基于优先级链表结构的大学排课算法设计与实现*

蒙焕念，黄良永
（柳州师范高等专科学校 教务处，广西 柳州 545004）

摘 要：针对计算机解决大学课程表问题的难点，提出使用优先级链表解决课表问题的贪心策略。该策略定义了特有的数据优先级权重，并以权重为基础生成排课数据的优先级链表，以优化设计编码，实现了一种基于链表操作的贪心排课算法。

关键词：大学课程表；链表；贪心算法

中图分类号：TP301 **文献标识码**：A **文章编号**：1674-7720(2012)21-0014-03

The design and realization of the university course scheduling algorithm based on the priority structure of linked list

Meng Huannian, Huang Liangyong
(Dean's Office of Liuzhou Teachers College, Liuzhou 545004, China)

Abstract: This paper talks about the difficulties of the university time table problems, and proposes using a priority list of the greedy strategy to solve the schedule problems. In this strategy, a specific priority weight of data is defined, and has generated the priority linked list structure data of university course scheduling based on priority weight, in that case the code has been optimized and a course scheduling by means of the greedy algorithm based on linked list operations has been realized.

Key words: university course scheduling; linked list; greedy algorithm

大学课程表问题是一种时间表问题TTP（Time-Table Problem），是公认的一种NP（Non deterministic Polynomial）困难问题[1]。几十年来，人们对计算机排课方法做了很多尝试，但实现的效果并不是很理想[2]。研究表明，解决大规模的课表编排问题仅靠数学方法是行不通的[3]。排课问题是一个多目标条件组合的优化选择问题，需要实现课程、学生、老师、上课时间和地点的优化组合，加上大学排课约束因素的多样性，如教师的特殊要求、课程的特殊要求和学生的特殊要求等，致使排课工作复杂而繁重。

目前，用于解决课程表问题的算法很多，但各种算法同时也存在自身的不足。如遗传算法复杂，实现难度大；蚁群算法计算时间长，容易出现停滞现象；模拟退火算法控制参数选择困难，易发生算法迭代不收敛的情况[4]。通过对多种算法的优劣性分析，本文基于贪心算法的思想，以特定的优先级链表为主要数据结构，提出并实现了一种改进的排课算法。

*基金项目：柳州师范高等专科学校科研基金项目（No. LSZ2008B005）；广西教育厅科研面上项目（200911MS294）。

1 解决问题的策略

贪心算法也称为“贪心策略”，它对解空间进行搜索时，不是从整体上考虑最优，其所做出的选择只是在某种意义上的局部最优解。换句话说，贪心算法进行的每一次贪心选择，都是对当前条件做出的最佳选择。虽然这种局部最优选择并不总能获得整体的最优解，但是通常能获得近似的最优解。由于算法考虑的信息量较小，一旦选择后就不需再改变（不回溯），所以算法效率比较高。

由于大学课程的专业性强，科目多，通常情况下，课程的开课班级和任课老师只能通过人工指定。所以在大学课程表问题中，计算机排课要处理的关键问题就是安排课程的上课时间和教室，实现上课时间和教室的冲突检测和智能选择。因此，本设计的排课采取的贪心策略为：先为所有的课程、上课时间和教室指定排课优先级[5]，然后从优先级最高的课程开始排课，在当前的条件约束下，选择符合条件的、优先级最高的

上课时间和教室。

2 数据结构与算法实现

2.1 数据结构

2.1.1 待排课程信息(Course)

一般来说,可以从各专业的教学计划自动生成原始课程信息,然后人工进行合班、指定任课教师,就可以得到待排课程信息,其数据结构可定义如下:

```
struct Course
{
char CourseID;                    //课程教学任务号,唯一值
char TeacherID;                   //教师工号
char ClassWeek;                   //上课周,格式如1~18
int Period;                       //总学时
int PeriodPerWeek;                //周学时
char Term;                        //学期
int Number;                       //人数上限(合班或选课人数)
char RoomType;                    //请求教室类型
int Priority;                     //排课优先级权值
char ClassInRow                   //连排节次,如2节,5节
char FinishFlag;                  //排课完成标识
struct Course*Next;               //下一记录指针
}
```

课程的排课优先级可根据学校的具体情况调整,优先级权值大的课程优先排课。优先级权值由4项数据决定,每一项数据权值系数可能取值为0~9,总权值表达式为:课程性质权值系数$\times10^3$+教室类型权值系数$\times10^2$+人数容量权值系数$\times10$+学分权值系数。我校的课程性质和教室类型优先级权值系数设置如表1、表2所示,人数容量系数=人数/40取整,如果人数大于360,则系数取最高权值9;学分权值系数用课程本学期学分直接表示(本学期学分不能大于9)。这样的优先级设计保证了公共必修课必须先排课,然后优先考虑紧缺教室类型的安排和大班教学排课的情况,如果以上三个条件计算出多个课程的优先级权值都一样,则由课程的学分决定最终的课程优先级。

