A Multi-Agent System for Course Timetable Generation

Zina Houhamdi [1], Belkacem Athamena [2], Rani Abuzaineddin [1], Mohammad Muhairat [3]

[1] *Software Engineering Department, Al Ain University of Science and Technology, Al Ain, Abu Dhabi, UAE*
[2] *Business Administration Department, Al Ain University of Science and Technology, Al Ain, Abu Dhabi, UAE*
[3] *Software Engineering Department, Alzaytoonah University, Amman, Jordan*

Abstract **– In the university, course scheduling and preparation for each semester can be defined as the process of determining what courses to offer, the number of sections needed for each course, assigning of a faculty member to teach each section, and allocating a timeslot and a classroom for each section to avoid clashes. The output of this activity (which is a timetable) affects every faculty member and student in various departments. This process is essentially broken down into three main stages: determining the courses to be offered as well as their section numbers, assigning faculty members to different sections, and scheduling of the sections into timeslots and classrooms.**

This paper investigates each of these steps and congregates them in a scheduling and Decision Support System (DSS). The DSS is used to make easy the process of course offerings by taking into consideration the students' suggestions because the department resources are limited. The faculty member preferences are also considered in the assignment of sections for the sake of lessening disappointments in the department. Also, the couples (faculty, section) are planned into university timeslots based on faculty member preferences. Our proposed system considers student suggestions and preferences and the time availability of a faculty member since it minimizes disappointments and avoids conflicts between faculty members and classrooms and courses.

DOI: 10.18421/TEM81-30
https://dx.doi.org/10.18421/TEM81-30

Corresponding author: Belkacem Athamena,
Business Administration Department, Al Ain University of Science and Technology, Al Ain, Abu Dhabi, UAE
Email: athamena@gmail.com

Received: 22 January 2019.
Accepted: 21 February 2019.
Published: 27 February 2019.

Keywords **– Multi-agents, Scheduling problem, Course timetable.**

1. Introduction

Nowadays, creating a functional schedule is exceptionally important. Scheduling is a commitment to enabling a reliable execution of planned tasks for an individual person and entire industrial systems as well. Timetabling is a really hard task that belongs to the larger area of scheduling and can be defined as daily schedules of railroad transport, weekly periodic schedules of flights considering multiple airports, best course schedules in academic institutions, etc. In these problems type the timeslot, representing the interval needed to perform tasks, is known in advance. Therefore, reducing the schedule length is not generally the main goal and however other criteria are proposed to estimate the schedule quality. For instance, in case of academic institutions, the timetable schedule should satisfy the following constraints: reduction of working day length, reduction of the whole number for students and faculty members, maximizing the satisfaction of preferences for both students and faculty members in term of time and classroom location, etc.

Academic timetabling is a special case of timetabling and belongs to NP-complete problems [1]. It involves scheduling a group of resources over restricted timeslots under certain constraints called hard and soft constraints. The staff, students, courses, classrooms, and timeslots represent the resources. According to Burke et al. [2], this problem is formalized regarding four finite sets: resources, timeslots, constraints, and meetings. The main difficulty is how to assign resources and timeslots to the meetings in order to overdue as many constraints as possible.

The solution to timetable scheduling issues is generally time-consuming, iterative and has clashed preferences (this makes the problem of searching an optimal solution harder). To arrive at the solution to this problem, it is essential to find a compromise between all the preferences, generally conflicting.

This study investigates and proposes a solution to a particular case of university course timetable scheduling by adopting the agent technology. Our purpose is to reduce the actual gap by implementing a system that imitates the work of a human planner in reality. The collaboration between agents is considered by proposing protocols required to solve conflicts (like humans do in the real world), and also because all agents search intended for solutions, the collaboration can reduce each other's problems. Thus, in this paper, we will investigate two research issues: the application of the agent technology for the allocation of the resources in order to satisfy the hard constraints needed to serve the teaching tasks and address the soft constraints that correspond to real-world requirements.

2. Background

The agent technology presents appropriate mechanisms to address the timetable scheduling problems [3],[4],[5]. Since the intelligent agent is able to perform a task autonomously, it is qualified to do the work of its representative in order to reach its goals. As mentioned in [6], by relying on agents' features, a multi-agent system can be developed to work out the conflicts between assumptions and constraints in the timetabling problem were set of agents to negotiate and collaborate with each other. Moreover, Carter [3] said that the distributed software architecture describes appropriately the timetable scheduling problem. He agreed that the agent technology supports the distributed timetabling situation, where the distribution arises from the existence of different and autonomous agents which don't interact in every detail, although need to communicate more rude grained information. The solution comes from the gathered information after an agreement between separated agents. Therefore, the multi-agent model provides remarkable and robust characteristics that deal with real-world requirements and challenged circumstances in the timetable scheduling problem domain.

Multi-Agents Systems (MASs) focus on the behavior coordination between a set of autonomous software agents (e.g., intelligent agents) that operate in a context. Sometimes, a software agent is conceived to adapt its private concerns with the other agents' constraints. The MAS complexity is usually bigger than the traditional software systems, but the MAS success is a consequence of well-designed and properly-tested subsystems. Furthermore, in a specific situation of timetable scheduling, the MAS is able to obtain an optimal or a sub-optimal solution by taking advantage of the collaboration between agents (with a minimum number of passing messages).

3. Academic Timetabling

The academic timetable schedule includes all tasks required to create a timetable in an academic institution. It fits in the NP-complete problem class. Consequently, it is improbable to find an approach that solves this problem in a polynomial period of time [7]. Academic timetabling consists of scheduling a list of courses to a particular timeslot and classrooms, under several constraints. A summary of computerized timetabling classifying the academic timetabling into three categories depending on the institution type and the constraint type [8].

