Adam Cornelissen
Benjamin Mader
Martijn Sprengers
22.04.2010

OPUS - ROSTERING MODULE

PRELIMINARY REQUIREMENTS

Assumptions:

User has to be authenticated, this is done by logging in to the OPUS system.

Rostering is done before the beginning of a semester.

2 rosters have to be made, 1 for lectures and 1 for exams (Who sets date? are they independent?)

Lecturer (Teacher) has to be available all week (Mo - Fr)

University Day consists of *n* lecture hours of fixed length and time (Format?).

Constraints:

Hard Constraints (more important than soft - weights?):

I. Student can‘t have 2 lectures/exams at the same time

II. rooms can‘t be booked multiple times in the same timeframe

III. Lectures have a certain timeframe (Begin - End semester)

IV. Student can not have more than *n* exams on the same day

Soft Constraints:

I. Student should not have lecture and exam at the same time

II. There should not be 2 lectures of the same course on the same day for one student.

III. Gaps between lectures should be avoided if possible.

Required Data:

I. Course/Exam names

II. how many lectures per week

III. Desired length of lecture in lecture hours.

IV. lecture type (lecture, practical session, ...)

V. number/id of registered students (with access to their other course registrations)

VI. available rooms with their capacity/configuration

Target users:

Administrators:

Before the beginning of the semester, an admin uses this module to generate a complete roster (lectures and exams) for the next semester.

Functionality:

Generation of a complete roster from the available data, taking into account the constraints and keeping their violations as low as possible (measurable).

Desired Output:

Stored somewhere (DB), accessible for timetabling module

All lectures/exams with their assigned date/time/room

Implementation Thoughts:

Memetic algorithm based on fitness function.

Fitness function evaluates the fitness of a given roster in terms of violations of constraints and overlapping. Measure of fitness could be: #violations per student.

2 interfaces, 1 input, 1 output. Output has to be accessible by the TT module.