AgentTime: A Distributed Multi-agent Software System for University's Timetabling

Eduard Babkin[1,2], Habib Adbulrab[1], Tatiana Babkina[2]

[1]LITIS laboratory, INSA de Rouen. Rouen, France.
[2]State University — Higher School of Economics. Nizhniy Novgorod, Russia.
babkin@hse.nnov.ru, abdulrab@insa-rouen.fr, bts@sandy.ru

Abstract. In the course of researching distributed timetabling problems this article applies the multi-agent paradigm of computations and presents a correspondent mathematical model for university's timetabling problem solution. The model takes into account dynamic nature of this problem and individual preferences of different remote users for time and location of classes. In the framework of that model authors propose an original problem-oriented algorithm of multi-agent communication. Developed algorithm is used as a foundation for the distributed software system AgentTime. Based on multi-agent JADE platform AgentTime provides friendly graphical interface for online design of time tables for universities.

Keywords: timetabling, multi-agent algorithms, distributed systems.

1. Introduction

In modern society time scheduling plays the outstanding role. Any schedule is the obligation, which enables to carry out authentic planning of activities for a separate person, and whole industrial systems as well. Timetabling represents an important research activity in the scheduling theory, and focuses on such problems as optimal lecture's schedules in educational institutions, week cyclic schedules of plane's flights within the framework of several airports, week or daily schedules of railway transportation, etc. For all these problems the interval of time, inside which the given set of jobs should be fulfilled, is known beforehand. Thus, the minimum of the schedule's length is not usually considered as a primary goal, - other criteria are used for estimation of quality of the schedule having been built. For example, in an educational institution the timetable design process should achieve the following goals: minimization of maximal length of a working day, minimization of the number of the "holes" in the schedules of groups and professors, maximal satisfaction of personal professor's preferences to the time and location of classes, etc. In the current situation, when many educational institutions rapidly grow in size, and distribution scale, wide application of effective software systems for distributed solution of timetabling problems becomes very important.

At present there are many various algorithms for drawing up the time tables in universities. The fundamental approaches are based on the well-known linear and integer programming paradigms [2, 3, 4, 5, 6]. However, several researches of 80s have shown that the integer programming is not equally effective from the point of view of the calculations volume. The high computational costs make integer programming poorly attractive to the large tasks of timetable design, because that method does not guarantee productivity, when the sizes and complexity of tasks grow [3, 7].

Last twenty years have shown the increased interest of the researchers to development of the approaches for the design of timetables with use of various metaheuristics [3], like simulated annealing, Tabu Search, genetic algorithms (GAs), and their hybrids [8, 9, 10, 11, 12, 13, 14, 19]. It is affirmed, that among others, GAs have larger capacity, and allow to find the greatest number of the feasible solutions [15, 16, 17]. Nevertheless, when GAs are exploited, there are difficulties in the description of controlling parameters, in definition of exact roles of crossover and mutations, as well as in analysis of convergence [18].

Also, it should be noted, that the majority of the considered approaches follow the paradigm of centralized systems, they do not allow the remote users govern the process of timetable design. In complex distributed and evolving systems like modern virtual universities and peer-to-peer communities, that shortcoming makes impractical classical methods, and demands new timetabling principles, which take account of real-time user's preferences in complex changing environments. A multi-agent approach represents a successful paradigm for those kinds of problems, when an optimal or quasi-optimal solution is built in the result of interaction of large number of autonomous computational entities.

In general, for timetabling applications several types of multi-agent algorithms are suitable. The first type of algorithms includes economics-based models of interaction [1, 20, 21]. The second one consists of various generic algorithms for solution of Distributed Constraint Satisfaction Problems (DCSP) [22, 23]. But the most effective algorithms, comprising the third type, were specifically designed for a particular scheduling problem. Such specialized algorithms apply all domain- or problem-specific information and show unbeaten productivity. The results are known [24, 25, 26, 27], where authors propose problem-specific algorithms of agent's interaction for the meeting scheduling. Although such algorithms fit well the timetabling model, and have attractive computational efficiency, their direct application for university's timetabling is not so straightforward and requires additional efforts.

In the given article authors propose new multi-agent algorithm for university's timetabling, and describe basic principles of the distributed system AgentTime, based on that algorithm. Presentation of the results has the following structure. Section 2 describes the mathematical model of university's timetabling, which includes user's preferences. Section 3 gives overview of the corresponding multi-agent algorithm for the design of timetables. In section 4 certain topics of AgentTime's software implementation are considered. Overview of results and discussion are presented in Section 5. Section 6 contains references.

2. The Proposed Mathematical Model For the University's Timetabling Problem

The exact mathematical statement of the university's timetabling problem forms the basis of our own multi-agent algorithm for design of the educational schedule. For the sake of generality we use the term 'teacher' to denote different kinds of university employees (e.g. professors, instructors, etc), the term 'stream' to denote a stream, and the term 'subject' to denote different kinds of student's subjects. Also we give the same name of user to all of the stakeholders of the schedule (e.g. the teachers and student's groups). In our mathematical model we will use the following designations.

