Annales UMCS Informatica AI 5 (2006) 59-67

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Agent based approach to University Timetabling Problem

Paweł Peryt*

Institute of Applied Computer Science, Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

Abstract

A concept of agent-based approach to timetabling problem is presented. Based on the problem description and with its formalization the term "agent" is introduced. Agents act on behalf of entities taking part in the timetabling process (activities, rooms and students) and they interact to maximize their own utility. Also a brief overview of existing approaches is presented.

1. University Timetabling Problem

At the typical university students attend various courses (activities). These activities take place in designated rooms with some equipment and a maximum capacity. Some activities require dedicated equipment (e.g. beamers, oscilloscopes, etc.) available only in several rooms (like labs). Students attend activities they enrolled on. It would be also preferable if a constructed timetable reflected students' preferences (e.g. to time). On the other hand, the enrolment is guarded by some formal criterion (like paying the tuition or finishing prerequisite courses).

The university timetabling problem (UTP) can be defined as follows: having the set of defined timeslots, the set of available rooms and the list of activities to be planned, the aim is to assign activities to rooms in particular timeslots in such a way that activities take place in properly equipped rooms and all the students have a possibility to attend their classes. To be more precise:

- every student attends no more than one activity at any timeslot,
- no more than one activity takes place in any room at any timeslot,
- every student may attend his or her classes
- ach activity takes place in a room with appropriate equipment
- at any time no more students than the maximum capacity of a room attend a course.

* *E-mail address*: pawel.peryt@pwr.wroc.pl

The aforementioned criteria are so called "hard constraints" – breaking them is not allowed.

However, it is often preferable to address also other requirements – criteria that are not critical, but it would be good to have them kept. These are so called "soft constraints", for example:

- it is preferable that no student attends more than 3 classes in a row,
- the number of students attending to a class taking place in a particular room should be close to maximum capacity of the room (to avoid a situation when class for 10 students is held in room for 100 students),
- the overall cost of renting rooms is minimized.

The UTP is solved when there is such a plan constructed, that no hard constraints are broken and preferably soft constraints are satisfied [1]. Because of space limitation only the problems which have a solution – i.e. it is possible to find a timetable with all hard constraints satisfied (in the conclusions section an approach to the case with no possible solution will be tackled) are considered here.

If we treat students as a special kind of resources, the problem may be formulated as assignments of activities to resources over time in a way that specified constraints are kept.

In the following sections a more formal definition of the problem will be presented, and then a concept of agent will be introduced.

2. Constraint satisfaction problem

Hard constraints may be specified in terms of constraint satisfaction problem (CSP) defined as [3] $\Pi_{cs} = (X, D, C)$, where:

$X = \{x_1, \ldots, x_n\}$ – set of domain variables x_i over domains D_i constituting the set $D = \{D_1, \ldots, D_n\}$

$\lambda: X \rightarrow D_1 \cup \ldots D_n$, $\lambda = (v_1, \ldots, v_n) \in D_1 \times \ldots \times D_n$ – labeling associating a value $v_i \in D_i$ with each domain variable x_i;

$C = \{C_1, \ldots, C_m\}$ – set of constraints (relations of type $2^{Di1x \ldots xDik}$) limiting the set of possible (partial) labelings.

The problem is to find a labeling $\lambda \in \Lambda(C)$ in solution space Λ given by C in a way that all constraints are satisfied.

In UTP variables may refer for example to:

- a time when a particular activity takes place in a room,
- time and activity assigned to a student at particular time.

Soft constraints may be modeled by imposing a partial order on values of variables. Then if more than one value of a variable satisfies the constraint, the one taking the precedence over the other will be chosen.

3. Distributed constraint satisfaction problem

The variables and constraints in CSP are derived from entities and relationship between them present in the problem statement. Often more than one variable and/or constraint refers to a particular object from the considered domain. This relationship may be more formally stressed by defining for a given CSP a grouping function for constraints and variables. It is said that variables and/or constraints are in the same group if values of associated grouping function are the same. The grouping reflects logical relationships from the problem domain.

This way the CSP has been extended to distributed constraint satisfaction problem [2] – DCSP – $\Pi_{dcs} = (X,D,C,A,\varphi)$, where:

- X, D, C – refers to variables, domains and constraints (as in definition of CSP),
- A – set of groups,
- φ: $X \cup C \rightarrow A$ – assigns variables and constraints to groups.

