A MAS Approach to Course Offering Determination

Fuhua Lin, Alex Newcomb, A. J. Armstrong

This paper has been accepted by the 2012 International Symposium on The Intelligent Campus (IC12) *iCampus,* Theme: Intelligent Educational Environments for the Next Generation, within the 2012 World Intelligence Congress, December 4-7, 2012 Macau, China.

The paper is based on many years' efforts and innovative work by the team led by the applicant during 2009-2011. During that time period, Alex Newcomb was an undergraduate student of University of Alberta and he worked as an USRA student for this research. A. J. Armstrong was a master student of MSc in Information Systems of Athabasca University during 2009-2011.

A MAS Approach to Course Offering Determination

Fuhua Lin Alex Newcomb
School of Computing and Information Systems
Athabasca University
Athabasca, Canada
oscarl@athabascau.ca

A.J. Armstrong
Computer Engineering Technology
Northern Alberta Institute of Technology
Edmonton, Canada
aja@nait.ca

***Abstract*— Course-offering determination (COD) for educational programs is the complex task of deciding what subset of courses an academic department or program should offer in a given academic term or semester. In this paper, we first model COD decision settings, e.g. modeling students as a group of self-interested agents, and then use a group decision-making protocol voting theory to aggregate the preferences of the different participants toward a single joint decision. Finally, we show how agent-to-agent negotiation techniques can be used to offer courses to mutual benefit between the department and the body of the students.**

Keywords: multiagent systems, course-offering determination

I. INTRODUCTION

Course offering determination (COD) is concerned with the optimal assignment of the courses for a specific academic semester by program administration to meet the needs of students within budget and other resource constraints. Students in degree programs have various course selection preferences and priorities. A department of an academic institution may not be able to offer all courses in a program every semester, especially under contemporary fiscal and staffing constraints. Courses are only arranged every other semester or even less frequently. Therefore, the program administrator needs to determine course offering for the coming semester using the past experience, historical course registrations of the program, and the availability of instructors. In the current course offering workflow, after a course delivery schedule for a new semester become available as the registration period is approaching, the student can select courses to be taken in the coming semester. The competing or even adversarial goals of students and the department as well as the mutability of those goals mean that COD is a complex constraint-satisfaction problem. Evidence shows that COD without adequate consideration of the students' needs and priority often results in unsatisfied students, low enrollments, and delayed graduation. Effective COD permits the efficient assignment of limited resources like faculty, labs, and classrooms while satisfying the desires of most students.

The multiagent system (MAS) approach allows the representation of every principal in the system as a single autonomous agent with unique goals and permits decision-making based on the preferences of multiple agents [1]. The agents can react with the needed flexibility to changes and disturbances through pro-activeness and responsiveness [2]. Therefore, the MAS approach is used to solve the COD problem because it has the following characteristics: (1) its optimal solution can change during calculation; (2) The relation between a user and the scheduling system lasts for a long period of time, which features a high degree of repetition, and may count for the possibility of learning by feedback; (3) The course scheduling is time consuming; (4) The process of scheduling for course-offering involves different parties, e.g. program administrators and students. Individuals have particular preferences, resulting in conflicting goals and therefore leading to conflicts of interest between them. These conflicts should be resolved in a fair cooperative decision making manner; (5). Program administrators consider job markets and the unpredictable nature of preferences students, which generate the need in course scheduling to adapt fast and flexibly to environmental variables and their changes.

II. LITERATURE REVIEW

The majority of Artificial Intelligence applications in education address pedagogical tasks related to tutoring and personalized instruction, such as [3-4]. Significantly fewer publications are related to the decision-making or administrative tasks in education such as school choice [5], academic advising [6], and course timetable scheduling [7].

