فهرست مطالب

مقدمه و توضیحات پروژه

تولید برنامه زمان بندی برای نهادهای مختلف از جمله دانشگاه ها همیشه یکی از پیچیده ترین و زمان برترین فرایندها بوده است که مستلزم صرف وقت و انرژی زیاد برای کارکنان است. درانجام این کار بصورت دستی ابتدا یک جدول اولیه ساخته میشود و سپس براساس محدودیت های دانشگاه و اساتید تا جایی تغییر پیدا میکند که تناقضی در آن باقی نماند.

در چند دهه گذشته الگوریتم های تکاملی از جمله ژنتیک در مسائل زیادی ورود کرد اند که به دلیل ماهیت آن ها در مسئله رمان بندی هم میتوانند استفاده شوند.

در این پروژه بنا براین است که با داشتن منابع استاندارد دانشگاه برای ساختن برنامه درسی شامل : اساتید،کلاس ها ، زمان ها، دروس و ... و محدودیت های سخت و نرم شامل: محدودیت اساتید،محدودیت زمان، محدودیت مکان و .... برنامه کلاسی تولید کنیم که هیچ محدودیت سختی را تقض نکند و محدویت های نرم را نیز تا حد امکان رعایت کند.

در انجام این پروژه از زبان برنامه نویسی پایتون و ویژگی شئ گرایی آن استفاده شده است همچنین از یک کتابخانه برای ایجاد یک محیط کاربری ساده جهت گرفتن اطلاعات از کاربر کمک گرفته شده است. برای نمایش جدول نهایی در خروجی نیز از کتابخانه های رسم پلات استفاده گردیده است.

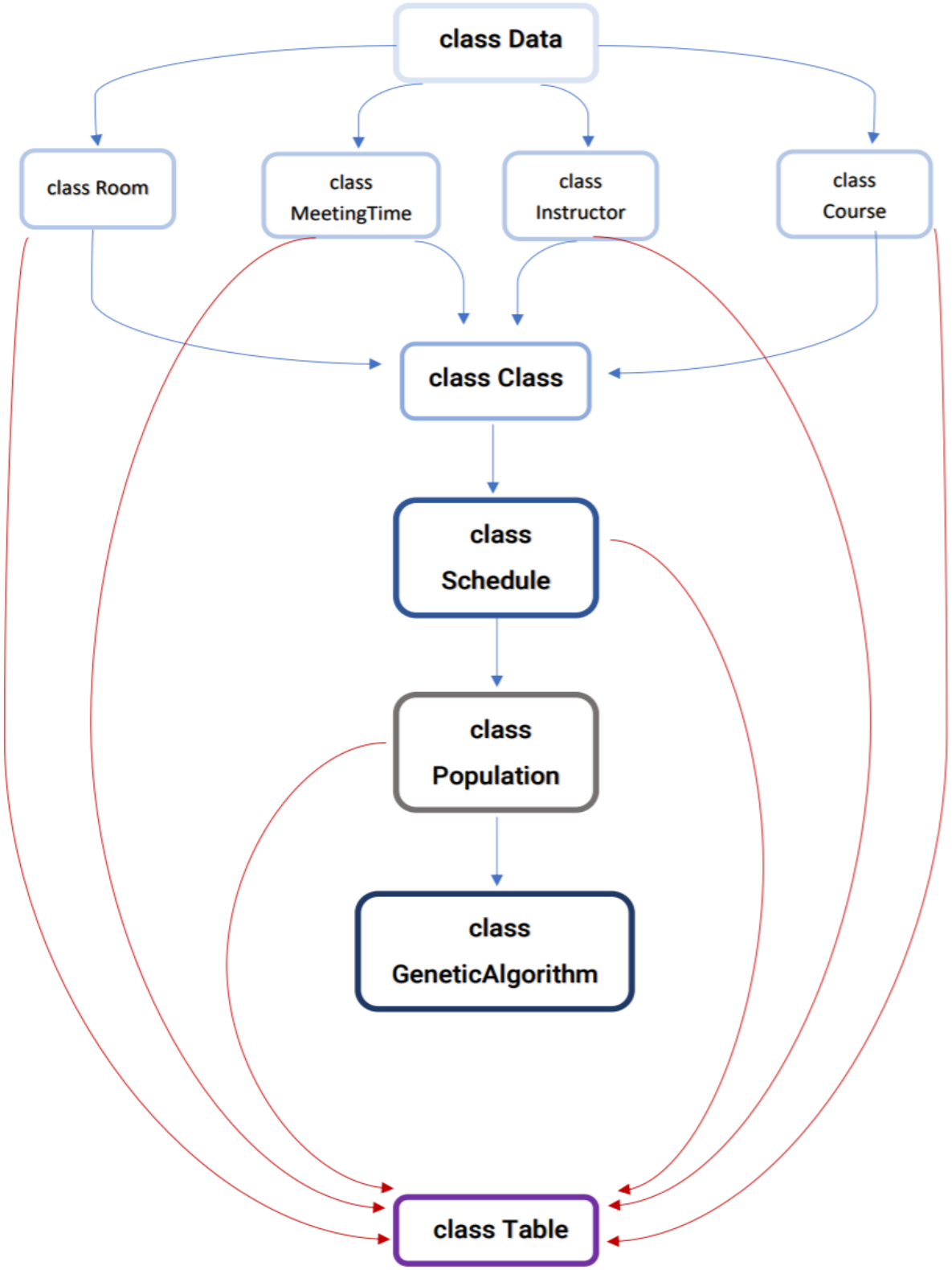
نمونه ایی از گرفتن ورودی های و نمایش خروجی بصورت زیراست:

