**Team 13 – Project Description: University Course Scheduling**

**General Problem Description**

The problem in general is to construct a schedule of classes at a university for a single term / semester. This schedule is often referred to as a “university timetable” and it includes four key elements: courses, lecturers, classrooms, and time slots. The courses, lecturers and classrooms have additional information (such as classroom capacity, course maximum enrollment, lecturer maximum teaching load, etc.) which form a set of constraints on the overall schedule.

This is generally considered a constraint satisfaction problem, with multiple hard constraints as well as some soft constraints or preferences. Although there appear to be some available open-source and commercial applications that will generate a timetable, scheduling in general is not a solved problem and is an active area of research in the AI community.

Depending on the size of the university, there could be a very large CSP to solve. For example, at Purdue University, in a typical term there are approximately 9000 classes offered in over 500 teaching spaces. The classes vary in size as do the classrooms. Some of the classroom spaces are shared across the different departments, while others are dedicated to a specific department.

**Scope, Goals, Experiments**

Scope

We will limit the scope of the project to a smaller data set, something on the order of a course plan for a College of Engineering.

Goal

The goal is to take a set of input data (a set of courses, lecturers, classrooms, and time slots) and produce a course schedule that satisfies the constraints.

The objective is to develop several approaches to solving the CSP (i.e. producing a schedule) and to compare the performance of the algorithms.

Experiments

We intend to investigate a number of approaches to solving the problem. Thus, the main experiments will involve running the different algorithms against a number of different baseline data sets and comparing the results. Additional experiments may include running the algorithms against a data set of larger problems, in order to determine how well the algorithms scale in runtime and memory.

In order to assess many approaches to the problem, we will look for open source algorithm implementations that we can leverage.

Performance

We will assess the performance of the algorithms in 3 ways: 1) using a score for a generated schedule (described below), 2) the relative runtime performance of the algorithm and 3) the ability of the algorithm to scale as the problem grows.

Scoring

The example problems provided my our main data source (described below) include a number of hard constraints as well as many soft constraints. In general, solutions exist which meet all of the hard constraints and the goal is to minimize the number of broken soft constraints.

We will develop a score which appropriately penalizes broken hard and soft constraints.

**Potentially Applicable AI Technologies and Approaches**

* Constraint Satisfaction techniques, including all the tricks (backtracking search, minimum-remaining value, degree and least constraining variable heuristics, node/arc consistency, etc.).
* Genetic algorithms and evolutionary/nature-inspired algorithms (such as ant colony optimization, bee colony, etc.).
* There are also some multi-agent approaches (where an agent represents a course, or a lecturer, or a classroom, and the multiple agents have to arbitrate and negotiate) which might be worth a look.

**Data Sets / Sources**

We will continue to look for existing data. At the moment we have two sources of timetabling data. To begin we will use data from the International Timetabling Competition (2002-03, 2007, 2011, 2019). Datasets are available from the 2007 and 2011 competitions, and we will use data from 2007 (<http://www.cs.qub.ac.uk/itc2007/>) in the first phase of the project.

An additional data source is UniTime.org. This provides some benchmark data sets for larger scale scheduling problems.

Listed below is an example data set for a Toy Problem from ITC 2007. There are 4 main sections which include:

* Courses: <CourseID> <Teacher> <# Lectures> <MinWorkingDays> <# Students>
* Rooms: <RoomID> <Capacity>
* Curricula: <CurriculumID> <# Courses> < CourseID > ... < CourseID>
* Unavailability\_Constraints: <CourseID> <Day> <Day\_Period>

***Example Input:***

Name: ToyExample

Courses: 4

Rooms: 2

Days: 5

Periods\_per\_day: 4

Curricula: 2

Constraints: 8

COURSES:

SceCosC Ocra 3 3 30

ArcTec Indaco 3 2 42

TecCos Rosa 5 4 40

Geotec Scarlatti 5 4 18

ROOMS:

A 32

B 50

CURRICULA:

Cur1 3 SceCosC ArcTec TecCos

Cur2 2 TecCos Geotec

UNAVAILABILITY\_CONSTRAINTS:

TecCos 2 0

TecCos 2 1

TecCos 3 2

TecCos 3 3

ArcTec 4 0

ArcTec 4 1

ArcTec 4 2

ArcTec 4 3

ITC-2007 also specifies a format for the solution. During the competition this was used to allow for auto-scoring of any submission, using a score defined by the competition coordinators. The auto-scoring code may be available for us to use.

In the output format, each line represents the assignment of the room and timeslot to one lecture: <CourseID> <RoomID> <Day> <Day\_Period>

***Example Output:***

SceCosC B 3 0

SceCosC A 3 1

SceCosC A 4 0

ArcTec B 0 1

ArcTec B 1 1

ArcTec B 1 2

TecCos B 0 0

TecCos A 0 1

TecCos B 2 2

TecCos B 4 2

TecCos B 4 3

Geotec A 2 2

Geotec A 2 3

Geotec B 3 0

Geotec A 3 1

Geotec A 4 2

Typical datasets from ITC-2007 have more courses, room or constraints. Here are the headers for two of the files:

Name: Fis0506-1

Courses: 30

Rooms: 6

Days: 5

Periods\_per\_day: 6

Curricula: 14

Constraints: 53

Name: Ing0304-2

Courses: 94

Rooms: 18

Days: 5

Periods\_per\_day: 5

Curricula: 78

Constraints: 463