Student's groups and Streams. $g \in \mathbf{G}$ – the unique identifier of the group. $\mathbf{G}$ – the set of group's identifiers. $|\mathbf{G}| = \gamma$ – the total number of groups. Each group belongs to one stream at least. Some streams can consist of a single group, but in most cases several groups form a stream with the following constraints:

1. All groups of the same stream exploit the same classrooms for lectures.
2. Lectures are delivered to all groups of the stream at the same time.
3. Each stream has as minimum one lesson.

$\mathbf{R}$ – the set of stream's identifiers. $|\mathbf{R}| = \rho$ – the total number of streams. $r \in \mathbf{R}$ – the unique identifier of the stream. Each single group can be treated as a separate stream, thus $\rho \geq \gamma$. $\mathbf{C_r} \subset \mathbf{G}$ – the stream. $\mathbf{C} = \{\mathbf{C_1}, \mathbf{C_2}, ..., \mathbf{C_\rho}\}$ – the set of streams.

Teachers. $\mathbf{P}$ – the set of unique teacher's identifiers. $p \in \mathbf{P}$ – the unique teacher's identifier.

Timetable users. Union of the group's set and the teacher's set gives us the set of the timetable users: $\mathbf{M} = \mathbf{G} \cup \mathbf{P}$, $m \in \mathbf{M}$ – the unique identifier of the timetable user.

Time. $\mathbf{W}$ – the set of the days of the week. $w \in \mathbf{W_g}$ – the certain day of the week. $\mathbf{W_g} \subset \mathbf{W} = \{1, 2, ..., 7\}$ – the set of learning days for the group g, $j \in \mathbf{J} = \{1, 2, ..., 8\}$ – the lesson's number. $\mathbf{T} = \{(w, j) \mid w \in \mathbf{W}, j \in \mathbf{J}\}$ – the set of timeslots, which are the elementary units in the timetabling problem. For example, the timeslot (1, 2) means the second lesson on Monday. For each timetable user *m* the set of free timeslots $\mathbf{T}_m^+ \subset \mathbf{T}$ is known. The set of denied timeslots $\mathbf{T}_m^- \subset \mathbf{T}$ is known also. We assume the obvious constraints are true (i.e. $\mathbf{T}_m^+ \cup \mathbf{T}_m^- = \mathbf{T}; \mathbf{T}_m^+ \cap \mathbf{T}_m^- = \varnothing$).

Subjects. In our model teachers conduct lectures and manage practical exercises. Lectures are delivered to the whole stream, while practical exercises are organized for a single group only. Also some practical exercises impose restrictions on allowable classrooms, like computer or chemistry labs. To describe all these pecularities, let's introduce the following mathematical structures.

$\mathbf{S_r} = \{1, 2, ..., \sigma_r\}$ – the set of lecture's identifiers delivered to the stream r. $s_r \in \mathbf{S_r}$ – the unique lecture's identifier; $\mathbf{Q_r} = \{1, 2, ..., \theta_r\}$ – the set of practical exercise's identifiers organized to the stream r. $q_r \in \mathbf{Q_r}$ – the unique exercise's identifier;

Each lecture's assignment can be uniquely identified by a pair $(r, s_r) \in \mathbf{RS}$, where

$$\mathbf{RS} = \{(r, s_r) \mid r \in \mathbf{R}, s_r \in \mathbf{S_r}\} \quad \textbf{(1)}$$

The total number of lecture's assignments is computed as follows:

$$|\mathbf{RS}| = \sum_{r=1}^{\rho} \sigma_r \quad \textbf{(2)}$$

Each certain exercise's assignment can be uniquely identified by a triple $(r, g_r, q_r) \in \mathbf{RQG}$, where

$$\mathbf{RQG} = \{(r, g_r, q_r) \mid r \in \mathbf{R}, q_r \in \mathbf{Q_r}, g_r \in \mathbf{C_r}\} \quad \textbf{(3)}$$

The total number of exercise's assignments is computed as follows:

$|\mathbf{RQG}| = \sum_{r=1}^{\rho} |\mathbf{C_r}| \cdot \theta_r$, where $|\mathbf{C_r}|$ – is the total number of groups in the stream $\mathbf{C_r}$.

For further analysis differences between lectures and practical exercises can be neglected and the united set of subjects **E** will be used:

$$\mathbf{E} = \mathbf{RS} \cup \mathbf{RQG} \quad \textbf{(4)}$$

Curriculum consists of subjects' assignments for each of the teacher during one semester (fall) in the following form:

$$\delta : \mathbf{E} \to \mathbf{P} \quad \textbf{(5)}$$

$$\delta(e) = \begin{cases} \delta_1(e), e \in \mathbf{RS} \\ \delta_2(e), e \in \mathbf{RQG} \end{cases}$$

$\delta_1 : \mathbf{RS} \to \mathbf{P}$, where $\mathbf{P}$ – the set of teachers; $\mathbf{RS}$ – the set of lecture's

assignments.

$\delta_2 : \mathbf{RQG} \to \mathbf{P}$, $\mathbf{RQG}$ – the set of exercise's assignments.

For example, $\delta_1(1,2) = 4$ means, that teacher 4 delivers lecture 2 for stream 1, and $\delta_2(1,2,4) = 7$ means, that teacher 7 manages practical exercise 2 for group 4, included into the stream 1.