Similarly to CSP, the problem is to find a labeling λ satisfying the constraints defined on the variables. In the case when φ is a constant function, DCSP becomes an ordinary CSP (sometimes referred to as centralized CSP).

If there is a local consistency predicate defined to indicate that all variables within a group do not break constraints belonging to that group, then the original problem is divided into sub-problems (defined by these predicates). In the special case, when there are no constraints between groups (i.e. constraints which referr to variables belonging to separate groups) the problem was divided into independent sub-problems and each of the sub-problems may be solved independently.

In a general case (and in most of the practical cases) such a division is not possible. Variables from different groups are tied with constraints. There can be two classes of constraints identified [3,4]:

- external constraints – those with variables belonging to separate groups,
- internal (local) constraints – constraints with variables belonging only to a particular group.

The concept of grouping is not only a formal construct, it also can be derived from the following practical premises [5,6,7]:

- in the case of naturally distributed problems the cost of gathering all constraints in one place may be too high (e.g. cost of sending, storing the data, but also converting it to a one dedicated format – especially important when data is gathered from many independent sources),
- gathering all information in one place may be unwanted because of security of privacy reasons,

– in the case of problems which are dynamic (in a sense that conditions are changing fast) there is no point in processing data in one place, because by the time data is gathered, it is outdated.

In the case of UTP we can enumerate also several reasons for not gathering all information about the problem in one place, for example:

– the full knowledge about time preferences of student is known only to a student and there is no point in revealing them,
– the full information about cost of renting a room is known to the authority which the room belongs to.

Local sub-problems may be solved independently without the need of looking at values of variables in other groups. When the local sub-problem is solved all local constraints are not violated and variables are given proper values. In the case that there exist more than one solution to a local problem, the best solution can be chosen according to some preferences. As it does not affect solution of other local problems the preferences can be set up for all groups independently.

The case with external constraints is more complicated and it requires that a special coordination mechanism has to be used. When a variable is given a value in one group the information have to be propagated to other groups that share a constraint.

In [6,7,9] there is an overview of algorithms for solving DCSPs and some practical problems that can be formalized in this way.

The grouping concept can be refined further – the groups can be combined together to formulate higher level groups. The constraints are then defined between such a higher level groups or within a particular group. Of course if the process is continued finally only one group will remain and the problem will become a classical CSP one.

4. An Agent-Based approach to DCSP

A term “agent” is widely used in AI related publications. Although there are many doubts about its precise definition there was a significant progress made in the area. Several important results were achieved in AI domain and in software engineering domain. However, there is still much work to be done in both the tools and the methodologies to construct reliable and scalable agent systems.

One of the general definitions of an agent will be adapted here – a (software) entity which is able to interact with environment (gather information from the environment and change the environment by means of actions) in order to fulfill its goals. An agent makes an autonomous decision about what action to take in order to achieve the goals.

In the case of Multi Agent Systems (MAS) there are many interacting agents which are acting independently of each other. Without going into too much details it can be said, that the agents’ goals do not have to be consistent or coherent. It is often the case that they are contradictory, so in order to allow

coexistence of such agents a special coordination mechanism has to be incorporated into the system [8]. In a special case, agents interact to achieve a common global goal – in this case this is so called cooperative problem solving [10,11]. Each agent may follow its own reasoning procedure and gather information from previous interactions. More detailed description of areas of interest in MAS can be found for instance in [13,14].

Referring to DCSP domain, an agent has information about all variables belonging to one group as well as about constraints relating to these variables. The actions that agent can take are to give a value to own variable and to exchange with other agents information about change in value of a variable bounded by external constraint. The goal of the agent is to find an acceptable assignment to all its variables. The assignment is acceptable when all external constraints are satisfied. An agent is also endowed with a preference function (utility function) that allows choosing among acceptable assignments.

In the case of UTP we can distinguish agents representing rooms, students and activities. Each agent is responsible for establishing its own timetable. In addition, agents have their own preferences (like preferred time, number of activities in a row, usage ratio, etc). The timetable constructing is guided by this local utility functions instead of global utility function defined for the whole timetable.

