A Multi-agent Pattern Based Timetabling System

Mohammed El-Helly, Yasser Abdelhamid, Mohamed Al-Wakeel

Information Systems Dept., College of Computers and Information, Tabuk University, KSA.
Central Lab. For Agricultural Expert Systems, ARC, Egypt

Computer Science Dept., Community College, Tabuk University, KSA

College of Computers and Information, Tabuk University, KSA

melhelly@ut.edu.sa, yabdelhamid@ut.edu.sa, malwakeel@ut.edu.sa

Abstract

The “Academic Timetabling System” is a research project that has been funded by the deanship of scientific research of Tabuk University. The project aims at developing and implementing an academic time tabling system for Tabuk University that is based on Multi-Agent technology. In this paper we introduce the “Academic Time Tabling System” that has been designed and implemented in Tabuk University. The introduced system has two research points: The first is the introduction of a new solution model for the time tabling problem that uses predefined class hours patterns according to the features and requirements of each scheduled course, and successively filters these possible patterns as the timetable is being built. The second is introducing the Multi-Agent technology as an alternative framework for solving timetabling problems. The proposed model has been implemented and applied on real environment at Tabuk University, and the results were satisfactory in terms of compactness and preferences satisfaction. Regarding to the preferences satisfaction the results show that about 95 percent of courses are scheduled into the preference time periods proposed by the instructors, referring to the compactness the results shows that about 97 percent of the studying hours is scheduled without gaps.

Keywords: Agent, Multi Agent, Timetabling Problem, Scheduling Agent, Intelligent Agent.

1. Introduction

Academic timetables generation has been classified as NP-hard problem [1] in the general case. This means that it is unlikely that it will be possible to find fast (i.e., polynomial-time) algorithms to solve this problem. In order to find optimal solutions to such NP-hard problems, it is usually necessary to consider all possible solutions to choose the best one. However, solutions to large-scale practical problems, including timetabling, are often desired and needed much more quickly than any exhaustive search algorithm could hope to provide. As a result, numerous heuristics have been developed for such problems which produce “near-optimal” satisfactory solutions in much less time. Over the last three decades many approaches and models have been proposed for dealing with a variety of timetabling programs. Most of these work employed operational research [2,3], human-machine interaction [4, 5], constraint programming [6-8], expert systems [9-13], or neural networks [14-16], multi-agent system [17-24] to solve the timetabling problem. However, there are still several problems that should be addressed [25]:

-23-
The timetabling problem varies significantly from institution to institution in terms of specific requirements and constraints. The timetabling systems should be easily reformulated or customized to support changes.

The lack of a generalized framework makes the development of a timetabling system slowly and expensively. A generalized methodology which can handle a wide range of timetabling problems is required.

It is hard to capture knowledge and incorporate it into the timetabling system. Expertise helps in reducing the search space and in fitting the solution to the context.

The difference between hard and soft constraints should be addressed. A constraints relaxation technique is needed for obtaining a feasible solution.

To alleviate above problems, we introduce a new solution model for solving timetables generation problem. This method is based on generating a series of possible class-hours patterns that satisfy both hard constraints (Essential) and soft constraints (Preferential) for each section/course, and successively filters these possible patterns as the timetable is being built. This method highly reduces the search space of possible timetable configurations and delivers “almost” complete and optimal solution. The introduced method concentrates on realizing the following features:

1. Rapid, automatic generation of timetables.
2. Minimum gabs between classes.
3. Maximum satisfaction of staff preferences.
4. Complete satisfaction of university requirements.

2. Related Work

A wide variety of approaches to the timetabling problem have been proposed, in recent years, a number of AI technologies, including constraint programming, expert systems, and neural networks, are used for solving the timetabling problems.

Constraint programming approaches formulate the timetabling problem as a constraint satisfaction model and then propagate constraints to obtain a feasible solution. Deris et al. [6] propose a constraint-based reasoning algorithm to model and solve the timetabling problem. The proposed system is implemented in an object-oriented approach and can thus be easily adapted to support changes. In [7], OR models and local search techniques are also used to assist the constraint programming search process by effectively reducing the solution search space. They propose a minimum cost matching algorithm to relax the constraint satisfaction model. Constraint logic programming [8] integrates logic programming and constraint solving to tackle combinatorial problems such as planning, scheduling, and resource allocation. This combination helps make constraint logic programs expressive and flexible.