- *School Timetabling*: provides a schedule for all courses offered by the school, usually arranged by week, and aims to eliminate situations like assigning two different courses to an instructor at the same timeslot and vice versa.
- *Course Timetabling*: provides a weekly schedule for all lectures of a subset of university courses and aims to eliminate situations like lectures overlap of courses on a specific program.
- *Examination Timetabling*: provides a schedule of exams for a subset of university courses, needs to eliminate situation like exams overlap of courses that are taken by the students in the semester, and aims to distribute the exams over the examination interval as much as possible.

Each timetabling category considers different perspectives. School timetabling focuses on instructor availability and reduction of inactive time for students and/or instructors, course timetabling focuses on finding a timetable without or with few conflicts, in order to enable the students from different program to enroll in multiple courses as they desire while examination timetabling focuses on ensuring that two successive exams for a student must be separated by enough time. The difference between examination scheduling and course schedule is: one classroom should be assigned to a unique lecture at any timeslot, whereas for exams the same classroom is usually occupied by students from different courses, and students in the same course are distributed across classrooms.

This paper investigates university course timetabling. The solution must satisfy several constraints. There are two types of constraints [7]:

- A hard constraint defines a set of constraints that must be fully satisfied. A timetable which satisfies hard constraints is considered as a possible or valid solution.
- A soft constraint defines a set of constraints which don't need to be completely fulfilled. A valid solution which satisfies a subset of soft constraints is called good solution. The

percentage of soft constraint satisfaction is used to differentiate between valid solutions (i.e., the optimal solution is a valid solution which satisfies more soft constraints) [9].

Table 1. shows a list of hard and soft constraints which are related to university course timetabling problems [5], [7].

Table 1: Hard and Soft Constraints to University Course Timetabling Problems

Hard Constraints	Soft Constraints
The timetable is valid for a semester	The instructor may prefer some timeslots
The classrooms number is fixed	The instructor may prefer a specific working day.
Classrooms have limited seats	Avoid gap between the lectures
Timeslot are defined by the university	Avoid to schedule course in the last timeslot
A classroom is assigned to just one lecture at a time	Course may need to be scheduled before/after another course
Limited resources during a specific timeslot	Spread the load smoothly over the week
	The instructor may prefer a specific classroom
Students and instructors can be in one classroom at a specific time	Some students may have an off-day during the week
	Instructors may prefer to teach more than one lecture per day

McCollum [10] has confirmed the existence of a gap between the practical and theoretical aspects of university course timetabling. He explains that the majority of investigations in this domain focus on the evaluation of proposed approaches and methods, and the computerized solution deals only with databases and disregards the human factors. Moreover, Carter has noticed that limited suggested methods for timetabling were implemented and used in real institutions [3]. The reason is that there were a few investigations of real-world issues and insufficient understanding of human approaches. Moreover, institutions have different sets of constraints during timetable generation, thus, determining a general relevant solution to this complicated problem, is really hard.

Accordingly, the previous studies can be considered as an initial attempt to solve the timetabling problem by providing strong search methods. Additional effort is required to investigate the problem formulation and modeling [11]. Computerized timetabling frequently strive to deal with, not just, increasing student numbers, but also with a set of events that should be incorporated in one component (such as lectures, tutorials, and practical and/or laboratory classes). Furthermore, getting the optimal solution is given up in order to obtain a valid solution. Table 2. shows a set of the problems of timetabling and the proposed solutions.

Table 2: Timetabling Problems and Solutions

Problems	Proposed Solutions
The increase of students number	Relevant human coordination
Event series in one module	Fuzzy algorithms, Meta-Heuristics, Events classification
Optimization goal	Optimization techniques and high-level heuristics
Timetabling of dependent events	No current solution

McCollum [10] singles out a list of relevant problems related to university course timetabling which need more investigation:

- Production of timetables within an institution that is equitable and fair to all concerned members.
- The balance between the section sizes when proposing maximum choices to the students.
- Generation of optimal solution that keeps a balanced satisfaction of all the stakeholders (students' choice, Instructor preferences, and classroom usage).

4. Multi-Agent System for Timetable Generation

This section starts by giving a brief description of the university structure. Then it presents the suggested architecture of the course scheduling by using a MAS. First, we describe our proposed system architecture; then we define the protocol to manage the communication between agents and finally we describe in depth the generation of the timetable by a collaboration of three agents.

4.1. University Organization

Since our goal is to mimic the real-world timetabling process; this subsection presents a university structure. The university defines the hierarchical organization as follows:

- *University Level*: a university is composed of a set of colleges.
- *College Level*: a college is composed of a set of departments.
- *Department Level*: department offers one or more programs.
- *Program Level*: a program has a study plan which is composed of a set of courses.
- *Course Level*: a course is composed of one or more sections. It is taught by an instructor who is specialized in the course. Usually, the course and the instructor belong to the same department.
- A *Shared Course* is a course that is common to various programs.

The course timetabling involves a set of resources: students, instructors, classrooms, courses, and timeslots. Every resource type belongs to one particular level of a university's structure.

- *Student*: student belongs to one particular program of the specific department.
- *Instructor*: instructor is a faculty member of one particular department.
- *Classroom*: classroom belongs to the college, which manages the classrooms directly. A classroom is represented by its name and its maximum seat number.
- *Course*: a course belongs to one program of a specific department, which takes in charge the scheduling of the course by assigning an instructor and setting up the course timetable. The same scenario for the shared courses, but the other departments must avoid conflicts with the shared timeslots during the organization of their owned timetables.
- *Timeslot*: usually, each working day is split into a set of timeslots. A period is defined as a couple of (day, timeslot).