Given the curriculum δ, we can easily compute the total number of subjects $\mathbf{E}_m$ assigned to the teacher (or the group) with identifier m:

$$\mathbf{E}_m = \{e \mid m \in \mathbf{P} \wedge \delta(e) = m\} \cup \{e = (r,s) \mid m \in \mathbf{C}_r \wedge s \in \mathbf{S}_r\} \cup \{e = (r,q,m) \mid m \in \mathbf{C}_r \wedge q \in \mathbf{Q}_r\} \quad \textbf{(6)}$$

Room's stock consists of laboratories, lecture halls and classrooms available for subjects in the university. It is modeled by the set **A** of unique room' identifiers. For each element of the set of subjects **E,** a subset of permitted rooms $\mathbf{A}_e$ is selected : $\mathbf{A}_e \subset \mathbf{A}$.

The primary goal of the timetabling problem in our model is formulated as looking for the feasible mapping from the set of subjects **E** to the set of timeslots **T**:

$$\tau : \mathbf{E} \to \mathbf{T} \quad \textbf{(7)}$$

For example, mapping $\tau(1,2) = (4,4)$ means that subject 2 for stream 1 will be given on Thursday during the fourth lesson.

Related with the mapping τ, the mapping α should assign a classroom for each subject:

$$\alpha : \mathbf{E} \to \mathbf{A}, \text{ where} \quad \textbf{(8)}$$

$\mathbf{E}$ – the set of subjects; $\mathbf{A}$ – the set of classrooms.

For example, mapping $\alpha(1,2) = 101$ means that subject 2 for stream 1 will be conducted in the room 101.

Constraints for the university's timetabling problem are defined as follows.

1. The teacher can conduct only one subject at the single timeslot.

$$\forall p \in \mathbf{P}, \forall e_1, e_2 \in \mathbf{E} : e_1 \neq e_2 \wedge \delta(e_1) = \delta(e_2) = p \Rightarrow \tau(e_1) \neq \tau(e_2) \quad \textbf{(9)}$$

2. In one classroom only one subject can be given at the single timeslot.

$$\forall a \in \mathbf{A}, \forall e_1, e_2 \in \mathbf{E} : e_1 \neq e_2 \wedge \alpha(e_1) = \alpha(e_2) = a \Rightarrow \tau(e_1) \neq \tau(e_2) \quad \textbf{(10)}$$

3. Each group has no more than one subject at the single timeslot.

$$\forall g \in \mathbf{G} : \left(e_1 = (r_1, ...), e_2 = (r_2, ...) \in \mathbf{E} \wedge g \in \mathbf{Cr_1} \wedge g \in \mathbf{Cr_2} \wedge e_1 \neq e_2\right) \vee \\ \vee \left(e_1 = (..., g), e_2 = (..., g) \in \mathbf{E} \wedge e_1 \neq e_2\right) \Rightarrow \tau(e_1) \neq \tau(e_2) \quad \textbf{(11)}$$

Subject's priority. It is obvious, not all subjects have identical importance within the framework of educational process. As such, it is necessary to set the priority order among different subjects, so subjects with higher priority will borrow the best time and location. In our model the priority is modeled as the partial order on the set of subjects **E**:

$e_1 \succ e_2 \Leftrightarrow U(e_1) \geq U(e_2)$, where **(12)**

$e_1, e_2 \in \mathbf{E}$; $U(e) = |\mathbf{M}_e| + k_1(e) + k_2(e) + k_3(p_e)$ – the "utility" of the subject e;

$\mathbf{M}_e = \{m \mid \left(e = (r, s) \in \mathbf{RS} \wedge m \in \mathbf{C}_r\right) \vee \left(e = (r, q, m) \in \mathbf{RQG}\right)\}$ – the total number of groups for those the subject *e* is given;

$k_1(e) \in \{0, 5, 10\}$ – the measure of subject's importance for the stream (0 – optional, 5 – important in general, 10 – important for stream);

$k_2(e) \in \{0, 2\}$ – 0 – undergraduate, 2 – graduate;

$p_e = p, \delta(e) = p$ – the teacher's identifier;

$k_3(p_e) \in \{0..5\}$ – the estimation of the novelty level of the material given by the teacher p_e.

User's Preferences comprise the important part of our model. Each preference is modeled by a numeric value from the range [0,1]. Value 0 corresponds to the least desired alternative, and value 1 corresponds to the most desired alternative. The model includes two kinds of the user's preferences:

- preferences of the user $m \in \mathbf{M}$ for the time of subjects:

$$f_m^1 : \mathbf{E}_m \times \mathbf{T}_m^+ \rightarrow [0,1] \quad \textbf{(13)}$$

- preferences of the user $m \in \mathbf{M}$ for the location of subjects:

$$f_m^2 : \mathbf{EA}_m \rightarrow [0,1], \text{ where} \quad \textbf{(14)}$$

$\mathbf{EA}_m = \{(e, a) \mid a \in \mathbf{A}_e \wedge e \in \mathbf{E}_m\}$ – the set of feasible pairs "subject-classroom".

We use an evident representation of the user's preferences in the form of graphics tables (table 1, 2). The darker color denotes the more preferable alternative (in respect of time or location).

