基于遗传算法的排考系统①

龙 恒[1], 谭彩明[2]

[1](茂名职业技术学院 教育信息与网络中心, 茂名 525000)
[2](茂名职业技术学院 计算机工程系, 茂名 525000)

摘 要: 排考是高校教学管理一项非常重要的工作. 随着高校规模的日渐增大, 参与考试的学生数量和课程数量也成倍增加, 由工作人员手工完成排考工作将非常困难. 提出了一种使用遗传算法实现排考的方法, 通过对变异算法进行优化, 实现算法的快速收敛. 即使对于非常复杂的考试表, 也可以取得很高的成功率和很快的收敛速度.

关键词: 遗传算法; 排考; 变异; 快速收敛; 优化

Examination-Arrangement System Based on Genetic Algorithm

LONG Heng[1], TAN Cai-Ming[2]

[1](Education Information and Network Center, Maoming Polytechnic, Maoming 525000, China)
[2](Department of Computer Engineering, Maoming Polytechnic, Maoming 525000, China)

Abstract: Examination Arrangement is an important work task in higher education management information system. With the increasing number of students, the number of students who participate in the examination and course is also multiplied, the task of manual testing examination arrangement will become very difficult. In this paper, an algorithm using genetic algorithm is proposed. The algorithm achieves fast convergence by optimizing the mutation operation. Even for the complex examination table, it can also obtain a very high success rate and a convergence speed.

Key words: genetic algorithm; arrangement examination; mutation; fast convergence; optimization

排考是高校教务人员工作的难点之一, 随着参加考试的学生人数以及课程数量的增加, 排考人员的工作量及工作难度也直线上升, 例如一张有 500 条记录的考试表, 由一个工作人员使用电子表格软件 excel 进行处理, 至少也需要一天时间才能完成, 另外手工排考也容易出现错误. 排考问题是一种 NP 完全问题, 除了使用穷举法之外无法保证找到最优解, 但是穷举法搜索的空间太大, 无法用来解决复杂的问题. 遗传算法是一种用来解决 NP 完全问题的常用方法, 并且被证明具有良好的效果, 在满足一定的性能要求的情况下能找到较为合理的解. 我们把遗传算法应用到高校排考系统中, 经过试验, 在设定合适的参数的情况下, 在不使用回溯法的情况下, 对实际中常用的考试数据, 可以在较短的时间内得出正确的结果. 即使在处理人为设定的冲突特别严重、数据量特别大的考试表的时候, 也能取得较高的成功率.

1 排考的相关问题

排考问题是根据学生参加考试的情况, 由教务人员安排好相应的场次, 将相应的课程安排到正确的场次中, 然后将各场次安排在不同的时间段进行考试, 要求满足以下三个条件:

① 安排的场次尽量少;

② 同一门课程必须安排在同一个场次进行考试;

③ 每位学生同一个时间段最多只能参加一场考试, 因此两门课程有学号相同的学生参加考试, 这两门课程就不能安排在同一个场次;

根据条件 3, 如果参加两门不同的课程考试的学

① 收稿时间:2013-06-09;收到修改稿时间:2013-07-08

生当中有学号相同的, 就把这两门课程称为冲突课程. 对上面需求的分析可知, 可以把排考问题归结为尽可能将相互之间没有冲突的课程安排到同一个场次之中的问题, 其难点是满足条件 1, 因为如果没有条件 1 的限制, 即使使用手工进行排考, 只要场次足够多, 难度也不会很大.

为了简化算法的实现, 把一门课程作为一个实体, 参加该课程考试的学生看作课程的属性, 在决定将某门课程安排到某个场次中的时候, 只需确定所有参加该课程考试的学生是否已经出现在该场次之中即可.

2 遗传算法

遗传算法原本用于研究自然和人工系统的自适应行为问题, 现在已经被广泛应用在了组合优化、人工智能等方面[1].

遗传算法是一种随机搜索算法, 它借鉴生物学中的进化现象中的一些概念, 通过若干步骤在随机确定的解的集合内进行搜索[2], 虽然不能保证找到最佳答案, 但是大部分情况下都能得到最优解或者近似的最优解, 可以在这些近似最优解的基础上使用一些传统算法, 例如回溯法, 进一步求得最优解.

