**《程序设计课程设计》实验报告**

**实验名称 《Jobshop车间调度设计》**

**班 级 2017211307**

**组 号 24**

**姓 名 雷一鸣 刘翔宇 张开舜**

# 1．用户界面设计

## 1.1 用户界面（略）

# 1.2 操作元素和操作效果（略）

# 3 高层数据结构设计

**3.1全局常量/变量定义**

double型MUTATION\_RATE (变异率）

double型CROSSOVER\_RATE（交叉率）

int型ITERATION\_COUNT（迭代次数）

int型POPULATION\_SIZE（种群大小）

**3.2 模块常量与变量定义**

遗传算法GA模块内的全局变量

1. product\_count 记录产品数量
2. proced\_count 记录每个产品的工序数量
3. chromo\_len 染色体长度，其值为(1)\*(2).
4. machine\_count 记录机器数量

**3.3 全局数据类型**

DVector模板类，用作安全数组.

**3.4 自定义数据类型**

(1) typedef (int) TIME.

(2) typedef DVector<DVector<T>> DVector2D<T>

(3) LAUNCH\_PARAMETRE结构，用于传递运行参数

公有成员:

布尔型from\_file, to\_file

字符串infile\_name, outfile\_name

(4) CHROMO结构，用于保存单个染色体

公有成员：

DVector<int>型chromo, 记录染色体内容

TIME型time，记录染色体代表的工作时间

(5) PROCEDURE结构，用于保存工序

公有成员：

int型machine，记录机器号

TIME型duration，记录工序时长

(6) MACHINE\_TASK结构，用于保存机器的工作单位

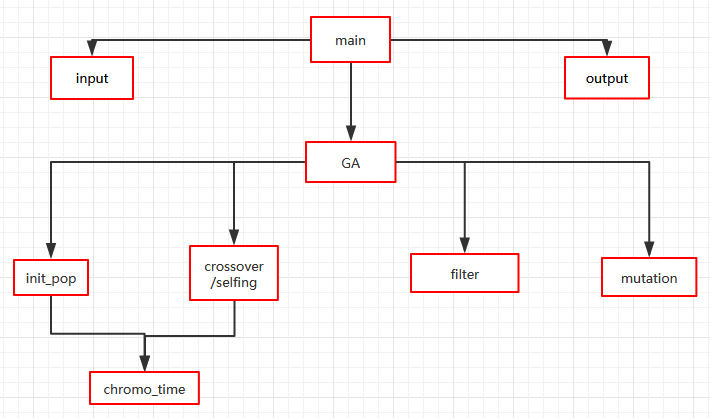
公有成员：

TIME型start, end，记录开始和结束时间

unsigned int型prod, proced，记录工件和工序号

# 4 系统模块划分

## 4.1 系统模块结构图



**模块划分思路说明：**

1. **模块名称：input**

功能说明：文件输入和命令行输入，图形化输入

1. **模块名称：output**

功能说明：文件输出和命令行输出，图形化输出

1. **模块名称：GA**

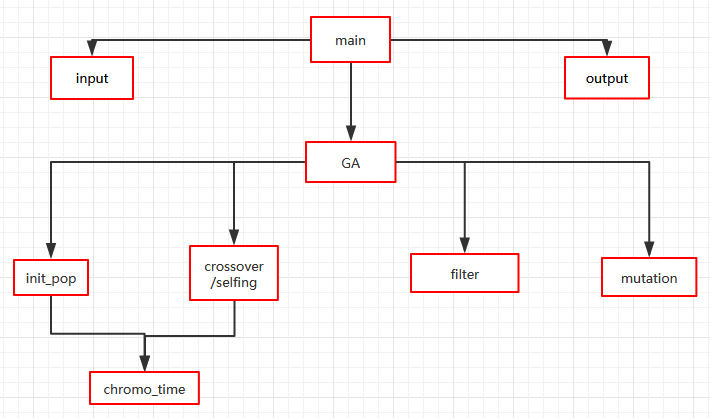
功能说明：遗传算法的实现

1. init\_pop: 初始化种群
2. communication: 种群间基因交流
3. crossover/selfing: 杂交或自交
4. filter：筛选
5. mutation：变异
6. chromo\_time：计算单个染色体代表的工作时间