There are three different agents in MATTS. In literature, we found diverse existing systems designed to develop a timetable. However, some of them are desktop systems and some lack intelligence approaches, while some are not implemented. Fang [12] proposes a UML based course scheduling system that has many limitations such are: it is not an intelligent system, and it is a desktop system. These same limitations are found in the system presented by Takegami [13]. This system uses spread-sheets and was developed for commercial organizations. Oprea [14] suggests a MAS system for university timetable scheduling by presenting its framework and showing coordination protocol through several diagrams without validating his proposed system. We propose Multi-Agent Time Table System (MATTS) which is a web-based system, to assist managerial people such as Head of Department (HoD) or directors of academic college in working out an infallible timetable with no face-to-face meeting with concerned people. MATTS overcomes all previously mentioned problems. Particularly, MATTS is intelligent, and it is a web-based system.

4.2. System Description

Software architecture shows the relationship between system components. The main objective of the software architecture is reduction of the software development cost and establishment of an agreement between the developer and the system customer [15]. First, this section presents system architecture and then it gives the details of three-tier architecture.

a) *System Architecture*: Figure 1. depicts the suggested system architecture.

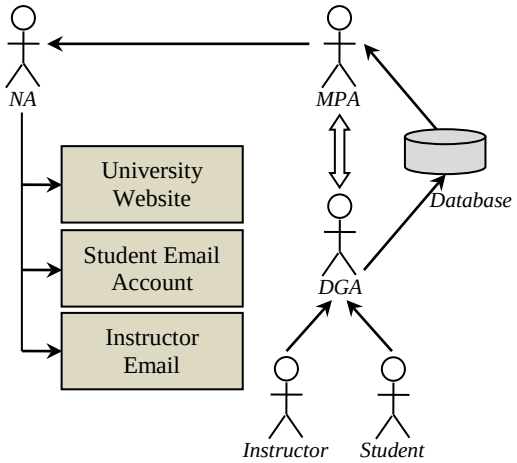

Figure 1. System Architecture

The MAS contains three agents and a database:

- *Data Gathering Agent* (DGA): This agent gathers data from instructors (including Head of Departments (HoD)) and saves it in a database. In fact, the context changes when the DGA stores user data in the database. The DGA communicates and collaborates with MPA and sometimes may save data in a database on the MPA request. Also, the DGA can call MPA (upon an instructor or HoD request) to execute a particular task such as report generation and information announcement. For instance, generation of a report about who is teaching the same course. DGA then forwards the request to MPA which collects needed data from the database; puts it in a report shape and at the end call NA to send the report to the HoD.
- *Notification Agent* (NA): This agent sends emails (e.g., notification, announcement, alert, etc.) to instructors, students, and HoD and publishes information on the website. He performs this job on the request of MPA.
- *Managing and Processing Agent* (MPA): This agent represents the heart of the system. It executes all important activities of the system. MPA supervises its context continuously and reacts immediately by taking suitable action if the context changes. Context changes in case the database are modified by DGA. Thus, MPA reacts in two different situations: the first situation is related to the reception of a message directly from an instructor or HoD requesting a specific task execution. The second situation relates to the database (context) modification.
- *Database*: Actually, this database represents the agents' context. It contains the data essential to MPA in order to execute different tasks. Any modification in this database forces MPA to react

by performing a pertinent action. The database contains data grouped into different tables with appropriate relationships. Figure 2. shows the Entity Relationship Diagram of the database. MPA observes this database around the clock.

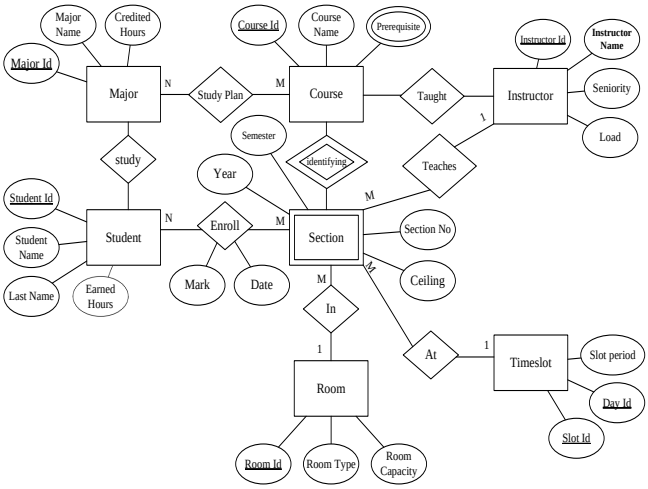

Figure 2. The Entity Relationship Diagram

b) *Three-Tier Architecture*: Figure 3. presents the three-tier architecture of our suggested system. Because NA publishes data to the user, it is located in the presentation layer. The application processing layer contains MPA and DGA. As illustrated in Figure 3., DGA collects data from the presentation layer and preserves it in the database. Also, DGA cooperates with MPA to generate reports requested by the HoD. MPA acquires data from the data management layer, converts it to the specific form and sends to NA to post it on the website.

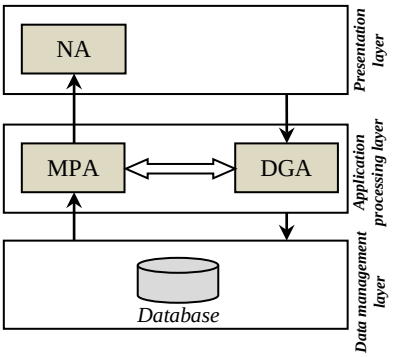

Figure 3. Three Tier Architecture

c) *Agents communication*: Table 3. shows KQML communication between MPA and DGA agents. This communication is established when DGA performs a task for HoD. In case the HoD asks the DGA to inform the faculty members who want to teach a specific subject for example "Artificial Intelligence". DGA uses *ask-one* KQML to send a query to MPA which uses *tell* KQML to reply that Dr. Zina wants to teach "Artificial Intelligence" (see Figure 4.).