遗传算法有如下几个基本步骤: 确定初始种群、选择、交叉、变异以及计算适应度[3], 其工作流程如图 1 所示.

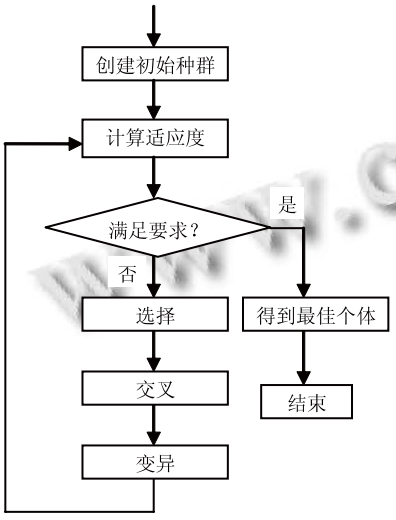

图 1 遗传算法工作流程图

2.1 染色体编码

在本文中, 把一个考试时间安排表作为一条染色体, 每一个场次作为一个基因, 其结构如图 2 所示.

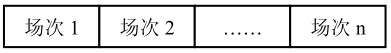

图 2 染色体编码结构图

本文染色体不使用二进制的编码方式, 而是每个基因都使用一个自然数表示, 在实现中一个染色体通常都是保存在一个一维数组中的, 因此可以直接使用数组下标表示基因. 场次的数量一般都是由工作人员指定一个初始值, 这样也就确定了一个染色体中基因的数量.

2.2 初始种群的创建

初始种群的创建是否合理对后面的步骤有较为重要的影响, 如果初始种群的适应度比较高, 后面的进化过程就会收敛得比较快, 在限定的进化代数内就有更大的机会得出最优解.

本文采用的方法是将所有课程随机分配到每个场次中, 经过测试, 如果产生的随机数比较均匀, 初始种群的适应度就比较高, 初始种群的适应度的高低与考试表的大小以及安排的场次数量也有直接的关系, 考试表越大, 在其他条件不变的情况下, 初始种群的适应度越小; 而场次越多, 初始种群的适应度普遍都比较大.

2.3 适应度

适应度是判断结果是否已经达到要求的依据, 在本文中适应度的计算方式是用各场次中没有冲突的课程数量的和除以课程总数, 得到的商就是适应度, 如果适应度等于 1 就表示所有场次的课程都已经没有冲突了, 也就是找到了最优解.

遗传算法是一种随机搜索算法, 每次运算的结果都有可能不同, 而且不能保证找到最优解[4], 因此在实际运算过程中为了避免出现死循环, 并不以适应度是否为 1 作为结束条件, 而是使用限定进化的代数、运算的时间或者适应度在一段时间内没有变化作为结束的条件, 然后对得到的最优个体, 也就是考试时间安排表, 使用传统的回溯法或者手工调整来得到正确的结果.

2.4 选择

选择运算用于模拟生物界的优胜劣汰的过程, 根据一定的规则从当前的种群中选择一些优秀的个体进行繁殖, 使得其优秀的基因得以遗传到下一代, 从而

得到适应度更高的新个体.

本文中使用轮盘赌的方法从种群中选出 x 对个体, 然后分别对每对个体进行交叉和变异运算. 由于轮盘中的每个区域是根据个体的适应度的大小按比例进行分配的, 因此轮盘赌算法从理论上可以使得适应度高的个体有更大的机会被选中.

另外还必须从当前种群中淘汰 y 个适应度差的个体, 使得种群得以引入新的基因, 本文采取的方法是将当前种群的所有个体按适应度的大小降序排序并保存在一个链表中, 每轮运算将生成的 y 个新个体, 替换掉种群中适应度最差的同等数量的个体. 每轮运算后, 那些进行了交叉和变异运算的个体、新生成的个体以及那些被保留的个体就构成了新一代的种群.

2.5 交叉

交叉运算借用的是生物在繁殖的时候两个同源染色体的交叉重组概念, 从两条染色体中分别截取一些基因片段组合成一条新的染色体.