گرفتن ورودی از رابط کاربری

Class #	Course (number, max # of students)	Room(capacity)	Instructor(id	Meeting Time(id)
0	pattern(C1, 25)	Arshad1(25)	Dr Mohseni(p4)	yekshanbe-seshanbe 10:30 - 12:00(T4)
1	ML(C2, 30)	Arshad3(35)	Dr Eftekhari(p2)	yekshanbe-seshanbe 09:00 - 10:45(T3)
2	DM(C3, 25)	Arshad1(25)	Dr Mohseni(p4)	shanbe-doshanbe 10:00 - 11:30(T2)
3	N.N(C4, 32)	Arshad3(35)	Dr Saeed(p5)	shanbe-doshanbe 10:00 - 11:30(T2)
4	EC(C5, 15)	Arshad2(15)	Dr Afsari(p1)	yekshanbe-seshanbe 09:00 - 10:45(T3)
5	IP(C6, 35)	Arshad3(35)	Dr Saeed(p5)	yekshanbe-seshanbe 10:30 - 12:00(T4)
6	Fu(C7, 15)	Arshad2(15)	Dr Eftekhari(p2)	shanbeh-seshanbeh 19:00 - 20:30(T5)

نمونه ایی از جدول نهایی خروجی

فلوچارت کلی الگوریتم:



در فصل های آینده قسمت های با اهمیت کد به تفصیل توضیح داده خواهند شد

تولید یک برنامه کلاسی اولیه بصورت تصادفی

برای نوشتن این الگوریتم ژنتیک از ۱۰ کلاس به شرح زیر استفاده کردیم :

(۱) Class data
(۲)Class schedule
(۳) Class population
(٤)Class Geneticalogorithm
(٥)Class course
(٦)Class Instructor
(۷)Class room
(۸)Class meetingtime
(۹)Class class
(۱۰)Class table

از کلاس های نام برده شده مهم ترین کلاس ها Schedule و GeneticAlgorithm هستند که ابتدا به شرح کلاس Schedule میپردازیم.

این کلاس دوبخش دارد که در بخش اول پس از تعین پارامترهای مورد نیاز و گرفتن دادن ها یک برنامه اولیه بصورت تصادفی تولید میکند که این کار به صورت زیر انجام میشود:

```python
class Schedule:
    def __init__(self):
        self._data = data
        self._classes = []
        self._numbofConflicts_H = 0
        self._numbofConflicts_S = 0
        self.finall_numbofconlicts = 0
        self._fitness = 0
        self._classNumb = 0
        self._isFitnessChanged = True
    def get_classes(self):
        self._isFitnessChanged = True
        return self._classes
    def get_numbofConflicts(self):
        return self.finall_numbofconlicts
    def get_fitness(self):
        if (self._isFitnessChanged == True):
            self._fitness = self.calculate_fitness()
            self._isFitnessChanged = False
        return self._fitness

    def initialize(self):
        # new code after delete dept :
        courses = self._data.get_courses()
        # for i in range(0,len(depts)):
        # courses = depts[i].get_courses()
        for j in range(0, len(courses)):
            newClass = Class(self._classNumb, courses[j])
            self._classNumb += 1
            newClass.set_meetingTime(data.get_meetingTimes()[rnd.randrange(0, len(data.get_meetingTimes()))])
            newClass.set_rooms(data.get_rooms()[rnd.randrange(0, len(data.get_rooms()))])
            newClass.set_instructor(courses[j].get_instructors()[rnd.randrange(0, len(courses[j].get_instructors()))])
            self._classes.append(newClass)
        return self
```

در کد بالا ابتدا داده ها دریافت میشوند و سپس متغیرهایی برای تعیین تعداد تناقض در محدودیت های سخت و نرم تعریف شده اند که این متغیر های سخت و نرم در بدنه اصلی کد که در انتها آورده شده است باید صفر شوند.