The MAS approach has proven an important and effective framework for intelligent educational systems, for example iHelp [8], program planning [9], Time Table Scheduling [10], and personalized study planning [11]. To the best knowledge of the authors, there is no significant extant work on solving the COD problem. In this paper, we first model COD decision settings, e.g. modeling students as a group of self-interested agents, expressing their preferences [12] and then use a group decision-making protocol --- voting theory [13] to aggregate the preferences of the different participants toward a single joint decision. Finally, we show how the negotiation techniques [14] is used to generate a set of course offerings for the next term that reflects those preferences, academic requirements, and economic necessities to mutual benefit between the department and the students.

III. SYSTEM ARCHITECTURE

We designed a MAS system that consists of an administrator (AD) agent, a group of student (SA) agents, and a student representative (SR) agent. This correlates with an academic program, where a course scheduling process is initiated by having the program administrator determine the

priority of courses available in the program, based on expressed student needs, preferences and goals. The agents have distinct areas of concern and intent, but collectively interact to generate a set of recommendations for courses to be offered that will be satisfactory to most students while fitting within the operational limitations of the offering program. The interactions amongst our agents are depicted in Figure 1.

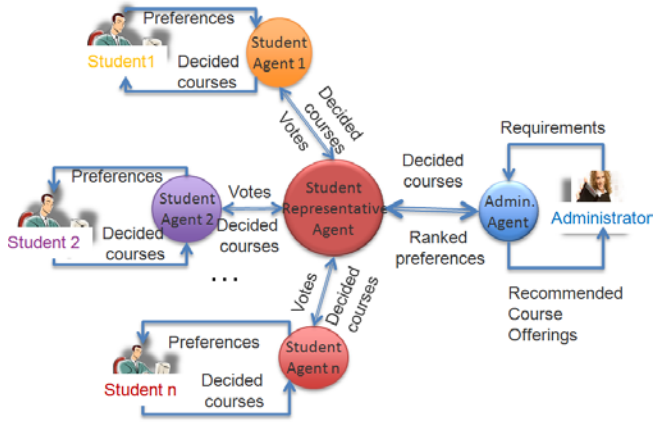

Figure 1: Interactions among the agents.

A. *Agent Types*

SA Agent: An SA agent has three major responsibilities: (1) representing the student principal's interest in interactions with other agents; (2) generating plans for their principals; and (3) generating course selection requests for their principals.

SR Agent: An SR agent is instantiated for any identifiable student group with shared interests and resources. Typically, this would be one per academic program. The SR agent has the following major responsibilities: (1) Managing voting among student agents, ensuring fairness. (2) Representing the student body to other agents, particularly the AD agent.

AD Agent: The AD agent is the representative of the program administration's calculating the needs of the academic department as determined by factors such as course delivery policies, budget, and resource availability. It is responsible to: (1) Provide executive control and oversight for the system; (2) Enforce resource and other course availability constraints; (3) Inform other agents of those constraints; (4) Negotiate with the SR agent to provide an optimal set of course offering recommendations to the offering academic program.

B. *Agent Interactions*

Within this system, for each semester, before the course registration starts, the program administrator of the department delegates the task of initial selection to his/her AD agent, which performs this task using course dependency graphs, past offerings, and departmental obligations. The AD agent collects requirement information from the program administrator, which determines a set of required courses (C_0) for the next term as well as the proposed budget for course delivery in the next term. At the same time, each SA agent generates a study plan based on the program study preferences of the student, identifies "*ready-to-take*" courses of the student, and captures course selection preferences of the student. Once all SA agents complete the actions mentioned above, they send their votes to the SR agent. And then the SR agent aggregates the votes and generates the set of all ranked preference ordering over courses as a group decision. Once the AD agent has this information, it initiates a one-to-one agent negotiation with the SR agent.

IV. Vote Generation by SA Agent

A. *Preference representation*

Our initial investigation was through interviews and anecdotal discussions with current students, staff and faculty. We had strong indications that the desirability of a particular course was not independent of other course's availability. It appeared that course selection preferences are more properly thought of as conditional preferences. Furthermore, there were a variety of ways and degrees of complexity that individual students might express the way in which they determine which courses they want to take in the upcoming term.