在本文中, 交叉运算是通过定义若干个交叉点, 在运算的时候同时扫描被选择进行交叉的两个染色体, 在到达交叉点位置的时候, 轮流将两个染色体的当前交叉点与上一个交叉点位置之间的基因复制到新的染色体当中, 图 3 是一个定义了 3 个交叉点的例子.

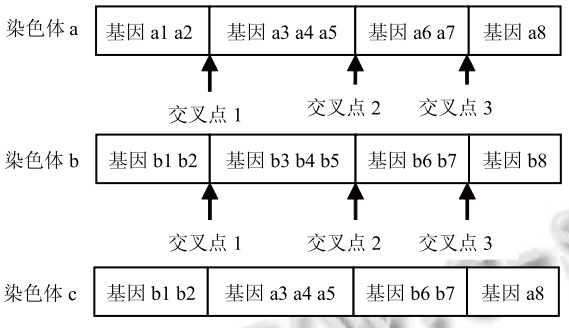

图 3　定义了 3 个交叉点的例子

图 3 显示了通过交叉运算从染色体 a 和染色体 b 生成染色体 c 的结果, 在软件的实现当中, 一个场次被当成一个基因, 因此交叉运算实际上执行的是将进行交叉的考试时间安排表的场次中的课程放到新表中相同场次中. 在实现当中, 每个考试时间安排表都保存了一个按某个关键字进行排序的课程表, 并且每个考试时间安排表中的这个课程表的课程顺序都是相同的, 每次交叉都相当于从同一个课程表中取出课程, 因此不会出现将相同的课程排入新的考试时间安排表的情况.

为了增加随机性, 每次交叉运算可以通过某些随机的算法确定首先从哪条染色体中取出基因, 图 3 的例子就是首先从第二条染色体中取出基因.

2.6 变异

变异是借用了染色体进行复制的时候发生差错的概念, 这种变异有可能会产生积极的影响. 变异运算是在交叉之后进行的, 在本文中, 变异运算是通过将一门课程从一个场次调整到另一个场次中来实现的.

尽管生物界的变异的概率通常是很小的, 但是对于本文的应用, 经过测试后发现加大变异的概率可以大幅提高算法的收敛速度, 特别是数据量较大的考试表, 在将变异概率调整到 80%以上能取得很好的效果.

在编写变异算法的时候曾经使用过两种方法, 第一种是每次变异时, 在进行变异的染色体中随机选中一个场次, 在这个场次中随机选择一门课程, 将这门课程放入另一个随机选中的场次当中.

第二种方式是, 为每个场次创建一张冲突表, 当将课程加入某个场次的时候, 如果该课程与该场次中的课程有冲突, 就将相互冲突的每门课程都加入冲突表中, 另外每次课程从该场次中移出的时候都要更新一次冲突表. 每次变异运算的时候随机选择一个课程存在冲突的场次, 在冲突表当中取出一门课程移动到另外的场次中.

经过测试, 使用第二种方法对提高算法的收敛性也有非常明显的作用, 特别是在处理数据量较大的考试表的时候, 第一种方法基本上无法得到适应度为 1 的个体, 对这一点本文将在第 3 节的测试中给出测试数据.

2.7 参数

在实现中为算法设定了几个参数, 除了在选择运算中提到的参数 x、y 外, 还将种群大小 p、进化的最大代数 g、交叉的概率 c、变异的概率 m 也做为参数.

另外还有一个重要的参数就是排考的场次数 n, 最理想的场次数是参加考试课程最多的学生所考的课程数 N, 经过手工排考验证, 并不是每张考试表都能排出 N 场, 真实的场数将会从 N 开始进行检测, 逐次增加 1, 直到排出正确的场次为止.

每次排考时, 根据考试表的实际情况而使用不同的参数, 对提高算法的性能以及收敛性有重要的作用. 在实际应用中发现, 变异概率 m 的大小对提高算法的收敛性有重要的影响; 而进化的的代数可以控制算法

的总体运行时间.