برای ساختن یک برنامه اولیه تصادفی به تعداد Course(درس ها) هر بار یک کلاس ایجاد میکنیم و از منابع: زمان ها،کلاس ها و اساتید بصورت رندوم هر بار یکی را برمیداریم و داخل کلاس قرار میدهیم.

در بخش دوم این کلاس میزان شایستگی کلاس ساخته شده بر اساس نقض کردن محدودیت های سخت و نرم سنجیده میشود که کد آن بصورت زیر است:

```python
def calculate_fitness(self):
    self._numbofConflicts_H = 0
    self._numbofConflicts_S = 0
    self.finall_numbofconlicts=0
    classes = self.get_classes()
    for i in range(0,len(classes)):
        if (classes[i].get_room().get_seatingCapacity() < classes[i].get_course().get_maxNumbofStudents() ):
            self._numbofConflicts_H += 1  # hard constraint now 1
        for j in range(0,len(classes)):
            if (j >= i):
                if (classes[i].get_meetingTime() == classes[j].get_meetingTime() and
                    classes[i].get_id() != classes[j].get_id()):
                    if (classes[i].get_room() == classes[j].get_room()):
                        self._numbofConflicts_H += 1  # hard constraint now 2
                    if (classes[i].get_instructor() == classes[j].get_instructor()):
                        self._numbofConflicts_H += 1  # hard constraint now 3
        m1= classes[i].get_instructor()
        # if (str(classes[i].get_instructor())== "Dr Eftekhari" and  "MT2" == classes[i]._meetingTime.get_id()  ):
        if (str(classes[i].get_instructor()) == sc[0][0] and sc[0][1] == classes[i]._meetingTime.get_id()):
            self._numbofConflicts_S+=1   # soft constraint now 1

        m2=  classes[i].get_meetingTime()

        # if (str(classes[i].get_instructor())== "Dr Afsari" and  "MT5"  ==  classes[i]._meetingTime.get_id() ):
        if (str(classes[i].get_instructor()) == sc[1][0] and sc[1][1] == classes[i]._meetingTime.get_id()):
            self._numbofConflicts_S += 1  # soft constraint now 2

        if (str(classes[i].get_instructor())== sc[2][0] and sc[2][1] ==  str(classes[i]._room.get_number() )):
            self._numbofConflicts_S += 1  # soft constraint now 3

    self.finall_numbofconlicts = (self._numbofConflicts_H*1)+(self._numbofConflicts_S*0.8)  # 0.8 and 1 are factors for soft and hard constranints

    # we have soft constranint now!!!
    return 1 / (self.finall_numbofconlicts + 1)# 1 to avoid divide by zero
```

در این جا ابتدا سه متغیر برای تعین تعداد تقض محدودیت های سخت و نرم ساخته میشوند و سپس سه محدودیت سخت که بترتیب:

۱- اگر حداکثر ظرفیت کلاس از تعداد دانشجوهای یک درس(ظرفیت درس) کمتر باشد.
۲- اگر زمان دو کلاس مجزا یکی باشد و اتاق ها نیز یکی باشند.
۳- اگر زمان دو کلاس مجزا یکی باشد و استاد نیز یکی باشد.

سه محدویت فرضی نرم بصورت زیرهستند:

۱- استاد اول در ساعت اول کلاس نداشته باشد.
۲- استاد دوم در ساعت دوم درس نداشته باشد.
۳- استاد سوم در کلاس دوم (مثلا ارشد ۲) درس نداشته باشد.

