

A WEB-BASED SYSTEM FOR MULTI-AGENT INTERACTIVE TIMETABLING

João Pedro Pedroso Nelma Moreira Rogério Reis
DCC-FC & LIACC, Universidade do Porto
R. do Campo Alegre 823, 4150-180 Porto, Portugal
{jpp, nam, rvr}@ncc.up.pt

Abstract—We propose a web-based timetabling system for a typical situation in universities, where agents (usually departments or faculties) compete for a set of resources (class rooms) on a given number of time slots.

Each agent (typically a person, on the behalf of a department) proposes the placement (room and time) for events. A dispatching system decides which event should be scheduled next, based on a pre-established set of rules, and asks its placement to the corresponding department.

The system also includes a solver that suggests the placement of an event to each agent, thus allowing a completely automated timetable construction.

We describe a prototype implemented at the Faculty of Sciences, University of Porto.

I. INTRODUCTION

Typically in a timetabling problem one needs to assign to every element of a set of events, each requiring a set of resources, to a time slot and to a room. In this assignment, a set of constraints must be satisfied; some constraints are hard (they cannot be violated for feasibility) and some other are soft (for which violation should be avoided, but is not prohibited).

In most universities, a subset of the rooms available for classes is shared by several departments. On the other hand, each department normally has its own rules for constructing the timetables: student preferences, lecturer preferences, breaks, etc. can be handled differently by different departments.

Our experience shows that when there is not a room authority which controls access to the rooms by the departments, each department tends to produce its own timetables, on its own rooms, independently. This causes problems to departments which own an insufficient number of rooms and therefore are forced to use rooms on scatter slots left by the others.

The aim of the current paper is to propose a method and a data specification which enable a set of departments to simultaneously construct their timetables. In this context, we call *department* to the entity which has the responsibility of preparing the timetables for a set of courses.

We will also make use of an authority that controls the rooms, and which determines the order through which departments request rooms to events, based on a set of pre-established rules. This way, a department can construct its timetable independently of the others (with the exception of room occupation), while keeping a high degree of “fairness” on the room attribution.

We will assume that at any point of the construction there is complete information, i.e., at any time every department knows the partial solution of all the others. This enables each department to construct its solution taking into consideration the (partial) solutions of all the others, in order to avoid room clashes, consider courses which are attended by students of more than one department, etc.

II. TIMETABLE CONSTRUCTION PROCESS

The process of construction of a timetable is the following (fig.1):

- An *iteration* corresponds to the attribution of a room, at a given time, to an event.
- Room dispatching is made according to known criteria, established by the set of departments.
- Based on those criteria, the dispatcher selects the next event to be scheduled, and asks its department a placement for it. A placement specifies a room and a starting time. The room will be reserved for the duration of the event.
- Departments formulate their requests with full information of the previously dispatched events.
- The room authority dispatches only requests that do not lead to room clashes and comply with all the other hard constraints.
- Before starting a new iteration, the system verifies that no request of placement exchange is present. All existing exchange requests must be dealt before the beginning of the new iteration.

- A department may request the release of a room previously assigned to any of its events. The release will be done immediately, and the event will be included in the unplaced events list.

A complete solution will include the specification of the room and time for all the events. Additionally, there is a set of events (a set of repeated classes for a given course), of which a student (or group of similar students) has to attend exactly one; the solution must specify which of these events each student attends.

Timetables are constructed by each of the departments independently, except for room attribution.

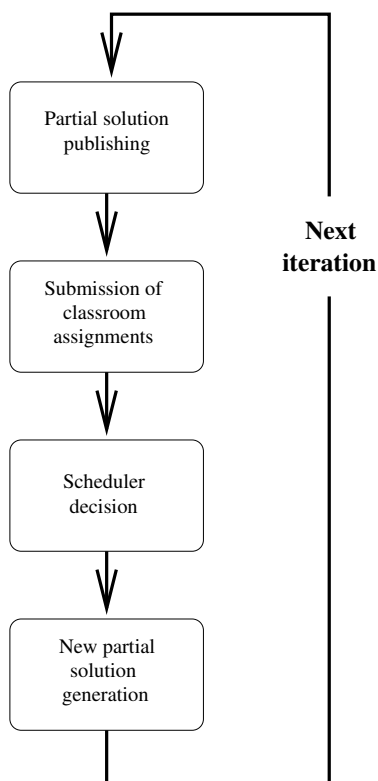


Fig. 1. Timetable construction process

III. DISPATCH RULES

The actual rules used by the dispatch authority might be different from university to university, although the concepts proposed here should be general. The ideas supporting the rules should be mostly common sense.

