# 概述：

遗传算法（Genetic Algorithm）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，而一个种群则由经过基因（gene）编码的一定数目的个体(individual)组成。每个个体实际上是染色体(chromosome)带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码，初始种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择（selection）个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding），可以作为问题近似最优解。

其他详细的介绍与资料，可以到网上搜索，这里不再作进一步的介绍。概括地讲，遗传算法主要解决四个方面的问题，即编码操作，选择操作，交叉操作以及变异操作。以下几个章节简单介绍一下遗传算子操作。

# 编码方式

TSP问题编码一般有五种不同的方式：

* 基于二进制的编码。比较传统的编码方式，但是这种方式的编码，在经过遗传操作以后很难保证后代还是一个可行解，还需要另外的修正操作；
* 基于矩阵的编码。比较复杂，并且需要占用大量的内存，应用的不是很广泛；
* 基于邻接的编码。同上；
* 基于索引的编码。同上；
* 基于路径的编码。 目前最普遍用的编码方式。因为它最直观最容易理解，操作起来也比较方便。

显然，对于二进制编码来说，其遗传操作是比较容易实现，这里不再详细解释。以下将主要介绍基于路径的编码方式以及相关的遗传操作，作为今后的资料索引。（可以继续补充）

# 选择操作

## 轮盘赌选择法

轮盘赌选择法是最常用的选择法，依据个体的适应度值计算每个个体在子代中出现的概率，并按照此概率随机选择个体构成子代种群。轮盘赌选择策略的出发点是适应度值越好的个体被选择的概率越大。因此，在求解最大化问题的时候，我们可以直接采用适应度值来进行选择。但是在求解最小化问题的时候，我们必须首先将问题的适应度函数进行转换，以将问题转化为最大化问题（TSP中可以将适应度设置为路程的倒数）。下面给出最大化问题求解中遗传算法轮盘赌选择策略的一般步骤：

(1)  将种群中个体的适应度值叠加，得到总的适应度值==1 ，其中 为种群中个体个数。

(2)  每个个体的适应度值除以总的适应度值得到个体被选择的概率

(3)  计算个体的累积概率以构造一个轮盘。

(4)  轮盘选择：产生一个[0,1]区间内的随机数，若该随机数小于或等于个体的累积概率且大于个体1的累积概率，选择个体进入子代种群。

重复步骤(4)次，得到的个体构成新一代种群。

## 随机遍历抽样法

该方法同样像轮盘赌一样计算选择概率，只是在随机遍历选择个体，设npoint为需要选择的个体数目，等距离的选择个体，选择指针的距离是1/npoint，第一个指针的位置由[0，1/npoint]的均匀随机数决定。

## 锦标赛选择法

锦标赛方法选择策略每次从种群中取出一定数量个体，然后选择其中最好的一个进入子代种群。重复该操作，直到新的种群规模达到原来的种群规模。具体的操作步骤如下：

(1) 确定每次选择的个体数量(本文以占种群中个体个数的百分比表示)。

(2) 从种群中随机选择个个体(每个个体入选概率相同) 构成组，根据每个个体的适应度值，选择其中适应度值最好的个体进入子代种群。

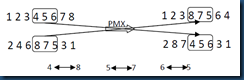
(3) 重复步骤(2)次，得到的个体构成新一代种群。

需要注意的是，锦标赛选择策略每次是从个体中选择最好的个体进入子代种群，因此可以通用于最大化问题和最小化问题，不像轮盘赌选择策略那样，在求解最小化问题的时候还需要将适应度值进行转换。

# 交叉操作

## 部分匹配法（Partially Matching Crossover，PMX）

以两个父代个体为例：（1 2 3 4 5 6 7 8）和（2 4 6 8 7 5 3 1），随机选择两个交叉的点，假如第一个点为位置4，第二个交叉点为位置6，那么在两个点之间的位置将进行交叉，其它的位置进行复制或者用相匹配的数进行替换。在此实例中，第一个父代个体中4 5 6被选中，第二个父代个体中，8 7 5被选中。那么4 与8，5与7，6与5相匹配。匹配过程和如图所示。

