成绩：

指导教师：

基于着色理论的排课问题研究

软件工程专业　2020414327　林德松

指导教师 胡春美　　依托课程 数据结构课程设计

摘要：在高校各项教学管理工作中，排课一直是最基本的、最重要的工作，其实质就是给教学计划中设置的课程安排合适的时间和地点，保证整个教学工作能够顺利地进行;同时，排课工作也是一项很复杂的工作，排课是一个 NP 完全问题，就是始终找不到一个最优的方法能够解决的问题，因为这个问题涉及了多种因素进行组合规划，有教师、学生的因素，也有教室的因素，尤其在目前各高校规模不断扩大，教学资源面临紧张，教师总数不足的前提下，排课工作问题更为凸出。

关键词：图论 着色　动态规划法　边着色　排序算法

引言 随着计算机科学技术的不断发展，各个行业信息化、科学化不断推进。高校如何才能提高办学的效率，这是每个高校都会面临的，也是每个高校需要迫切解决的问题。而采用信息化手段来代替传统的教学管理模式是一个重要的途径。

1．绪论

1.1 研究背景和意义

排课问题是 S.Even 在 1975 年证明了的 NP 难问题,在排课过程中需要综合考虑很多要素,比如年级、课程种类、教室容量、教师类型、上课时间、教师等等,以实现教学资源的合理规划。因此,如何对高校课程进行优化排列,提高学校教师、

教室以及配套教学资源的利用率是各个学校追求的目标。排课问题涉及到教学资源的配置问题,更关系到各学校教学目标能否实现,已经成为国内外众多学者关注的焦点。高校排课问题属于时间表问题,该问题的解决不仅可以有效处理高校扩招、学分制推行等改革过程中遇到的资源紧缺问题,还能对其他时间表类问题给予一定的借鉴意义,如教育系统的考试安排问题、行政或商业部门的会议安排问题、交通部门的车辆时刻安排问题等。课程表是编排课程的表。科学的课程安排才能调度全校师生的教学活动,使学校的教学工作有序进行。所以,高校相关机构和人员需要对课程进行合理编排。早先,高校是通过人工的方式进行课程编排,不仅浪费了大量的人力和精力,还可能会由于某种情况考虑不周出现排课冲突。

1.2 国内外研究现状

1.2.1 国外研究现状

排课问题是一个多目标有限资源、带有约束条件的组合规划问题。排课系统已经成为国内外众多高校及软件公司的研究课题，取得了许多方面的理论。国外针对排课问题展开的研究较早。1963 年 Gotlieb 在他的文章《The Construction of Class-Teacher Time-Tables》中提出了课表编排的数学模型。虽然之后人们对课表问题的算法等问题做了很多的探索，但是大多是都是在 Gotlieb 提出的数学模型的基础上简化或者是补充。1976 年，Bondy 用图的边染色算法来解排课表问题。排课问题被 S.Even 等人证明具有 NP 难度，并被理论化。1976 年 S.Even 在其论文《The Complexity of Timetable And Multi Commodity Flow Problem 》中，第一次证明了课表问题是 NP 完全的。S.Even 的论证进一步地将人们对课表问题复杂性的认识提高到理论高度。

1.2.2 国内研究现状

国内对排课问题的研究较晚，开始于 20 世纪 80 年代初期，1984 年，清华大学在《清华大学学报》上发表了林漳希和林尧瑞在该课题上的实验性研究成果《人工智能技术在课表编排中的应用》。伴随着计算机普及、教务管理信息化的脚步,许多的学校也进行了一些排课自动生成软件的研究,相应的教务管理软件如雨后春笋般地涌出,成型的系统早期的有南京工学院的 UTSS (A University Timetable Scheduling System).清华大学的 TISER (TimerableSchedu1ER),

大连理工大学的智能教学组织管理与课程调度系统以及大连理工大学的课程调度系统等。总体来说，这些排课系统可分为两类，一类适合投课时同相同，单班教学的学校情况，显然，这与很多高校的实际教学不一致，所以不具有大众性。一类适合以“教学班”的形式投课，这一类总体比第一类而言，结合了图论中的调度问题，且多与图论中染色问题有关。第二类排课系统的出现，在当时影响很大，虽然有些没有连具体的模型都没有，但是确实与很多学校的实际情况更接近了一步。其次，各种不同算法的数学模型也相应而生，黄干平等提出使用模拟退火算法求解课表间题，采用该方法对中学排课问题进行了实验，该方案的不足之处在于对“温度”设定单调递减，禁止出现预计中的回火问题。束礼菊提出整数规划模型，该模型根据高校村课系统中各位教师对每门课的乐教度与熟练度，统计出相应的绩效评价系数，在此基础上，列出了排课系统中的六项目标，以实现教师群体最优绩效为总目标，以其他目标为约束条件，采用整数规划模型将指派教师讲授各班级特定的课程这一人文决策过程转化为数学组合问题:来优化高校排课，该模型只把教师群体最优绩效为总目标，没有考虑学生群体的特点，因此模型还须进一步改进，李琪等采用遗传算法对排课系统进行建模，得到各班级的课程安排和时间安排、最后在此基础上通过矩阵运算得到上课地点，不足之处在于对部分约束条件未能完全与算法结合。动态规划理论被应用于高校教学管理当中，得出了教学管理中重要的环节-排课的最优策略，另外，图论算法也被用于排课系统中来设计模型。

1.3 本文主要任务

本文从排课问题的研究背景、意义及研究现状，分析排课问题的必要性，研究探讨了几种解决排课问题的算法，结合实例分析比较它们的优点和缺点，最终结合图论着色理论给出一种排课算法。

第一章绪论。介绍本文的研究背景及意义、排课问题研究现状和本文研究内容。

第二章图论基本概念和理论。概述图论有关概念以及着色理论，包括图的分类、点染色、边染色和全染色等着色理论。

第三章排课问题算法分析与比较。分析研究几种经典的排课问题算法，比较其优缺点。

第四章排课问题研究。分析排课问题因素和约束条件，并且结合学校排课实例给出图染色排课算法的应用，解决教师、班级、教室的时间冲突现象。

第五章总结与展望。总结本文工作，指出本文研究不足之处，以及下一步要做的进一步研究工作。

2．图论基本概念和理论

2.1 图的分类

2.1.1 无向图：边集 E(G)为无方向边的集合，任意一条边都代表 u 连 v，以及 v 连 u，每条边都是双向的。如图 1 所示。

2.1.2 有向图：边集 E(G)为有方向边的集合，每条边都是单向的，即只能由一个点指向另一个点。如图 2 所示。

2.1.3 带权图：边集 E(G)是每条边上有权值的边的集合，同理，带权图也有有向带权图和无向带权图之分，当然，不加权的图可以看成所有边上的权值都是 1。如图 3

