免疫算法

1. 免疫算法的总体介绍：

免疫算法的目的：免疫算法是一种具有对多峰值函数进行多峰值搜索和全局寻优的算法，多峰值函数：多峰值函数优化是一种典型的复杂优化问题，其主要特点是目标函数的局部最优解数量多，全局最优解难以确定.在工程实际中，往往不仅要求能够确定问题目标函数的全局最优解，而且要求搜索出所有局部最优解。传统数值优化的方法：最速下降法、共轭梯度法等等虽然收敛快、精度高，但难以找到所有的局部最优解。

1. 基本概念

自体 指生物体自身的细胞

异体 指来自生物个体之外的细胞

抗原 指在病原体上表达的模式(主要)

抗体 指用于识别抗原的细胞(主要)

亲和力 指抗体和抗原之间的匹配程度

记忆细胞 指亲和力大于指定阈值的抗体

亲和性：匹配程度。

应用免疫算法求解实际问题时，常将抗原、抗体、抗原和抗体之间的亲和性分别对应于优化问题的目标函数、优化解、解与目标函数的匹配程度。

1. 免疫算法的基本原理：
   1. 多样性和亲和性

多样性：算法中为了表明全体中的抗体的多样性，引入信息熵的概念。免疫系统的多样性可由以信息熵为参数的公式所表示，多样性，对抗体的克隆和变异有助于产生新的抗体。

亲和性：抗原和抗体之间、抗体和抗体之间的匹配程度可以用亲和性描述。抗原和抗体之间的亲和性：用于表明抗体对抗原的识别程度。

* 1. 疫算法的基本步骤：

1. 抗原的识别阶段：输入目标函数和各种约束作为免疫算法的抗原
2. 初始抗体的产生阶段：在解空间中用随机方法产生抗体
3. 亲和力的计算：分别计算抗原和抗体 V 之间的亲和性及抗体 V 和抗体 W 之间的亲和性
4. 记忆单元的更新：将与抗原亲和性高的抗体加入到记忆单元，并用新加入的抗体取代与其亲和性最高的原有抗体（抗原的原有抗体，因为为了找到局部最优解）
5. 促进和抑制抗体的产生：计算每个抗体的期望值，抑制期望值低于阈值的抗体；可以知道与抗原间具有的亲和力越高，该抗体的克隆数目越高，其变异率也越低。与抗原亲和性高的抗体或低密度的抗体生存机率较大。由于高亲和性的抗体得到促进，而高密度的抗体受到抑制，体现了免疫系统的多样性。
6. 抗体的产生：通过交叉和变异产生进入下一代的抗体，重复步骤3-6，直到收敛判断依据满足
7. 终止记忆细胞的迭代：在达到指定阈值的时候终止记忆细胞的生成和选取；

免疫算法一般包括3类：克隆选择算法、免疫网络算法、阴性选择算法。

1. 克隆选择算法

克隆选择算法是一个迭代的过程。

算法步骤如下：

1. 在随机空间中生成数量为n的抗体。计算抗体中解的值。
2. 进入迭代过程，如满满足终止条件则终止，否则一直重复步骤3-6。
3. 对n个抗体进行克隆变异，即对每一个抗体进行克隆，每个抗体可能克隆出多个抗体，对克隆的抗体进行变异。
4. 将变异后的抗体集合加入原抗体集合中，然后从集合中选取出最优的n个抗体。
5. 然后生成一定数量的随机抗体，与上一步选取出的最优抗体放到一个集合中，在该集合中选取出最优的n个抗体。
6. 判断终止条件。
7. 在最优的n个抗体中选取出最优的抗体，输出。
8. 使用克隆选择算法进行虚拟网络映射使得成本开销的例子：

首先介绍下主要函数的作用：

**public** ArrayList<Integer> random\_solution(Graph virtual, Graph physic)：生成随机解，即将虚拟网络的各个节点映射到物理网络节点上。

**public** **int** object\_function(**int** hop, **int** virtual\_link\_bandwidth)：计算一条虚拟链路映射到一条物理链路后物理链路的成本。

**public** **void** evaluate(ArrayList<Po> pop, Graph physic, Graph virtual)：计算解空间中的各个解(抗体)的值。

**public** **double** num\_clones(**int** size, **double** clone\_factor)：计算每个解需要克隆出几个解

**public** **void** calculate\_affinity(ArrayList<Po> pop)：计算解空间中每个解的亲和度

**public** **double** calculate\_mutation\_rate(Po po, **double** mutate\_factor)：计算解的变异值

**public** ArrayList<Integer> point\_mutation(ArrayList<Integer> solution, **double** m\_rate,

Graph virtual, Graph physic)：对一个解进行变异

**public** ArrayList<Po> clone\_and\_hypermutate(ArrayList<Po> pop, **double** clone\_factor,

**double** mutate\_factor, Graph virtual, Graph physic)：对解空间中所有解进行克隆和变异

**public** ArrayList<Po> random\_insertion(ArrayList<Po> pop, **int** num\_rand, **int** pop\_size

, Graph virtual, Graph physic)：生成随机数量num\_rand的解，与原始解空间pop合并,并选取出最优的pop\_size个解

**public** Po search(Graph physic, Graph virtual, **int** max\_gens, **int** pop\_size,

**double** clone\_factor, **int** num\_rand, **double** mutate\_factor)：克隆选择算法的主体，不断克隆选择迭代，最终选取出最优解。

**其算法步骤是这样的：**

1. 随机生成固定数量pop\_size的解(抗体，即映射方案)，并利用函数evaluate计算各个解的值(即虚拟网络映射到物理网络后，物理网络消耗的成本)
2. 判断是否满足终止条件，本题我设置的终止条件为迭代max\_gens次，不满足则重复步骤3-
3. 在函数clone\_and\_hypermutate中对初始解空间中的解进行克隆变异，首先对于每一个解，使用num\_clones计算该解需要克隆出几个解，用calculate\_affinity计算该解的亲和度，基于亲和度和变异因子利用函数calculate\_mutation\_rate计算该解的变异值，然后基于变异值，利用函数point\_mutation对该解的克隆解进行变异。所有解的克隆变异后的解放入变异解空间clones中。
4. 计算变异解空间中的解的值。
5. 合并变异解空间和原始解空间中的解，排序，选取前pop\_size个解(最优pop\_size的解)重新赋予原始解空间，此时原始解空间中只有这前pop\_size个解。
6. 利用函数random\_insertion生成随机解，与原始解空间合并，并选取前pop\_size个解(最优pop\_size的解)重新赋予原始解空间。
7. 满足迭代条件，输出原始解空间的第一个解，即最优解。

克隆选择算法难以找到全局的最优解，它利用克隆选择，在迭代多次后，在庞大的随机解中一步步走向优化目标(成本低)，最终得到一个满足条件的近似最优解的较优解。

当然，如果迭代次数非常多，随机解空间十分庞大，有可能得到最优解。