Table 1. Preferences of the user *m* for the desirable time of subjects, f_m^1,

		Lesson's number, *j*							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Days of the week, *w*	**Mon**								
	Tue								
	Wed								
	Thu								
	Fri								
	Sat								
	Sun								

Table 2. Preferences of the user *m* for the desirable location of subjects, f_m^2

Classroom, *a*	1	2	3	4	5
f_m^2					

Criterion of timetable quality generalizes several partial criteria, and evaluates the solution found, namely the pair of mappings $\tau(e), \alpha(e)$. The first partial criterion evaluates the sum of the user's preferences for the time of the subject *e*:

$$F_e^1(\tau) = \sum_{m \in \mathbf{M}_e} f_m^1\left(e, \tau(e)\right) \to \max \qquad \textbf{(15)}$$

The second partial criterion evaluates the sum of the user's preferences for the location of the subject *e*:

$$F_e^2(\alpha) = \sum_{m \in \mathbf{M}_e} f_m^2\left(e, \alpha(e)\right) \to \max \qquad \textbf{(16)}$$

The generalized criterion is constructed as follows:

$$F(\tau, \alpha) = \sum_{e \in \mathbf{E}} \left(F_e^1 + F_e^2\right) \to \max \qquad \textbf{(17)}$$

The solution of the described problem consists of the found mappings τ, α, assuming that all constraints are satisfied, and the generalized criterion has a maximum value.

3. The Multi-Agent Algorithm For Timetable Design

We took for the basis of our algorithm the well-known multi-agent algorithm MSRAC for meetings scheduling by A. Ben Hassine et all. [27]. Although some correspondences still remain, our algorithm is specifically designed for a quite different problem of university's timetable design, and together with time schedule it gives also an occupancy schedule for classrooms.

In our algorithm we recognize two roles of agents: agents-organizers and agents-participants. The agent's structure also mimics the application domain, so we classify all agents as teachers, groups and classrooms. Agents-teachers play the role of organizers; agents-groups and agents-rooms play the role of participants. The numbers of agents-teachers and agents-groups correspond to the real numbers of the teachers and the groups in the university. One agent-room corresponds to all classrooms in the context of the single time table. Collective search for the best time and location of the study involves communication between different agents. For each study the agent-teacher performs a set of actions, comprising the following state diagram (fig. 1).

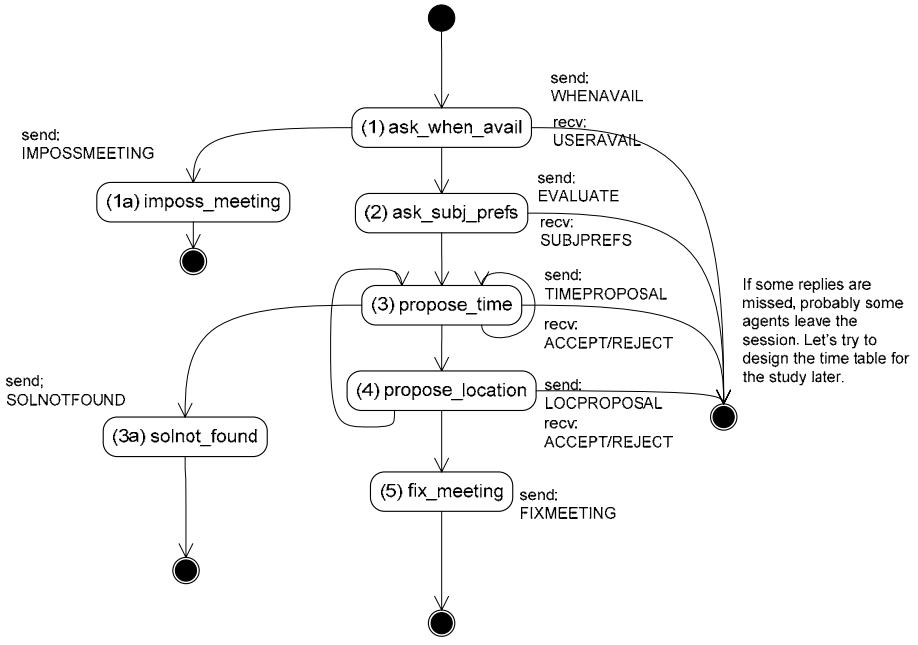


Fig. 1. The state diagram of the agent-teacher's algorithm.

The agent-teacher performs state transitions in accordance with the description given below.

1) *Ask_when_avail.* That is the first state in the algorithm. The agent-teacher sends to all agents of groups the query WHENAVAIL with the study's identifier, requesting available time for that study. The agents of groups answer by the message USERAVAIL, in which they inform when the agent is free, and has available time for the study. If all agents have informed the answer, then the agent-teacher finds intersection on time. If the intersection is empty, then the agent comes the final state imposs_meeting.

1a) *Imposs_meeting*. In that state the agent-teacher founds itself if intersection of available for other agents times is empty and the total solution was failed. The study is marked as "having no solution".

2) *Ask_subj_prefs*. In that state the agent-teacher requests preferences for time and location (the message EVALUATE). The agents of groups reply own preferences in the message SUBJPREFS. The agent-teacher sorts received preferences for time and for location in accordance with criteria (15) and (16).

