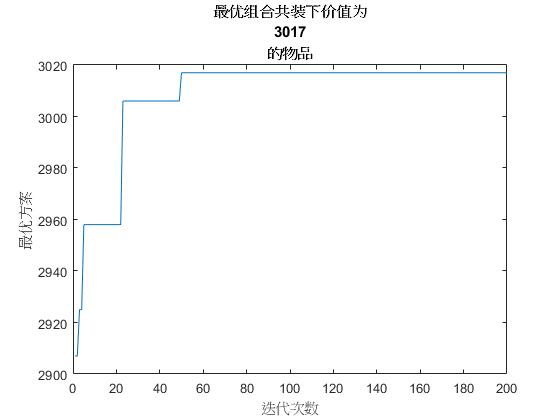
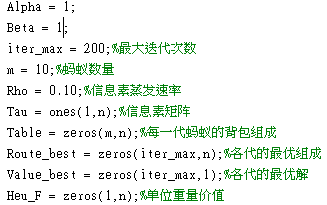
以下是只有蚁群算法的多次计算后取得的中位数左右的值

参数一：

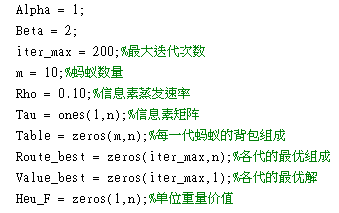


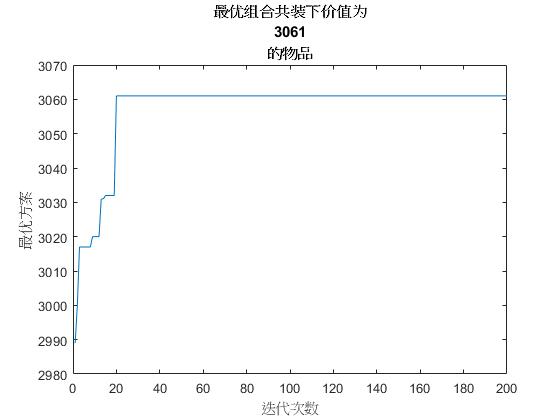


参数二：

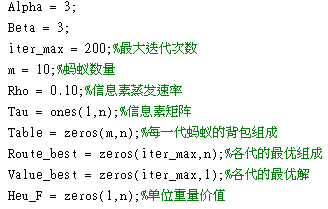


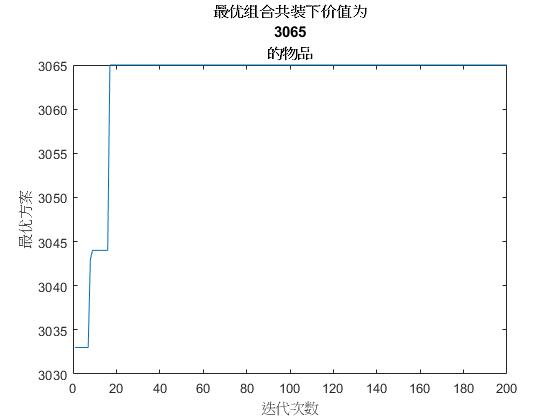
参数三：





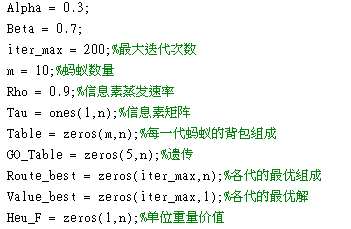
参数四：

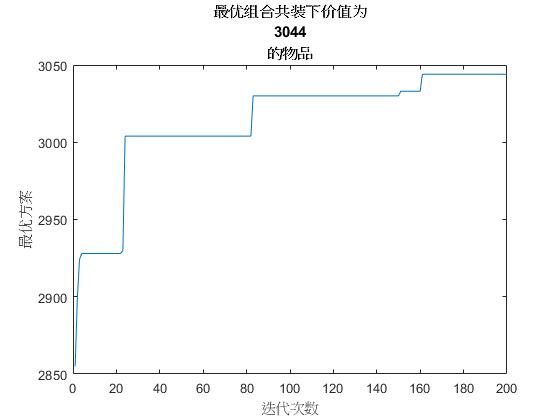