Expert systems capture human expertise and use it in the solution process. Since the model employed more closely resembles the real world situation, its solutions could also better fit the context. Gunadhi and his colleague [9] introduce an automated timetabler that combines data model and knowledge base, developed using object-oriented methodology. The separation of data, knowledge, and algorithms provide flexibility to deal with changes, and the incorporation of human expertise helps in reducing the search space for a feasible
solution. Isaai and his colleague [10] introduce a lookahead, constraint-based algorithm which is designed using an object-oriented approach. In their approach, expert knowledge is used as heuristics for finding practical solutions and combined with the constraint-propagation technique. Solotorevsky et al. [11] develop a rule-based language, called RAPS, for specifying resource allocation problems and timetabling problem. The language enables the specification of a problem in terms of resources, activities, allocation rules, and constraints, and thus provides a convenient knowledge acquisition tool.

In the area of multi agent, several systems were proposed for the development of time table, some of them have the lack of intelligence approach while some are not implemented [12]. Luca Di Gaspero [14] proposes a multi-agent system for searching a local solution and negotiating about resources with other departments and managing relevant information. Nunohiro [15] proposes a multi-agent system consisting of agents representing the requirements and restrictions for professors and departments. He solves the problem using negotiation between agents together with the hill-climbing method. Oprea [16] proposed a multi-agent system for solving Time table, his work focused on analyzing the benefits of using multi-agent approach for the university course timetable scheduling and avoiding conflicts that may arise using negotiation between agents. Nandhini [18] has tried to use the agents approach to solving timetables problem, he presents a Steepest ascent hill climbing algorithm on generates the maximum possible combinations and formulates an evaluation function that is applied on generated timetable configuration.

3. Class Timetabling Problem Description

Through our work, we consider a timetable model which is followed in one semester of a computer science curriculum of Faculty of Computers and Information Technology of Tabuk University as an example. The problem is characterized as follows:

- The curriculum consists of a number of levels.
- Each level has a number of courses.
- Each course is defined as a triple pattern of (Lec, Lab, Tut), where Lec is the theory class hours, Lab is the laboratory class hours, and Tut is the tutorial class hours.
- Time slots of a course are allocated according to a predefined pattern. For example, if the course has 3 lecture hours, and 2 lab hours (3-2-0) then it can be assigned one of the alternative time slots indicated in Figure 1.
- One Course has to be assigned to one staff member, and at most 3 staff members.
- Lab slots and Tut slots are at least 2 contiguous hours in the afternoon, while Lec slots can be scheduled in 1 to 4 distinct time slots in a week based on the credit hours of the curriculum.

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. Sample course-hours patterns
3.1 Hard Constraints

In order to build valid timetable the following constraints should be satisfied:

- One room cannot be allocated more than one class at the same time.
- One staff member cannot be allocated more than one class at the same time.
- Classes of the same level should not be intersected.
- Rooms should be allocated according to the type of the lecture (Lec, Lab, and Tut).
  For example practical lectures should not be conducted in a lecture room.

3.2 Soft Constraints

In order to build an optimal timetable the following are some of the constraints that might be satisfied:

- The maximum number of class hours of a staff member should not exceed 6 hours/day.
- Course lectures should be assigned according to the preferences of the staff member.
- The maximum number of class hours for a student should not exceed 6 hours/day.
- Gaps between lectures should be minimized as possible.

4. Proposed Framework of the Timetabling System

An overview of the system framework is shown in Figure 2. Four main components are presented: User Interface, Data Repository, Application Component, and Problem Solving Component. Data Repository is the subset of the system where static data about courses in each level and their prerequisites (Curriculum), available rooms and labs with their capacities and facilities, and lecturers and the courses taught by each of them. The user interface component acquires the requirements from the end user, manages data repository, and posts the end user requests to the problem solving component. It works as a front end that appears to the end user. The end user provides the sections to be included in the timetable to the user interface and may respond to whatever requirements posted by the user interface.

Figure 2. An overview of the scheduling system
Application Component has two modules; the first one is Pattern Generation Module which is responsible of handling the definition and manipulation of time slot patterns suitable for each course. As specified before, each course is identified by a triple of (Lec, Lab, Tut) hours, which indicates the number of lecture hours, laboratory hours and tutorial hours successively. According to the requirements of Tabuk University, these hours should be distributed on the week days according to certain criteria. It helps the user in generating alternatives for distributing lecture hours, lab hours and tutorial hours for each course, in a manner that conforms to the requirements of the university. It is important to note that patterns are reusable among courses that have the same (Lec, Lab, Tut) signature. The second module is the Preference Selector Module; it provides the opportunity for the staff members to put their preferences for the courses/sections they teach, the staff member selects only the patterns that suit his preferences. The most suitable patterns are selected first to indicate highest priority, and other selected patterns indicate less preferred ones.

Problem Solving Component is the most important component that generates time tables for the courses of a selected level according to the hard constraints and the soft constraints and preferences of each course. It solve the time tabling problem by converting the development process into a number of distributed processes for generating an acceptable timetable that satisfies all the predefined hard constraints and the maximum possible of soft constraints.