2.1.4 完全图：对于图中任意两个顶点都是相邻的，称之为完全图，如图 4 所示。

2.1.5 无向完全图：在阶无向图中，图中任意两个顶点间都有一条边相连，则该图称为无向完全图，如图 4 所示。

2.1.6 有向完全图：在阶有向图中，图中任意两个顶点间都有方向相反的边相连，该图称为有向完全图,如图 5 所示。

2.1.7 二部图：设无向图图中,如果图中顶点集 V 可划分为两个互不相交的非空子集 X 和 Y，并且图中的每条边有一个端点属于子集 X，另一个端点属于子集 Y，则称图 G 为一个二部图,如图 6 所示。

2.1.8 简单图:在无向图中，如果关联一对顶点的无向边多于一条，则称这些边为平行边，平行边的条数称为重数。而自环是两端连接着同一端点的边，那么既无环也无平行边的图就是简单图。如图 1 所示。

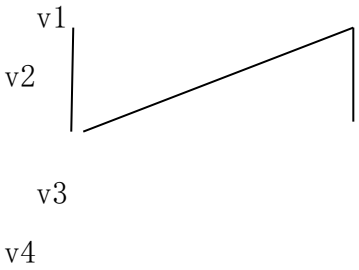


图 1 无向图（简单图）

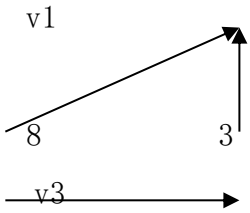


图 2 有向图

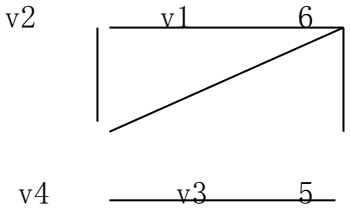
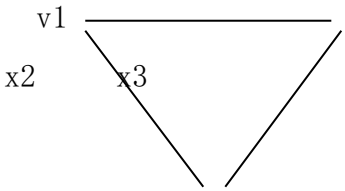
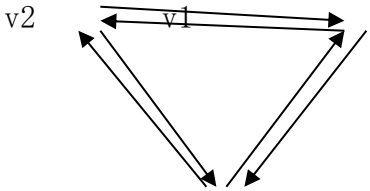
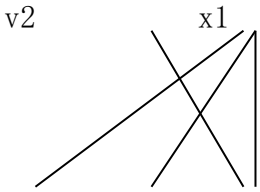


图 3 带权图







V2

v3

y2

y3

图 4 完全图（无向完全图）

图 5 有向完全图

图 6 二部图

2.2 着色理论——图的染色

关于图的染色类型有很多，比如图的点染色、边染色、全染色、邻接边染色等，其中最具有代表性的是点染色、边染色和全染色。

2.2.1 点着色

给定图 G = (V, E),设顶点集$V(G)=\{v_0,v_1,\ v_2,...,v_{n-1}\}$,则其点着色是指将图 G 中顶点集 V(G)中的每个元素进行染色，顶点全部染色之后，要求使每个邻接顶点的颜色不同。换句话说，在给定颜色集 C(k) = {1,2, … , k}之后，可以将顶点集划分成多个相互独立的子集形式$V(K)=\{v_1,v_2,...,v_{n-1},v_n\}$,要求每个子集中的顶点之间无邻接顶点，然后在给定的颜色集合 C(k)中选取一种颜色对每个子集进行染色，使每个子集的颜色互不相同，这也被称为图 G 的正常 k-染色。图 G 的全体正常 k-染色构成的集合通常记为$C_{vk}(G)$,简记为$C_k(G)$。若$C_k(G)\neq\varnothing$，即图 G 中至少有一个正常 k-点染色，就称图 G 是正常 k-点可染色，使图 G 正常 k-点可染色的 k 的最小值称为图 G 的色数,用 x(G)来表示图 G 的色数,若 x(G) = k,则称图 G 是 k-色图。

那么对于 k 有没有取值范围呢?若对于完全图 G 是正常 k-点可染色的，很显然，k ≥ x(G),x(G) = Δ(G) + 1, Δ(G)表示图 G 中的最大度数。1941 年 Brooks 给出了一个重要的结论：若 G 是一个非完全连通图且Δ(G) ≥ 3，则称图 G 是Δ(G)-点可染色的。并且由此结论可以导出另一个结论：若非完全连通图 G 是 k-点可染色的，那么 k 的取值范围是 x(G) ≤ k ≤ Δ(G)，但由于对任一平面图，可以构造出顶点度数任意大的平面图，所以该结论的范围太大。

2.2.2 边着色

类似于邻接顶点一样，对于图 G 中也有邻接边的概念：图 G 中的任意两条边e_1和e_2，若这两条边有且仅有一个公共顶点，则称这两条边是邻接边。若把图 G 的顶点染色看成是从顶点集 V(G)到颜色集 C(k)的一个映射，即 f：V(G)→C(k),设定$\forall v_1 \in V(G), v_2 \in V(G)$,并且$v_1$和$v_2$是邻接结点，那么该映射 f 需要满足$f(v_1) \neq f(v_2)$。

仿照这种定义，同样地，我们也可以把图的边染色用映射来定义：把图 G 的边染色看成是图 G 的边集 E(G)到颜色集 C(k)的一个映射 f：E(G)→C(k)，设定$\forall e_1 \in E(G), e_2 \in E(G)$,并且$e_1$和$e_2$是邻接边,那么该映射 f 需要满足$f(e_1) \neq f(e_2)$。用简记的$C_e(G)$来表示图 G 的正常 k-边染色。若$C_e(G) \neq \varnothing$，即图 G 中至少存在一条正常 k-边染色，则称图 G 是 k-边可染色的。用$x'(G)$来表示图 G 的边色数，即使图 G 为 k-边可染色的最小值。若$x'(G) = k$，则称图 G 是 k 色图。

1964 年 Vizing 和 1966 年 Gupta 分别独立研究完成了边色数$x'(G)$的取值范围。

性质：设在简单图 G 中，v_1和v_2是图 G 中两个不相邻的顶点，γ是 G 的正常 k 边染色，若对该染色γ，v_1和v_2以及与v_1相邻的点均至少缺少一种颜色，则 G + $v_1 v_2$也是 k 边染色的。

定理：若图 G 是一个简单图，则其边色数$x'(G)$= Δ(G)或者$x'(G)$= Δ(G)+1。

2.2.3 全着色

图的全着色（total coloring）是指给图的顶点和边都分配颜色

定义：图 G = (V, E)的一个 k-全染色是从$V \cup E$到 C(k) = {1,2, … , k}的一个映射 f：假设对$\forall e_1$，$e_2 \in V \cup E$，其中e_1和e_2是两个相邻或相伴的元素(这里e_1和e_2并不是表示边，是表示$V \cup E$中的元素),有$f(e_1) \neq f(e_2)$，则称 f 是图

