Applying Multi-Agent Algorithm to a Class Scheduling System

Eiji Nunohiro[1] and Kenneth J. Mackin[1]
[1] Department of Information Systems, Tokyo University of Information Sciences
1200-2 Yatoh-cho, Wakaba-ku, Chiba, 265-8501, Japan
{nunohiro, mackin}@rsch.tuis.ac.jp

Abstract- **We propose a multi-agent algorithm to solve a scheduling problem. The proposed application takes into account various restrictions relative to Japanese university class scheduling in particular. A multi-agent system consisting of agents representing the requirements and restrictions for professors and departments are introduced. Scheduling is solved using negotiation between agents together with the hill-climbing method. Through a software simulation, we were able to show that our proposed method successfully applies the self-organizing nature of agents to solve the scheduling problem.**

I. Introduction

Emergence and self-organized behavior in agent systems has become an important tool or feature in solving complex real-world problems, where straight-forward system design fails to yield satisfactory or sufficient results.

For this research, we aim at applying a self-organizing agent system to a particular real-world problem, namely the class (course) scheduling problem for the author's university. Our direct goal is to create a satisfactory solution the complex real-world scheduling problem, but our underlining motive is to study the self-organized behavior in such a working agent system. Our hope is to determine key aspects of self-organizational behavior and understand the basic mechanism of the resultant complex behavior.

Class scheduling is a scheduling problem where limited resources (i.e. professors, classrooms, time periods) must be assigned appropriately. Especially for the case of university class scheduling, there exists specific restrictions and constraints, for example constraints on the days or hours a specified professor can teach, requirements on hardware used for class, or ordering of courses. The various constraints make university class scheduling a complex and time consuming problem to solve.

For this research we solve the university class scheduling by using negotiation between agents. One type of agent, the curriculum agent, tries to solve the scheduling problem regarding placement of professors and appropriate classrooms for each class year of each department. Another type of agent, professor agent, tries to solve the scheduling problem regarding constraints for each professor. The class schedule is evaluated using a violation point method, and the agent negotiation strategy is based on a hill-climbing method to lower the violation points.

In the proposed system, the class scheduling problem is solved using the following 3 steps;

1) Initial allocation of classes and professors taking into account course constraints.
2) Curriculum agent negotiation to solve constraints for professors and classrooms.
3) Professor agent negotiation to solve constraints for professors.

This paper is organized in the following order;
First we give a description of self-organized agent behavior. Next the problem domain of the class scheduling problem is described. Following, the agent system specification and schedule negotiation is given. Finally we close with discussion of results and future topics.

II. Self Organization

It has been reported by various researchers[1] that design of effective organization of multiagents is crucial for the construction of an efficient multiagent system. The Distributed Artificial Intelligence (DAI) community and other disciplines have begun to apply evolutionary processes in order to achieve self-organization of multiagent systems[2].

Self-organization seen in nature is the phenomena in which complex structures or systems are created from apparently self-induced organization of simple components. The complex crystalline structure of snow flakes is an example. For this paper, self-organization in multiagent systems is defined as the emergence of macroscopic organized behavior of a multiagent system from an initial unorganized state, achieved through the microscopic interaction of the constituent agents. In a self-organized multiagent system, the final state of the total system achieved is more organized or complex than the initial state, but this self-organization is achieved by a bottom-up approach, in which each agent only has local goals and a limited view of the total system. Through the local interaction or microscopic behavior of the agents, the global interaction or macroscopic behavior of the total system is constructed.

Table 1. Course Data Example

Department	Year	Course No	Course Name	Requirement type	Constraint	Period	Prerequirement	Hardware Requirements
Inf. Systems	2	1201a	Software Basics	Required	Pair with 1201b	2		
Inf. Systems	2	1201b	Software Basics Lab	Required	Pair with 1201a	2		PC
Inf. Systems	2	1202	System Design	Required		1		
Inf. Systems	2	1203	Artificial Intelligence	Elective		1		
Inf. Systems	3	1301a	System Programming	Required group Elective	Pair with 1301b	1	1201a	
Inf. Systems	3	1301b	System Programming Lab	Required grou Elective	Pair with 1301a	2	1201b	PC

Table 2. Scheduling Constraints

Category	number	constraint	violation points
Curriculum	p1	same professor scheduled to simultaneous time slots	100
	p2	two classes require the same class room (lab) hardware.	100
	p3	two classes require the same size class room.	100
	p4	classroom cannot seat expected number of students	100
	p5	classroom does not have required hardware	80
	p6	course assigned outside of specified day or time	60
Professor	p7	course which is not the professor's specialized area is assigned.	40
	p8	class is assigned on day which the professor is not available.	40
	p9	6 or more time slots per week allotted to professor.	(time slots-5)*60