As a simple possibility, we propose the following dispatching rules for ordering events:

- room size required for the number of students that attend the corresponding event;

- the number of compatible rooms with the event (i.e., rooms large enough and with all the required features);
- the number slots still available in the whole of these rooms.

The preceding rules are goals; this means that the second rule is only used for deciding on ties of the first rule, and so on.

These rules allow the dispatcher to select the next event to be scheduled, and to ask the corresponding department its placement. In this situation, a department only has to find the placement for one event at a time.

Another concern is the way that exchanges can be done in the current (possibly partial) solution. We propose that exchanges take precedence over regular placements, and that when there are several requests for exchanges they are fulfilled using the order through which they have been dispatched.

There is also the possibility of releasing a room previously assigned to some event, at any time, by demand of the department which requested it. The event will be added to the unplaced events list with its initial priority.

IV. DATA MODEL

In this section we describe the data model that supports the information available on our timetabling system. No explicit difference will be made between *input data* and *output data*, or between *global data* and *department local data*. The data model should be expressive enough both for the production of *human-readable* input-output information and for system manipulations.

The moments at which events might start are defined by pairs (p, s) , where:

- any p belongs to a **scope**, which is an ordered set of **periods**;
- any s belongs to an ordered set of **slots** available for each period.

Entities are:

- **Persons**, who can have a limited number of time **preferences**, with a commitment level and polarity (positive or negative). Each person may have a role as a teacher or as a student.
- **Groups** are sets of students that must attend together a given set of lessons or courses. Groups are used for simplifying event assignment.
- **Departments** are the entities responsible for determining a timetable for a set of events, and are therefore characterised by this set. They additionally might own a set of rooms, which might be shared with the other departments or not.

- Events might belong to **courses**; a course has **classes** with several kinds of lessons (lectures, laboratories, precepts, etc), each possibly occurring several times a period and during a number of time slots. Additionally, each class might have to be repeated a number of times, as there might be an upper bound on the number of students attending it.
- **Rooms** have a **capacity** (maximal number of students), might belong to a department, and can have several features from a user-defined set.

Events correspond to class repetitions. An event can have a set of room features to be **satisfied** or **avoided**, or even a set of **preferred** rooms. Each person or group **attends** to events or courses, as a teacher or as a student. If a student attends to a course, he/she must attend one of the class repetitions, for each kind of lesson and occurrence.

The timetabling system, must provide an **assignment** to a room and a time slot for each event.

These assignments can generate violations of a set of **constraints**.

The constraints are organised by **goals**, whose list is specified by each department. Each constraint has a **weight**, with which its violations will be accounted on the goal to which it belongs. The first goal corresponds to the hard constraints.

The semantics of each constraint are not defined in this data model; they are described somewhere else. However some information must be provided on the way violations to constraints should be accounted, which will drive through the construction process.

As an example of what we define as data, we can select the following:

- the **break times** for each department, i.e slots that should not have any event assigned
- for each person, the **maximum number of assigned slots** per period
- for each person, the **maximum number of consecutive assigned slots** per period
- time ordering between events: events that can be **consecutive**, or **simultaneous**, or one **before** the other, or one **after** the other, etc.

After an event is actually placed—i.e., the corresponding request has been deferred by the dispatcher—, it is marked on the data as *fixed*, and will be allowed to change only in case the underlying department releases or exchanges it.

For the description of the data model we defined an XML language [?]. This language is a first attempt towards a general language for describing timetabling problems and

solutions, that will allow a better comparison between different approaches and an easier data exchange between systems (see also [?], [?]).

In the section VIII we represent a DTD (*Document Type Definition*) for a fragment of the above data model.