در انتها برای محاسبه مقدار شایستگی از فرمول ابتکاری زیر با تغیراتی استفاده شده است:

$$Fitness = \frac{K}{0.8 \times \sum H_i \times hard_i + 0.2 \times \sum S_i \times soft_i + 1}$$

بدنه اصلی الگوریتم ژنتیک

تمامی عملگر های الگوریتم ژنتیک شامل: crossover,mutation,selection در این کلاس تعریف شده اند.

```
class GeneticAlgorithm:
    def evolve(self, population):
        return self.mutation(self.crossover(population))

    def crossover(self, pop):
        crossover_popu = Population(0)

        for i in range(POPULATION_SIZE ):
            crossover_popu.get_schedules().append(pop.get_schedules()[i])
            i = 0  # nothing is mysteries now !! dast dast !!
        while i < POPULATION_SIZE:
            crossoverShedule = Schedule().initialize()
            sci1 = self.selection(pop).get_schedules()[rnd.randrange(0, 3)]  # its random
            sci2 = self.selection(pop).get_schedules()[rnd.randrange(0, 3)]  # its random
            i += 1

            for i in range(0, len(crossoverShedule.get_classes())):
                if (rnd.random() < Crossover_rate):
                    crossoverShedule.get_classes()[i] = sci1.get_classes()[i]
                else:
                    crossoverShedule.get_classes()[i] = sci2.get_classes()[i]
                crossover_popu.get_schedules().append(crossoverShedule)

            return crossover_popu

    def selection(self, pop):
        out_selection = Population(0)
        for index in range(0, TOURNAMENT_SELECTION_SIZE):
            tournament_pop = Population(0)
            tournament_pop.get_schedules().append(pop.get_schedules()[rnd.randrange(0, 4)])
            tournament_pop.get_schedules().sort(key=lambda x: x.get_fitness(), reverse=True)
            out_selection.get_schedules().append(tournament_pop.get_schedules()[0])

        return out_selection
```

در این جا ابتدا به تعداد Popsize عملگر Selection صدا زده میشود که هربار الگوریتم selection که براساس انتخاب مسابقات است تعدادی فرد(به تعداد Tournoment selection size) به صورت مرتب براساس شایستگی برمیگرداند سپس دو فرد از این جمعیت(دوبرنامه کلاسی) به صورت رندوم انتخاب میشوند و مراحل Crossover شروع میشود. دراین مرحله با چک کردن یک عدد رندوم با نرخ همبری که در ابتدای کد تعریف شده است یکی از والدین به مرحله جهش میرود.

در مرحله جهش دوباره با چک کردن یک عدد رندوم با نرخ جهش در صورت بزرگتر بودن عدد رندوم یکی از کلاس های برنامه با مقداری تصادفی که در لحظه تولید میشود جایگزین میگردد یا در واقع برنامه جهش پیدا میکند. تابع جهش در زیر قابل مشاهد است:

```
def mutation(self,pop):
    schedule = Schedule().initialize()
    mutatation_pop= Population(0)
    for i in range(0,POPULATION_SIZE):
        self.muto= (pop.get_schedules()[i])
        for i in range(0, len(self.muto.get_classes())):
            if (MUTATION_RATE > rnd.random()):
                self.muto.get_classes()[i] = schedule.get_classes()[i]
        mutatation_pop.get_schedules().append(self.muto)

    return mutatation_pop
```

نتایج

در این قسمت به بررسی نتایج حاصل از اجرای الگورینم میپردازیم. با توجه به ماهیت تصادفی الگوریتم ژنتیک در اجراهای متوالی تعداد نسل های ساخته شده یکسان نیستند و در هر بار اجرا میتوانند متفاوت باشند. همچنین با تغیر تعداد قیدها و اندازه منابع سرعت اجرای الگوریتم تغییر میکند به گونه ایی که با اضافه کردن قیدها سرعت پاین می آید(نسل ها زیاد میشوند) ولی با اضافه کردن منابع مانند اضافه کردن تعداد کلاس ها و اساتید سرعت اجرا بالاتر میرود درواقع دست الگوریتم برای ساختن یک برنامه کلاسی بازتر گذاشته میشود.

با یک بار اجرا الگوریتم و داده های ورودی زیر:

id - instructor	room (capacity)	id - Metting Time	id - course (instructors, capacity)
p1 - Dr Afsari	Arshad1 (25)	T1 - shanbe-doshanbe 09:00 - 10:30	C1 - pattern (Dr Afsari, Dr Mohseni ,25)
p2 - Dr Eftekhari	Arshad2 (15)	T2 - shanbe-doshanbe 10:00 - 11:30	C2 - ML (Dr Eftekhari ,30)
p3 - Dr Nik nafs	Arshad3 (35)	T3 - yekshanbe-seshanbe 09:00 - 10:45	C3 - DM (Dr Nik nafs, Dr Mohseni ,25)
p4 - Dr Mohseni		T4 - yekshanbe-seshanbe 10:30 - 12:00	C4 - N.N (Dr Mohseni, Dr Saeed ,32)
p5 - Dr Saeed		T5 - shanbeh-seshanbeh 19:00 - 20:30	C5 - EC (Dr Afsari ,15)
			C6 - IP (Dr Mohseni, Dr Saeed ,35)
			C7 - Fu (Dr Eftekhari, Dr Nik nafs ,15)