Recently there has been some research in modeling self-organization in multiagents. Far et. al.[3] proposes that in a purposeful (i.e. not random) organization, Organizational Intelligence (OI), is a property of interaction among agents, and can only be ascribed to at least a pair of agents. Goldman et. al.[2] has used the Game of Life to analyze the patterns of self-organizing agents. Parunak et. al.[4] proposes an entropy model for self-organization in multiagent systems. Using the entropy model, it is possible to achieve self-organization in the macro level of the system while the system shows an increase in entropy (loss of organization) at the micro level among agents. Parunak notes that in order to model self-organization, it is important to view the system explicitly in terms of macro and micro levels. Horling et. al.[1] proposes a model for analyzing and diagnosing the effectiveness of organizational structures for multiagent systems.

III. University Class Scheduling Problem

Class scheduling is a real-world complex scheduling problem. For this research we take the case of an actual Japanese university system, but efficient algorithms solving the class scheduling problem can be applied to a wide range of similar scheduling.

Class scheduling implies the necessary scheduling of classes (courses), classrooms, and professors, all within limited class times. In this case, we take the case of class scheduling for the author's university, Tokyo University of Information Sciences..

Table 1 shows examples of the course data used for scheduling. Laboratory courses require specific ordering of classes and

specified hardware in classrooms, and create complex constraints in the scheduling.

Assumptions for the class and professor assignment include;
1. Classes(courses) each professor teaches has been predefined.
2. Which semester the classes (courses) will be held has been predefined.
3. Every class is assumed to be held (i.e. minimum students will take the class)
4. There are enough classrooms and time allotments in the whole week to allow all necessary classes to be held in the semester.
5. There is a wide range of classrooms available, from different seating capacities and hardware (i.e. projectors)
6. There is a shortage of classrooms so that most classrooms will be used throughout the day for the whole week.
7. There are 5 class days a week. 5 class periods (90 minutes each period) in a day.

Table 2 shows the constraints for the scheduling problem.

IV. Multi-agent System

For this research, a multiagent system is applied to solve the scheduling problem. Here we define a multiagent system as a system in which autonomous software entities, called agents, cooperate to perform some global task or fulfill a system goal. For this research we define an agent to have only local goals, have a limited view of the environment, and can communicate or negotiate with neighboring agents.

The class scheduling system outline is shown in Figure 1.

For the scheduling agent system, 2 types of agents were used; an agent type representing the requirements of professors, and an agent type representing the requirements of the department curriculum.

For the professor agent, 1 agent is added to the system as a representative (agent) for each professor. Each agent carries information regarding the requirements and requests specified by each professor, such as the preferable times and dates for each class, which days of the week the professor cannot hold classes, etc. The Professor agent's goal is to negotiate with other professor agents in order to satisfy the requests and requirements of each individual professor the agent represents.

For the department curriculum agent, 1 agent is added to the system for each class year of each department curriculum. Classes are classified as either 1st year, 2nd year or 3rd year classes, so for each department there are 3 agents, one for each class year. The department agent carries information regarding the requirements and requests specified for each department curriculum, such as requirements regarding ordering of courses, constraints in courses held in the same time slot, etc. The department agent's local goal is to negotiate with department agents to create a department course schedule that will satisfy the requests and requirements of the department.

A simple blackboard model is used to schedule the courses. The agents use the blackboard to share a single class schedule plan. Each agent can enter/change the course for which the professor is assigned into a class slot. If the class slot for a particular time and classroom is already filled with another class, the agent may choose to select a different classroom or time slot, or may choose to negotiate with the agent of the class already in the desired class slot.

V. Schedule Negotiation

The class schedule is evaluated using the following equation:

$$W = F (B (C, T, R), P) \quad (1)$$

W: Total Constraint Violation
F : Constraint Violation Evaluation Function
B : Time table for each student year
P : Violation Point
C : Course data
T : Professor data
R : Classroom data

Total Constraint Violation

Total Constraint Violation points is the total of violation points ($w_{c,i}$) for each Department Agent(i=1,...,n) and violation points (j=1,...,m) for each Professor Agent (j=1,...,m).

$$W = \sum_{i=1}^{n} w_{c,i} + \sum_{j=1}^{m} w_{t,j} \quad (2)$$

Constraint Violation Evaluation Function F

The constraint violation evaluation function is the collection of Professor Agent constraint violation evaluation function (f_t) and Department Agent constraint violation evaluation function (f_c). The constraint violation points for each Professor Agent and Department Agent is calculated by the constraint violation evaluation functions for each type of Agent.

$$w_{t,j} = f_t(\Sigma v_{j,k} * p_k) \quad (3)$$

(constraint k = 1～7)
j : Professor Agent number
$v_{j,k}$: Constraint Violation point for constraint k against Professor Agent j
when v = 0, there is no violation
p_k : Constraint Violation point for constraint k

