**高级算法设计**

主思路：基于遗传算法的动态调度

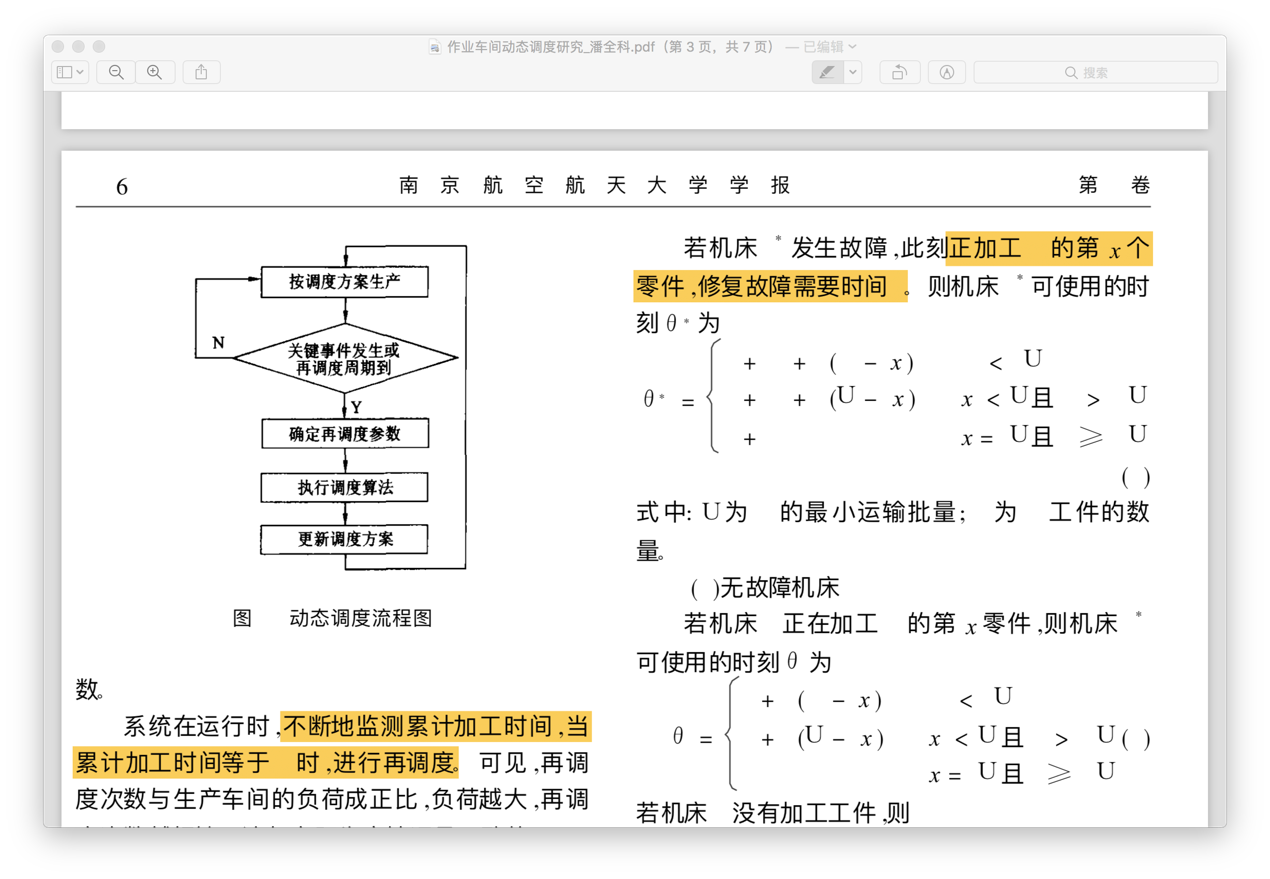
1. 简介：

遗传算法（GA）是一种高度并行、随机和自适应化的算法，他将问题的解决用染色体描述，再通过选择、交换和变异等操作对种群中的个体作用，是种群进化来进行全局优化搜索，特别适用于搜索较大的解空间。