G = (V, E)的一个正常全染色。同时，图 G 的全色数记作$x_\gamma(G)$，$x_\gamma(G)$可以用下式表示：

$$x_\gamma(G)=\min\{k|\text{存在一个正常}k-\text{全染色}\}$$

3.排课问题算法分析与比较

早期解决排课问题，更多是使用传统算法和精确算法，包括动态规划法、贪婪算法等等。这类算法在面对规模较小的问题时可以获得较优解，但遇到大规模问题需要花费大量时间，并且不能获得理想的结果。 目前普遍使用的是现代启发式算法,又称为智能算法。在解决复杂优化问题时，启发式算法可以获得令人满意的解，并且花费的时间在可接受范围内。遗传算法、蚁群算法、模拟退火算法都是目前常用的启发式算法。

3.1 动态规划法

动态规划是运筹学领域中一个重要的分支，一般用来解决决策过程的最优化。20 世纪 50 年代初，美国数学家 R.Bellman 等人通过对多阶段决策过程优化问题的研究，提出了“最优化原理”理论。该理论的思想是在解决复杂的多阶段过程问题时，先把复杂问题分解为更小的、相似的子问题，再利用子问题之间的关系逐个求解，解决最优化问题的动态规划法也因此诞生。1957 年 R.Bellman 出版了关于动态规划法的第一本著作《Dynamic Programming》目前，动态规划被广泛应用于多个领域，例如路径规划、资源分配、航天飞机飞行控制等。

动态规划法解决排课问题,需将问题逐级分解,直至子问题能直接求解为止，在分解后，降低了问题的复杂程度。动态规划法求解排课问题首先需要描述问题的特征，其次递归最优解，并自底向上的计算最优解，最后依据每一步获得的局部最优解，构成全局最优解。

3.2 模拟退火算法

1953 年，美国物理学家 N.Metropolis 发表了关于研究计算复杂系统能量分布的文章,在研究中采用蒙特卡罗模拟法计算多分子系统中分子的能量分布在研

究中采用蒙特卡罗模拟法计算多分子系统中分子的能量分布。Kirk patrick 等学者借鉴 Metropolis 的方法探讨一种旋转玻璃态系统时，发现其物理系统的能量和组合最优问题的成本函数相当类似，即寻求最低成本相当于寻求最低能量。由此，他们提出以 Metropolis 方法为基础的一种新型算法，并用其来求解组合问题的最优解。1982 年，Kirk patrick 等人在《Science》发表文章《Optimization by Simulated Annealing》，该文章中首次提到模拟退火算法。

模拟退火算法来源于退火原理，退火是指将固体加热到足够高的温度，使分子呈随机排列状态然后逐步降温使之冷却，最后分子以低能状态排列，固体达到某种稳定状态。目前，模拟退火算法在图像识别、神经网络、生产调度问题中得到广泛的应用。模拟退火算法求解排课问题首先要初始化排课问题关联的参数，设计排课约束条件，其次根据约束条件生成初始课表，并采用初始课表作为初始解，然后构建解决问题的目标函数，最后根据目标函数求解极小值。

3.3 遗传算法

1967 年美国 Michigan 大学 John Holland 教授的学生首次在其论文中提出“遗传算法”一词，并在此后几年时间，多次发表关于遗传算法研究的论文。直至 1975 年，J. Holland 出版了《Adaptation in Natural and Artificial Systems》，这是第一本讲述遗传算法的专著，因此也把 1975 年作为遗传算法诞生的一年。同年 K.A.DeJong 发表了他的博士论文《An Analysis of the Behavior of a Class of Genetic Adaptive System》该论文将他的实验测试与 J. Holland 的理论结合，对遗传算法解决最优化问题具有重要的意义。近些年我国学者对遗传算法进行了深入的研究，王冠等人提出了使用并行遗传算法解决大规模组合规划问题的方法，与基本遗传算法相比在搜索解的速度上有了提高。牛慧兰等学者在解决背包问题时采用遗传算法，并结合禁忌搜索的思想，试图用遗传算法做全局搜索，禁忌搜索辅助做局部搜索，这一尝试在获取全局最优解方面有了的进展。

遗传算法是启发式算法的一种，它模拟了生物世界中适者生存、优胜劣汰的自然进化过程。作为成熟的全局优化算法，遗传算法相比于传统算法有更好的优势。主要因为遗传算法拥有更好的自行组织搜索能力，在自适应和自学习能力方面表现突出，并且算法不需要求导或其他辅助知识，寻优的过程中由概率决定，而不是确定性的，同时遗传算法可以直接应用，具有高扩展性。但同时遗传算法也存

在不足，体现在以下几方面：遗传算法基本参数的选择与解的质量相关，但目前这些参数的选择大部分是依赖经验，且需要根据具体问题的规模进行实验后才能确定，普遍适用性较差；编码实现过程较为复杂，当获得最优解后还需对其进行解码；问题求解规模较大，遗传算法容易过早收敛于局部最优解。目前，遗传算法更多应用于排课、生产调度、机器学习、图像处理和模式识别等问题。但求解规模过大时遗传算法容易陷入早熟，因此现在也常用遗传算法和其它算法协同使用解决问题。

3.4 蚁群算法

20 世纪 90 年代初，意大利学者 Marco Dorigo 在欧洲人工生命会议上提出一种通过模拟蚂蚁觅食行为的算法——蚁群算法，由此蚁群算法第一次在人们的视线中出现。1996 年，Marco Dorigo 等学者发表文章《Ant system:optimization by a colony of cooperating agents》，在文中对蚁群算法的核心思想进行了进一步的阐述，Marco Dorigo 等人还在 1988 年组织举办了蚁群算法的国际会议，此后会议每两年召开一届。蚁群算法作为正反馈算法，常用于处理优化问题。蚁群算法在问题空间内展开独立的解集搜索，每只蚂蚁在搜索过程中都是一个独立的个体，它们之间只通过信息素进行沟通，因此蚁群算法有优秀的全局搜索能力。蚁群算法自身的优势很多，但是遇到群体规模较大的求解问题时，蚁群算法容易出现早产或者信息素更新停滞的现象，需要花费大量的时间才能找到最优的解。蚁群算法最早用来解决路径规划问题，并取得了一定的成效，目前该算法多用于解决图像边缘检测、网络路由、网络路由和车辆调度问题等。

3.5 禁忌搜索算法

禁忌搜索算法（Tabu Search，简称 TS）在 1986 年由美国科罗拉多大学的 Fred Glover 教授提出。它是一种全新的启发式算法，不再像以往算法一样在局部邻域内搜索，而是在全局范围内开展逐步寻优策略。TS 算法的优点是具有“记忆”功能，在搜索过程中遇到局部循环时，可以依靠禁忌表和特赦操作，准确地跳出局部最优，进而逐步指向全局最优。

和其它启发式算法相比，TS 算法最大的优点莫过于其较强“爬山”能力，从而能够有效避免局部最优。而能够有效避免局部最优。在每一步搜索时，新的可行解不是随机生成的，通过和禁忌表的对比，能够找到截至目前步骤的最优解，