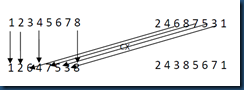


PMX操作

首先将4 5 6与8 7 5分别加入到子代2和子代1中相应的位置，然后将其他位置上的数字直接复制到相应的后代中，如果该数字已经在该子代中已经存在，则用相应的匹配法则进行替换，例如子代1中将7复制进去的时候，发现7已经存在于子代中，通过查找相应的匹配法则，发现，7与5匹配，然后复制5，又发现5也已经存在于该子代中，在此查找匹配法则，发现5与6匹配，将6复制，6不存在于该子代中，所以可以将6复制进去，如此反复，直到子代中城市的数目达到定义的长度，该子代创建完成。

## 循环交叉法（Cycle Crossover，CX）

以两个父代个体为例：（12345678）和（24687531），在构造子代的过程中，首先从父代1中选取第一个元素，然后查找父代2中对应位置的元素在在父代1中的位置，将这个位置对应的元素加入到子代1中，如此反复，直到找到一个元素对应的父代2元素等于起始元素，次循环结束，然后将剩余未填充的位置用父代2相应的位置的元素进行填充，这样就可以完成子代1的创建。子代2创建方法类似。CX操作过程如图所示。



CX操作

首先选择父代1中的第一个元素1，将它加入到子代1中，然后检查父代2中对应位置，该位置元素为2，在父代1中查找该元素，该元素在父代1中的位置为2，将2加入到子代1的第二个位置，再次检查父代2中第二个位置的元素，它为4，然后查找它在父代1中的位置为4，将4加入到子代1的第四个位置，然后将其加入到子代1中对应的位置4，在检查父代2中该位置的元素，它为8，查找到它在父代1中的位置为8，然后将其加入到子代1中位置8，再次查找父代2中位置8的元素，它为1，等于起始选择的元素，循环结束，然后将子代1中3567元素为空的位置，用父代2对应位置的元素进行填充，这些元素为6753，所以得到的子代1为（12547538）。同样的方法，得到子代2为（24385671）。

## 次序交叉法1（Order Crossover，OX1）

以两个相同的父代个体为例：（12345678）和（24687531），随机选择两个交叉的点，假如第一个点为位置3，第二个交叉点为位置5，那么在两个点之间的位置将进行交叉。然后从第二个交叉点开始，将原来相应的父代按照顺序进行填充，如果选择的元素已经存在于该子代中，跳过该元素，选择下一个元素，这种过程反复进行，知道所有的城市都被选择一次。在此实例中，第一个父代个体中345被选中，第二个父代个体中，687被选中。匹配过程和如图所示。

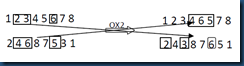
image

OX1操作

首先，将选择的位串进行替换，即345换到子代2中，687换到子代1中。现在从第二个交叉点开始，将父代1中的基因插入到子代1中，所以，1插入到第六个位置，2插入到第七个位置，3插入到第八个位置，第二个交叉点后面已经填充满，将剩下的插入到第一个插入点前面的位置，所以4插入到第一个位置，5插入到第二个位置。这样，子代1构建完成。同样地方法，可以构建子代2。当遇到子代中已经存在的基因，就跳到下一个。

## 次序交叉法2（Order Crossover，OX2）

以两个相同的父代个体为例：（12345678）和（24687531），随机选择几个交叉的点，假如第一个点为位置2，第二个交叉点为位置是3，第三个为位置6。首先找到父代2中相应位置的基因在父代1中位置，然后将其用父代2中选择的位置的基因进行替换插入到子代1中相应的位置中。然后将其它的位置用OX1相似的方法进行填充。这样就可以得到子代1。同样，替换角色，可以得到子代2。具体的操作过程如图所示。



OX2操作

首先找到父代2中第，2，3以及第六个位置的基因，它们为4，6和5，这三个基因在父代1中的位置为4，5和6，将它们替换成465，加入到子代1中相应的位置，然后将父代1中的基因按照顺序插入到子代1中，如果该基因已经存在在位串中，则跳过该基因，继续下一个。这样就可以构建完子代1。子代2也可以以相同的方法构造完成。

## 基于位置的交叉法（Position Based Crossover，POS）

还以两个相同的父代个体为例：（12345678）和（24687531），随机选择几个交叉的点，假如第一个点为位置2，第二个交叉点为位置是3，第三个为位置6。将两个父代中这些选择基因按照顺序交换，并且插入到子代1和2中。然后对其它位置的基因，按照顺序插入。具体操作过程如图所示。