3) *Propose_time.* The agent-teacher selects the first timeslot from the ordered list of the preferences, and sends it along with the study's identifier to the agents of groups inside the message TIMEPROPOSAL. In response the agents of groups analyze own agendas. If the proposed timeslot is free in the agent's agenda, the agent gives the positive answer, sending the message ACCEPT. Else the agent compares the priority of the study in the agenda with the priority of the study in the message. If the priority of the message's study is greater, then the agent accepts new proposal and sends the message ACCEPT. In the opposite case the agent sends the message REJECT. The agent can apply the metropolis criterion [27] for decision making when the priorities are equal. In the case of total acceptance of the proposed timeslot, the agent-teacher passes to the next state Propose_location; in default the agent remains in the state propose_time, and chooses the next timeslot to negotiate. If all timeslots were rejected, it means that the decision for the currently selected study does not exist, and the agent-teacher passes to the state (3a) Solnot_found.

3a) *Solnot_found*. In that state the agent-teacher founds itself if all proposed for timeslots were rejected by the agents of groups, and the total solution was failed. The study is marked as "having no solution".

4) *Propose_location.* The agent-teacher sends the sorted list of classrooms to the agent of classrooms inside the message LOCPROPOSAL. That message contains also the study's identifier and the timeslot's identifier. Using own occupancy list, the agent of classrooms searches for the first classroom in the list, which is available for the timeslot given or occupied by the study with a lower priority. If the search was successful, and the classroom is found, in reply to the agent-teacher the agent of classrooms sends the message ACCEPT with the identifier of the room found. In the failure case the agent of classrooms sends the reply REJECT. Once the positive ACCEPT reply is received, the agent-teacher moves to the next state. In the result of REJECT receiving the agent returns to the state propose_time for selecting the available timeslot for the study.

5) *Fix_meeting.* If the agent-teacher occurs in that state, it means that both the timeslot and the classroom for the study were successfully found. In the result the agent-teacher sends to all other agents the message FIXMEETING with the identifiers

of the study, the timeslot, and the classroom. If the agent-group does not have assignment for the received timeslot, the timeslot is fixed. By a similar way the agent of classrooms fixes the location. If the received timeslot (or the classroom) is occupied, the agent discards assignment of the study with lower priority, and sends to the agent-teacher the message CANCEL MEETING, which is forwarded further to other agents in order to modify their agendas.

In the states (1), (2), (3) and (4), if some agents did not send the answer during a predefined time period, the agent-teacher places the study, being under consideration, into the list of the cancelled subjects, to retry attempts later. Once all agents-teachers finish state processing, the common schedule is considered to be complete. One important feature of our algorithm is that of the partial timetable is always available. The complete timetable, including all the subjects, sometimes simply does not exist. In such a case, however, the considered algorithm will build the consistent time table, with some subjects of low priority ignored.

4. Implementation Details of the Software System AgentTime

The described mathematical model and the multi-agent algorithm were applied in the course of design and development the software system for time tabling called AgentTime. AgentTime uses rich communication and agent-life cycle capabilities of Java-based JADE multi-agent platform [29], and has highly distributed software architecture (fig.2). Flexible multi-tier architecture of the system supports simultaneously multiple timetable design sessions and interaction of multiple agents.

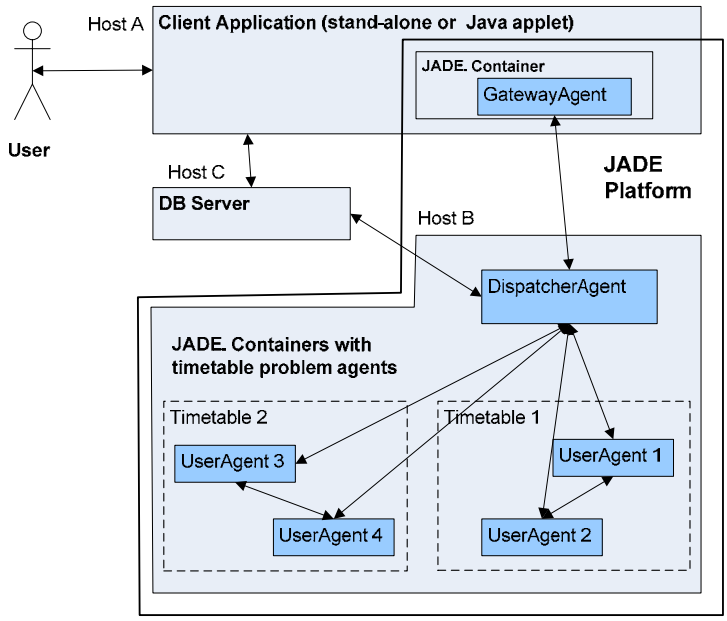


Fig. 2. The software architecture of AgentTime timetabling system.

In AgentTime agents indirectly communicate with each other by passing the messages in accordance with a problem-specific ontology (table 3).