Table 3: KQML dialogue between DGA and MPA

(*ask-one* :sender DGA :content "course-owner (Artificial Intelligence, Course instructor)" :receiver MPA :reply_with instructor-query :language Prolog :ontology MATTS)	(*tell* :sender MPA :content "course-owner (Artificial Intelligence, Dr. Zina)" :receiver DGA ;in_reply_to instructor-query :language Prolog :ontology MATTS)

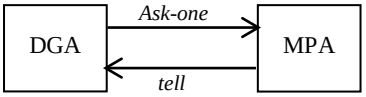

Figure 4. Communication between DGA and MPA

4.3. Timetable Generation Process

The timetable is generated by executing the timetable development algorithm. The proposed algorithm can be decomposed into two main steps. These steps are related to DGA and MPA.

Step1: Identification of possible courses to be offered. The Agent MPA will access the registration database and identify the list of possible courses to be offered in the upcoming semester. The identification of courses is done by assigning a weight for each course in the study plan. The weight of the course is determined by the number of students who have taken the course prerequisite(s), but still did not take the course itself, and courses with no prerequisite should be considered each semester for newly registered students. MPA compiles this information and obtains the list L of possible courses to be offered.

Agent MPA also computes the maximum number of courses to offer (*Max*) by getting the total instructors load and comparing it to the number of courses in the list L where the weight of the course is equal or more than the required number of students in a section N. In our prototype, the minimum number of students in a section to be opened is five. If a course has a weight less than thirty, then one section will be opened, and if the weight exceeds thirty, then two sections would be opened of that course.

If the instructor number is sufficient to cover all courses in the list $L(Max \geq length(L))$ then the final list of offered courses is equal to all the courses in the list of possible courses. Otherwise $Max < length(L)$ agent MPA will reduce the list L to become equal to *Max* by asking agent NA to send emails for students to provide their suggestions. Each student can select courses from L that he is eligible to register. After the period specified for students to provide feedback is over, the system will determine the final list of offered courses L' by assigning a high

weight for courses that have been chosen by students. If students did not provide sufficient feedback, the system will go back to the previous list *L* and select the remaining courses to fulfil the load of instructors.

Step 2: Assignment of courses for the instructors. Once the list of lectures is developed, agent MPA will ask agent NA to send emails to instructors to provide their preferences about the course(s), classrooms and timeslots. Then DGA saves the instructor's preferences data in the database. Agent DGA will notify highest seniority instructors first to provide their preferences during a specific period of time; once they have provided their feedback or their time is done it will notify the lower seniority instructors and so on. Preferences are the lectures (representing a section of an offered course), its timeslot, day and classroom. From the offered courses, each instructor will be able to select only the courses from his major and courses he taught before and not preferred by a higher seniority instructor. Instructors are not allowed to choose the same timeslots and day for two different lectures. Instructors are obligated to select a specific number of lectures to complete their load.

At each seniority level the MPA will allocate courses chosen by instructors from that seniority, and in case two instructors have chosen the same course the system will allocate it for the instructor who has taught it a number of times, and in case both instructors have taught it the same number of times the system will assign it for the instructor who has joined the university first. Once a course is assigned to an instructor, the instructor will enter the timeslot, day and classroom that he prefers for that lecture. Instructors whose preferred courses have been assigned for another instructor will have the preferred time, day and classroom preferred for that course saved by the DGA and assigned for another course.

The MPA will look for instructors who are still under load (because one of their preferred courses was allocated for a more eligible instructor, or did not provide any feedback) and assign the remaining courses (courses which were not chosen by any instructor) for instructors of the same major. And in case a load of instructors of the same major is full it will be assigned for any instructor in the department.

For instructors that didn't provide any feedback, the MPA will assign time, day and classroom for courses that have been assigned to them in such a way that it does not create clashes with already existing timeslots in the timetable. Now, the MPA has the initial timetable. The MPA will look for clashes and try to resolve them by performing the following steps: at this point, a clash is identified as two courses have the same timeslot and the same day in the same classroom. To maintain instructors' preferred time and day as much as possible the MPA will search for available classrooms in that timeslot and day and assign that classroom for one of the clashed courses. In case there is no available classroom, it will try to change the timeslot in a way that doesn't create more clashes. If there isn't any available timeslot, the system will change the days of that course.

Once all clashes are resolved the timetable is ready, and MPA will ask the NA to publish it on the website. Figure 5. shows the process of timetable development which involves the cooperation and collaboration of three agents DGA, MPA, and NA.

The system sequence diagram of MATTS is shown in Figure 5.

Figure 6. shows the MATTS use case diagram.

5. Results and Discussions

Our system provides an interface for instructors to enter their preferences about the *courses*, *day*, *timeslot*, and *classroom*. This interface, shown in Figure 7., provides a list of offered courses which are related to the instructor's major or have been already taught by the instructor in previous semesters and not selected by instructors with high seniority. This list is labeled as '*course list*'. An Instructor chooses a course which he prefers to teach and moves to the list labeled as '*preferred courses*' by clicking the '*select*' button. An instructor can select between three and five courses in a semester according to his load. '*unselect*' button is used to move back the selected course from the '*preferred course*' list to the list '*course list*'. '*show*' button below the '*course list*' is used to display information about a course which has already been selected by other instructors. The information includes *classroom*, *day*, *time* and *instructor information*. When an instructor decides to select the '*show*' button, he will be asked to select a course that has already been chosen by another instructor, the details of which are displayed in a new window. Furthermore, an instructor can select the preferred time, day and classroom. '*submit*' button requests the DGA to store data in the database.