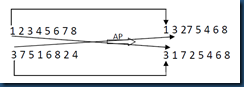
image

POS操作

首先将236和468交换，分别插入到子代2和子代1相应的位置2，4，6中，然后将父代1和2中的基因按照顺序插入到子代1和2中，如果该基因已经存在在位串中，则跳过该基因，继续下一个，直到位串长度等于定义的长度。

## 交替位置交叉法（Alternating Position Crossover，APX）

以两个父代个体为例：（12345678）和（37516824），APX是一种简单的交叉操作方法，它是轮流选择父代1和2中的基因，直到子代的长度达到定义的长度为止。具体操作如图所示。



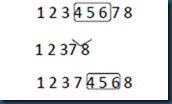
APX操作

首先选择父代1中的第一个元素1，加入到子代1中，然后选择父代2中的第一个元素3，它不等于1所以也加入到子代1中，然后再选择父代1中的第二个元素2，它也不包含在当前的位串中，所以也加入进去，然后再选择父代2中的第二个元素，……，直到子代1长度达到8。同样的方法，可以得到子代2。

# 变异操作

## 替换变异（Displacement Mutation，DM）

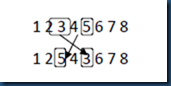
DM先从父代个体中选择一个子位串，然后再随机在剩下的位串中选择一个为止，并插入该子位串，如图所示。



DM操作

## 交换变异（Exchange Mutation，EM）

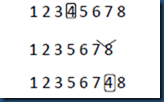
EM是从父代个体中选择两个基因位置，然后互换这两个位置的基因。如图所示。



EM操作

## 插入变异（Insertion Mutation，IM）

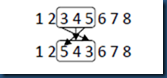
IM和DM类似，唯一的区别在于它是从父代个体中只选择一个基因，然后随机的插入到剩下的基因串中。如图所示。



IM操作

## 简单倒位变异（Simple Inversion Mutation，SIM）

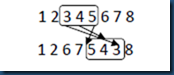
SIM先从父代个体中选择一个基因串，然后将这个基因串中所有的基因倒位放置，如图所示。



SIM操作

## 倒位变异（Inversion Mutation，IVM）

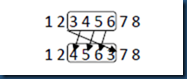
IVM在SIM的基础上，随机选择一个插入位置，将这个到位后的基因串在插入到其中。如图所示。



IVM操作

## 争夺变异（Scramble Mutation，SM）

SM非常简单，先随机从父代个体中选择一个基因串，然后将除了第一个基因外的基因均向前移动一位，将第一个基因移到最后一位。具体操作过程如图所示。



SM操作

**以下是pdf:**

**Clever Algorithms Nature-Inspired Programming Recipes**

**中第三章关于进化算法的内容解释与说明：**

# 进化算法pdf内容

从3.2到3.11这些算法一些区别在于：编码方式（都是二进制串，只是有的把运算符也进行编码）、（亲代/子代）形成新种群的方式、遗传算子的选择（是否有交叉、变异、删除、复制，以及如何选择操作对象）、是否进行表现型的解码、映射并进一步计算成本函数等。具体细节当然有所不同。要使用的话可以先看一下代码清单提供的例子，清楚直观。简单分类如下：

3.2-3.11的十个小节可以分为四种类别的目标函数求解：

① 3.2 是求解MaxOne问题（没有编程实现的必要，直接归到**一元单目标函数**中）。

② 3.3，3.7，3.8求解一元多项式最值，只有一个变量x，并对四则运算符号也进行编码。**（一元单目标函数）**

③ 3.4，3.5，3.6求多项式最值，但是不对运算符编码，变量是一个x向量（也可理解为多元函数）。**（多元单目标函数）**

④ 3.10与3.11只有一个变量x，对多个函数目标的优化求解，即帕累托最优。**（多元多目标函数）**

**（本部分使用该pdf中的章节编号）**

## 3.2 遗传算法

遗传算法是一种自适应策略和全局优化技术。遗传算法是遗传规划（3.3节），进化策略（3.4节），进化规划（3.6节）和学习分类器系统（3.9节）等其他进化算法的同胞。遗传算法是许多不同的变体技术和子领域的父类。