3 测试结果

在测试中主要针对有重要影响的算法和参数进行验证, 需要进行两方面的测试, 一是在变异算法中是否使用冲突表, 二是变异的概率对算法的影响. 为此选择了我校教务处使用的两张考试表进行测试, 这两张考试表的信息如表 1 所示.

表 1 考试表信息

表名	记录数	课程数	学生数
T1	477 条	106 门	331 个
T2	1997 条	197 门	1320 个

选定的测试环境是一台处理器类型为 Pentium4 , 主频为 3.0G 赫兹, 内存为 2GB 字节的计算机.

首先测试的是在变异运算的时候不使用冲突表的情况, 共进行三轮测试, 每轮运算 100 次, T1 表的各轮参数如表 2 所示, T2 表的各轮参数如表 3 所示.

表 2 测试 T1 表的参数

轮数	p	x	y	g	c	m	n
1	50	8	5	8000	80%	10%	7
2	50	8	5	8000	80%	60%	7
3	50	8	5	8000	80%	100%	7

表 3 测试 T2 表的参数

轮数	p	x	y	g	c	m	n
1	50	8	5	8000	80%	10%	9
2	50	8	5	8000	80%	60%	9
3	50	8	5	8000	80%	100%	9

使用 T1 表进行测试的各轮的结果如表 4 所示, 从结果可以看到, 这次测试只取得了非常低的成功率, 而使用 T2 表进行测试则一次都没有成功.

表 4 不使用冲突表的 T1 表测试结果

	第 1 轮	第 2 轮	第 3 轮
成功率	4%	1%	0%
平均进化代数	2337 代	610 代	x
平均花费时间	120 秒	6 秒	x
最少花费时间	3 秒	6 秒	x
最多花费时间	83 秒	6 秒	x

接着测试在变异算法中使用冲突表的情况, 并且检验变异概率的不同对结果的影响. 一共进行三轮测试, 每轮运算 100 次, 依然使用表 2 和表 3 的参数, T1 表的测试结果如表 5 所示, T2 表的测试结果如表 6 所示.

表 5 使用冲突表的 T1 表测试结果

	第 1 轮	第 2 轮	第 3 轮
成功率	97%	100%	100%
平均进化代数	428 代	143 代	111
平均花费时间	5.16 秒	1.64 秒	1.18 秒
最少花费时间	1 秒	1 秒	小于 1 秒
最多花费时间	15 秒	3 秒	3 秒

表 6 使用冲突表的 T2 表测试结果

	第 1 轮	第 2 轮	第 3 轮
成功率	20%	90%	99%
平均进化代数	3083 代	767 代	599
平均花费时间	89 秒	18.5 秒	12.76 秒
最少花费时间	28 秒	7 秒	6 秒
最多花费时间	223 秒	68 秒	53 秒

从测试结果可以得知, 在变异运算是否使用冲突表对算法会有决定性的影响, 不使用冲突表的算法只在考试表数据量特别小的情况下才有效. 另外从表 5 和表 6 可以得知, 在其他条件相同的情况下, 变异概率的提高会显著提高算法的成功率和收敛速度.

4 结束语

笔者在使用遗传算法实现排考系统之前, 一直是使用回溯法解决排考问题, 但是回溯法经常无法排出最优的场次. 使用遗传算法之后, 排出的场次大部分都优于回溯法, 特别是在考试表数据比较多的时候, 遗传算法的优势更为明显. 另外由于遗传算法的随机性, 不能保证每次都能获得最优解, 但是排考人员只要在软件中设定重复运算若干次, 在绝大部分情况下都能得到满意的结果.

本算法可以很好地解决教务人员的排考问题, 具有收敛速度快, 成功率高的优点, 是一个较为实用的算法.

参考文献

1 王小平,曹立明.遗传算法—理论、应用与软件实现.西安:西安交通大学出版社,2002.3–15.
2 张文修,梁怡.遗传算法的数学基础.西安:西安交通大学出版社,2000.2–30.
3 玄光男,程润伟.遗传算法与工程设计.北京:科学出版社,2000.1–3.
4 Z.米凯利维茨.演化程序—遗传算法和数据编码的结合.北京:科学出版社,2000.11–14.