Our system provides an interface for students to enter their suggestions about the courses to be offered in the upcoming semester. The interface shown in Figure 8. provides a list of courses to be offered for the next semester. For each student, the list labelled as '*course name*', contains the list of courses for which the student is eligible to register (prerequisite requirements are satisfied). The student is allowed to select between three and six courses. The remaining buttons are similar to Figure 7.

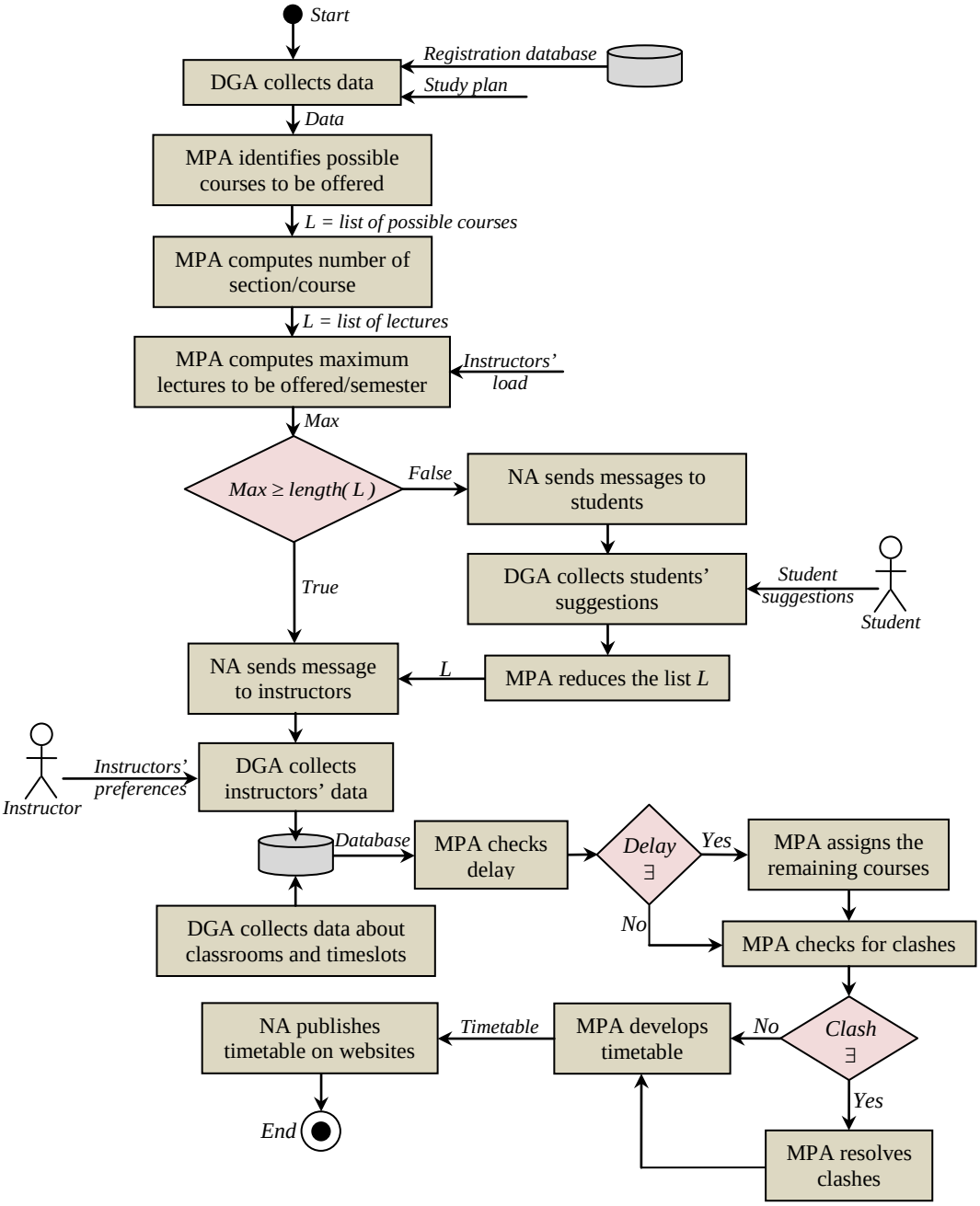

Figure 5. Timetable Development Flowchart

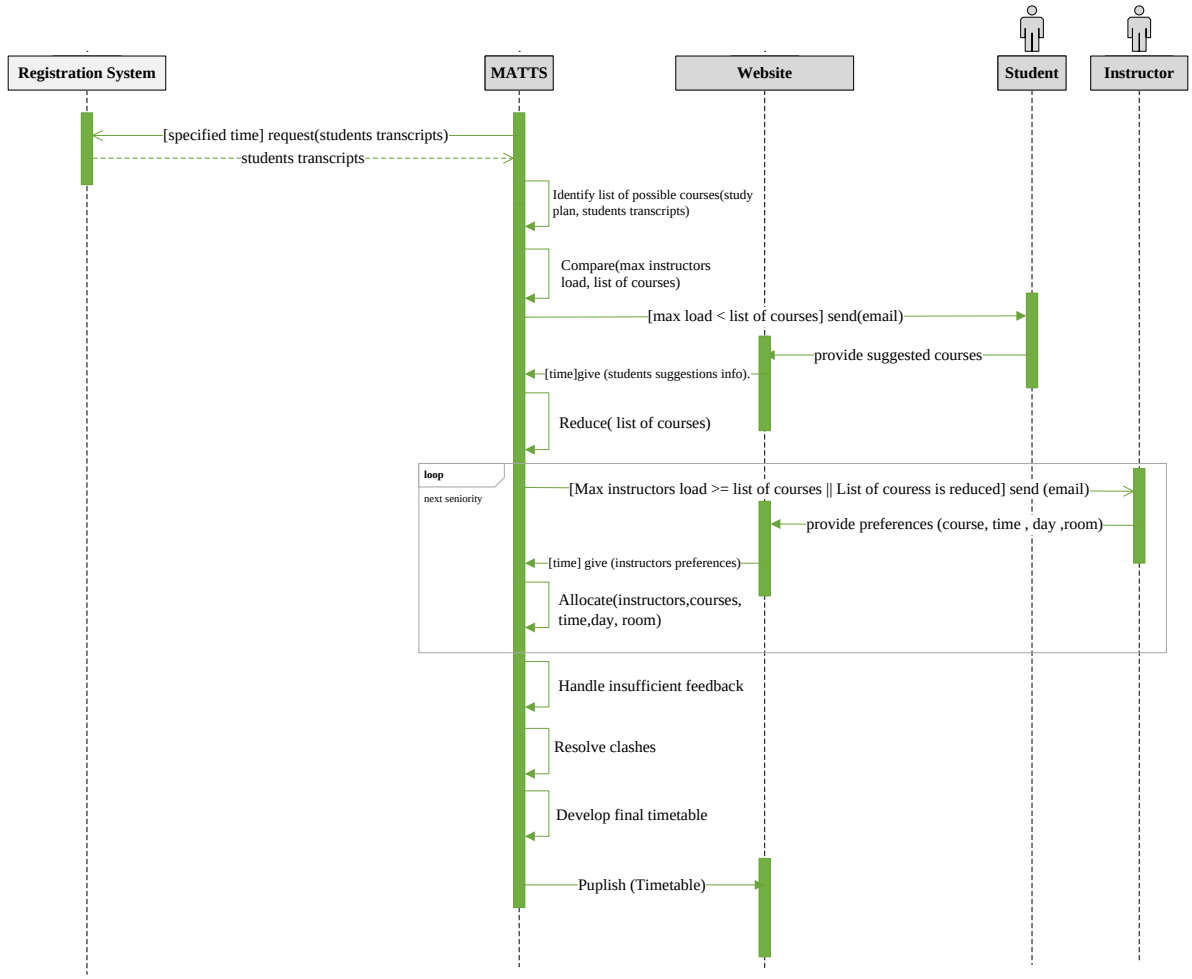

Figure 6. System Sequence Diagram

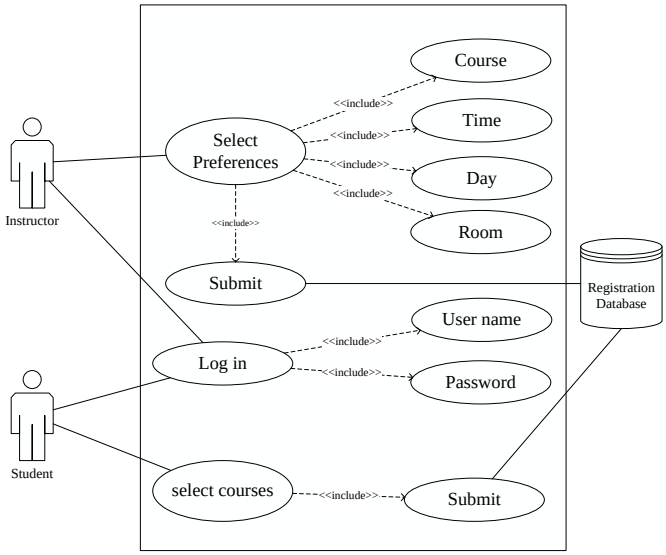

Figure 7. Use Case Diagram

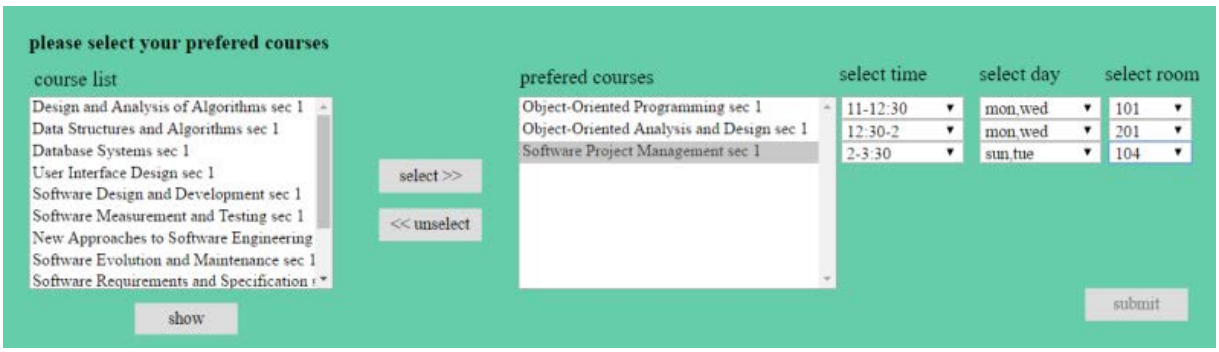

Figure 8. An Interface for Instructor Preferences

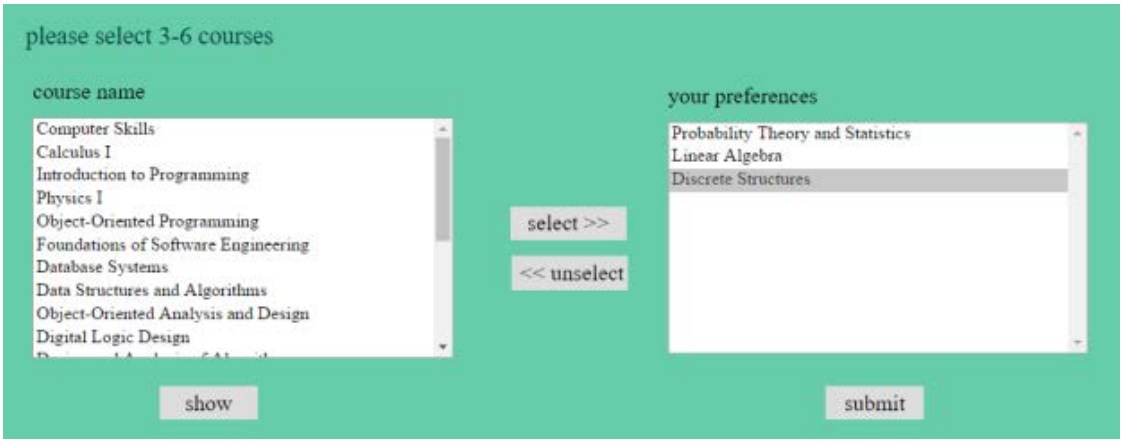

Figure 9. An Interface for Student Suggestions

COURSE_NUMBER	COURSE_NAME	SECTION_NUMBER	INSTRUCTOR_NAME	DAY	TIME	ROOM	ROOM_TYPE
107101	Calculus I	1	maha rahrouh	mon,wed	11-12:30	202	lecture
107101	Calculus I	2	reyaz ahmad	mon,wed	11-12:30	103	lecture
107104	Calculus II	1	talha ahmad	mon,wed	12:30-2	102	lecture
101120	Computer Skills	2	maha rahrouh	mon,wed	2-3:30	103	lecture
101120	Computer Skills	1	reyaz ahmad	mon,wed	2-3:30	202	lecture
103201	Data Structures and Algorithms	1	zina houhamdi	mon,wed	11-12:30	101	lecture
101302	Database Systems	1	ayman oudeh	mon,wed	2-3:30	104	lecture
102305	Design and Analysis of Algorithms	1	mohammad daoud	mon,wed	11-12:30	102	lecture
102203	Digital Logic Design	1	talha ahmad	mon,wed	3:30-5	104	lecture
109206	Discrete Structures	1	reyaz ahmad	mon,wed	12:30-2	202	lecture
109206	Discrete Structures	2	maha rahrouh	mon,wed	12:30-2	103	lecture
103305	Formal Specifications and Design Methods	1	talha ahmad	mon,wed	5-6:30	103	lecture