V. ALGORITHMIC BACKGROUND

The architecture used in this system allows each department to use its own preferences for choosing the placement to the event that has been requested by the dispatcher, i.e., its room and time. Even though the actual decision is taken by the user, we propose some guidelines on an algorithm to support it.

The rationale is that the definition of the relevant constraints is too intricate for the normal user. To overcome this problem, the system supplies a set of predefined constraints, with documented semantics. The user must indicate which of them are relevant, and to what extent. For this end, the user defines a list of goals, ordered by relevance. Then, for each constraint the user states the corresponding goal, and how violations of this constraint should be weighted. Hence the first goal corresponds to the hard constraints, with higher weights on the criteria that are considered more relevant (e.g., room, teacher, and student clashes with high weights, and classes on lunchtime with low weight; see [?] for a different treatment of soft constraints). The last goal corresponds to the softest constraints (like having no classes on the first hours of each day).

To suggest the placement for an event, the solver uses a greedy system: it selects the best room and the best time for the event. By best we mean the one which leads to less weighted violations on the first goal, or a tie on this goal with less weighted violations on the second goal, and so on [?]. The user might then accept this move, and submit it to the dispatcher, or ask for the next suggestion.

VI. WEB BASED MODEL

Because there is need of a two way negotiation protocol, both in the normal event placement and the exchange placement consent, a pure HTTP server model cannot be used. Instead, dedicated client programs are used by each department agent to interact with the dispatcher server (fig.2). A timetabling solver program—the one proposed or any other—, can either assist the choices or take full control over them. In the first scenario, at any time, the partial solution can be consulted on the Web server with a normal browser (fig.3). If an automated solver is used it can get an updated partial solution from the client program. Authentication is only needed between client and server programs; it is implemented via public key cryptography. All other communication is done with normal, public HTTP connections.

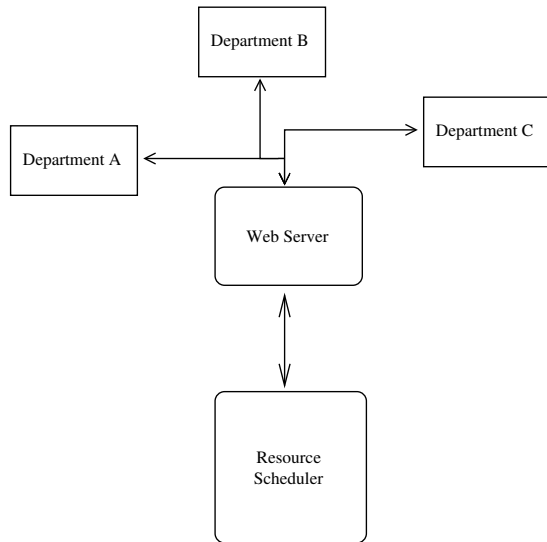


Fig. 2. Web model

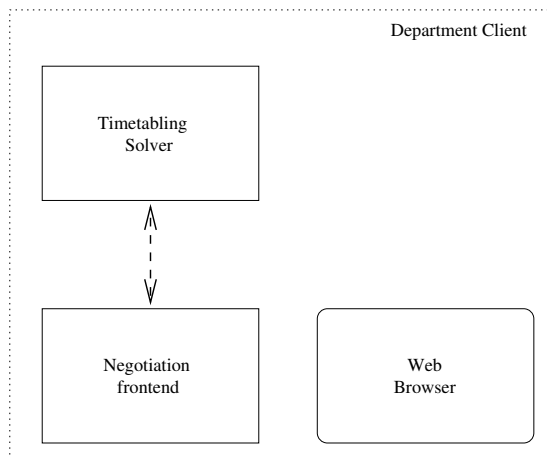


Fig. 3. Department Client scheme

VII. CONCLUSION

In this paper we introduce the concept of a room authority, responsible for dispatching requests for event placement. To the best of our knowledge, this is the first time that this concept is used on timetabling. Nevertheless, on situations where there is competition for common rooms by independent timetabling agents, this notion is crucial for having a reasonable solution method.

We specify data management and solution architectures. Their computer implementation is quite immediate, though the implementation in timetable production requires political decisions which might not be straightforward.