所以 TS 算法的优化能力要强于其它启发式算法。不过，TS 算法也有缺点，在处理较大规模的时间表问题时，TS 算法的时间效率并不高，并且算法很依赖邻域的设计。

4．排课问题研究

4.1 排课问题因素

在排课问题中,关系到的因素较多，并且各因素之间相互存在着多种制约,因此增加了排课问题的复杂性。总之,教师、教室、班级、时间和课程是排课问题中要解决的五类重要因素。

4.1.1 课程因素：课程编排的时候安排时间的一个重要对象就是课程,每个学校对课程有着严格的规定,要严格按照教学计划对课程进行合理的设置，一般来说,每门课程对象都有自身的属性,如:课程号、课程名、学分数、课程类别、课时数等。由于每门课程的各个属性的不相同, 因此排课时就会影响到课程安排的最终结果。

4.1.2 教学班因素:教学班是排课设置中的最小单位,在排课的时候是按照教学班为单位进行教师资源、教室资源和时间资源的分配。每个班级有其自身的属性，如班级编号和班级名称。并且班级与学院之间应该是一对多的关系,并且要求在某个具体的时间,每个班级仅能安排一门课程。

4.1.3 教师因素:教师作为课程的主体也是需要全面考虑的重要因素。每个教师都有自己的教工号、姓名、职称等属性,并且教师与课程是一对多的关系，但是对于同一名教师,在某个具体的时间，每个教师只能教授一门课程。

4.1.4 教室因素:教室是完成课程教学的主要场所,教室也分为小班教室、大班教室，机房和实验室等类型。每个教室都有所在楼名称、楼层门牌号和容纳人数等属性。但是,在某一个具体时间段内,每个教室只能安排一门课程,且要求尽量不要资源浪费，如按照授课人数安排合适的教室。

4.1.5 时间因素:在编排课程表时会严格根据教学计划中课程所需课时数的属性,相应的也会涉及到星期，周，学期等时间的概念。

4.2 排课约束条件

综上所述，一个合理的排课方案必须满足相应的约束条件，约束条件由两部分

组成：硬约束条件和软约束条件。硬约束条件是指在排课过程中必须遵守的条件，也可称其为规则；软约束条件是根据现实情况选择满足的条件。 硬约束条件也是衡量排课方案是否可行的准则，只要其中任意一个条件没有满足，排课工作就无法顺利完成。那么，一般硬约束条件有以下几点：

同一时间同一教师不能教授一门以上的课程；

同一时间同一教室不能安排一门以上的课程；

同一时间同一班级不能安排一门以上的课程。

教室的座位数不能小于班级人数；

教室所属类型必须与课程类型相匹配。

通常软约束条件包含以下几点：

1.一门课程在一周有多节的情况下，上课的间隔时间要合理安排； 2.当教师或者班级有连续课程时，安排的教室距离不应太远； 3.专业课或者难度较大的课程最好安排在学生思维活跃的时间段； 4.体育课应安排在下午或者上午的第二大节，因为体育课后人体疲惫，不适宜安排课程。

在遵循硬约束的前提下，根据学校的现实情况进行软约束条件的选择，这样编排的课表不仅科学合理，而且能满足大部分教师、学生的需求。

4.3 排课问题的求解目标

实际高校排课过程中， 教学计划均为已知， 也就是班级要上的课程、课程相应的教师也均为已知。所以在排课工作中，每位教师授课课程是固定的，同时每个班级上课课程也是固定的，因此排课问题的决策向量主要由时间和教室两个因素构成。时间方面体现在课程具体上课的时间安排在周几，每周几节课等，教室方面体现在课程被安排的教室类型、座位数等。

4.4 基于着色理论的排课算法及实例

4.4.1 教师与班级建立关系图

教师排课管理中时间的安排其实质是指让教师在指定的时间内去给某个班级上课。在这当中， 需要满足以下几个要求:首先， 在同一个时间范围内教师只可以给一个班级上课;其次， 不同班级在同一时间也只能有一个老师教课。运用图论方法（关系图）来解决这一问题， 如:有 N 个老师， 分别是 X1,…,XN 另外有

M 个班级， 分别是 Y1,....,YM， pij 表示教师给 xi 班级 yj 上课的次数,其中某个老师要教某个班级就要将这两者用线连接起来， 若是一个星期之内某个老师给某个班级上了两次课，则用两条线相连，依次增加。

例如：如果一天内，有 4 个老师分别给 5 个班的学生上不同的课，则其中 Xi (i=1,2,3,4)代表 4 个不同的教师，Yj(j= 1,2,3,4,5)代表 5 个不同的班级，其教师与班级的关系图如图 7 所示。

	y1	y2	y3	y4	y5
x1	1	1		1	
x2	1		1		
x3		1			1
x4			1		1

图 7 教师班级关系图

对应的关系矩阵如图 8

$$Pij=\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

图 8 关系矩阵

从这个教师班级关系图中，我们可得到一个顶点集 X = {x1,x2,x3,x4}和顶点集 Y ={y1,y2,y3,y4,y5}，以 P 表示教师和班级关联的图，也可以用图 9 表示关联图，现在可通过图的边染色理论对授课分配时间段。

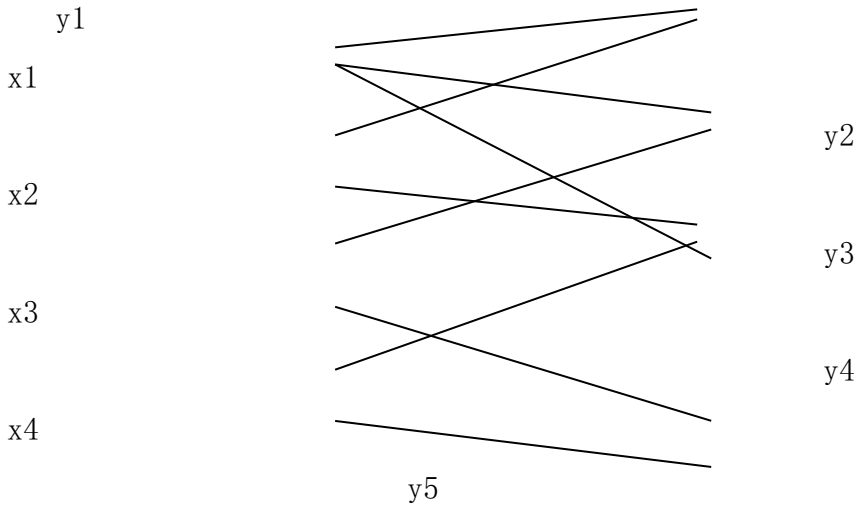


图 9 教师班级初始关联图

4.4.2 边着色理论分配时间

众所周知， 同一个顶点的边是相邻边。对每条边用不同颜色上色， 一种颜色指代一段时间， 拿大学为例， 大学课程一般为两课时一节课， 一天四节， 一星期五天， 因此在教师排课管理中边色数应为二十， 指代的是二十段时间， 一样的颜色指代的是一个时间段是由于相同时间当中每个老师只可以去一个班级教课，不同班级在同一时间也只能有一个老师教课。相邻边指代的是老师或学生是相同的， 不能安排在同一时间段内上课， 这就要求相邻边不可以使用相同颜色标明。