Table 3. Multi-agent ontology for timetable design

Message	Semantics
WHENAVAIL	Inquiry to the agent-group for available timeslots.
USERAVAIL	Agent's response to the message WHENAVAIL. The message contains the vector with available timeslots. Format : (a_{11} a_{12} … a_{18} a_{21} a_{22} … a_{28} … a_{71} a_{72} … a_{78}), where $a_{ij} \in \{0,1\}$, 1 – working day, 0 weekend.
IMPOSSMEETING	The message to the server agent about impossibility to find a time table for the subject with id sbj_id. Format: (sbj_id).
EVALUATE	Inquiry to the agent of group for time and location preferences related with the subject sbj_id. Format:(sbj_id).
SUBJPREFS	Agent's response to the EVALUATE. Format: (sbj_id (w_{11} w_{12} … w_{18} w_{21} w_{22} … w_{28} … w_{71} w_{72} … w_{78}) ((L_1 p_1) (L_2 p_2) …(L_n p_n))), where $0 \le w_{ij} \le 1 (i = 1..7, j = 1..8)$ – evaluation of i-th day of week and , j-th lesson; $0 \le L_k \le 1, k = \overline{1,n}$ – the number of the classroom; missing classrooms have the priority with value $0; 0 \le p_k \le 1, 0 \le k \le n$ – the preference of the classroom L_k.
TIMEPROPOSAL	The agent-teacher proposes time for the subject sbj_id. Format: (sbj_id (d p)), where d – the day of the week; p – the number of the lesson.
LOCPROPOSAL	The agent-teacher proposes location for the subject sbj_id. Format: (sbj_id (L_1 L_2 … L_m)), where $L_k, k = \overline{1,m}$ – the identifier of the classroom. The classrooms are sorted in accordance with the preferences.
ACCEPT/REJECT	Agent's response to the message TIMEPROPOSAL (ACCEPT or REJECT). If the proposal is accepted the message contains the classroom's identifier. Format: (L), where L – is the id of classroom.
FIXMEETING	Inquiry to fix the timeslot and location for a certain subject sbj_id. Format: (sbj_id (d p) L), where d – the day of week; p – the id of the lesson; L – the id of the classroom.
CANCELMEETING	Notification about cancelling a conflicting subject. Format: (sbj_id (d p) L), where d – the day of the week; p – the id of the lesson; L – the id of the classroom.

Interaction of the agents during the design of timetable can be illustrated by the UML sequence diagram in fig. 3. In AgentTime apart from previously mentioned types of the agents we use the dedicated ServerAgent which is responsible for communication with external data sources, logging and other technical tasks.

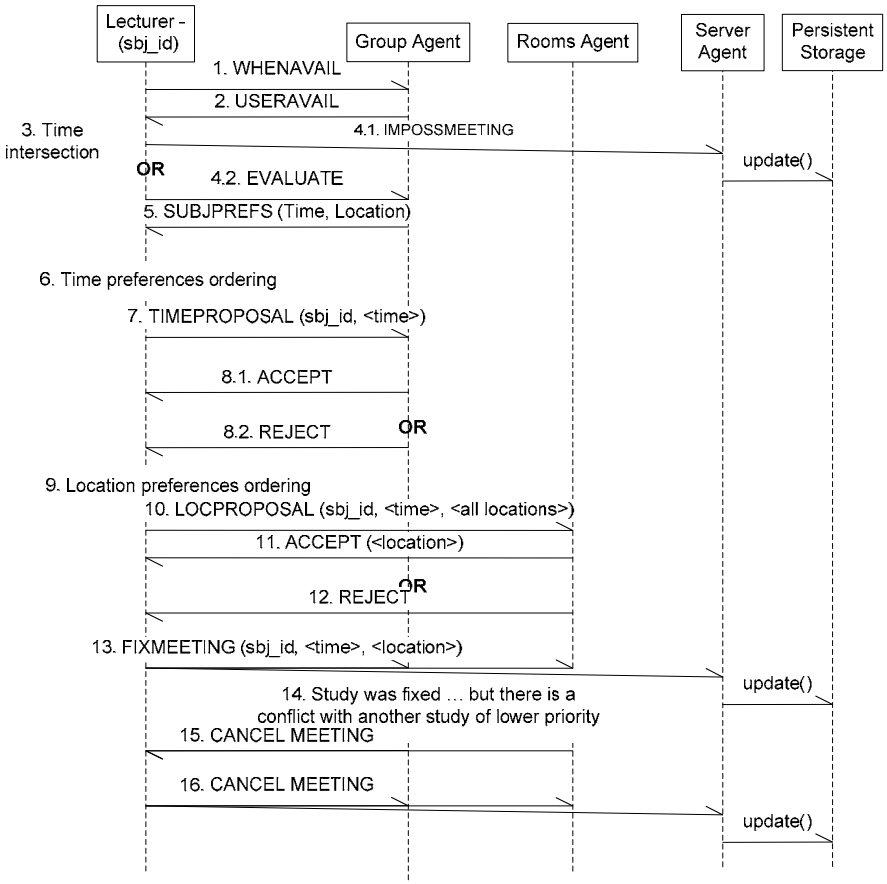
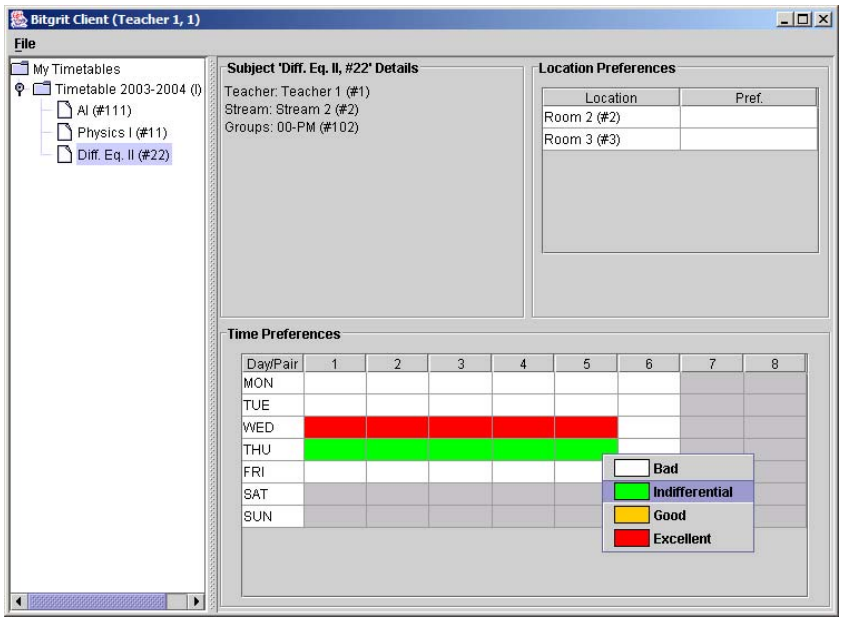
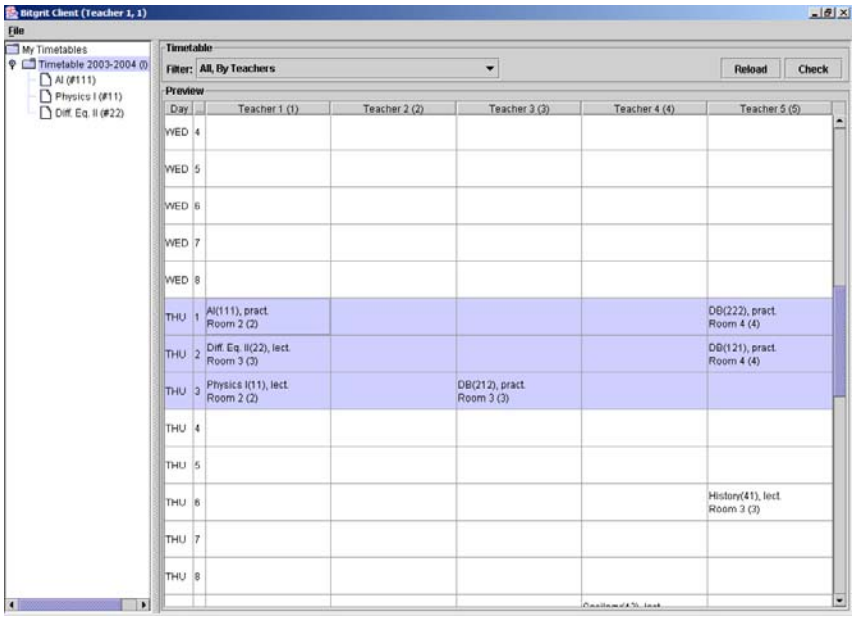