We identified three distinct models of how students described their preferences with regard to course offerings, which we have termed: *precedence*, *grouping*, and *progression*. The first, *precedence*, simply referenced a most-preferred course, a next-most preferred, etc. However, after a comparatively short list of courses, the students lapse into a don't-care state along the lines of "if none of those are offered, then it doesn't matter". This sort of preference model is common in cases where the student only plans on taking one or a very few courses. The next model was more common amongst students planning on taking several courses: *grouping*. In this case, students express their desires in terms of sets - a group of courses is desired en masse, and if not all courses are available, and then the remainder are less desirable. Finally, there is the case of those students that plan their program *progressions* sequentially – their desire is to complete some set of courses, then progress to another set, etc, which we have termed the progression model. This sort of preference closely approximates the way in which academic departments model academic progression and is common in the case where students are in a full-time degree program or wish to systematically complete a program. This progression model is also most similar to that expressed in the CP-net [13].

B. *Generating Fractional Votes*

Based on the preferences expressed by the principal on one or more of our preference models, the SA agent automatically generates a set of fractional votes for each round of an election. Prior to the voting process, the SA agent determined the list of all courses in the program that that student can legally and preferably take, which consist of all incomplete courses of the student for which all pre-requisites have been completed. The SA agent receives a list of courses from the AD agent that will not be offered. The SA agent also receives a list of courses that will already be recommended from the AD agent. These are similarly removed from the "legal" course set –but are treated as though they will be offered for purposes of other decisions.

From the progression interface, all legal courses that are either in the first block of a progression or whose predecessor block contains only courses that have already been taken are added to the list. From the grouping interface, all courses from groups in which all member courses can legally be taken are added to the list. From the precedence interface, the highest ranked course that can be preferably and legally taken is added.

Each of the courses on the list then gets a relative fraction of the agent's single vote for that round, with repeat courses getting a proportionately higher portion of the vote. The vote is recalculated for each round, and can be affected by the results of previous votes. The use of multiple fractional votes also provides another advantage – it makes the systems sufficiently computationally complex that it is resistant to manipulation [14].

C. Election Protocol

For each round of agent negotiation, the SR agent uses the dynamic information about student preferences and the current state of negotiation to generate an ordered list of desired courses — corresponding to a vote — for the next term from the set of courses that each student could possibly take provided by the AD agent, as determined by course dependencies and program selections, as well as by the set of courses that it has already been determined will be offered.

For each round of agent negotiation, the SR agent collects the updated ordered list of desired courses from all the participating student agents and determines the aggregate student preferences for the near-term course-offerings. To this end, we construct the SR agent behavior for coordinating the election process with an algorithm based on Single Transferable Vote (STV) algorithm [11]. The reason for using STV is as follows: (1) the SA agents vote for the courses they want, but they will often have several courses that will work for them, so using a straight one-vote-for-one-student-for-one-course system does not adequately represent the actual desires of students. Alternatively, having the students vote for each course they can take will produce an overabundance of votes, which might give a misleading picture of which courses will actually be taken in the upcoming terms, and does little to address the preferences of the student. A system would need to consider all the courses available to students, while still taking into account the students course preferences. (2) It has been shown that STV, in comparison with other voting schemes in actual use, is computationally resistant to manipulation [16].

D. STV-based Election Process

The overall STV-based election process is managed by the SR agent. It is a multiple-round plurality election. In each round, electors revise their votes through partitioning their votes across multiple candidates using previous results in formulating a new vote:

L_0: the set of courses that will be offered, regardless of student preferences – perhaps because of contractual obligations or because a course has not been offered for some time. L: the ordered list of courses. r: the number of rounds for the election process.

A1: [Initialization, SR agent] $r \leftarrow 1$; $L \leftarrow L_0$.

A2: [SA agents] do the following two steps **until** either the list of courses reaches some pre-determined maximum number **or** until no agents are still voting:

A2.1: [all SA agents re-vote] re-generate their set of fractional votes and send them to the SR agent.