从图 9 中我们可以看出该图共有顶点 9 个，若现在用 K 种颜色对其进行染色以后，使得图中每个顶点相邻的边都染上不同的颜色，我们称之为 K-正常边染色。比如在图 9 中与顶点 x1 相连的边有三条，分别是 x1y1 ，x1y2 ，x1y4,所以在染色以后，这三条边都会被染上不同的颜色，每种颜色表示一种授课关系，且图中最多有三条边相邻，因此，在一张有 K（K≥3）个时间段的课表上上课，就可以满足在同一时间段内，每个教师会到不同班级授课的情况，而且在同一个班级内，不会出现多个教师来授课的情况。

针对上面的例子，假设以 3 个时间段为基础，给出一种基于二部图边染色算法的排课算法:

(1) 先以教师与班级的授课关系，给出一张教师班级关系图，根据教师班级关系图，给出一个表示教师的顶点集为 X = {x1,x2,x3,x4}和一个表

示班级顶点集为 Y ={y1,y2,y3,y4,y5}的二部图，如图 9 所示，在图 9 中寻找一条最多只包含 3 条边的对集 M1，即 M1 中不能出现任意两条边有相邻的情况，得到 M1={ x1y1, x2y3,x3y5}，这些边可以用同一种颜色进行染色，即把这些边的授课关系安排在同一时间段内，也就是 M1 时间段。

(2) 把（1）中已经染色的边（M1 中的边）删去，得到一个新的关联图，如图 10，接下来继续从该图中找一个边数不超过 3 的对集 M2={x1y2,x2y1,x4y3},用第二种颜色对这个对集里的边进行染色，安排在时段 M2。

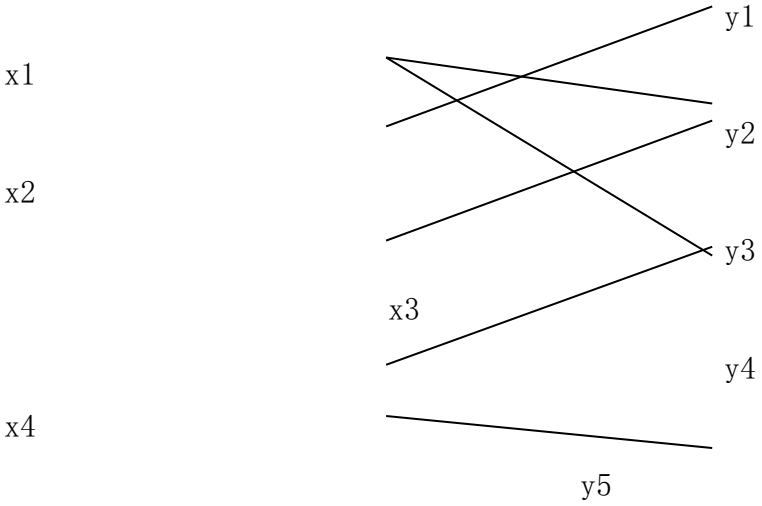


图 10 去掉对集 M1 后的关联图

（3）在上图中去掉 M2 中的边又得到一个新图，如图 11，显然去掉 M2 的边之后就得到第三个时间段的授课关系 M3={x1y4,x3y2,x4y5}

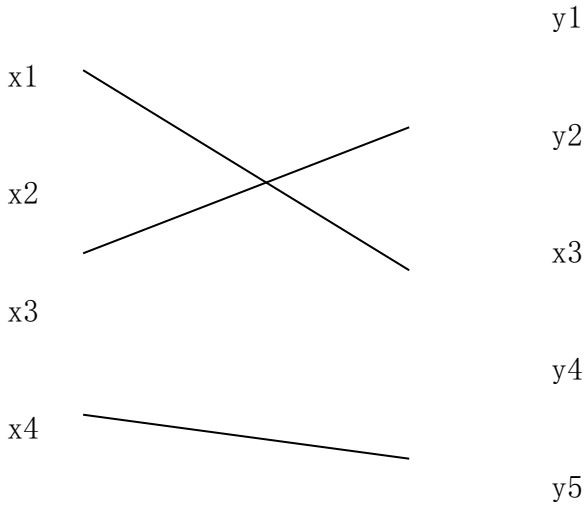


图 11 去掉对集 M2 后的关联图

（4)通过以上三步对集的寻找，我们可以得到三个时间段的授课表，如图 12，M1={x1y1, x2y3,x3y5},M2={x1y2,x2y1,x4y3},M3={x1y4,x3y2,x4y5}。

	M1	M2	M3
x1	y1	y2	y4
x2	y3	y1	
x3	y5		y2
x4		y3	y5

图 12 一个 3 个时间段的授课表

4.4.3 实例

接下来以大学一天四个课时（上午两个课时，下午两个课时，晚上一般是自习或者选修课）为基础，还是假设有 5 个老师分别给 6 个班的学生上不同的课，其中 X i(i= 1,2,3,4,5)代表 5 个不同的教师，Yj(j= 1,2,3,4,5,6)代表 5 个不同的班级，其教师与班级的关系图如图 13 所示。

	y1	y2	y3	y4	y5	y6
x1	1			1		1
x2	1	1	1		1	
x3			1	1		1
x4			1		1	
x5		1				

图 13 教师班级关系图

从这个教师班级关系图中，我们可得到一个顶点集 X = {x1,x2,x3,x4,x5}和顶点集 Y ={y1,y2,y3,y4,y5,y6}，以 P 表示教师和班级的关联图，也可以用图表示关联图，现在可通过图的边染色理论对授课分配时间段。

（1）从图 14 中我们可以看出该图共有顶点 11 个。在图 14 中与顶点 x1 相连的边有三条，分别是 x1y1 ,x1y2 ,x1y4,所以在染色以后，这三条边都会被染上不同的颜色，每种颜色表示一种授课关系，且图中最多有四条边相邻，因此，在一张有 K（K≥4）个时间段的课表上上课，可以满足在同一时间段内，每个教师会到不同班级授课的情况，而且在同一个班级内，不会出现多个教师来授课的情况。那么，针对该例子，假设以 4 个时间段为基础，根据教师班级关系图，给出一个表示教师的顶点集为 X = {x1,x2,x3,x4}和一个表示班级顶点集为 Y ={y1,y2,y3,y4,y5}的二部图，如图 14 所示，在图 14 中寻找一条最多只包含 4 条边的对集 M1，即 M1 中不能出现任意两条边有相邻的情况，可得到 M1={x1y1,x2y2,x3y3,x4y5}，这些边可以用同一种颜色进行染色，即把这些边的授课关系安排在同一时间段内，称作 M1 时间段。