The method proposed in this paper needs to be validated, firstly with benchmarks, then with real instances, prior to actual use by the departments.

VIII. APPENDIX

We present here is a DTD for a fragment of our data model.

```

<!ELEMENT timetabling
(scope|slots|person+|group+|course+|class+|
event+|room+|feature*|department*|attends*|
assignment*|preference*|room_prefer*|
break_time*|max_occupation*|
max_consecutive*|time_order*|
goals*|constraint*)>
<!ELEMENT scope (period+)>
<!ELEMENT period (#PCDATA)>
<!ATTLIST period id ID #REQUIRED>
<!ELEMENT slots (slot+)>
<!ELEMENT slot (#PCDATA)>
<!ATTLIST slot id ID #REQUIRED>
<!ELEMENT person(#PCDATA)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT group EMPTY>
<!ATTLIST group id ID #REQUIRED
number CDATA #REQUIRED>
<!ELEMENT preference EMPTY>
<!ATTLIST preference person IDREF #REQUIRED
period IDREF #REQUIRED
slot IDREF #REQUIRED
level CDATA #REQUIRED
positive (1|0) #REQUIRED>
<!ELEMENT department (#PCDATA)>
<!ATTLIST department id ID #REQUIRED>
<!ELEMENT course (#PCDATA)>
<!ATTLIST course id ID #REQUIRED
department IDREF #IMPLIED>
<!ELEMENT class EMPTY>
<!ATTLIST class
id ID #REQUIRED
course IDREF #REQUIRED
type CDATA #REQUIRED
type_n CDATA #IMPLIED
nslots CDATA #IMPLIED
repetition CDATA #REQUIRED>
<!ELEMENT event EMPTY>
<!ATTLIST event id ID #REQUIRED
class IDREF #REQUIRED
repetition CDATA #REQUIRED
require IDREFS #IMPLIED
avoid IDREFS #IMPLIED>
<!ELEMENT attends EMPTY>
<!ATTLIST attends course IDREF #IMPLIED
event IDREF #IMPLIED
person IDREF #IMPLIED
role (teacher|student) "teacher"
group IDREF #IMPLIED
fixed (0|1) #IMPLIED>
  
```

```

<!ELEMENT room (#PCDATA)>
<!ATTLIST room capacity CDATA #REQUIRED
            id ID #REQUIRED
            department IDREF #IMPLIED
            features IDREFS #IMPLIED>
<!ELEMENT feature (attribute,value)>
<!ATTLIST feature id ID #REQUIRED>
<!ELEMENT attribute (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT room_prefer EMPTY>
<!ATTLIST room_prefer event IDREF #REQUIRED
            room IDREF #REQUIRED>
<!ELEMENT assignment EMPTY>
<!ATTLIST assignment event IDREF #REQUIRED
            period IDREF #IMPLIED
            slot IDREF #IMPLIED
            room IDREF #IMPLIED
            fixed (0|1) #IMPLIED>
<!ELEMENT break_time EMPTY>
<!ATTLIST break_time period IDREF #REQUIRED
            slot_start IDREF #REQUIRED
            nslots CDATA #IMPLIED
            department IDREF #IMPLIED >
<!ELEMENT max_occupation (#PCDATA)>
<!ATTLIST max_occupation person IDREF #IMPLIED
            role (teacher| student) #IMPLIED>
<!ELEMENT max_consecutive (#PCDATA)>
<!ATTLIST max_consecutive person IDREF #IMPLIED
            role (teacher| student) #IMPLIED>
<!ELEMENT time_order EMPTY>
<!ATTLIST time_order ev_id IDREFS #REQUIRED
            operator (before|after|
            simultaneous|consecutive) >
<!ELEMENT goals (goal+)>
<!ATTLIST goals department IDREF #REQUIRED>
<!ELEMENT goal (#PCDATA)>
<!ATTLIST goal id ID #REQUIRED>
<!ELEMENT constraint EMPTY>
<!ATTLIST constraint name CDATA #REQUIRED
            id ID #REQUIRED
            goal IDREF #REQUIRED
            weight CDATA #REQUIRED>

```