A2.2: [SR agent aggregates preferences of SA agents] Find the course with the highest number of votes ("wins"), denoted as *c*; $L \leftarrow L + \{c\}$; $r \leftarrow r + 1$.

Because agents cannot vote for courses that are already being offered, in the worst case the voting will proceed until all the courses that the program could offer have been ordered by desirability. There are a number of rules that help to ensure that this protocol is fair and will always terminate, for example, "*there are a finite number of courses that can be voted on*." As there are finite courses available for voting on, and this set is reduced each round, the system will always converge on a single recommended list. Because the number of rounds is strictly determined by the number of courses, the system's complexity is linear on that axis. It is further linear with the number of students – even for a large number of courses or students, this model is computationally feasible.

V. NEGOTIATION PROTOCOL

We developed the negotiation protocol that governs the interactions between the AD agent and the SR agent for determining a recommended set of course offerings in several heterogeneous steps:

(1) Determining Non-Negotiable Courses Initially, the AD agent identifies a set of courses C_0 that must be offered (irrespective of negotiation) and the initial number n_0 of courses that should be offered, as governed by budget constraints.

(2) Determining not-to-offer courses C_{-1} The AD agent also prepares an excluded list containing courses that absolutely may not be offered (C_{-1}).

(3)Determining Negotiable Courses Once the required courses have been identified and the initial budget target has been determined, this information is presented by the two agents as an initial position. The negotiation then proceeds iteratively between the two agents, the AD agent attempting to maximize the course enrolments and minimize delivery overhead, and to maximize staff efficiency of the offering, and the SR agent attempting to maximize the availability of courses desired or needed by students to complete their programs. Consequently, the SR agent will primarily agitate for the inclusion of all courses that students have expressed a desire to take, in order of popularity (as determined by voting). Based on a monotonic concession protocol [16], a negotiation protocol to determine negotiable courses is developed.

A. Disutility function of the AD agent:

The AD agent calculates the (dis)utility of the current offering list according to the formula below:

$$U_{AD} = \left| \sum_{n=1}^{n_c} Cost(c_n) - Cost_{ideal} \right|$$

where n_c is the number of courses in the offering list and $Cost(c_i)$ is the cost of delivering course c_i. $Cost_{ideal}$ is a putative ideal expenditure on course delivery, typically governed by the departmental budget. It is assumed that departments will prefer to be on-budget rather than to under- or over-shoot the target, so the utility function reflects this. Obviously, this function might be modified to meet the particular circumstances of a delivering department. It should also be noted that in this case, the goal is to minimize the value U_{AD}, which can thus be thought of as a disutility value.

In addition to the particular (dis)utility of a given offering, the AD agent has a set variance from the ideal that it will accept as satisfactory (the amount that a given offerings cost can vary from the ideal budget amount, positive or negative). This value U_{AD} determines the value of a simple satisfied predicate (A_{sat}) that is simply the truth of the statement $U_{AD} \leq \Delta U_{AD}$. In iterations after the first iteration, the current offering list may be updated by adding the content of the proposal of the SR agent, if this does not cause the (dis)utility (U_{AD}) to rise. The fixed excluded list, current offering list and the satisfaction predicate A_{sat} for that offering list is transmitted to the SR agent at the beginning of each iteration of the negotiation.

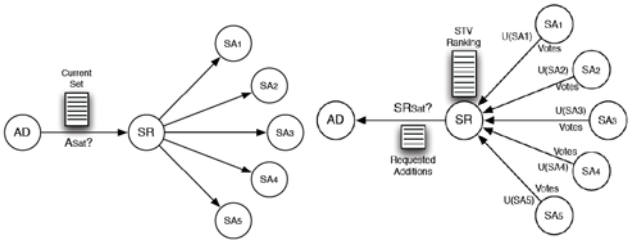

Figure 2: Message passing (a) Downstream; (b) Upstream.