تغیر نسل ها به صورت زیر است:

Generations	Fitness - Conflicts
9 - C1,Arshad2,p1,T3, C7,Arshad3,p2,T1C2,Arshad3,p2,T2, C7,Arshad3,p2,T1C3,Arshad1,p4,T2, C7,Arshad3,p2,T1C4,Arshad2,p4,T4, C7,Arshad3,p2,T1C5,Arshad2,p1,T3, C7,Arshad3,p2,T1C6,Arshad2,p5,T2, C7,Arshad3,p2,T1	F = 0.147 - C = 5.8
0 - C1,Arshad1,p1,T2, C7,Arshad1,p2,T1C2,Arshad2,p2,T1, C7,Arshad1,p2,T1C3,Arshad2,p3,T1, C7,Arshad1,p2,T1C4,Arshad2,p4,T4, C7,Arshad1,p2,T1C5,Arshad3,p1,T3, C7,Arshad1,p2,T1C6,Arshad2,p4,T3, C7,Arshad1,p2,T	F = 0.128 - C = 6.8
1 - C1,Arshad2,p4,T2, C7,Arshad3,p3,T4C2,Arshad2,p2,T4, C7,Arshad3,p3,T4C3,Arshad2,p4,T2, C7,Arshad3,p3,T4C4,Arshad2,p5,T1, C7,Arshad3,p3,T4C5,Arshad2,p1,T1, C7,Arshad3,p3,T4C6,Arshad2,p5,T2, C7,Arshad3,p3,T·	F = 0.091 - C = 10.0
Generation 1605	
0 - C1,Arshad1,p4,T4, C7,Arshad2,p2,T5C2,Arshad3,p2,T3, C7,Arshad2,p2,T5C3,Arshad1,p4,T2, C7,Arshad2,p2,T5C4,Arshad3,p5,T2, C7,Arshad2,p2,T5C5,Arshad2,p1,T3, C7,Arshad2,p2,T5C6,Arshad3,p5,T4, C7,Arshad2,p2,T5	F = 1.0 - C = 0.0
1 - C1,Arshad2,p4,T1, C7,Arshad3,p2,T4C2,Arshad3,p2,T1, C7,Arshad3,p2,T4C3,Arshad1,p4,T4, C7,Arshad3,p2,T4C4,Arshad2,p5,T3, C7,Arshad3,p2,T4C5,Arshad2,p1,T1, C7,Arshad3,p2,T4C6,Arshad2,p5,T5, C7,Arshad3,p2,T4	F = 0.2 - C = 4.0
2 - C1,Arshad2,p4,T4, C7,Arshad1,p3,T5C2,Arshad3,p2,T1, C7,Arshad1,p3,T5C3,Arshad1,p4,T3, C7,Arshad1,p3,T5C4,Arshad2,p5,T3, C7,Arshad1,p3,T5C5,Arshad2,p1,T3, C7,Arshad1,p3,T5C6,Arshad3,p4,T3, C7,Arshad1,p3,T5	F = 0.2 - C = 4.0
3 - C1,Arshad2,p1,T1, C7,Arshad3,p2,T4C2,Arshad3,p2,T3, C7,Arshad3,p2,T4C3,Arshad1,p4,T3, C7,Arshad3,p2,T4C4,Arshad2,p5,T3, C7,Arshad3,p2,T4C5,Arshad2,p1,T1, C7,Arshad3,p2,T4C6,Arshad3,p5,T5, C7,Arshad3,p2,T4	F = 0.2 - C = 4.0
4 - C1,Arshad2,p4,T2, C7,Arshad1,p2,T1C2,Arshad3,p2,T3, C7,Arshad1,p2,T1C3,Arshad3,p4,T3, C7,Arshad1,p2,T1C4,Arshad3,p5,T4, C7,Arshad1,p2,T1C5,Arshad2,p1,T3, C7,Arshad1,p2,T1C6,Arshad2,p5,T2, C7,Arshad1,p2,T1	F = 0.2 - C = 4.0
5 - C1,Arshad1,p4,T2, C7,Arshad2,p3,T4C2,Arshad3,p2,T2, C7,Arshad2,p3,T4C3,Arshad1,p4,T3, C7,Arshad2,p3,T4C4,Arshad1,p5,T4, C7,Arshad2,p3,T4C5,Arshad1,p1,T1, C7,Arshad2,p3,T4C6,Arshad3,p4,T2, C7,Arshad2,p3,T4	F = 0.179 - C = 4.6
