Negotiation among Scheduling Agents for Distributed Timetabling

Eliezer Kaplansky and Amnon Meisels

Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, 84-105, Israel
{kaplan_e,am}@cs.bgu.ac.il

1 Introduction

Multi-Agent System (MAS) technology has become a new paradigm for modeling, designing, and implementing software solutions. Agents are sophisticated computer programs that act autonomously on behalf of their users, across distributed environments. A Multi-Agent System is a loosely coupled network of software agents that interact to solve a global problem that is beyond the individual capacities or knowledge of each agent.

Many real world *Timetabling Problems* (TTPs) are composed of organizational parts that need to timetable their activities in an independent way, while adhering to some global constraints. The timetables of all the parts must be combined to yield a coherent, consistent solution. To achieve this goal negotiations among multiple agents and requests for changes in their local solutions are needed.

A well-defined example of a real-life *distributed TTP* (DisTTP) is the construction of university course timetables (CTT). The departmental problem consists of scheduling a set of lectures for each course within a given number of time periods and rooms. Since each department owns different resources, has different teaching requirements, different preferences and follows different strategies for utilizing their teaching space, each department needs to solve a different problem and can use a different solver that is tuned up to its needs. The agents that generate the schedules of the departments are termed *Scheduling Agents (SAs)* [1].

Each course belongs to one of the university departments. However, it is almost always the case that a significant part of the curricula of one department is given by another department. If a course is given to more than one department it must be scheduled at the same time slot in all departmental timetables that use this course. These courses are termed *Shared Courses*. This relationship can be considered as a *supplier - consumer* relationship.

Our model uses *Constraints Satisfaction Problem* (CSP) techniques for the agents of a DisTTP. A common CSP model for the CTT problem, represents lectures as variables and the time slots as the domain values for the SAs. The required class rooms form the domain of the *Rooms Agent* (RA) [4]. In our model the Scheduling Agents interact via messages in order to coordinate their timetables to be consistent with inter-departmental constraints. The proposed model is based on a well defined negotiation process that always ends in an agreement (not ensured to be the optimal).

When the above real world problem is represented by a constraints satisfaction problem (CSP), it is an instance of the *Distributed Constraints Satisfaction Problems* (DisCSP) (see Yokko's book [5] for an introduction). One can think of a DisTTP as a network of scheduling agents. The DisCSP instances that have appeared in the literature are modeled so that each agent has only one variable. However, the local timetabling problem of each SA is very complex and the global

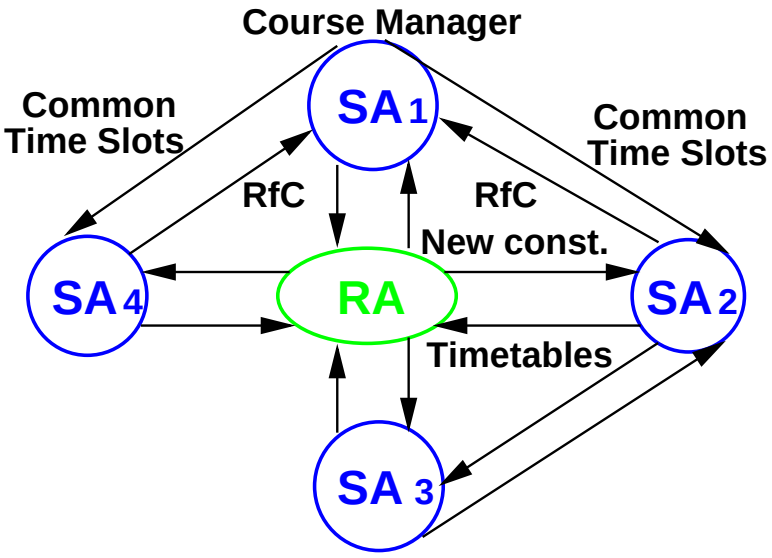

Fig. 1. First the scheduling agents ($SA_i s$) conduct negotiation for global timetable. Next the Rooms Agent (RA) adds new constraints to SAs. The SAs solve the modified problem and send back a new timetable.

problem includes relatively fewer constraints [2, 3]. Therefore, a new solution approach need to be devised for DisTTPs. The focus of this paper is on negotiation procedures that can detect and avoid the occurrence of conflicts between Scheduling Agents that are sharing a course, and between each SA and the Rooms Agent.

2 Inter-agent Negotiation

The proposed approach uses a multi stage inter-agent negotiation procedure.

In our model, inter agent negotiation consists of three main stages. The first stage (steps 1-3 below) focuses on the search for a local solution for the timetabling problem of each agent. In stage 2 (steps 4-8), the target is a stable global timetable that eliminates any inter agent conflict between the SAs. In the third stage the SAs negotiate with the *Rooms Agent* for rooms for all the courses under their management.

Step 1 - Self assessment: Each agent estimates the rate of finding solutions for its local problem, expressed by the *Average Computation Cost* (ACC) per solution found. ACC is measured by the average number of *constraint checks* per solution found. In this stage, the time slots of the *Shared Courses* are still unknown. The agents take one of the following intelligent guess:

Last year: Use last year's time slots for the shared courses. This is the option taken today when the timetables are done manually.

Random: Use random time slots with some known guidelines.

Manually: Let the user set these slots manually.

In the second part of the self assessment step the agent calculates its *Agent Minimal Cost* (AMinC). This is the cost of the best solution the agent can find when it is free to place the *shared courses* anywhere. $AMinC$ represent an absolute lower limit of the cost of any solution for the agent problem.

Step 2 - Direction of the inter agent constrains: The common practice is to base the order of decision making on the basic *supplier - consumer* relationship so the supplier of a course is the *course manager* that make the final decision.