**伪代码大意**

输入：种群规模、问题规模、交叉概率、变异概率

输出：最优解

目标：最小化成本函数（或者说适应度）

迭代过程：

选择亲代

迭代过程中亲代进行交叉（0.95-0.99），子代进行变异（1/L）

亲代全部舍弃，只留下子代

选出最佳子代

**代码清单**

OneMax问题：

目标函数仅提示二进制候选字符串中正确位数的数目，而不是正确的比特位置。

遗传算子操作：

二进制锦标赛选择法。

单点交叉：在二进制编码中，随机选择一个点，以这个点为界限，相互交换变量。如 ：  
父个体1      01111111**0000000000**   
父个体2      00000000**1111111111**  
如粗体前边位置为所选择的交叉点，那么生成的子个体为：  
子个体1      011111111111111111  
子个体2      000000000000000000

点变异：某一点0/1的变换

## 3.3 遗传规划

遗传规划算法是一个演化算法的例子。属于进化计算与领域，是更广泛的计算智能和生物启发计算。遗传编程算法是遗传算法（3.2节），进化策略（3.4节），进化规划（3.6节）和学习分类器系统（3.9节）等其他进化算法的同胞。从技术上讲，遗传规划是遗传算法的一个扩展。遗传规划算法的目标是使用归纳来设计一个计算机程序。 这是通过在具有树结构的候选程序上使用演化算子来实现的，可以改善候选程序和目标函数之间的自适应匹配。

**伪代码大意：**

遗传规划算法用于归纳式自动编程，非常适用于更广泛的函数逼近下的符号回归，控制器设计和机器学习任务；

使用类LISP语言的符号，称为S-expresssions，表示具有功能节点和终端节点的程序图，并采用深度优先遍历；

功能节点使用的所有功能都必须返回可用结果。例如，除零必须返回一个合理的值（例如零或一）。

选择过程应该在随机选择和贪婪选择之间取得平衡；

高概率的交叉（≥90％）和低概率的变异（≤1％）；

复制和架构变更则为中等概率；

架构变更操作不限于子结构的重复和删除；

目标：最小化成本函数

输入：种群规模、功能节点、终端节点、交叉概率、变异概率、复制（reproduction）概率、变更（alteration）概率

输出：

子代集合初始化

迭代过程（子代数目小于种群数目）：

选择遗传算子（输入的四个概率）

如果算子满足交叉算子（应该是概率满足）

选择两个亲代

交叉产生子代

加入子代集合

如果算子满足变异算子（应该是概率满足）

选择一个亲代

变异产生子代

加入子代集合

如果算子满足复制算子（应该是概率满足）

选择一个亲代

复制产生子代

加入子代集合

如果算子满足变更算子（应该是概率满足）

选择一个亲代

架构变更（Alter Architecture）产生子代

加入子代集合

从子代集合选出最优解

**代码清单：**

示范问题是一个符号回归的例子，其中必须设计一个函数来匹配一组观察值。 在这种情况下，目标函数是二次多项式x^2+x+1，x∈[-1,1]。

（整个式子当作染色体啊？）

此时功能集合为{+，-，\*，/}，终端节点集合为{X,R}，x ∈ [−5, 5]，R为静态随机变量。设置除以0返回1。通过对随机输入值的范围进行评估并计算均方根误差（RMSE）来计算候选解的适应度。该算法配置有90％的交叉概率，8％的复制概率以及2％的变异概率。为简洁起见，该算法不实现改变遗传操作的架构，并且不会将交叉点偏向终端上的功能。

## 3.4 进化策略

进化策略是一种全局优化算法，是进化计算领域的进化算法的一个实例。 进化策略是其他进化算法（如遗传算法（3.2节），遗传编程（3.3节），学习分类器系统（3.9节）和进化规划（3.6节））的兄弟技术。进化策略算法的流行后代是协方差矩阵适应进化策略（CMA-ES）。

进化策略算法的目标是最大限度地从域中目标函数的上下文中收集候选解的适用性。目标是通过采用动态变化来经典地实现的，其中变量是基于性能的启发式动态调整的。当前方法是通过与候选解决方案共同调整控制变量的数量和偏差的参数。