102405	Introduction to Computer Graphics	1	talha ahmad	mon,wed	2-3:30	201	lecture
103409	Introduction to Distributed Systems	1	ayman oudeh	mon,wed	5-6:30	202	lecture
107102	Linear Algebra	2	maha rahrouh	mon,wed	3:30-5	103	lecture
107102	Linear Algebra	1	reyaz ahmad	mon,wed	3:30-5	202	lecture
103411	New Approaches to Software Engineering	1	ayman oudeh	mon,wed	3:30-5	201	lecture
103407	Object-Oriented Analysis and Design	1	samir mohammad	mon,wed	12:30-2	201	lecture
101202	Object-Oriented Programming	1	samir mohammad	mon,wed	11-12:30	105	lecture

Figure 10. Time Table Developed by Collaboration of DGA, MPA, and NA

DGA is responsible for the storage of all entered data in the database. Once the data is saved in the database, the MPA starts its work. If a delay or a clash in offering a course by any instructor is detected, the MPA attempts to work out the problem. In case the MPA isn't able to find a solution, then it notifies and allows the HoD to solve the problem. In the end, the MPA calls the NA to publish the timetable on the website. A timetable developed in interaction and coordination of the three agents' looks like Figure 9.

6. Similar Works

Several solutions were suggested for approximately fifty years of research in timetabling domain, as example, Genetic algorithms [16], Tabu Search [17], Graph Coloring Algorithms [18], Ant Colony [19], and Constraint Programming [11]. The existence of gaps between theory and practice was noticed especially in the university timetabling. This is a direct consequence of human factor of ignorance in computerized timetabling during the evaluation of methods and approaches. Nevertheless, a useful timetable needs to fulfil the hard constraints required by the academic activities and also the soft constraints reflecting the real world requirements of both instructors and students.