However, preliminary experiments had shown that the complexity of the local problem of each agent is a much more significant factor for choosing an agent to be the course manager of a Shared Course.

To implement this method, each agent sends its ACC, it had calculated in step 1, to all its constraining agents.

Agents agree on the direction of the constraint between them in such a way that the more complex (slower) agent is the *Course Manager Agent* and the other agents are the *Course User Agents*. To avoid cycles the result over the whole network of SAs is forced to be a DAG.

Step 3 - Calculate best local solution: According to the order established in the previous step, each agent waits to get the time slots of its set of shared courses from all its *Course Manager Agents*. Time slot conflicts at the receiving agent can occur when a few course managers ask to use the same time slot. To resolve such conflicts, the agent approves the request of the most complex agent (as established in step 1) and sends new constraints to the other *course manager agents*.

Now, the agent computes its best local solution with the shared courses set at fixed time slots.

Finally, the agent assigns the cost of the best solution found, as its *Agent Maximal Cost* (AMaxC).

Step 4 - Initial suggestion: Every course manager sends the time slots of the *Shared Courses* it manages to the agents that use these courses on its outgoing links.

In response, each *Course User* Agent sends its *Agent Cost Range* ($ARangeC = AMaxC - AMinC$) to the agents that manage its set of shared courses.

Step 5 - Normalizing Agents Costs: To enable the course manager to resolve conflicts between agents sharing the same course, the manager generates a common scale by normalizing the complexity of all agents using this course. For each shared course, the course manager agent computes a set of *course user weights*, that scales the *ARangeC* of each agent using this course to 1.

Step 6 - Requests for Change: Each agent tries to improve its local timetables by changing the time slot of one of its shared courses and re-solving its local problem. When it finds a better solution, it sends a *Request for Change* (RfC) to the *Course Manager Agent*. The RfC message is accompanied by an *Expected Gain* (EG) and a list of suggested *Alternative Time Slots* (ATS) (part of its open time slots). This agent is termed *Initiator Improvement Agent* (IIA).

Step 7 - Approve change: When a *Course Manager Agent* receives a *Request for Change*, it searches for the best solution using one of the slots of the ATS list accompanying the RfC. If the *Manager Change Cost* is lower than the *Expected Gain* an *Approve Change Procedure* is started:

1. The course manager sends a *Change Offer* to all agents using this course.
2. All *Course Users Change Costs* are collected.
3. If the sum of all *Change Costs* is lower than the *Expected Gain* then this *Request for Change* is approved. Otherwise, all involved agents move to the next step - the bid.

Step 8 - The bid: The *Initiator Improvement Agent* attempts to buy its RfC from the *Course Manager Agent* using its *bid tokens*. The *Course Manager Agent*, tries to use these tokens to buy this RfC from all the other agents that use these courses. If the bid succeeds, the RfC is approved.

Step 9 - Negotiations with the Rooms Agent When a university wide stable global timetable is reached, each agent sends a *Rooms Request* for each course it manages to the *Rooms Agent*.

If demand for rooms exceeds supply for some specific time slot, the Rooms Agent send a *Room Refusal Message* to **all** agents that request rooms for this time slot accompanied by ATS.
The agents try to change the time slot of any course that is assigned to this *Refusal message time slot*. If the RA gets enough response messages agreeing to change - the problem is solved.
Otherwise the *Rooms Agent* initializes an auction for rooms for this time slot. A separate auction process is managed by the *Rooms Agent* in order to resolve each case of *Not enough rooms*. The details of the procedure of the rooms negotiation are presented in a separate paper.

3 The DisTTP system at BGU

The approach and the results reported in this paper represent ongoing work to investigate possible solution methods for networks of SAs.

A project to implement a distributed timetabling (DisTTP) system for Ben-Gurion University has been started a year ago. The first module that has been implemented enables the department head to assign teachers to courses. We had installed the Alpha Version of this module in three departments. This enables us to collect real relevant departmental data, mainly:

a) The courses, the TA sessions and the list of labs.
b) The Teachers' and TA personal constraints and preferences.
c) The list of the assignments of teachers and TAs to courses.

The second module solves the timetabling problem of each department. These two modules realize our concept of a Scheduling Agent.

The first version of the *Rooms Agent*, which deals with the constraints of rooms capacity and features among agents, is now being tested in the medical school.

The overall DisTTP system at BGU will include three software layers. The first layer is the *Scheduling Agents*. The second layer supports the negotiation protocol for generating a university wide schedule. The third layer implements the negotiation protocol between the SAs and the *Rooms Agent*. The main effort is currently concentrated on investigating different deployments for the negotiation protocol among the SAs. Current investigation uses real data of this spring semester. The Beta Version will to be a university wide system and is expected to be used to generate the timetable of the coming fall semester of BGU.

References

[1] E. Kaplansky A. Meisels. Scheduling agents - distributed employee timetabling (detp). In *In Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling PATAT'02, Ghent, Belgium*, pages 166–80, July 2002.

[2] C. Eisenberg. A distributed breakout algorithm for solving a large-scale project scheduling problem. In *In Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS2002) Workshop on Distributed Constraint Reasoning, Bologna, Italy*, pages 104–118, July 2002.

[3] S. Lin. A broker model for distributed constraint satisfaction and optimisation. In *In Proceedings of the Tenth International Conference on Artificial Intelligence: Methodology, Systems and Applications (AIMSA'02)' Varna, Bulgaria*, pages 193–202, September 2002.

[4] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87 – 127, 1999.

[5] M. Yokoo. *Distributed constraint satisfaction: foundations of cooperation in multi-agent systems.* Berlin Springer, 2001.