**伪代码大意：**

进化策略算法的实例可以用自定义的术语来简洁地描述，表达形式为（μ，λ）-ES。μ是亲代中候选解决方案的数目，λ是从亲代产生的候选解决方案数目。λ>=μ.选择策略是从亲代子代集合中选择。

1 ≤ μ ≤ λ；μ与λ的比率影响算法施加的选择压力（贪婪）的量。

1/5规则：成功变异的比率应该是所有变异的1/5。

输入：μ、λ、问题规模

输出：最优解

目标：最小化成本函数（或者说适应度）

初始化种群

评价种群

得出最优解

循环过程：

子代集合初始化

从0到λ遍历

选择亲代

解集合初始化

变异产生问题解（标记）

变异产生策略解（标记）

加入子代集合

评价子代集合

从子代集合与最优解中选出最优解

亲代、子代重新形成种群

选出最佳子代

**代码清单：**

进化策略算法的实例是寻求连续函数f（x）的优化，其中。最优解显然是。算法是一个（30,20）-ES，要能够修改策略变量，并包含一组额外的适应策略参数来影响变异的方向。

## 3.5 差异进化

差异进化是一种用于优化问题的启发式算法，包含变异和交叉操作。它是一种随机直接搜索和全局最优化算法，是进化计算领域的一个算法实例。但同时相较于遗传算法的选择操作，差异进化算法采用一对一的淘汰机制来更新种群。跟PSO更类似，是单个个体的进化。（从伪代码看感觉更像向两个方向进化的样子）

差异进化算法要维持候选解决方案的种群，要经过重组、评估和选择的迭代过程。重组方法涉及创建新的候选解决方案，这是基于第三成员与两个随机选择的成员之间的加权差异。

差异演化有一个专门的术语来描述采用的配置，即DE/x/y/z。x表示待处理的解（随机或最佳）。y表示随机选择的两个个体的差异（向量表示）。最后，z表示重组运算符（二项式或是指数。）

伪代码大意：

差分进化适用于非线性、不可微分的连续函数优化；

加权因子F ∈ [0, 2]控制差异变化的放大，建议值为0.8；

交叉率CR=0.9；

初始种群应为随机生成；

第一段：DE/rand/1/bin

目标是最小化成本函数

输入：种群规模、问题规模、权重因子、交叉率

输出：最优解

初始化种群（P）

评价种群

得出最优解(候选集S)

循环过程：

初始化新种群

遍历种群中的每一个个体（Pi）

Si<-NewSample（个体，种群，问题规模，权重因子，交叉率）

如果Si成本小于Pi

Si加入新种群

否则

Pi加入新种群

结束遍历后新种群就是目前要操作的种群了

评估种群得出最优解（继续循环）

第二段：NewSample函数

输入：P0，种群，NP，F，CR

输出：S

找出非P0的个体P1、P2、P3

得出切点（NP问题）

S<-0

遍历切点可能性

如果i等于切点并随机<CR

Si=P3i+F（P1i-P2i）

否则

Si<-P0i

**代码清单：**



## 3.6 进化规划

进化规划是一种全局优化算法，是进化计算领域的进化算法的一个实例。简答地说，只考虑变异过程，将亲代子代放一起选出最优解。.

采用锦标赛选择法

样本量（Bout规模）通常在人口规模的5％至10％；

目标是最小化成本函数

输入：种群规模、问题规模、Bout规模

输出：最优解

初始化种群（P）

评价种群

得出最优解(候选集S)

循环过程：

初始化子集

对于每一个亲代，变异产生子代然后加入子代集合

评价子代集合，得出最优解

将原本种群与子代合成新集合

遍历新集合每个个体Si

遍历抽样集合BoutSize

随机选择抽样集合中的个体Sj

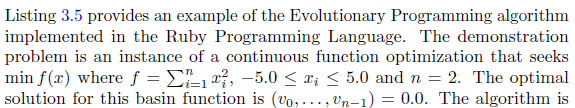
如果Si成本小于Sj，Si的胜出次数+1

得到新种群（选出胜出次数最多的那些个个体组成，种群规模即为上述的种群规模）

得到最优解

**代码清单：**

例子同遗传规划相同



## 3.7 语法进化

语法进化是一种全局优化技术。语法进化算法的灵感来自用于从遗传物质产生蛋白质的生物学过程以及更广泛的遗传进化过程。基因组由DNA组成，可以被转录成RNA。RNA密码子又被翻译成氨基酸序列并用于蛋白质中。在其环境中产生的蛋白质是表现型。举例来说就是一个二进制子串可以映射为一个 “表现型”，即可变长度的二进制字符串可以演化为符号的形式，并映射为符号和结构良好的形式。