During the last ten years, researchers used the agent technology to solve the university timetabling problem, such as, Meisels and Kaplansky [20] suggested a model with two collaborative agents with different roles that are Room agent and Scheduling Agents. Whereas Oprea [14] proposed four agents working together consistent with the university's organization, which are Instructor Agents, Person Agents, Department Agents and University Agent. Finally, Strnad and Guid [21] designed a model with three types of collaborative agents that are Central Agents, Course Agents, and Scheduling Agent. Each author suggested a different architecture that fulfils distinct constraints at the time of the timetable. Still, some of them take into consideration the hard constraints; only whereas others consider both hard and soft constraints that are related to their perspective.

In this paper, we proposed to use a system which is an extension of MUTSS proposed by Tariq et al. [22] by taking into consideration students suggestions. The system imitates the timetable scheduling process used by a human planner in the real world; each agent makes its decisions according to its own needs under any circumstance at that given moment. Negotiation and collaboration are embodied within a set of protocols designed for solving conflicts, and for collaborating among the agents in order to reach the best solution. The results help to narrow the gap between theory and practice in university course timetabling research.

The datasets which are used in this research are taken from the real world. They have been selected from a large set of interesting cases from a real-world institution (Al Ain University of Science and Technology) in the UAE. The experimental results show that the principles used in our agent model are able to achieve the goal and also satisfy the defined hard and soft constraints. That means the model could be working in a real-world situation.