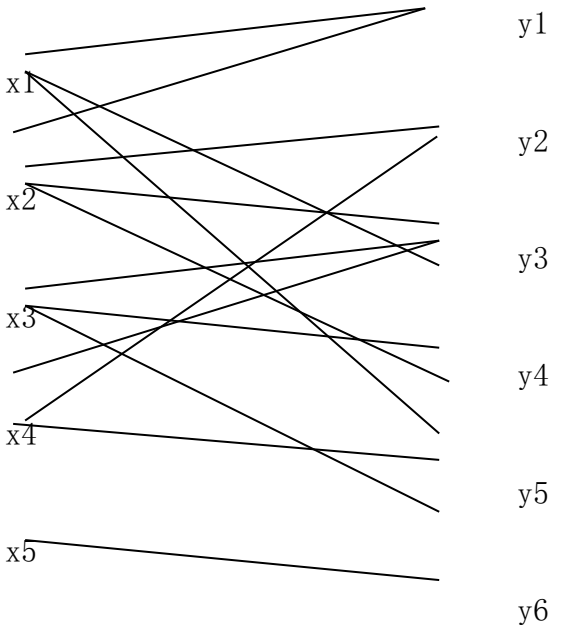


图 14 教师班级初始关联图

（2）把（1）中已经染色的边（M1 中的边）删去，得到一个新的关联图，如图 15,接下来继续从该图中找一个边数不超过 4 的对集 M2={x1y4,x2y1,x3y6,x4y3}，用第二种颜色对这个对集 M2 里的边进行染色，安排在时段 M2。

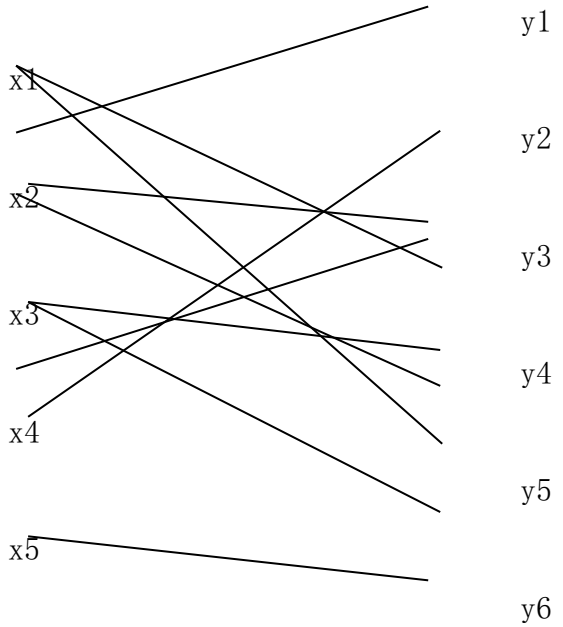


图 15 去掉对集 M1 后的关联图

（3）在上图中去掉 M2 中的边又得到一个新图，如图 16，显然去掉 M2 的边之后我们可以得到第三个时间段的授课关系 M3={x1y6,x2y3,x3y4,x5y2}。

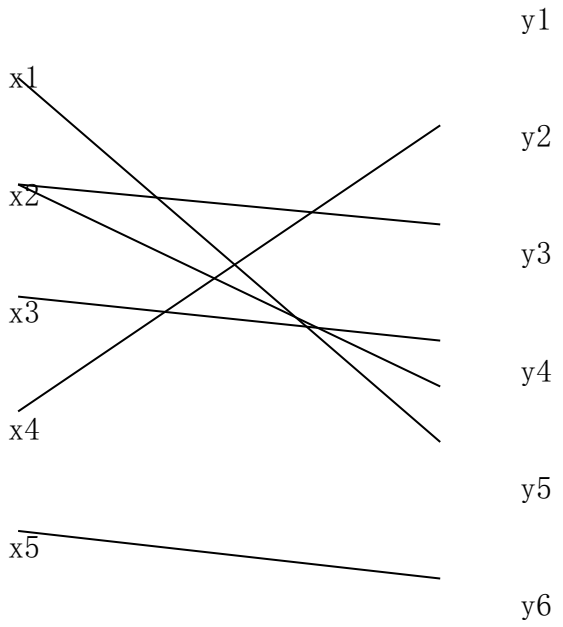


图 16 去掉对集 M2 后的关联图

（4）同理，在上图 16 中去掉 M3 对集里的边得到图 17，显然得到了一个第四个时间段的授课关系 M4={x2y5,x5y6}

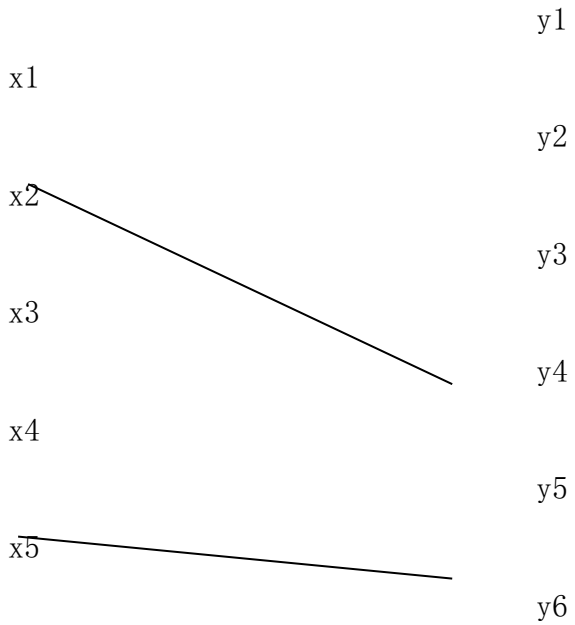


图 17 去掉对集 M3 后的关联图

（5）通过以上四步对集的寻找，我们可以得到四个时间段的授课表，如图 18，M1={x1y1,x2y2,x3y3,x4y5},M2={x1y4,x2y1,x3y6,x4y3},M3={x1y6,x2y3,x3y3,x5y2},M4={x2y5,x5y6}。

	M1	M2	M3	M4
x1	y1	y4	y6	
x2	y2	y1	y3	y5
x3	y3	y6	y4	
x4	y5	y3		
x5			y2	y6

图 18 一个 3 个时间段的授课表

4.4.4 排课模型图的优化

按照上文的分析， 在进行排课的时候， 如果单纯使用边着色理论， 只能确保从时间角度教师与班级不发生冲突。无法保证可以满足教学方面的需求。另外，

在进行排课的时候还要考虑到一些特殊情况。这样一来，便需要对排课的效果进行优化设计。

4.4.5 排课模型图的赋值

首先应该设计出权值函数，结合教学效果对排课模型图中的一些边赋了权值，选出权值最大加权图，最终得出最优的排课方案。

4.5 Java 编程实现排课问题的解决

4.5.1 设计过程

通过着色图论的分析和设计，可以使用 Java 编程排课算法来实现排课问题的解决，要求从文本文件导入教室情况（大小，数量）、培养方案（课程、学生年级、人数、教师）、约束条件（有些约束从培养方案中可知，如一个教师可以教授两门课不可以安排在同一时间上，其他包括课程先修关系，教师的特殊情况如不能上上午一二节课等），系统最终生成课表或给出无法排课的错误提示。使用多种存储结构，使用多种常用算法，编写复杂系统以此来解决排课问题