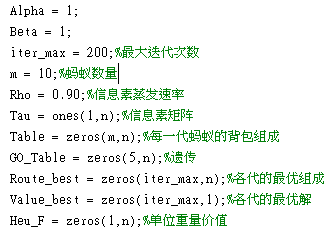
以下是蚁群算法和遗产算法的混合算法

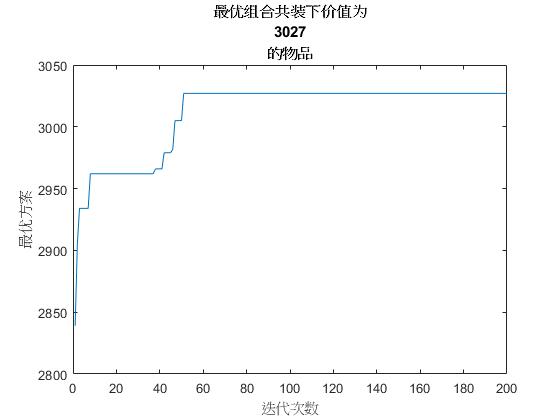
参数一：



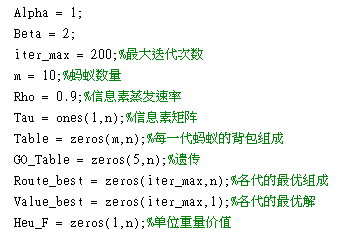


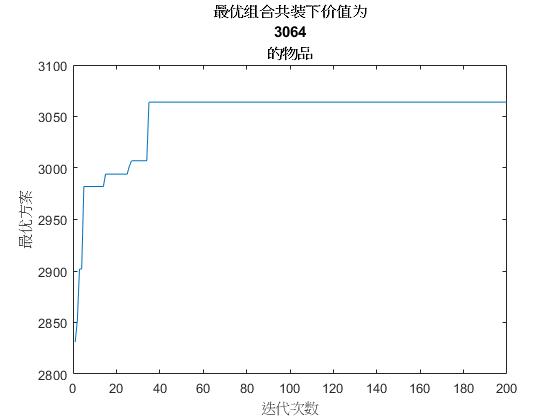
参数二：



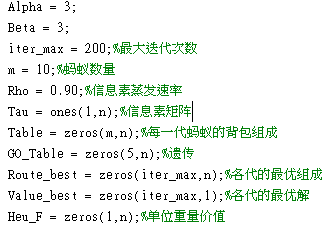


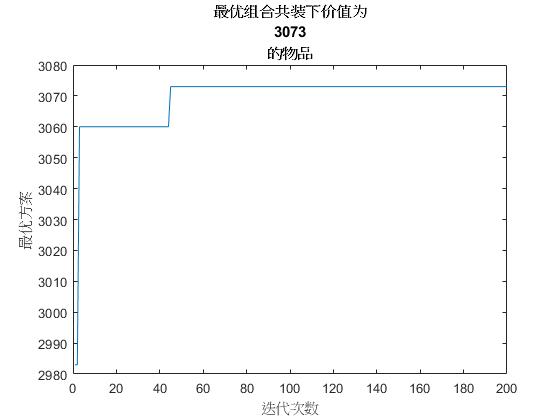
参数三：





参数四：





显而易见的可以看出无论是哪种算法，最终的参数都是

Alpha = 1或3;

Beta = 2或3;