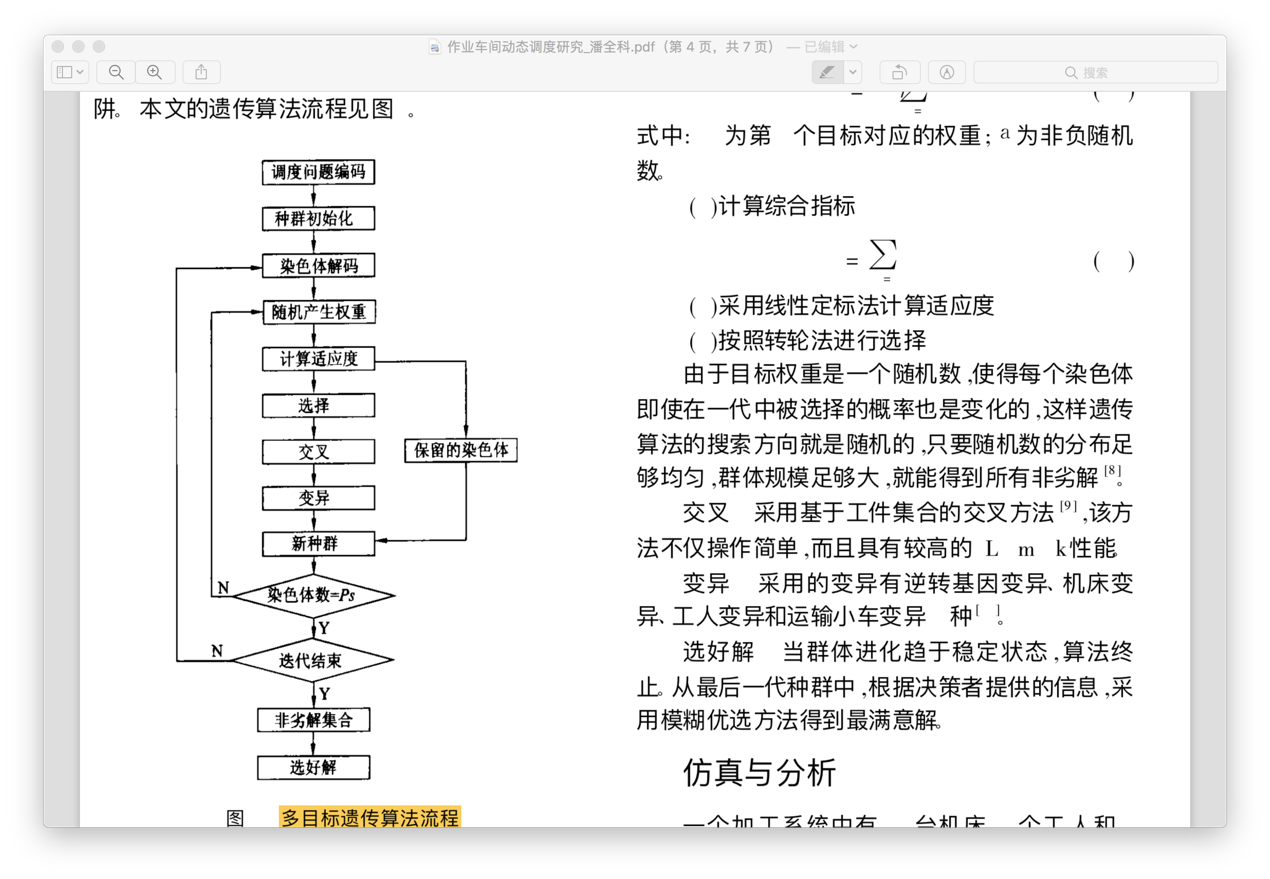
基于事件调度对加工过程进行再调度，以贴合工程中突发事件的实际性。

2.流程框架

考虑检修的整个加工过程：



其中，基础调度方案基于遗传算法框架：



3.主要概念介绍：

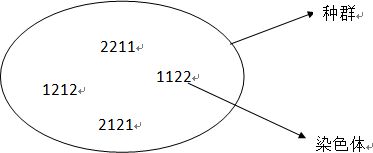
（1）染色体编码

染色体的编码是将车间调度问题转化为遗传算法可处理形态的关键步骤，所谓染色体即是将各加工订单的编号按先后顺序排列后形成的整数串。

（2）种群的创建

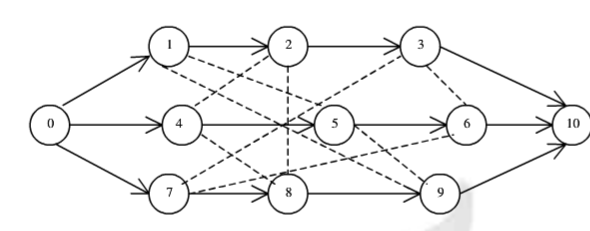
在编码之后随机生成的所有工件号的排列的集合称之为种群。

例如，假设当前有两个工件需要加工，分别用1和2表示，每个工件有两个加工步骤（工件编号出现次数代表该工件的加工步骤数），则下图就表示了该情形下所有染色体排列所形成的种群。



（3）析取图

析取图是描述jobshop的常用工具。对于n个工件、m台机器的问题，对应的析取图G=(V,A,E)。如图，其中V代表所有操作构成的顶点集，包括0和N+1两个虚拟操作（分别表示开始和结束）；A为n\*(m+1)条子边（实线）构成的边集，子边表示某工件按约束条件再所有机器上从开始到结束的加工路径；E为n\*m条子弧（虚线）构成的弧集，子弧表示在同一机器上加工的各操作的连接。



若以最大完成时间为指标，则对jobshop的求解就归结为找到各子弧围成的环（环代表机器）上作为优先决策的各操作的一组顺序，其最大完成时间即为关键路径长度。

（4）染色体的交叉

交叉是遗传算法中的一个重要操作，他的目的是从两条染色体中各自取出一部分来组合成一条新的染色体，交叉操作是遗传算法中决定全局搜索能力的主要因素。交叉操作的思想是保留并充分利用已知的优良模式，使得交叉操作向进化的方向前进。