首先定义一个课程、学期、老师类，这些类包含了课程、班级、教师、教室、星期、周次几个属性。课程、班级、教师是安排好了的，教室、星期、周次是通过对集合对象元素处理的算法来安排，其程序目录如下：

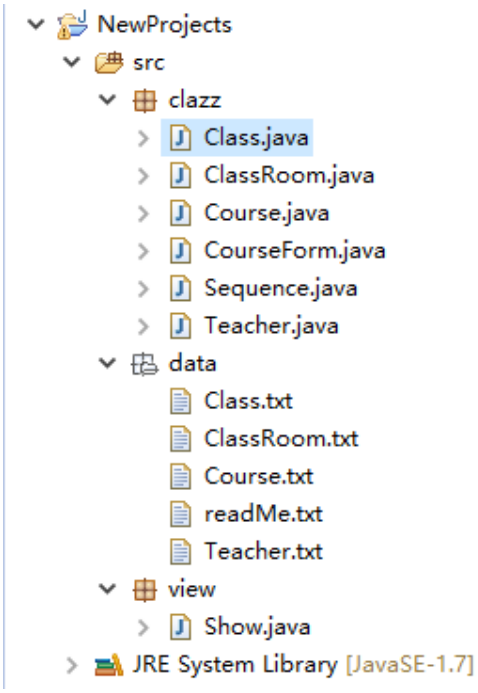


图 19 工程目录

Class.txt 配置文件说明：每列第一个数为班级 id，第二个字符串为年级名称，第三个数为年纪人数。(数据为本班级每学期的教学数据)

ClassRoom.txt 配置文件说明：每列第一个数为教室 id，第二个字符串为教室名称，第三个数为教室容量。(数据为本班级每学期的教学数据)

Course.txt 配置文件说明：每列第一个数为课程 id，第二个字符串为课程名称，第三个数为周课时，之后如果有则为对应的先修课程。(数据为本班级每学期的教学数据)

Teacher.txt 配置文件说明：每列的第一个数为教室 id;第二个字符串为教师姓名；之后字符串为老师所教课程，结尾可输入老师无法上课的时间代号。(数据为本班级每学期的教学数据)

这个 Sequence 类用来存储计算课表种群的冲突。当被测试课表冲突为 0 的时候，这个课表就是个符合规定的课表。冲突检测遵循下面几条规则：

同一个教室在同一个时间只能有一门课。

同一个班级在同一个时间只能有一门课。

同一个教师在同一个时间只能有一门课。

排课算法核心是集合对象LinkedList双向循环链表和Set集合实现的。队列：先进先出的数据结构。栈：后进先出的数据结构。

用 LinkedList 实现队列：队列(Queue)是限定所有的插入只能在表的一端进行，而所有的删除都在表的另一端进行的线性表。表中允许插入的一端称为队尾(Rear)，允许删除的一端称为队头(Front)。队列的操作是按先进先出(FIFO)的原则进行的。队列的物理存储可以用顺序存储结构，也可以用链式存储结构。

用 LinkedList 实现栈：栈(Stack)也是一种特殊的线性表，是一种后进先出(LIFO)的结构。栈是限定仅在表尾进行插入和删除运算的线性表，表尾称为栈顶(top)，表头称为栈底(bottom)。栈的物理存储可以用顺序存储结构，也可以用链式存储结构。

利用 BufferedReader 类从文本文件读取到排课信息数据，然后再将各元素对象存入到 LinkedList 集合对象中。用循环遍历将 LinkedList 集合对象中的元素按照约束条件来进行判断。将存放教师与课程的集合对象交集,取出教师和课程的一定次数的随机组合，以此达到最终排课问题的解决。

程序核心代码

```
1. /**
```

```
2.  *开始排课
3.   */
4.  private static void Arranging() {
5.   //contains（）,该方法是判断字符串中是否有子字符串。如果有则返回 true，如果没有则返回 false。
6.   for (Course cou : courseList) { // for 循环遍历课程集合
7.     Teacher tea = null; // 老师实体类初始化
8.     Class cla = null; // 班级实体类初始化
9.     // 将 set 对象集合的内容作为条件进行分支判断
10.    if (preCouse.contains(cou.getName())) {
11.      for (Class x : classList) { //遍历班级对象
12.                                  if  (x.getGrade()  ==  1)  {

13.                                      cla  =  x;
14.                                      break;
15.                                  }
16.                          }
17.      for (Teacher x : teacherList) { //遍历教师对象
18.        if (x.teachCourse.contains(cou.getName())) {
19.                                      tea  =  x;
20.                                      break;
21.                                  }
22.                          }
23.
24.      order(tea, cla, cou);
25.
26.   } else if (having_preCourse.contains(cou.getName())) {
27.                          int  max  =  0;
28.      for (Class x : classList) { //遍历班级对象
29.        if (x.getSq().getList().size() > max && x.getGrade() != 1) {
```

```
                                        cla = x;
            max = x.getSq().getList().size();
                                    }
                                }
        for (Teacher x : teacherList) {//遍历教师对象
          if (x.teachCourse.contains(cou.getName())) {
                                        tea = x;
                                        break;
                                    }
                                }
        order(tea, cla, cou);
                    } else {
                                int max = 0;
        for (Class x : classList) {//遍历班级对象
          if (x.getSq().getList().size() > max) {
                                        cla = x;
            max = x.getSq().getList().size();
                                    }
                                }
        for (Teacher x : teacherList) {//遍历教师对象
          if (x.teachCourse.contains(cou.getName())) {
                                        tea = x;
                                        break;
                                    }
                                }
                                //调用排课算法
        order(tea, cla, cou);
                    }
          }
} /**
 * @param te 教师对象
```

```
3.  * @param cl 班级对象
4.  * @param co 课程对象
5.  * 排序算法
6.    */
7.  @SuppressWarnings("unchecked")
8.  private static void order(Teacher te, Class cl, Course co) {
9.    temList1 = (LinkedList<String>) (te.getSq().getList().clone());
     //temList1 存放原教师空闲时间
10.   te.getSq().getList().retainAll(cl.getSq().getList()); //求教师与学生时间交集
11.   if (te.getSq().getList().size() < co.getTimesWeek()) {
12.     JOptionPane.showMessageDialog(null, "老师和同学公共可利用时间不足安排", "安排失败", 1);
13.     System.out.println("老师和同学公共可利用时间不足安排" + co.getName());
14.         }
15.   temList2 = randList(te.getSq().getList(), co.getTimesWeek());
//temList2 存放教师与学生时间交集,取出教师和学生的一定次数的随机组合
16.   cl.getSq().getList().removeAll(temList2); //移去被分去的时间
17.   te.getSq().setList(temList1); //恢复 temList1 中时间
18.   te.getSq().getList().removeAll(temList2); //移去被分去的时间
19.   // System.out.println(cl.getSq().getList().size());
20.   for (String x : temList2) {
21.                   int   max   =   1000;
22.     ClassRoom selectRoom = null;
23.     for (ClassRoom y : classroomList) {//课程教室地点分配，循环遍历教室对象
24.       if (y.getSq().getList().contains(x)
25.           && y.getCapacity() - cl.getSum() < max
```

```
26.             && y.getCapacity() - cl.getSum() >= 0) {//将集合内的对象
    进行比较
27.         max = y.getCapacity() - cl.getSum();
28.                                     selectRoom  =  y;
29.                             }
30.                 }
31.     if (selectRoom != null) {
32.       selectRoom.getSq().getList().remove(x);
33.                 }  else  {
34.       System.out.println("教室资源不够。");
35.       JOptionPane.showMessageDialog(null,"教室资源不够。","提示
    ", 1);
36.                             return;
37.                 }
38.     //向教师实体类中的数组存入排课后的信息内容
39.     te.observed[x.charAt(0) -'a'] = co.getName() + "" + cl.getName
    ()
40.         + "" + selectRoom.getName();
41.     cl.observed[x.charAt(0) -'a'] = co.getName() + "" + te.getName
    ()
42.         + "" + selectRoom.getName();
43.          }
```