B. Disutility function of the SA agent

Given a list of offering courses, a SA agent compares the preference (vote) of the student to calculate its (dis)utility value. For example, we can define it as

$$U_{SA} = \sum_{m=1}^{j} (R_m - m),$$

where j is the number of courses the student is planning on taking in the upcoming term. R_m is the rank on the current offering list of the m_{th} course in the vote of the student that is ranked by the vote value. In the case that no course from the current offering is represented in the preference list, a value one greater than the total number of courses in the preference list is used. The SA agents report U_{SA} back to the SR agent. They also provide a vote for the STV process that will be used to generate the next offering in the negotiation (if there is one). This vote is the ordered list of all courses the student agent could potentially take in the next term, less any courses currently in the offering list (as they will already be offered) and any courses in the excluded list (as they will not be offered). This vote and the current disutility for that student agent commence the initial upstream message passing (see Figure 4b).

C. Negotiation process

The SA agents initiate the upstream messages to the SR agent. These consist of U_{SAi} the current offering list, and the vote list to be used in calculating the next proposal. The SR agent receives these values and calculates a collective disutility based on the statistical content of U_{SAi}. Our concern was that the collective disutility not merely reflect the average disutility experienced by the various SA agents, but also the degree to which individual agents varied from that average. For example, a low average disutility which nevertheless had a few agents that were extremely dissatisfied might not be preferable to a slightly higher disutility where all agents were near that value. Consequently, we added a term to the collective disutility to capture the variance in reported values, as measured by the standard deviation of the values:

$$U_{SR} = l_1 \bar{x} + l_2 \sigma ,$$

where U_{SR} is the collective disutility, $\bar{x}$ the arithmetic mean and σ the standard deviation of the reported U_{SAi}. The weighting constants l_1 and l_2 ($l_1 + l_2$=1) are used to tune the contribution of each component to the calculated disutility.

In a manner similar to the calculation done for the AD agent, the SR agent determines a satisfied predicate, SR_{sat} which indicates whether the aggregate disutility is less than the permitted variance $SR_{sat} = (U_{SR} \leq \Delta U_{SR})$. If SR_{sat} is true, the SR agent is satisfied with the current offering.

The SR agent also calculates the courses for the next proposal even if it is already satisfied. The number of courses for the STV is either a standard initial value, the size of the most recent accepted proposal to the AD agent, or one less than the most recent rejected proposal to the AD agent. This allows the protocol to converge more rapidly on an acceptable solution by initially jumping by several courses before dropping down ultimately to one course at a time for the final negotiations. It also allows us to use STV more optimally, as its value is more readily apparent in elections where several candidates will be elected.

The SR agent sends its current satisfaction value to the AD agent, along with the current proposal, which is the output from the STV election for the specified number of candidates.

The AD agent recalculates U_{AD} based on a projected course list that is the sum of the current offering list and the courses in the proposal. If that disutility is smaller than the previous disutility, the proposal is accepted and added to the current offering list. This step occurs even if both agents are already satisfied, to allow the possibility of a final improvement on disutility before concluding negotiation.

If A_{sat} is true and the SR agent has indicated that the last round satisfied it, the negotiation is concluded and the AD agent records the final offering list and sends it to the SA agents. If the A_{sat} is false, the negotiation will always continue. If the AD agent is satisfied, but the SR agent is unsatisfied, the negotiation will continue until the count of courses in a proposal reaches zero (recall that the SR agent will iteratively decrease the number of courses after a rejected offer). In the case that the AD agent still declines to add a course (rejects even a one course proposal) and the A_{sat} is false (which is only

possible in the case that an earlier iteration overshot by adding too many courses), the SR agent must continue by submitting offers that involve removing courses. It does so by iteratively holding an election with only the current offering list courses as candidates. The loser (first eliminated course) is sent as a removal proposal. This continues iteratively until the AD agent indicates satisfaction. Once the count of courses in a proposal reaches zero, the SR agent must accept the last proposal that satisfied the administrator, even if this is the required courses only. In this way, the negotiation converges fairly rapidly to a set that is near the budgeted amount for the department, while trying to add courses most likely to fit into the plans of the majority of students.