Fig. 3. The state diagram of the agent-teacher's algorithm.

Different users of AgentTime can interact with the system using different end-user tools, including web-browsers and PDAs. The mostly used way of interaction assumes application of applet-based graphical interfaces (fig.4), but also JSP-based interfaces are available.



a)



b)

Fig. 4. Examples of AgentTime's graphics interfaces: a- assignment of preferences; b – representation of the ready timetable.

5. Discussion

This article focused on the important problem of time tables' design for educational institutions. To tackle this problem in the context of modern distributed and highly dynamic universities we proposed the mathematical model and correspondent multi-agent algorithm for iterative timetabling in presence of different subjective preferences for time and location of subjects. The theoretical considerations become a foundation for development of the multi-agent software system AgentTime. That system facilitates distributed time planning and allocation of timeslots and classrooms.

The developed algorithm belongs to the class of domain-specific multi-agent algorithms and shows good performance metrics. Analysis shows that in the case of single computational node computational complexity of the algorithm *C* for allocation of timeslots and rooms can be estimated as follows:

$$C \le S \cdot \log_2 S \cdot T_0\left(n_g, n_l\right), \text{ where}$$

S – is the number of subjects, $T_0\left(n_g, n_l\right)$ - a constant determined by the problem's conditions. If AgentTime is distributed among $P \le S$ computational nodes, then estimation of processing time t_p will be $t_P \le \frac{C}{P} = \frac{S}{P} \cdot \log_2 S \cdot T_0$. In the extreme case, when $P = S$, t_p will not be greater then $T_0 \cdot \log_2 S$.

Comparing our results with other known approaches to multi-agent timetabling like the algorithm MSRAC [27] , we can note that our system is capable of solving a more general problem, allocating not only timeslots, but classrooms also. With a few modifications proposed model and algorithm will be suitable for managing other important resources as well. At the same time we need to improve theoretical background of our algorithm to rigorously prove the optimality of the solutions found in terms of the criteria (16) and (17).

In the nearest time we are going to perform wide-area field experiments with AgentTime to test its robustness and quality of timetabling in real conditions of the complex university. We are also interested in extending the proposed mathematical model and software implementation of AgentTime with other approaches to multi-agent coordination. In this context application of the paradigm "Controller-Variable Agent " [28] is seemed to be very promising.

