# 第五章详细设计

本章主要的内容是介绍系统的详细设计。介绍了本监控系统关键模块的详细设计。

## 智能排课模块详细设计

智能排课模块是本排课系统的核心模块，主要包括排课参数设置、排课数据输入、课表计算以及课表生成四个功能。下面就对应该模块的相关设计进行详细的描述。

### 关键类

在智能排课模块中只要包含排课的关键类AutoScheduler、保存个体信息的类IndividualInfo、工具类Utils、以及与排课参数管理相关的类BaseFitnessManager、PFManager、PCFManager、DCFManager、ASCManager。其类图如下：



图5-3 智能排课模块中的关键类

下面对每一个类的功能进行描述：

* BaseFitnessManager：排课相关优度参数的管理基类，定义了优度参数操作的相关接口，同时也实现了所有优度管理的一些通用接口。主要三个子类：PFManager、PCFManager、DCFManager。
* PFManager：继承了BaseFitnessManager，负责管理一个星期每一个节次的优度值。
* PCFManager：继承了BaseFitnessManager，负责管理每一天中节次组合的优度值。
* DCFManager：继承了BaseFitnessManager，负责管理一个星期中不同星期的日组合优度值。
* ASCManager：负责排课其他参数的管理，包括节次优度因子、时间分布优度因子、节次组合优度因子、课室利用率优度因子、变异因子、交叉因子、日组合优度因子、空间分布优度因子、种群大小、最大进化代数的管理。
* Utils：定义了排课流程操作相关的函数，包括适应度函数、交叉算子、变异算子以及课表生成函数。
* IndividualInfo：定义了个体的相关信息，包括个体中课程的范围和实验室的范围，该信息主要是用于个体对应的三维数组的遍历、交叉、变异等操作。
* AutoScheduler：智能排课的核心类，负责整个排课的流程。由于排课过程需要占用的资源比较多，为了不影响整个系统的运行，该类使用了单例的设计模式以保证同一时间只有一个实例在运行。

### 关键方法

智能排课模块是最核心也是最复杂的模块，下面对它的关键方法进行介绍：

* 创建个体的函数utils.create\_individual

实现流程：

1. 调用get\_empty\_individual方法创建一个空的个体individual（全部值为0的三维数组）；
2. 选择个体中的一个课程；
3. 设定一个空的列表chosen来保存课程选择的节次；
4. 根据课程的课时选择连续的节次并保存到chosen；
5. 如果上一次的安排已经选择了实验室，那么判断该实验室是否可用，如果可以就使用该实验室；否则，随机选择一个容量大于课程人数的课室；
6. 根据选择的节次和实验室组合成一个课程时间安排记录；
7. 按照3、4、5、6步骤完成课程的安排，得到一个课程安排的时间表schedule；
8. 根据该schedule将个体individual中对应的点置为1，完成第一个课程的安排；
9. 选择下一个课程，跳到第2步，直到所有课程遍历完位置
10. 个体初始化完成，返回初始化后的个体。



图5-4 create\_individual活动图

* 适应度函数utils.get\_fitness

适应度函数是最复杂的函数，整个过程涉及到个体中每个课程的日组合优度、节次优度、节次组合优度等的计算，同时还要进行个体的冲突检测，因此，该函数运算的时间也比较长。具体实现的活动图如图5-5。



图5-5 get\_fitness活动图

具体的实现流程是：

* + 1. 进行个体的冲突检测，包括实验室资源冲突检测、教师资源冲突检测、上课学生冲突检测以及云资源冲突的检测；
    2. 如果存在冲突，那么个体的适应度为0；否则进入下一步；
    3. 计算个体中课程的节次优度的平均值；
    4. 计算个体中课程的节次组合优度的平均值；
    5. 计算个体中课程的日组合优度的平均值；
    6. 计算个体中课程实验室利用率优度的平均值；
    7. 计算个体中课程空间利用率优度的平均值；
    8. 计算个体的时间分布优度值;
    9. 根据适应度计算公式，计算出个体的适应度。

## 课表调整模块详细设计

### 关键类

课表调整模块使得管理员可以随时对课程安排进行修改，并及时通知相关的授课老师，该模块中最大的难度是课表调整时冲突的检测以及调整课程中的管理员与系统的UI交互。该模块主要涉及到的类如下图。