（5）染色体的变异

变异操作通常发生在交叉操作之后，他的操作对象是交叉得到的新染色体。

例如：变异前的染色体：ABCABCABC

随机选取两个位置：ABCABCABC

变异后的染色体：AACABCBBC

（6）染色体解码

解码操作是利用染色体中的信息形成与其唯一对应的有向无环图，而此情况下的最优解则是从起点到终点中所有路径中总完成时间最长的路径对应的完成时间。

4.主要函数设计：

|  |  |  |
| --- | --- | --- |
| 函数名 | 函数功能 | 变量传入与返回值 |
| int\*\* InitPopulation(\*JOBPTR) | 生成初始种群的编码 | \*JOBPTR（传入存有所有工件信息的数组）；  返回随机生成的染色体组 |
| void Crossover(\*\*int,int,int) | 选取两条染色体进行交叉 | 传入染色体数组，以及需要进行交叉操作的两条染色体编号 |
| void Mutation(\*\*int,int) | 挑选一条染色体进行变异 | 传入染色体数组，以及需要进行变异操作的一条染色体编号 |
| （返回值格式待定）ComputeDAG(\*\*int,int) | 进行染色体的解码，构造析取图 | 传入染色体数组；返回记录节点信息和节点的指向及相对位置的信息 |
| int ComputeTime(\*\*int,int) | 根据析取图计算该染色体编码下的对应时间 | 传入染色体数组和特定的染色体编码，调用ComputeDAG函数计算时间；返回所需时间 |

5.部分框架伪代码

…时间累积

if 时间达到检修时间{

更新检修机器状态参数； //？？？？？？

更新调度方案；

while 时间在检修时间内{

执行调度算法；

}

还原检修机器状态参数； //？？？？？？

}