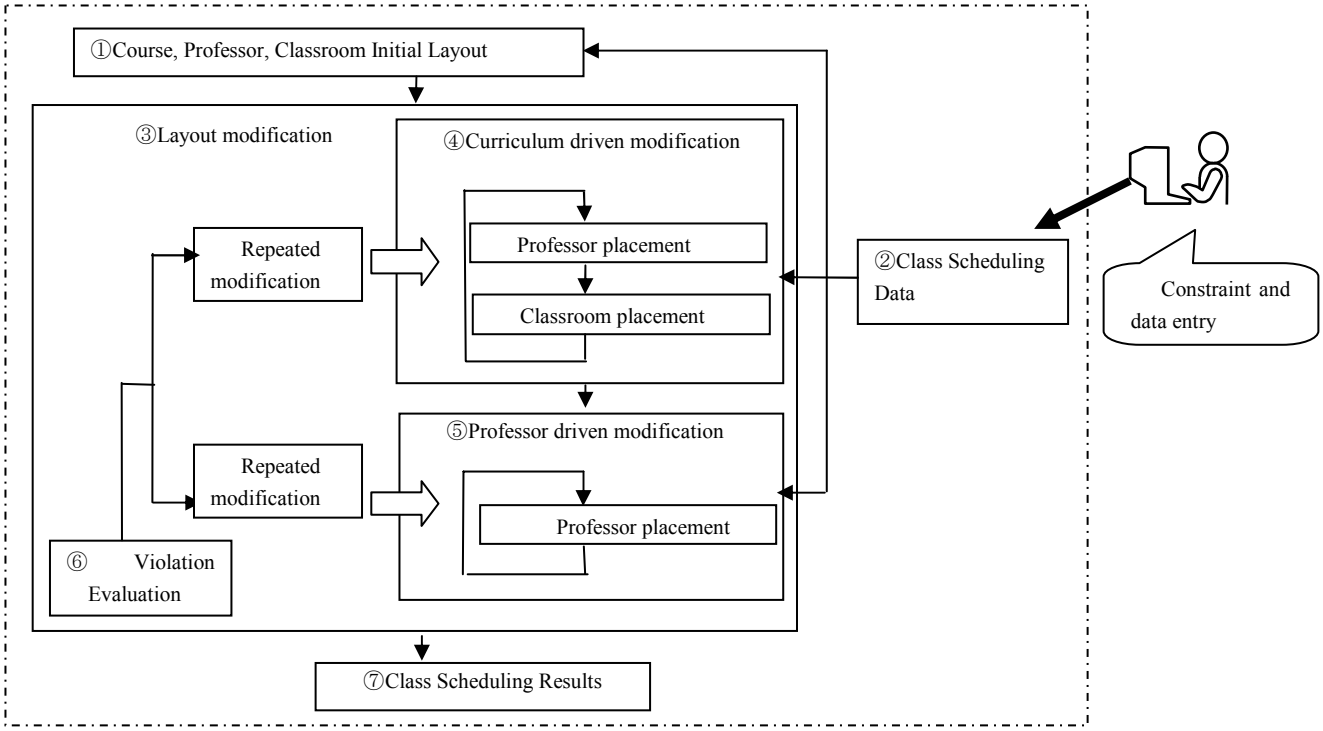

Figure 1. Class Scheduling System Configuration

$w_{c,i} = f_c(\Sigma v_{i,k} * p_k)$ (4)
(constraint k = 8～13)
i: Department Agent number
$v_{i,k}$: Constraint Violation point for constraint k against Department Agent i
when v=0, there is no violation
p_k : Constraint Violation point for constraint k

Below, we describe the scheduling algorithm.

Step 1) Initial Assignment
1-1 Input course data for each class year.
1-2 Select course data according to priority
1-3 Assign courses with time slot specifications
1-4 Assign courses with no time slot specifications
1-5 Assign professors and classrooms for allotted courses.

Step 2) Schedule Modification by Curriculum Agents
2-1 Read current class schedule.
2-2 Mark conflicts of professors and conflicts of classrooms.
2-3 Calculate constraint violation points for each curriculum agent
2-4 Select curriculum agents to negotiate according to violation points
2-5 Continue curriculum agent negotiation until high priority violations are cleared, or until total violoation points are not decreased.

Step 3) Schedule Modification by Professor Agents
3-1 Read current class schedule.
3-2 Calculate the constraint violations for each professor agent.
3-3 Select professor agents to negotiate according to violation points.
3-4 Continue professor agent negotiation until total violation points are not decreased or after a predefined number of negotiations.