5. Proposed approach

In the presented description of UTP three kinds of agents may be identified:

- representing students (A_S) – responsible for constructing a timetable for a student assuring that individual preferences are fulfilled. Each agent has an initial amount of money,
- representing rooms (A_R) – responsible for providing appropriate room allocations (taking into account its utilization, availability). For each timeslot there is an initial price for renting each room defined,
- representing activities (A_A) – responsible for assuring that each activity takes place in properly equipped room, and each student enrolled on the activity may attend it.

Agents interact in order to label all defined variables. Variables local to a particular agent may be labeled independently on other agents. In the case of external dependencies all concerned agents have to be informed about the current labeling. Each proposal of a change to the labeling have to be agreed before it is set. As the change may cause a violation to local constraints of other agents, or other external constraints that the initiator is not aware of, it has to have means of convincing others. A straightforward model may be assumed that there exist a kind of “money” in the system. The utility function of each agent is somehow dependent on it. Each agent representing a student may freely allocate

its money to buy a place on activity at the specified time. The money is allocated according to an individual preference of an agent. Agents do not have to explicitly express they preferences – it is done only implicitly by revealing bids.

When A_S wants to buy a place, it informs appropriate A_A about this fact, and gives information about amount of money it wants to spend (a bid). The A_A accumulates information from all interested A_S agents and seeks for an appropriate A_R agent. The A_R agent adjusts its initial renting price according to incoming requests and preferences relating for example to utilization ratio and kind of offered facilities.

The timetable construction is performed in two alternate phases:

1. Allocation of students' activities in time.
2. Allocation of rooms to activities.

The process ends when there is no agent willing to change its plan, or if there is no way (no money) for conducting the process further.

When there is a change in problem conditions (for example one of the rooms become unavailable) all concerned agents are informed by A_R and they can react. It is not necessary to rework the whole timetable.

6. Related work

One approach to CSP in context of agent-based systems can be found in [12]. In [15] there is a timetabling problem described, where also lecturers are taken into account – a lecturer is capable of conducting only a limited set of courses and it is not allowed that a lecturer has more than one class at the same time. Students are not treated separately; instead predefined groups of students are introduced. In addition, the walking distances between various rooms are modeled and the following quality measures for the plan were defined:

- compactness – the number of free time slots between subsequent classes for lecturers and students have to be kept minimal,
- there is a daily limit for the maximum number of classes per lecturer or student,
- the rooms should be assigned in such a way that the walking distance between them is minimized.

The initial timetable is constructed using a centralized approach – one central planning agent solving CSP. In the next step the timetable is refined by adjusting assignment of activities in time and then activities to rooms. During the process the following agents interact:

- representing teachers,
- representing groups of students.

Agents negotiate among themselves in turns. In each turn agents are selected in a predefined order and may propose a change to its timetable. The aim is to:

- maximize compactness of the own timetable,

– minimize the overall number of free time slots in the own plan (the sum of all free slots within an entire plan may remain the same).

During the negotiation process it may happen, that a change in a timetable proposed by one agent decreases the utility of timetable of another agent. In such a case an initiator of a change may propose some "money" as a refund to the second agent. Initially, each agent has a particular amount of money – proportional to the number of free timeslots in its timetable. This gives a natural preference to agents with worse timetable. To prevent the occurrences of cycles during negotiations there is a special parameter limiting the number of free timeslots that can be inserted into a timetable during the negotiation process. To increase the stability of a timetable the value of this parameter decreases over the time according to the simulating annealing scheme. Additionally, agents have to pay an entrance fee (or tax) at the beginning of each negotiation turn – the overall amount of money in the system decreases over time. If during a negotiation there is a better timetable constructed, it becomes the solution.

In [16] there is a problem room utilization considered in context of multi-faculty university timetabling. Each faculty has to construct a timetable for its activities using (preferably) own resources (rooms). There is cost associated with each resource. If it is not possible to construct a timetable for a faculty using only its own resources, resources from other departments can be used. A task of coordination of timetables between faculties in order to allow resource sharing is delegated to a special broker agent. The broker is responsible for optimal allocation of university resources. The solution refined by a broker is sent back to each department.

In [5] a problem of timetabling on multi-department university is also considered. Similarly, resources between departments may be shared and the constructed timetable should minimize the overall cost. Constructing a timetable for a particular department is a responsibility of a Scheduling Agent (SA), which is aware of all faculty specific constraints. The coordination between SA's is done by one dedicated agent – Central Agent (CA).