## 4.2各模块函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
| 1 | LAUNCH\_PARAMETRE read\_under\_cmd(  DVector2D<PROCEDURE>& table, bool from\_file,  std::string infile\_name); | 在命令行下输入数据 | table(引用)：空工序表  from\_file：是否从文件输入  infile\_name：从文件输入的文件名，默认空 | LAUNCH\_PARAMETRE结构.包含运行参数. |
| 2 | void  print\_under\_cmd(  const DVector2D<MACHINE\_TASK>& res\_table,  bool to\_file,  std::string outfile\_name); | 在命令行下输出结果 | res\_table(常引用): 结果表  to\_file：是否向文件输出  outfile\_name：向文件输出的文件名，默认空 | 无 |
| 3 | CHROMO jobshop\_GA(  const DVector2D<PROCEDURE>& jobtable); | 实现遗传算法 | jobtable(常引用):工序表 | CHROMO型.含得到的最好调度编码及时间 |
| 4 | void init\_population(  DVector<CHROMO>& population, const DVector2D<PROCEDURE>& jobtable); | 生成初始种群 | population（引用）:空种群表  jobtable（常引用）：工序表 | 无 |
| 5 | void crossover(  const DVector2D<PROCEDURE>& jobtable,  DVector<CHROMO>& population); | 种群交配 | population（引用）：种群 | 无 |
| 6. | void filter(  DVector<CHROMO>& population); | 筛选种群 | population（引用）：种群 | 无 |
| 7 | void mutate(  DVector<CHROMO>& population); | 种群变异 | population（引用）：种群 | 无 |
| 8 | TIME chromo\_time(  const DVector2D<PROCEDURE>& jobtable,  const DVector<int>& chromo); | 种群变异 | jobtable（常引用）：工序表  chromo（常引用）：染色体数组 | TIME型，时间 |

## 4.3 函数调用图示及说明



main函数调用input output 函数实现输入输出

main函数调用GA函数实现遗传算法

GA 函数调用init\_population函数实现随机种群的产生

GA 函数调用crossover函数实现种群的交叉

GA函数调用filter函数实现对种群的筛选

GA函数调用mutation函数实现对种群的变异

GA函数调用communication函数实现对种群间的基因交流

init\_population和crossover函数调用chromo\_time函数计算时间

# 5 高层算法设计

**（用伪代码、NS图或者自然语言描述清楚核心算法的程序设计思路）**

**采用遗传算法**

一、部分名词说明

①染色体（个体）：每一条染色体都带有不同的基因序列，基因序列代表着各个工件的加工顺序，所以每一条染色体代表一个解决方案。

②种群：所有染色体（个体）的集合称作种群，即为所有解决方案的集合。（染色体的数目庞大，并不会用到所有染色体）

③适应度：判断染色体优劣的值。通过适应度来保留或者淘汰个体（染色体）

④选择：一种基于适应度的优胜劣汰的过程。

⑤交叉：类比于有性生殖过程中基因的重组，对染色体部分进行交换，形成下一代。⑥变异：类比于生物进化过程中基因的编译，对染色体部分基因进行变异，形成突变个体。

⑦种群间交流：类比于群落进化时种群间的基因交换，可加快收敛速度。

二、算法主要内容

①编码：染色体长度为总工序数目，染色体序列的每一个元素是[1,N]之间的一个数，该数字在序列中第几次出现表示它所代表工件的第几道工序。

假设工件1，2，3分别有3道工序，那么我们可以得到其中一种染色体

https://lh4.googleusercontent.com/6X4QEDb-T5RymvTEnWT2yp_5XNaL4_t3fpHQzNbj5YcW6Ata0bu-oUD8BDfe4C6HaY0RSzkqYY88F3lG4xaVuwKdPCP9gBJ1G-CbadZM4CYGUslNojDz4C63cFG4kRAw-YvF2lPl