VI. Discussion

A software simulation of the scheduling multiagent system was constructed using Java. This was applied to a simulation for 4 departments and 52 professors. Figures 2-4 shows the changes of the violation points after negotiations rounds.

With the proposed algorithm, all high priority violations were cleared. The total violation points did not decrease to zero, which means that all of the professor agents were not perfectly satisfied, although the violation points were evenly shared among the professor agents. The actual class schedule results the system produced were realistic scheduling plans comparable to actual class schedules.

The proposed multiagent system used a large number of agents with very low level (local) goals, i.e. goals directly related to the agent, and a few agents with medium level goals, i.e. intermediate goals relating to the collection of agents. The system does not have direct control over the highest level goal,

i.e. the final goal of the system. Nevertheless, the multiagent system successfully satisfied the highest level goal or task through the self-organization of low level agents. Through several trials were able to confirm that the resultant organization depended strongly on changes made to the local goals, rather than changes to the intermediate goals. For future works, we intend to investigate more closely how changes in local goals affect the achievement of the self-organization.

For future works, we plan to use genetic algorithm techniques to create and select an optimized class schedule which produces the highest fitness satisfying the various constraints of class scheduling.

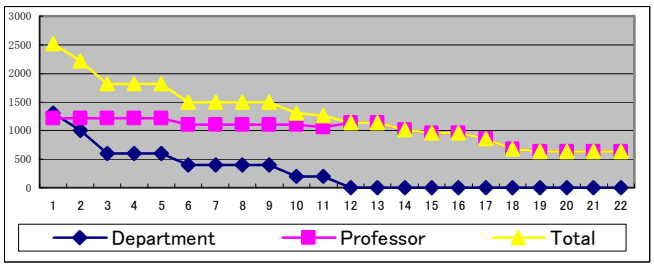

Figure2. Constraint Violation for Department and Professor Agents

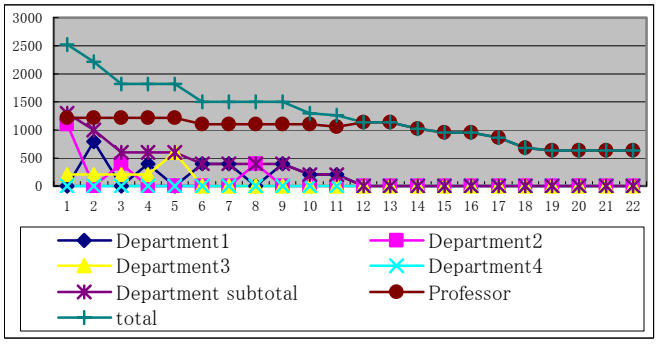

Figure3. Constraint Violation for Department Agents

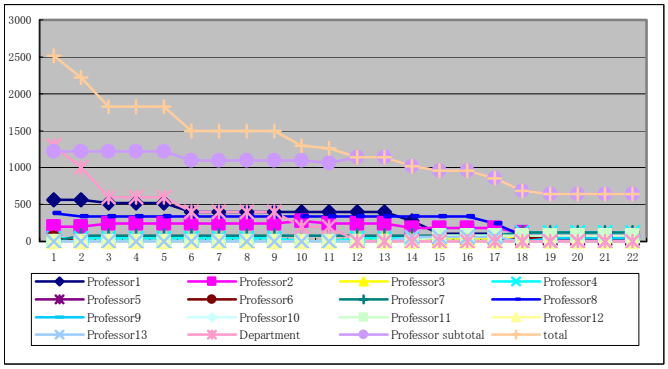

Figure 4. Constraint Violation for Professor Agents

References

[1] B. Horling, B. Benyo, V. Lesser. Using Self-Diagnosis to Adapt Organizational Structures, Proceedings of Agents'01, pp.529-536, 2001

[2] C.V. Goldman, J.S. Rosenschein. Evolving Organizations of Agents, Multiagent Learning : Papers from the AAAI Workshop, pp.25-30, 1997

[3] B.H. Far, H. Hajji, S. Saniepour, S.O. Soueina, M.M. Elkhouly. Formalization of Organizational Intelligence for Multiagent System Design, IEICE Trans. on Information and Systems, Vol.E83-D, No.4, pp.599-607, 2000

[4] H.V.D. Parunak, S. Brueckner. Entropy and Self-Organization in Multi-Agent Systems, Proceedings of Agents'01, pp.124-130, 2001

[5] M. Gianfranco, E. Pessa (eds.). Emergence in Complex, Cognitive, Social, and Biological Systems, Kluwer Academic / Plenum Publishers, New York , 2002

[6] W.C. Oon, A. Lim, Multi-Player Game Approach to Scheduling Problems, Proceedings of ISPAN'02, IEEE, pp.205-209, 2002