In [17,18] there is a multi agent system MedPAge presented. It is capable of constructing a timetable for medical activities in hospital. Various medical activities are interrelated – for example must be executed in a fixed order. Additionally, for some procedures the presence of a patient is unavoidable, so they cannot be performed in parallel. The schedule might be also affected by some unpredicted emergency cases. For some procedures there is dedicated unique equipment required. This can be shared between hospital divisions or it might be necessary to transport it from another hospital. The domain is modeled in the system using agents representing resources and patients. Each agent is endowed with a utility function and a goal to maximize it. Agents can sell or buy time slots in a pro-active manner: if an agent (buyer) wants to change the start of an execution of a task it contacts an agent responsible for the concerned

resource, then agents representing patients (sellers) are informed. The transaction takes place if the gain of a buyer is not less than the cost of establishing of a new timetable (alignment of patients' schedule, change in resources' allocation).

Conclusions

The relation between distributed constraint satisfaction problem and multiagent systems has been presented. On that basis a concept of a multiagent system for university timetabling problem was sketched. The model can be easily extended by introducing additional agents (e.g. representing teachers).The next step is to verify these concepts on a real system implemented using existing multiagent and CSP technologies.

There are some interesting problems left open, like the problem of relaxing constraints has to be refined: in the case that there is no solution with a given set of hard constraints some of them have to be relaxed. As it is hard to define a general approach guiding which constraints should be dropped, it was good if such a decision would be left to an agent itself. For example based on a concept of punishment an agent would have to pay a special amount of money when it violates a hard constraint.

References

[1] Abbas A.M., Tsang E.P.K., *Constraint-Based Timetabling – a Case Study*, (2001).
[2] Solotorevsky G., Gudes E., Meisels A., *Distributed Constraint Satisfaction Problems – A model and Application*, Kluwer Academic Publishers.
[3] Hannebauer M., *Their problems are my problems*, in [8]., (2001).
[4] Zhou L., Thornton J.,Sattar A., *Dynamic Agent Ordering in Distributed Constraint Satisfaction Problems*, (2003).
[5] Meisels A., Kaplanski E., S*cheduling Agents – Distributed Timetabling Problems (DisTTP)*, Practice and Theory of AutomatedTimetabling IV, Lecture Notes in Computer Science, Springer-Verlag Heidelberg., (2004).
[6] Yokoo M., Durfee E. H., Ishida T., Kuwabara K., *The Distributed Constraint Satisfaction Problem: Formalization and Algorithms*, IEEE Trans. on Knowledge and DATA Engineering, 10(5) (1998).
[7] Yokoo M., Hirayama K., *Algorithms for Distributed Constraint Satisfaction: A Review*, Autonomous Agents and Multi-Agent Systems, 3(2) (2000) 198.
[8] Tessier C., Chaudron L., Muller H.-J. (editors), *Conflicting Agents*, *Conflict Management in Multi-Agent Systems*, Kluwer Academic Publishers, (2001).
[9] Kumar V., *Algorithms for Constraint Satisfaction Problems: A Survey*, AI Magazine, 13(1) (1992) 32.
[10] Ferber J., *Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence*, Addison-Wesley Longman, (1999).
[11] Weiss G. (editor), *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, (1999).
[12] Calisti M., Neagu N., *Constraint Satisfaction Techniques and Software Agents*, Agents and Constraints workshop, AIIA'04, September 2004, Perugia, Italy, (2004).
[13] Murthy S., Akkiraju R., Rachlin J., Wu F., *Agent-Based Cooperative Scheduling*, (1997).
[14] de Weerdt M., *Transport planning and scheduling*, (1999).

[15] Anke K., Staudte R., Dilger W., *Producing and Improving Time Tables by Means of Constraint and Multi-agent Systems*, AAAI-97, WS Constraints and Agents, Technical Report WS-97-05, AAAI Press, Menlo Park, (1997).
[16] Richards E.B., Das S., Choi H., El-Kholy A., Liatsos V., Harrison C., *Distributed Optimisation, A Case Study of Utilising Teaching Space in a College*, Proceedings of the Expert Systems 96 Conference, SGES Publications, (1996).
[17] Bartelt A., Lamersdorf W., Paulssen T.O., Heinzl A., *Agent oriented specification for patient-scheduling systems in hospitals*, (2002).
[18] Paulssen T.O., Jennings N.R., Decker K.S., Heinzl A., *Distributed Patient Scheduling in Hospitals*, (2003).