This work was partially supported by Russian Fund of Basic Researches (grant # 07-07-00058).

6. References

1. Cheng J.Q., Wellman M.P. The WALRAS Algorithm: A Convergent Distributed Implementation of General Equilibrium Outcomes // Journal of Computational Economics, 12. 1998. pp. 1-23.
2. Sandhu K.S. Automating Class Schedule Generation in the Context of a University Timetabling Information System. PhD Thesis. 2001.

3. Petrovic S., Burke E. University Timetabling. University of Nottingham. 2003.
4. Akkoyunly E.A. A Linear Algorithm for Computing the Optimum University Timetable // The Computer Journal. 16(4). 1973. pp. 347-350.
5. Shaerf A. Local Search Techniques for Large High School Timetabling Problems. // IEEE Transactions on Systems, Man And Cybernetics 29(4). 1999. pp. 368-377.
6. De Werra D. An Introduction to Timetabling // European Journal of Operational Research. 19. 1985. pp. 151-162.
7. Taha T.R. Numerical schemes for nonlinear evolution equations // The College Journal of Science & Technology. Jerusalem. 2. 1987. pp. 105-116.
8. Dimopoulou M., Miliotis P. Implementation of a university course and examination timetabling system // European Journal of Operational Research. 130(1). 2001. C. 202-213.
9. Burke E., Ross P. (eds) The Practice and Theory of Automated Timetabling: Selected Papers from the 1st Int'l Conf. on the Practice and Theory of Automated Timetabling, Napier University, August/September 1995, Springer Lecture Notes in Computer Science Series, Vol. 1153. 1996.
10. Burke E., Erben W. (eds) The Practice and Theory of Automated Timetabling III: Selected Papers from the 3rd Int'l Conf. on the Practice and Theory of Auto-mated Timetabling, University of Applied Sciences, Konstanz, August 16-18, 2000, Springer Lecture Notes in Computer Science Series, Vol. 2079. 2001.
11. Burke E., De Causmaecker P. (eds) The Practice and Theory of Automated Timetabling III: Selected Revised Papers from the 4th Int'l Conf. on the Practice and Theory of Automated Timetabling, KaHo St.-Lieven, Gent, Belgium, 21-23 August 2002, Springer Lecture Notes in Computer Science Series Vol. 2740, 2003.
12. Back T. Evolutionary Algorithms in theory and practice. New-York: Oxford University Press. 1995.
13. Sharma D., Chandra N. An evolutionary approach to constraint-based timetabling. PRICAI Workshops 2000. pp.80-92
14. Sanchez E., Shibata T., Zadeh L. Genetic Algorithms and Fuzzy Logic Systems. Soft Computing Perspectives // World Scientific. 1997.
15. Buckles B.P., Petry F.E. Genetic Algorithms. IEEE Computer Society Press. 1992.
16. Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley. 1989.
17. Maxfield C. Genetic algorithms: programs that boggle the mind. EDN. 1997.
18. Srinivas M., Patnik L.M. Genetic Algorithms: A Survey. // IEEE Computer. 27(6). 1994. pp. 17—26.
19. Abramson D. Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms. Management-Science. 1991.
20. Wellman M.P., Walsh W.E., Wurman P.R., MacKie-Mason J.K. Auction Protocols for Decentralized Scheduling // Games and Economic Behavior 35. 2001. pp. 271-303.
21. Walsh W.E., Yokoo M., Hirayama K., Wellman M.P. On Market-Inspired Approaches to Propositional Satisfiability. Proceedings of Int. Conf. IJCAI-2000. 2001. pp. 1152—1160.
22. Yokoo M., Hirayama K. Distributed Constraint Satisfaction Algorithm for Complex Local Problems. Proceedings of the Int. Conf. ICMAS-98. 1998. pp.372–379.
23. Yokoo M., Hirayama K. Algorithms for Distributed Constraints Satisfaction: A Review. Proceedings of Int. Conf. AAMAS-02. Vol.3. No.2. 2000. pp. 185 – 207.
24. Garrido L., Sycara K. Multi-Agent Meeting Scheduling: Preliminary Experimental Results. Proceedings of the Int. Conf. ICMAS-96. Kyoto, Japan.1996.
25. Franzin M.S., Freuder E.C., Rossi F., and Wallace R. Multiagent Meeting Scheduling with Preferences: Efficiency, Privacy Loss, and Solution Quality. Proceedings of Intrl. Conf. AAAI-02 (workshop on preference in AI and CP). 2002.
26. BenHassine A., Ito T., Ho T.B. A New Distributed Approach to Solve Meeting Scheduling Problems. Proceedings of IEEE/WIC Int. Conf. IAT, 2003. pp. 588-591.

27. BenHassine A., D´efago X., Ho T.B. Agent-Based Approach to Dynamic Meeting Scheduling Problems. Proceedings of Int. Conf.AAMAS-04.V.3. 2004. pp. 1132 – 1139.
28. Al-Maqtari S., Abdulrab H., Nosary A. Constraint Programming and Multi-Agent System mixing approach for agricultural Decision Support System. Proceedings of In. Conf ECCS'O5, 2006, pp. 199-213.
29. Bellifemine F. L., Caire G., Greenwood D. Developing Multi-Agent Systems with JADE. Wiley. 2007.