4.5.2 效果截图

图 20 程序主界面

图 21 排课成功界面

信息与计算科学1601班大一下学期(2016/2017/2)的班级课表

时间	星期一	星期二	星期三	星期四	星期五
上午1-2节	管理学通论(1/8周) 徐建文 B205	中国近代史纲要(1/8周) 唐小芹 A101			概率论A(1/9周) 陈内萍 B205
上午3-4节		体育(四)(1-8周) 李幼萌体育馆前坪 ...	概率论A(1/9周) 陈内萍 A101	体育(二)(男)(1/8周) 彭志伟体育馆...	大学英语(二)(1/12周) 周佳 B205
下午5-6节				复变函数(1/9周) 罗智明 A101	
下午7-8节	复变函数(1/9周) 罗智明 B205		常微分方程(1/12周) 张文 A101	数值分析(1/12周) 罗智明 A101	

信息与计算科学1601班大二上学期(2017/2018/1)的班级课表

时间	星期一	星期二	星期三	星期四	星期五
上午1-2节	运筹学基础A(1/12周) 廖云华 A101		信息系统分析与设计(1/9周) 蔡妙娴...	应用写作(1/8周) 罗靖 B205	
上午3-4节	Matlab科学计算编程语言(1/9周) 王...	体育(二)(女)(1/8周) 易正兰体育馆...		运筹学基础A(1/12周) 廖云华 A101	应用写作(1/8周) 罗靖 A101
下午5-6节		马克思主义基本原理B(1/8周) 陈瑜 ...			体育(二)(女)(1/8周) 易正兰体育馆...
下午7-8节			操作系统B(1/9周) 史湘宁 A307	操作系统B(1/9周) 史湘宁 A307	

信息与计算科学1601班大二下学期(2017/2018/2)的班级课表

时间	星期一	星期二	星期三	星期四	星期五
上午1-2节			C语言程序设计(数)(1/15周) 曾强聪...		
上午3-4节	毛泽东思想与中国特色社会主义理...		算法导论(1/9周) 陈建文 B205	中国文化概论(1/8周) 刘明丽 A307	
下午5-6节	中国文化概论(1/8周) 刘明丽 A101			毛泽东思想与中国特色社会主义理...	算法导论(1/9周) 陈建文 A101
下午7-8节		数学分析(二)(1/18周) 张健 B205		数学分析(二)(1/18周) 张健 B205	JAVA程序设计(信)(1/12周) 胡德发 ...

图 22 课表截图

5. 总结与展望

随着计算机技术和人工智能的不断快速发展，图论相关理论和方法也逐渐融入到人们实际生活的很多方面，这意味着图论着色理论的作用也与日俱增，所谓的染色事实上就是对一些对象按照一定的方法进行分类，那么图染色就是指对图中的顶点、边和面等元素按照一定的规则进行分类，我们所制定的对象不同或者规则不同都会导致我们对图的染色方式不同。如今按照规则区分的图染色方式有成百上千种，比如从最初的点染色，边染色、面染色到如今发展到全染色、表染色、领强边染色、子染色、完备染色等等。众所周知，在我们实际生活中，很多问题的数学模型都可以用图的方式来建立，比如学校排课、库存管理、物流分配、通讯频率分配、寄存器分配等等这些数学模型都可以转化为图的形式来建立，因此我们只要能找到合适的染色算法就能帮助我们去很好的解决类似于这类有冲突的问题。

排课工作是高校教务工作中重要的一项工作内容，本文应用图论着色理论来研究排课问题，首先介绍了相关图论理论以及常见排课算法，其次分析排课问题以及基于着色理论的排课算法。由于时间和能力有限，本文研究内容存在很多不足，比如在本文中， 只适当考虑了某个工作日内的排课情况， 但是在实际应用中， 高校管理员在进行排课的时候需要考虑一周中五个工作日的排课情况， 与

此同时还需要考虑一些特殊情况的存在，比如单双周排课的问题. 因此，为了更好地解决高校排课问题，还需要进行进一步研究。

参考文献：

[1]C.C.Gotlieb. The Construction of ClassTeacher Time Tables. Proceeding IFIP[J].Congress62 ,1963:73-74.

[2] S.Even, A. Itai, A.Shamir. On the complexity of timetable andmulticonmmodity flowproble ms[J] .SIMA Joumal on Computing ,1976.5[41:691-703.

[3]J.A.Bondy and U.S.R.Murty.Graph theory with applications Joumal of the Operational Research,1977.

[4]王能斌,钱祥根.大学课程表调度系统——UTSS[J].计算机学报,1984(05):383-389.

[5]张清绵,徐明,王鸣,吴北雅,王凤兰.智能教学组织管理与课程调度系统[J].大连理工大学学报,1991(02):227-232.

[6]张清绵,郜荣春.用电子计算机自动构造高等院校课程表[J].大连工学院学报,1981(04):115-118.

[7]林漳希,林尧瑞.人工智能技术在课表编排中的应用[J].清华大学学报(自然科学版),1984(02):1-9.

[8]夏鹭平,方潜生.利用人工智能原理实现大学课程表调度——大学课程表调度模型及算法[J].微电子学与计算机,1992(10):46-49.