**伪代码大意：**

一个八位字符串就可以表示2^8个性状（解码为整数形式）。

单点交叉；

锦标赛选择；

目标函数：最小化成本函数

输入：语法、密码子数目、种群规模、交叉概率、变异概率（1/L）、删除概率（0.5/NC）、复制概率（1.0/NC）

输出：最优解

初始化种群

对于种群每一个个体Si

解码为整数形式->依据一定的语法映射->计算成本

得出最优解

循环过程：

选择亲代

初始化子代集合

对于两两的亲代

交叉，密码子删除，密码子复制，变异

加入子代集合

对于子代每个个体Si

解码为整数形式->依据一定的语法映射->计算成本

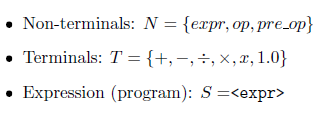
得出最优解

种群和子代形成新的种群

**代码清单：**

举例：

语法：



## 3.8 基因表达规划

（简述见pdf）

候选方案表示为线性的符号串（K表达式），每个符号都映射为一个功能节点或终端节点。以广度优先遍历的方法映射到表达树上（表现型是树）。K表达式具有固定的长度，由一个或多个子表达式（基因）组成，这些子表达式也用固定的长度来定义。 一个基因由两部分组成，头部可以包含任何功能或终端符号，尾部可以只包含终端符号。每个基因总是翻译成句法正确的表达树，其中基因的尾部提供确保表达关闭的遗传缓冲区。

**伪代码大意：**

目标是最小化成本函数

输入：语法、种群规模、头部长度、尾部长度、交叉概率、变异概率

输出：最优解

初始化种群（P）

对于种群每个个体Si，广度优先遍历解码，计算成本函数

得出最优解(候选集S)

循环过程：

选择亲代集合

初始化子代集合

对于亲代集合进行交叉、变异操作，得到子代集合

对于子代集合中的个体Si，广度优先遍历解码，计算成本函数

以当前种群与子代集合形成新的种群

得出最优解

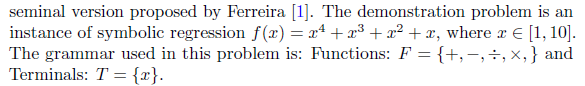
**代码清单：**

染色体长度（h+t），t=h（n-1）+1，n是功能节点最大长度

功能和终端节点在基因头部突变，而在基因的尾部只有终端端节点被替换，但是突变运算符取代基因组中的表达。

单点交叉；还有倒置、复制算子

例子：



## 3.9 学习分类系统

学习分类器系统算法既是来自进化计算领域的进化算法的实例，也是来自机器学习的增强学习算法的实例。学习分类系统是一个理论系统，有很多实现，可以使用遗传算法。实践和调查系统的两个主要方法是寻求优化整个分类器的匹兹堡风格和优化响应规则集的密歇根式风格。密歇根式学习分类器是最常见的，由两个版本组成：ZCS（零级分类器系统）和XCS（基于精度的分类器系统）。

学习分类器系统算法的目标是根据暴露于特定问题环境的刺激来优化报酬。 这是通过管理那些被证明有用的规则分配来实现的，并使用进化过程来寻找新规则和现有规则的新变化。系统的角色包括探测器、消息、效应器、反馈和分类器。系统使用探测器来感知环境的状态。信息是从检测器传送到系统的离散信息包。系统对消息进行信息处理，消息可能直接导致环境中的行为。效应器控制系统在环境内部和环境内的行为。除了通过检测系统主动感知外，系统还可以从环境中获得定向反馈（回报）。分类器是为消息提供过滤器的条件操作规则。如果消息满足分类器的条件部分，则分类器的动作会触发。规则充当消息处理器。消息一个固定长度的位串。处理过程如下：