6 - C1,Arshad1,p4,T4, C7,Arshad2,p3,T4C2,Arshad3,p2,T3, C7,Arshad2,p3,T4C3,Arshad1,p4,T2, C7,Arshad2,p3,T4C4,Arshad1,p4,T3, C7,Arshad2,p3,T4C5,Arshad3,p1,T3, C7,Arshad2,p3,T4C6,Arshad2,p4,T3, C7,Arshad2,p3,T4	F = 0.172 - C = 4.8
7 - C1,Arshad2,p1,T4, C7,Arshad1,p3,T4C2,Arshad2,p2,T4, C7,Arshad1,p3,T4C3,Arshad3,p4,T2, C7,Arshad1,p3,T4C4,Arshad3,p4,T5, C7,Arshad1,p3,T4C5,Arshad2,p1,T3, C7,Arshad1,p3,T4C6,Arshad1,p5,T4, C7,Arshad1,p3,T4	F = 0.167 - C = 5.0
8 - C1,Arshad1,p1,T2, C7,Arshad1,p3,T5C2,Arshad1,p2,T5, C7,Arshad1,p3,T5C3,Arshad3,p4,T2, C7,Arshad1,p3,T5C4,Arshad2,p5,T4, C7,Arshad1,p3,T5C5,Arshad2,p1,T4, C7,Arshad1,p3,T5C6,Arshad1,p4,T1, C7,Arshad1,p3,T5	F = 0.167 - C = 5.0
9 - C1,Arshad2,p1,T3, C7,Arshad3,p2,T1C2,Arshad3,p2,T2, C7,Arshad3,p2,T1C3,Arshad1,p4,T2, C7,Arshad3,p2,T1C4,Arshad2,p4,T4, C7,Arshad3,p2,T1C5,Arshad2,p1,T3, C7,Arshad3,p2,T1C6,Arshad2,p5,T2, C7,Arshad3,p2,T1	F = 0.147 - C = 5.8
0 - C1,Arshad1,p1,T2, C7,Arshad1,p2,T1C2,Arshad2,p2,T1, C7,Arshad1,p2,T1C3,Arshad2,p3,T1, C7,Arshad1,p2,T1C4,Arshad2,p5,T3, C7,Arshad1,p2,T1C5,Arshad3,p1,T3, C7,Arshad1,p2,T1C6,Arshad3,p5,T4, C7,Arshad1,p2,T	F = 0.147 - C = 5.8
1 - C1,Arshad2,p4,T2, C7,Arshad3,p3,T4C2,Arshad2,p2,T5, C7,Arshad3,p3,T4C3,Arshad2,p4,T2, C7,Arshad3,p3,T4C4,Arshad2,p5,T1, C7,Arshad3,p3,T4C5,Arshad2,p1,T1, C7,Arshad3,p3,T4C6,Arshad2,p5,T2, C7,Arshad3,p3,T·	F = 0.091 - C = 10.0

و در نهایت برنامه کلاسی نهایی ساخته میشود:

Class #	Course (number, max # of students)	Room(capacity)	Instructor(id	Meeting Time(id)
0	pattern(C1, 25)	Arshad1(25)	Dr Mohseni(p4)	yekshanbe-seshanbe 10:30 - 12:00(T4)
1	ML(C2, 30)	Arshad3(35)	Dr Eftekhari(p2)	yekshanbe-seshanbe 09:00 - 10:45(T3)
2	DM(C3, 25)	Arshad1(25)	Dr Mohseni(p4)	shanbe-doshanbe 10:00 - 11:30(T2)
3	N.N(C4, 32)	Arshad3(35)	Dr Saeed(p5)	shanbe-doshanbe 10:00 - 11:30(T2)
4	EC(C5, 15)	Arshad2(15)	Dr Afsari(p1)	yekshanbe-seshanbe 09:00 - 10:45(T3)
5	IP(C6, 35)	Arshad3(35)	Dr Saeed(p5)	yekshanbe-seshanbe 10:30 - 12:00(T4)
6	Fu(C7, 15)	Arshad2(15)	Dr Eftekhari(p2)	shanbeh-seshanbeh 19:00 - 20:30(T5)