图5-6课表调整关键类

下面对每一个类的功能进行介绍：

* Classroom：用于管理实验室数据。
* CourseTime：用来保存课程表，每次上课的时间都应该在这有对应的一条记录，每天记录应该包括上课都具体时间和地点。
* Course：用于管理课程数据。
* CourseSchedule：用于管理每天的节次安排数据，包括一天有多少节次，每个节次的开始时间和结束时间。
* Views：请求处理类，所有的请求都由该类来处理。
* Utils：工具类，定义了相关的业务逻辑方法，为Views服务的类。

### 关键方法

* Views.update\_coursetime

该函数的主要功能是处理用户修改课表请求。具体实现过程如下。



图5-7 update\_coursetime活动图

1. 用户提交修改课表请求；
2. 接收到请求后判断用户提交数据的方式是否为POST，如果是则进入下一步，否则，直接返回操作失败；
3. 获取用户的操作类型operation和数据data
4. 如果operation = 0，调用Utils.change\_coursetime(data)执行修改操作
5. 如果operation = 1，调用Utils.delete\_coursetime(data)执行删除操作
6. 如果operation = 2，调用Utils.add\_coursetime(data)执行添加操作
7. 如果operation是其他值，则操作失败
8. 返回操作结果

* Utils.check\_conflict

该函数用于检测新加的时间安排记录是否会造成课程表冲突，它的输入参数是新增的课程时间安排记录new\_coursetime。请参考图5-8。



图5-8 check\_conflict活动图

具体实现过程如下：

1. 设置p = new\_coursetime.start\_period；
2. 根据new\_coursetime的course\_date和p搜索出所有在该范围的课程安排记录coursetime\_list；
3. 判断coursettime\_list中是否有实验室资源冲突、授课老师冲突、上课学生冲突、云资源使用冲突，如果存在，则返回相应的冲突类型，检测结束；
4. 将p的值加1，重复执行第2、3步，直到p大于new\_coursetime.end\_period；

返回没有冲突。

## 调课申请模块详细设计

调课申请模块是一个优秀的排课系统不可缺少的模块，授课老师可以通过该模块提供的服务在线进行课程调整的申请，同时申请的审批也是在线进行的。在申请审批的过程中，系统能够根据授课老师提供的可选调整时间区间智能地计算出该区间最佳的调整时间和可用的调整时间，大大方便了调课审批的工作。

### 关键类

该模块主要涉及到的关键类如下图5-9：



图5-9调课申请关键类

* ScheduleAdjustment：是用于管理调课申请记录的，每一条申请记录有包含调课对应的课程、需要调整的时间、可调整到的时间区间、申请提交记录、申请状态以及调整结果。
* Views：负责业务请求处理，每个与调课申请相关的请求都由该模块进行处理。
* Utils：提供工具类函数，主要为Views服务。
* AutoScheduler.Utils：提供个体适应度计算接口get\_fitness。

### 关键方法

* Utils.do\_get\_usable\_time

该函数的作用是在申请审批的过程中，根据申请的可调整时间区间计算出可用的调整时间和最佳的调整时间。具体的实现流程如图5-10。



图5-10 do\_get\_usable\_time活动图

1. 根据申请的可调整时间区间计算出它所跨越的周数，并把每一周的开始时间和结束时间保存起来，保存为weeks；
2. 初始化最近调整时间best\_time = None，最佳调整时间的适应度best\_fitness=0，可用调整时间usable\_time = []；
3. 选择weeks中的第一周，并调用智能排课模块中的get\_empty\_individual初始化一个空的个体（三维数组）individual；
4. 搜索出这周所有的课程安排，并将individual中对应的坐标置为1；
5. 根据申请的可调整时间区间计算出在本周中的可选调整区间；
6. 在上一步计算出的时间区间的开始点选择对应的时间安排，然后将个体individual对应的坐标置为1，并计算当前的个体的适应度f，如果适应度大于0，则说明选择的时间可用，加入到usable\_time中。如果f大于best\_fitness，那么best\_fitness = f，且best\_time等于这次选择的时间。依次对该区间进行遍历，直到整个区间遍历完为止；
7. 选择weeks的下一周，重复4、5、6的操作，直到weeks遍历完成；
8. 输出best\_time和usable\_time。

### 关键过程