表1 课程性质权值设置表

公共必修	公共任选	专业必修	专业方向	专业任选	其他
9	5	8	6	7	4

表2 教室类型权值设置表

机房	语音室	实验室	多媒体教室	普通教室	其他
9	9	9	8	7	6

2.1.2 课程的上课班级(ClassInCourse)

该数据是对待排课程进行合班操作后生成的附属数据,用于指明一个课程的上课学生班级。把该数据与待排课程信息(Course)分开是考虑到当一个课程对应多个上课班级时,使关系数据仍能满足第三范式约束。ClassInCourse数据结构定义如下:

```
struct ClassInCourse
{
char CourseID;                    //课程教学任务号
char ClassID;                     //班级代码
struct ClassInCourse*Next;
}
```

2.1.3 上课时间(ClassTime)

ClassTime数据结构定义如下:

```
struct ClassTime
{
char ClassTimeID;                 //时间编码
char ChineseCharacter;            //时间中文含义
int Priority;                     //排课优先级
int UsedTimes;                    //排课使用次数
char IsUsed;                      //是否排课标志
int Period;                       //课时数,即该时间段包含小节数
struct ClassTime*Next;
}
```

为了获得时间遍历的高效性,可对时间编码进行技巧性设计。若每天有5个上课时段,可用"10"表示"星期一第一大节(1、2节)"、"11"表示"星期一第二大节(3、4节)"、"13"表示星期一第三大节(5、6节),如此类推到其他时段。这样,一周共35个时段,可定义数组$a[45]$、$a[i]$表示一个上课时段,则遍历一周的时间只需要一个循环:

```
For(i=10;i<=44;i++)
{
printf(GetChineseCharacter(a[i]));
                                  //使用编码获取时间的中文含义
}
```

数组之所以不使用a[1]~a[9],是因为1~9不好作相似运算(相似运算不能区分1和11)。对于各时间段的排课优先级,通常设上午时段和下午5、6节的优先级最高,下午7、8节次之,最后是晚上,星期六和星期天不能排课,IsUsed设为False。为了保证上课时间均匀分布,本设计用UsedTimes记录各时段被使用的次数(初始为零,每使用一次即加1),对于优先级相同的时间段,按时间使用次数升序排列。排课选择时间时,优先选择使用次数少的时间。这样,通常只需检测序列前面的几个时间段,就可以命中合适的时间对象,大大提高了选择时间的命中率。

2.1.4 排课地点(ClassRoom)

排课地点数据结构定义如下:

```
struct ClassRoom
{
char RoomID;                      //教室编号
char RoomType;                    //教室类型
char RoomName;                    //教室名称
```

```
int SeatingCapacity;                    //座位数
int Priority;                           //排课优先级
int UsedTimes                           //使用次数
char IsUsed;                            //是否排课标志
struct ClassRoom*Next;
}
```

为了避免出现大教室被小教学班占用的资源浪费现象，设计时按教室容量和教室类型分段划分教室的使用优先级，如表3所示。

表3 教室使用优先级安排表

人数区间	[1,50)	[50,80)	[80,120)	[120,∞)
普通教室	9	8	7	6
多媒体	5	4	3	2

与时间一样，设计时记录了每一个教室的使用次数(UsedTimes)，在教室的优先级相同时，优先选择使用次数少的教室，从而使教室资源得到有效利用，同时也大大提高了选择教室的命中率。

2.1.5 排课条件约束(Constraints)

排课条件约束是自动排课非常关键的数据结构，因为条件约束很大程度上决定了自动排课的成败和课表的科学合理性。常见的个性化排课约束条件有：(1)某个老师特定时间(或地点)排课/不排课；(2)某个课程特定时间(或地点)排课/不排课；(3)某个班级特定时间(或地点)排课/不排课；(4)课程的多个上课时间应有合理的时间间隔(1天以上)，而上课地点应尽可能相同。

可以看出，条件约束基本可以分为对课程、教师和学生的三种约束类型，约束项目主要有上课时间和上课教室(或教室类型)，其取值格式如表4所示。

表4 排课约束格式

约束类型	约束对象	约束项	逻辑符号	取值
课程	大学英语	上课时间	不等于	10
教师	李四	上课时间	不等于	11
学生	数学101班	上课教室	等于	1102

表4中，设“大学英语”、“李四”和“数学101班”分别是约束类型课程表、教师表和学生(班级)表中的唯一主键值，约束对象必须是约束类型表(待排课程信息，教师表和学生表)的一个主键值。为了简化程序，逻辑符号只有等于和不等于两种取值。表中第一行的含义为：安排课程“大学英语”时，上课时间不能选择“10(星期一第一大节)”。