In order to achieve the aforementioned goal, a group of agents has been developed according to the BDI model (Believes, Desires, Intentions) [21-23], each of them has its own knowledge about an aspect of the overall system and a set of goals to be accomplished to get the final solution, and a set of plans to be done to fulfill the specified goals. Figure2 presents four agents which are: Facilitator Agent, Curriculum Agent, Staff Agent, and Room Agent. The goal of the Facilitator is to allocate a time table for each proposed section, where a section is a class for a group of students of a certain courses to be introduced in the semester for which the time table is to be developed.

Figure 3. The Sequence Diagram
The Facilitator iteratively tries each time slots pattern for the selected section starting by the most preferred pattern to the least one. Each time it chooses a pattern it sends a message to the Curriculum Agent to make sure that there is no conflict at the level of the curriculum, as it is not acceptable to allocate a time table for two courses at the same level at the same time. The Role of the Curriculum Agent is to make sure that there are no conflicts between the times allocated to sections of courses within the same level in the curriculum. If there is a conflict, the curriculum agent replies to the message sent by the facilitator agent, declaring that the selected time allocation pattern cannot be used, so the facilitator can look for another pattern for the selected section. In case of no conflicts, a message is passed to the Staff Agent, which is responsible of resolving time conflicts for the faculty members, as it is not possible to allocate more than one class to the same faculty at the same time. In such case, the Staff Agent sends back a request to the facilitator to search for another time slots pattern for the current section.

If there were no conflicts, the request for allocation to the Room Agent, which looks for the suitable free room at the assigned time to the current section.

The Room Agent is responsible for assigning a room to the current section according to the reserved time, and according to the requirements of the class (lab., lecture room,…etc.), and the number of students in the class. In case of failure, the Room Agent sends back the request to the facilitator to look for an alternative time slots pattern for the current section. Otherwise, the Room Agent confirms the registration of the selected room and sends the time allocation entry to be saved in the time tables’ database. Figure 3 illustrates the scenario of interaction between these agents. The framework in Figure 2 is constructed using Microsoft Visual Studio 2010 as an object oriented paradigm and tested on the Faculty of Computers and Information Technology at Tabuk University. The user interface that represents the pattern generation and pattern selection depicted in Figures 4 and Figure 5 respectively.

![Figure 4. Pattern Generation](image-url)
5. Results

The proposed course timetabling system has been used two semesters for the Department of Computer Science at the Faculty of Computers and Information Technology at Tabuk University. The department has 50 courses of 62 sections have been scheduled for 8 levels and more than 240 students. In each semester, there are 45-50 subjects to be scheduled into 40 time slots and 8 classrooms. The results of course timetabling are shown in Figures 6 and 7. Figure 6 shows that about 95 percent of courses is scheduled into the preference time periods proposed by the instructors. Figure 7 presents that about 97 percent of the studying hours is scheduled without gaps. Sample of the generated timetable for level 3 and level 5 are presented in Figure 8 and Figure 9 respectively.
Figure 8. Schedule of Level 3

Figure 9. Schedule of Level 5
6. Conclusion

The Timetabling is a common problem that has been handled by many different approaches. In this paper we introduced the “Academic Time Tabling System”, a research project that has been funded by Tabuk University Scientific Research Deanship. The goal of this research project is twofold: Introducing a new solution model that reduces the search space of the problem, and introducing Multi-Agent Systems as an alternative paradigm for solving time tabling problems. According to the requirements of Tabuk University, class hours of a course should be distributed in the week using certain criteria. Augmenting these criteria in the search space of the timetable will move it to an even larger exponential dimension. In spite of dynamically generating possible and acceptable class time slots, we used predefined patterns for class time slots according to the characteristics of each course (theory, lab, and tutorial hours). Using Multi-Agent Systems technique, the time tabling problem has been divided into a set of simpler problems, each of them is handled by a separate agent. It is very easy to handle very complicated situations by augmenting the agent with the knowledge and rules to handle these situations. Another benefit of using Multi-Agents is reusability. Due to the inherent modularity of Multi-Agent Systems, it is easy to use the same agents in solving problems related to resource allocation and management problems in general. Expandability is another issue. It is possible to introduce another agent member that handles other issues like the co-ordination between different colleges that use the same computer labs. The “Academic Time Tabling System” has been applied on a real case at the Faculty of Computers and Information Technology at Tabuk University, where 50 courses of 62 sections have been scheduled for 8 levels. The generated timetables were optimal since all the hard constraints and soft constraints were satisfied. The generated timetables were compacted (without idle holes in the middle of the classes). Finally, the resources of the faculty (Rooms, and Labs) were utilized efficiently with sacrificing minor percent of the preferences of the faculty members and without sacrificing any requirements of the faculty.

7. Acknowledgements

We would like to acknowledge the support of the deanship of scientific research, Tabuk University, KSA for granting this research.

References