第一个3代表工件3的第一道工序，第二个3代表工件3的第二道工序，第三个3代表工件3的第三道工序，同理，第一个1代表工件1的第一个工序。

②初始化种群

假如我们对上面的例子进行随机排列，可以得到 9! / (3! \* 3! \* 3!) = 1680 种情况，当工序数增加的时候，这个数据会更加庞大，所以我们只选择随机排列后的一部分染色体组成初始种群。

③适应度值的计算

某条染色体的适应度值与该条染色体加工时间有关。根据帕累托最优原则，每个染色体的适应度值代表该染色体的优势排名。

我们面临两个问题：1.各个工件不同的工序之间有严格的先后顺序。

        2.同一台机器在某个时刻只可以加工一道工序。

第1个问题已经通过编码的方式解决，第2个问题解决方案如下：

我们采用二维数组 jobtable[ ][ ] 记录第 i 个工件第 j 道工序的机器号和耗费的时间

采用二维数组 end\_time[ ][ ]记录第 i 个工件第 j 道工序加工结束的时间

采用二维数组 every\_machine\_time[ ]记录每台机器的空闲时间点（总时间）

步骤如下：

1.染色体编码序列可知要加工工件的工件号 i ，工序号 j 。

2.由工件号和工序号从二维数组 jobtable[ ][ ] 获得第 i 个工件的第 j 道工序所使用的机器号和所耗费的时间

3.然后更新第 i 个工件第 j 道工序的开始时间为第 i 个工件第（j-1）道工序的结束时间（从end\_time[ ][ ]  获取）和该工序将要使用机器的最近空闲时间点的之中的较大者

4.更新第 i 个工件第 j 道工序的结束时间为该工序的开始时间和加工时间之和（end\_time[ ][ ]）更新该工序所使用机器的空闲时间点（every\_machine\_time[ ]）为第 i 个工件第 j 道工序的加工完成时间。

5.找到所有机器中时间的最大值即为加工时间 T。

④选择

选择过程使用 “轮盘赌算法”

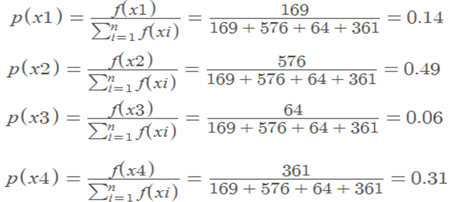
轮盘赌算法：

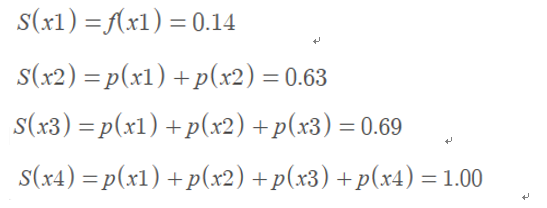
我们基于染色体适应度采用轮盘赌算法对种群进行筛选。

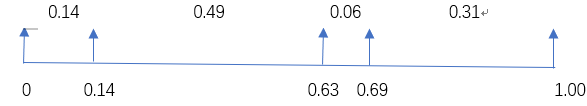
假设我们有四条染色体

https://lh3.googleusercontent.com/qInw_RMHSYQ1G5jFqxjRJzBlAp2wYz7RKsL4wqb8oc7H6JHtH1U3SiUaf00ZAi1Jc-uWXBA0FJapLW5MC1sMY8ezyzoglJaQrHc0WdG91OSw_zfaDM-WhdM502JhOZVM2aJys3jL

染色体的选择概率为：



染色体的累计概率为：



当我们产生多个[ 0 ,1]的随机数时候

我们可以发现适应度高的染色体被选中次数多，符合优胜劣汰原则

⑤交叉

采用“顺序交叉算法”或者“自交算法”