7. Conclusion

The agent technology is recognized as an effective approach to solving course timetabling problem. McCollum detected the existence of a gap between the theory and practice in university timetabling because the majority of the proposed solutions are interested in the evaluation of specific methods and techniques.

The proposed solution in this investigation concentrated on the timetable generation process from end-to-end by identifying the course offerings, assigning instructors to courses, and scheduling courses to timeslots. At the system level (considering the whole process), our system integrates a set of pieces together and generates an effective timetable. Our approach combines instructors' courses, day and time preferences, and student suggestions to generate an ideal timetable. With the integration of a decision support system, the time required to generate a timetable was extremely decreased. The prototype is tested with a real database, and the generated timetable succeeds to avoid overlaps. Accordingly, the students keep on track with their graduation progress, and consequently, delays due to conflicts are reduced.

As instructor's preferences were considered during the process, the feedback from instructors on the generated timetable was extraordinarily positive (with exceptions). Usually, with the manual process, around 50% of the instructors are unsatisfied with the generated timetable and consequently arduous modifications needed to be implemented otherwise the instructors will conciliate and ask for particular arrangements to adjust the generated timetable. If the assigned courses to instructors fit their enthusiasm and experience and the proposed schedule fits their lifestyle, certainly they will deliver high-quality lectures. Consequently, this strategy encourages the teaching spirit and passion that ensures the success of the education system.

To extend our system, we suggest thinking about a distributed timetable scheduling system by proposing a solution to the overlapping between common courses.

References

[1] Even, S., Itai, A., & Shamir, A. (1975, October). On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)* (pp. 184-193). IEEE.

[2] Burke, E. K., McCollum, B., McMullan, P., & Parkes, A. J. (2008). Multi-objective aspects of the examination timetabling competition track. In *Proceedings of PATAT* (pp. 3119-3126).

[3] Carter, M. W. (2000, August). A comprehensive course timetabling and student scheduling system at the University of Waterloo. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 64-82). Springer, Berlin, Heidelberg.

[4] P. De Causmaecker, P. Demeester, and V. Berghe,(2002).*Using Web Standards for Timetabling*, in *The fourth International Conference on the Practice and Theory of Automated Timetabling, PATAT 2002*, pp. 238–257.

[5] P. De Causmaecker, P. Demeester, V. Berghe, and B. Verbeke.(2004). *Analysis of real-world personnel scheduling problems*, in *The 5th Conference on the Practice and Theory of Automated Timetabling, PATAT 2004*, pp. 183–197.

[6] P. De Causmaecker, P. Demeester, P. Pauw-Waterschroot, and G. De Vanden Berghe,(2000). *Ontology for timetabling*, in *The third International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000*, pp. 481–482.

[7] Burke, E. K., & Newall, J. P. (1999). A multistage evolutionary algorithm for the timetable problem. *IEEE transactions on evolutionary computation*, *3*(1), 63-74.

[8] Schaerf, A. (1999). A survey of automated timetabling. *Artificial intelligence review*, *13*(2), 87-127.

[9] Woods, D., & Trenaman, A. (1999). Simultaneous satisfaction of hard and soft timetable constraints for a university department using evolutionary timetabling. *Department of Computer Science, National University of Ireland, Maynooth, Co. Kildare, Ireland*.

[10] McCollum, B. (2006, August). A perspective on bridging the gap between theory and practice in university timetabling. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 3-23). Springer, Berlin, Heidelberg.

[11] B. McCollum, T. Roche, and P. McMullan,(2007). *Optimizing Space Through Macro and Micro Planning and Scheduling*, in *SCUP–42 , Society of College and University Planners International Conference*.

[12] Fang, S. R. (2005). University Course Scheduling System (UCSS): a UML application with database and visual programming. *Journal of Computing Sciences in Colleges*, *20*(6), 160-169.

[13] T. Takeshi and T. Megumi,(2006). E-learn 2006 : World Conference on E-Learning in Corporate, Government, Healthcare, Higher Education, in *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare*, no. 1, pp. 3048–3052.

[14] Oprea, M. (2007). MAS_UP-UCT: A multi-agent system for university course timetable scheduling. *International Journal of Computers Communications & Control*, *2*(1), 94-102.

[15] Medvidovic, N., Rosenblum, D. S., Redmiles, D. F., & Robbins, J. E. (2002). Modeling software architectures in the Unified Modeling Language. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *11*(1), 2-57.

[16] Kanoh, H., & Sakamoto, Y. (2008). Knowledge-based genetic algorithm for university course timetabling problems. *International Journal of Knowledge-based and Intelligent Engineering Systems*, *12*(4), 283-294.

[17] S. Hooshmand, M. Behshameh, and O. Hamidi, .(2013). A TABU Search Algorithm With Efficient Diversification Strategy for High School Timetabling Problem, *Int. J. Comput. Sci. Inf. Technol.*, 5(4), 21–34.

[18] Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, *176*(1), 177-192.

[19] Thepphakorn, T., & Pongcharoen, P. (2012). Heuristic ordering for ant colony based timetabling tool. *Lecture Notes in Management Science*, *4*, 87-96.

[20] Meisels, A., & Kaplansky, E. (2002, August). Scheduling agents–distributed timetabling problems. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 166-177). Springer, Berlin, Heidelberg.

[21] Strnad, D., & Guid, N. (2007). A multi-agent system for university course timetabling. *Applied Artificial Intelligence*, *21*(2), 137-153.

[22] Tariq, M., Mirza, M., & Akbar, R. (2010). Multi-agent Based University Time Table Scheduling System (MUTSS). *Inter. J. of Multidisciplinary Sci. and Engg*, *1*(1), 33-39.