除了以上自定义的约束外，排课还应该遵循一些基本的常理性约束，如：(1)每个学生在同一时间只能上一门课；(2)每个教室同一时间只能上一门课程；(3)每个教师在同一时间只能在一个地点上课；(4)每个课程任务同一时间只能在一个地点上课；(5)每门课程安排的课时应等于计划的总课时等。

2.1.6 课程安排表(TimeTable)

课程安排表数据结构定义如下：

```
struct TimeTable
{
char CourseID;
char RoomID;                            //教室编号
char TimeID;                            //时间编码
char WeekFlag;                          //上课周标识
int Period;                             //不完全占用大节时标注学时数
struct TimeTable*Next;
}
```

2.2 算法描述

根据以上的数据结构来实现排课的贪心算法。算法步骤描述如下：

(1)初始化。按优先级从高到低生成待排课程链表(*Course)、时间链表(*ClassTime)和教室链表(*ClassRoom)；新建课程安排空链表(*TimeTable)。

(2)遍历课程链表。对应遍历的每一个课程，做以下3个操作：①为当前课程选择符合条件的、优先级最高且使用次数最少的上课时间(无论是否找到合适的时间，限定每个课程对时间链表的匹配次数不超过10次)。如果找到符合条件的课程，则把课程和时间加入TimeTable中，时间段使用次数加1；否则，说明该课程无法安排，跳转到下一门课程，重新执行3个操作；②对当前课程排定的时间，选择符合条件的、优先级最高且使用次数最少的教室(无论是否找到合适的教室，限定每个课程对教室链表的匹配次数不超过10次)。如果找到符合条件的教室，则把教室派给当前课程，教室使用次数加1；否则，说明该课程没有教室可排，跳转到下一门课程，重新执行3个操作；③如果当前课程已排时间节次小于课程的周课时，时间指针跳转到下一天时间段(保证同一门课上课时间间隔一天以上)，继续对当前课程执行3个操作；否则，说明该课程已经安排完毕，调整时间和教室链表排序，跳转到下一门课程(Course=Course->Next)，重新执行3个操作。

(3)课程链表遍历完成，返回*TimeTable，算法结束。

3 算法复杂度分析及使用效果

3.1 复杂度分析

本算法为每个课程的上课时间和教室进行了贪心选择，核心是使用了三层循环嵌套。通常情况下，三层循环嵌套算法的时间复杂度为$O(n^3)$。由于算法根据预先设定的优先级规则生成时间链表、教室链表，对时间和教室采取先排序，后选择的贪心策略，使算法总能在链表序列前部命中对象，保证循环每次命中目标所遍历链表节点的个数不大于10(如果超过10次未命中则跳出循环)。因此，算法的时间复杂度接近$O(n)$。

3.2 使用效果

本算法在安排课程的上课时间和教室时，总是选择优先级高且利用率低的时间和教室对象，保证了资源利

用的均衡化。但如果学校的资源确实紧缺或排课约束条件苛刻，则必须调整时间和教室的优先级规则，更改排课约束条件，再进行排课。如仍找不到合适的时间和教室排课，则需要人工调整排课。

本文使用 Visual C++6.0 和 SQL Server2000 实现本算法，对我校 42 个课程、12 个班级、36 个教室进行排课，每周排课总次数为 142，在 P4 双核 2.8、内存 1 GB 的机器上运行，所有课程都成功排课，共耗时 3.505 s。此外，还对我校 2010~2011 年第二学年的 1056 个课程、130 个班级、305 个教室级进行完全编排，整个过程耗时 60.121 s，比我校正在使用的某公司教务管理系统的排课耗时少了 15.826 s。实践证明，本文排课算法能够较好地解决大学排课问题，是一个实用而高效的排课算法。

参考文献

[1] 熊焱，王莉，李大卫，等.大学课程表问题的模型与算法[J].鞍山科技大学学报，2005，28(1)：26-29.

[2] 高喜玛，张萍.大学自动排课系统内核算法设计[J].南阳师范学院学报(自然科学版)，2003，2(12)：55-58.

[3] 潘以锋.高校智能排课系统的算法[J].上海师范大学学报(自然科学版)，2006，35(5)：31-37.

[4] 张晶，李广军，徐娟.智能排课算法综述[J].西南民族大学学报(自然科学版)，2009，35(3)：675-678.

[5] 冯思玲，李艳梅，梁瑜.基于分治和贪心相结合的排课算法研究[J].现代计算机(专业版)，2009(3)：11-13.

(收稿日期：2012-07-09)

作者简介：

蒙焕念，男，1981 年生，高级工程师，主要研究方向：计算机软件与数据工程，数据挖掘。

黄良永，男，1971 年生，硕士，副教授，主要研究方向：信息系统，计算机图形学。