1.顺序交叉算法 Order Crossover

产生两个[ 0, 染色体长度 ] 的随机数，设染色体长度为9，随机数为3和6

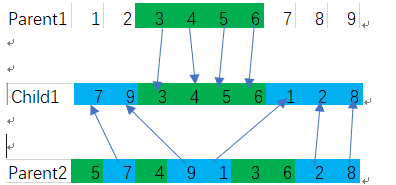
从父代1中“截取”该段染色体的基因序列

https://lh5.googleusercontent.com/8TQHHZUWuvBFjdn3GHgF_YjhLYsE_9oGmnk9J_On4qO5j-yjQZIPsiHx4-rKtXJgGxqDHVrkGmsm4i_68bVXOk1d9jRfPhla3bVC8NJTvbgPpkY3OjX5UnV4vC4gZSguQwdpWSWC.

存到子代1中

https://lh5.googleusercontent.com/2OjuoJVHPCy63lobHPZbPyJPGzfAhTlfRzwGebVnf_Eb7mGQRVgwoymzZYk_WLEvc2orlAbcqrtOC0bR_ed26nZ441DmYuWqd_OgHbN7ISdT9mNDXpIflMvJ-jqYgUuwrYSr2k7b.

然后将父代2中其他基因序列按原顺序存放到子代1中



同理，先截取父代2，再用父代1补齐，可以得到子代2，从而得到两个子代

2.自交算法

产生一个随机节点，交换随机节点前后的基因，即可产生子代

3.交叉产生的问题

在交叉的时候会有3个问题 ：1.什么时候交叉

                     2.交叉个体如何选择

                       3.旧种群和新种群的筛选

1.什么时候交叉

在自然界中动物的交配是存在一定的概率，同样我们设置一个交叉概率CROSSEROVER\_RATE，产生一个随机数，若随机数大于交叉概率（或者是轮盘赌算法的积累概率）进行交叉。

2.交叉个体的选择

Ⅰ.采用轮盘赌算法

Ⅱ.基于两优秀个体产生的后代优秀的概率较大，所以按适应度从高到低两两交叉

Ⅲ.随机选择两个进行交叉

同时我们遵循交叉双方不重复的原则，1和2交配，其他的就不能和1，2交叉了，需要查重。

具体采用哪种算法，我们会在测试中确定。

3.旧种群和新种群的筛选

在新种群和旧种群筛选的时候会产生2个问题：1.在交叉完后是否有重复的染色体,若有需要筛选。（至于是否要查重，会在测试中判定，查重会更好的模拟遗传的行为，但增加了时间复杂度）2.对新种群做适应度计算，和旧种群一起比较，筛选掉一半的染色体，保持种群大小。

⑥变异

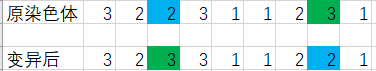
采用“逆转变异算法”和“精英保留算法”

变异在种群中的概率很低，我们设定一个较小的变异概率MUTATION\_RATE(具体数值会在测试中确定)产生随机数，当随机数小于MUTATION\_RATE时产生变异，那么下面解释逆转变异算法和精英保留算法。

1.逆转变异算法

产生两个[ 0, 染色体长度 ] 的随机数，比如染色体长度为9，产生的随机数为3 和 8

逆转（交换）第三个工序和第8个工序



2.精英保留算法

在变异之前的种群中会存在比较优良的个体，对这些个体不进行变异。至少要对最优秀的染色体进行保留，不进行变异。

⑦种群间基因交流

由于遗传算法本身具有的随机性，而进化过程中优秀个体的出现概率低，算法的收敛速度不可避免的受到“运气”的影响。而我们将多种群协同进化引入本遗传算法，一定代数后将优秀个体在种群间传递，大大提高了收敛速度，优化了算法效率。

**教师评语：**