时，算法收敛速度最快，而每种参数得到的迭代收敛次数，相较而言单一算法会好一点

混合算法的性能比起单一算法有些许提高，由于得到的解混合算法比起单一算法会好一点。

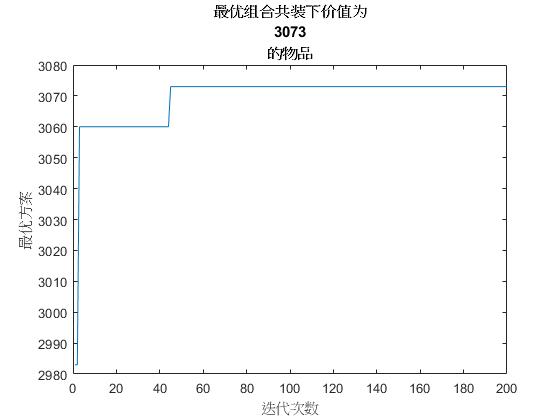
#### 4.2收敛验证

当参数选择为

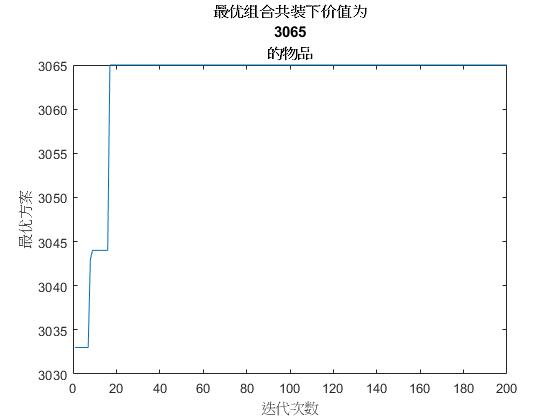
Alpha = 3

Beta = 3时，

混合算法最优组合值取值情况如下（多次验证取中位收敛速度）



单一算法最优组合值取值情况如下（多次验证取中位收敛速度）



显而易见单一算法收敛速度快在大概20次迭代时边收敛，而混合算法时常难以稳定收敛

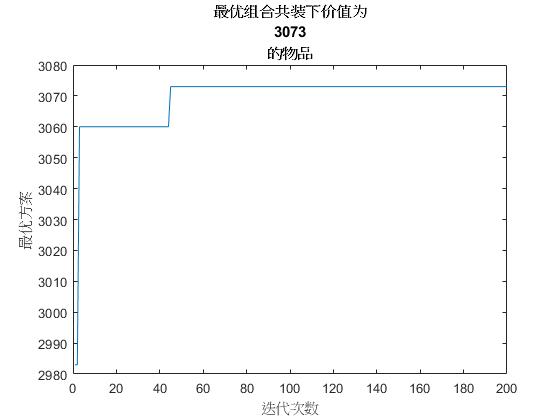
#### 4.3性能验证

当参数选择为

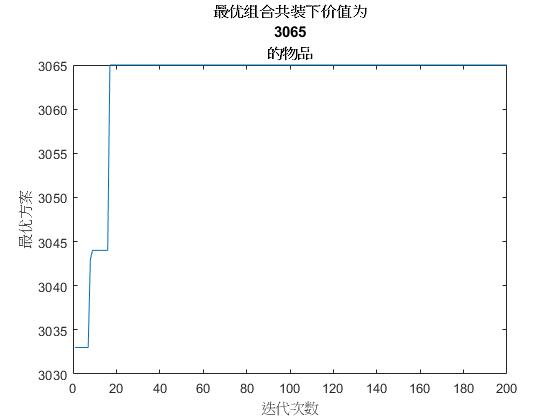
Alpha = 3

Beta = 3时，

混合算法最优组合值取值情况如下（多次验证取中位收敛速度）



单一算法最优组合值取值情况如下（多次验证取中位收敛速度）



混合算法的取值区间大多在3055以上，单一算法的取值区间大多在3050以上，可以得到混合算法的解质量更高。

## 结论

通过借鉴强化学习的思想，并且为了跳出局部最优，我选择了遗传算法作为蚁群算法的辅助算法，通过选择每一代蚂蚁得到的最优的几个解，获取相对而言的局部最优解，并且根据这几个局部最优解选择其中可能存在的最优子结构，通过交叉变异的方式跳出局部最优解的问题，本来的预期是可以在收敛速度上有所提升，但是反而在迭代次数上面比起单一算法落了下风，倒是最优组合的解的质量反而更高了，根据推测可能是遗传中的变异行为导致的智能算法的探索能力提高了，对于蚁群算法的缺陷部分做出了部分的改进，但是具体可以产生的效用还需再探讨。

## 参考文献

1. <https://blog.csdn.net/qq_26769591/article/details/80530398>蚁群算法学习总结
2. <http://www.elecfans.com/lab/code/20180202628570_2.html>蚂蚁算法计算过程实例 - 蚁群算法基本原理及其应用实例
3. <https://www.jianshu.com/p/ae5157c26af9>超详细的遗传算法(Genetic Algorithm)解析
4. <http://blog.chinaunix.net/uid-27105712-id-3886077.html>遗传算法(Genetic Algorithm)