1.来自环境的消息放在消息列表中。

2.检查每个分类器的条件，看它们是否被消息列表中的至少一条消息满足。

3.所有满意的分类参与竞争，获奖者将其行为发布到消息列表中。

4.所有发送到效应器的消息都被执行（在环境中引起动作）。

5.上一个周期的消息列表中的所有消息都将被删除（消息会持续一个周期）。

该算法可以根据主处理循环和两个子算法来描述：强化学习算法，例如桶式旅行（brigade）算法或Q学习，以及用于优化系统的遗传算法。

**伪代码大意：**

以XCS学习分类系统为例，适用于具有以下特征的问题：具有显着噪音的永久新事件，对动作的持续实时要求，隐含或不精确定义的目标，以及只有通过长序列任务才能获得的稀疏回报或强化。

分类器的学习率β所期望的回报、误差与适应度范围在[0.1,0.2]；（时刻1初始化为0）

遗传算法运行频率θGA为[25,50];

多步程序中的折扣因子γ为0.71；

分类器考虑的最小误差的精确度与最大回报的10%一致；

分类器中遗传算法的单点变异率为[0.01,0.05]；

Θdel的经验门槛为20；

Θsub的经验门槛为20；

选择随机动作的概率为0.5；

…

输入：环境描述

输出：种群

初始化环境

初始化种群

初始化t-1时刻的动作集、输入与奖励

循环过程：

输入t时刻环境

产生匹配集，预测，选择动作并加入到动作集合中，执行得到奖励

如果t-1时刻的动作集不为空

计算回报->执行学习->运行遗传算法

（不写了，不是很懂这个，先放着）

**代码清单：**

问题举例：布尔多路复用器（6路），2^6个模式的比特都是0/1的。例子中，前两位可能解码为地址，后四位可能解码为类别。其逻辑可以描述为：

3.10与3.11为多目标优化，简单介绍见

<http://blog.csdn.net/mimi9919/article/details/51234456>

<https://www.cnblogs.com/lomper/p/3831428.html>

## 3.10 非支配排序遗传算法

假设任何二解S1 及S2 对所有目标而言，S1均优于S2，则我们称S1 支配S2，若S1 的解没有被其他解所支配，则S1 称为非支配解。这是一个多目标优化（MOO）算法。多目标优化就是一个解要满足对多目标的优化，按对每个目标的优化程度排序。从总体中找出对多目标优化程度较好的解。在这个情境下，非支配的意思就是说一个解相对于另一个解来说，每个目标的优化程度都比较好。

另外，多目标规划中，由于存在目标之间的冲突和无法比较的现象，一个解在某个目标上是最好的，在其他的目标上可能比较差。Pareto 在1986年提出多目标的解不受支配解(Non-dominated set)的概念。其定义为：假设任何二解S1 及S2 对所有目标而言，S1均优于S2，则我们称S1 支配S2，若S1 的解没有被其他解所支配，则S1 称为非支配解（不受支配解），也称Pareto解。这些非支配解的集合即所谓的Pareto Front。所有坐落在Pareto front中的所有解皆不受Pareto Front之外的解（以及Pareto Front曲线以内的其它解）所支配，因此这些非支配解较其他解而言拥有最少的目标冲突，可提供决策者一个较佳的选择空间。在某个非支配解的基础上改进任何目标函数的同时，必然会削弱至少一个其他目标函数。

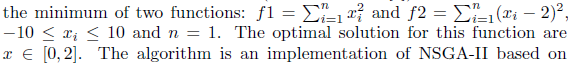
NSGA算法的目标是提高一组候选解对Pareto前沿的自适应性，该Pareto前沿被一组目标函数约束。该算法使用进化算子的演化过程，包括选择，遗传交叉和遗传变异。基于帕累托优势的排序，将种群分成多个等级的子种群。每个小组的成员之间的相似性在帕累托正面被评估，并且由此产生的小组和相似性措施被用来促进非主导解决方案的多样性前沿。

**伪代码大意：**

…（挖坑不填，大意也是通过遗传操作，得出一个一元多目标的最优解）

**代码清单：**

举例说明：



## 3.11 强帕累托进化算法

强度Pareto进化算法是多目标优化问题的遗传算法的扩展。它与同胞进化算法如非支配排序遗传算法（NSGA），矢量评估遗传算法（VEGA）和帕累托存档进化策略（PAES）有关。有两个版本的SPEA，原始SPEA算法和扩展SPEA2。其他扩展包括SPEA +和iSPEA。

该算法的目标是定位和维护非主导解决方案的前沿，理想情况下是一组帕累托最优解。这是通过使用进化过程（具有用于遗传重组和突变的替代程序）来探索搜索空间来实现的，并且选择过程使用候选解决方案被主导的程度（强度）和密度估计作为分配的适应度的帕累托前线。一个非支配集合的档案与进化过程中使用的候选解决方案的人口是分开的，提供了一种精英主义的形式。

# 问题代码实现

关于程序的解释，代码注释已经足够详细了，这里在总体上介绍一下。

## 2.1 TSP问题求解

本代码使用遗传算来解决TSP问题。该问题的实现代码放在工程文件夹package的TSP中

程序分成两个java类，Chromosome类和GA类。其中，Chromosome类是定义的染色体类，使用基于路径的整数编码来表示染色体，比如有六个城市{1，2，3，4，5，6}，那么用基于路径的编码(1 3 5 4 6 2)就可以表示为一个染色体。Chromosome类中，随机染色体（即随机解）的生成方法为randomGeneration()；适应度（即旅行的路程）的计算在方法calculatefitness()中。但是这里做了一下处理，将总路程的倒数作为适应度，以便选出适应度最大的（即总路程最小的），没什么实质影响。

此外，具体的遗传操作，包括选择、交叉和变异的方法都在GA类的evolve()方法中。每个方法都有详细的注释

主类main的实现代码为：

GA ga = **new** GA(100, 10, 100, 0.9, 0.05);

ga.solve();

## 2.2 多元单目标函数问题求解

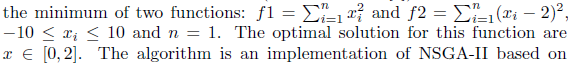
这里解决的是简单的多元函数求极值问题。虽然是多元单目标的，稍微修改一下就是单元单目标函数。该问题的实现代码放在文件夹package的functionExtremum中。

同样的，程序分成两个java类，Chromosome类和GA类。在Chromosome类中，用注释来举例了该程序要解决的函数表达式。该程序使用基于二进制的编码方式来表示染色体。

具体的遗传操作也都在GA类中（其实没有使用到Chromosome类）。关于遗传算法中的一些操作也都在GA类中，重要函数都有注释。相关内容与操作都类似同上。

## 2.3 多元多目标函数问题求解

这里解决的是简单的多元多目标的最值问题，即帕累托最优问题。该问题的实现代码放在文件夹package的paretoOptimization中。该程序解决的问题举例如下：



同样的，程序分成两个java类，Chromosome类和GA类。由于这个包的代码GA类代码是将pdf文件里的Ruby语言代码转译成java，所以看起来会怪怪的。其中，Chromosome类是定义染色体的类，基本根据ruby代码的一些提示与调用组合成的。该Chromosome类中也只定义了相关属性与get、set方法。

具体的遗传操作，包括选择、交叉和变异的方法都在GA类的中，使用基于二进制编码来表示染色体。遗传操作类似同上，从函数名称可以大概看出函数是用来干嘛的。

## 2.4 虚拟网络映射问题求解

该程序解决的问题是虚拟网络映射问题，实现代码放在文件夹package的virtualNetworkEmbedding中。

使用该程序，首先需要导入Graph包。它包含的是对物理网络与虚拟网络的定义与操作，包括对节点、链路等的定义。通过该包可以初始化物理/虚拟网络（读入文件P.txt与V.txt）。该图的实例方法dijkstra() 可以求得两个顶点之间的最短距离，并返回该最短路径的节点序列数组，同时消耗链路上的带宽（第三个参数指定）。refreshMatrix()方法用于更新图的邻接矩阵，使其恢复原样。

virtualNetworkEmbedding包中包含的类与TSP包类似，方法也类似。Chromosome类中随机染色体（即随机解）的生成方法为randomGeneration()，使用属性geneSequence[]来保存。适应度（映射的开销）的计算在方法calculatefitness()中。

GA类包含的就是对染色体的遗传操作。主函数操作方法类似于TSP包的主函数。

该程序流程为：

1. 初始化随机解，解用整型数组表示，索引表示虚拟网络节点编号，值表示映射的物理节点编号。
2. 进行选择、交叉与变异操作。适应度表达式为映射成本。
3. 迭代得出最优解。