VI. EXPERIMENTS

To evaluate the proposed approach, a prototype was implemented with JADE (http://jade.tilab.com), which allows simulating different scenarios by varying several parameters, such as the course number of the program, the divergence of preferences for course selection, or the must-offer courses due to emergence cases. We simulated a collection of MSc IS program students shown in Table 1 consisting of 88 students. Given that a course set $C = \{c_i\}_{i=1}^{n_c}$ to be offered in a semester in the program, the actual cost to be paid for the course offering is calculated as follows:

$$U_{AD} = \left|Cost(C) - Cost_{ideal}\right|, \qquad Cost(C) = b(n_c) + r\sum_{i=1}^{n_c} h_i \ ,$$

where b is the base salary for one course to be paid to the instructor for the course, e.g., b = \$5000; r is the amount of money to pay to the instructor for one student, e.g. r = \$500; h_i is the number of the students who will take course c_i for $1 \le i \le n_c$, and C_{ideal} =\$75,000.; n_c = 88 ; C_0 = {c_{501}, c_{503}, c_{504}, c_{601}, c_{695}}; C_{-1} = {c_{602}, c_{604}, c_{617}, c_{636} c_{637}, c_{682}}; ΔU_{AD} = \$100; ΔU_{SR} = 3.

Table 1: simulation results

Round		R_2		R_3		R_4		R_5	
ci	H_1	V_2	H_2	V_3	H_3	V_4	H_4	V_5	H_5
c_{501}	**23**	0.2	12		9		9		9
c_{503}	**17**	0.2	14		13		13		13
c_{504}	**11**	0.2	9		8		8		8
c_{601}	**20**	0.2	15		13		11		11
c_{695}	**6**	0.5	6		6		5		5
c_{607}		0.2		0.2		0.2		1.2	
c_{605}		5.0		**12**	10		10		10
c_{602}		1.6		1.6		1.6		1.6	
c_{610}		0.7		0.7		0.7		1.7	
c_{648}		2.0		2.0		**2.0**	4		3
c_{691}		**60.**	64	0.5	64		64		64
660		1.0		1.0		1.0		1.0	
c_{689}		1.0		1.0		1.0		**2.0**	11
c_{667}		1.7		2.0		**2.0**	12		12

In R1: $C_1 = C_0 \cup \{c_{602}, c_{605}, c_{607}, c_{610}, c_{648}, c_{660}, c_{667}, c_{689}, c_{691}\}$. Getting h_i shown in column H1 of Table 1. $U_{AD}(C_1)$ = \$11500.0. In R2: c_{691} is chosen in this round, with a voting result of 60. Getting h_{691}= 64 shown in H2 in Table 1 since it is the first course chosen, and many students have it as their first pick due to it being on many plans and not having many prerequisites. $U_{AD}(C_1)$= \$15000.0. Comparing C_{ideal}, the Administrator is still not satisfied with required courses. The SR agent is also unsatisfied with course offering as SR Weight: 10.31. After R3 and R4, in R5, the required 9 courses are: C_0, c_{691}, c_{605}, c_{648}, and c_{667}. The votes for this round are shown in V5 of Table 1. c_{689} is chosen in this round, with a result of 2.0. U_{AD} = \$48,000. So the AD agent is still unsatisfied with the list. The SR agent is now satisfied with course offering. The SR agent's weight is 0.86. Negotiation concluded with a list: $C_1 = C_0 \cup \{c_{691}, c_{605}, c_{648}, c_{667}\}$. The size limit of a class is not considered.

VII. CONCLUSIONS

COD is a complex task that requires the ability to search, schedule, and coordinate a set of participating entities under various constraints and uncertainties. We show that the MAS approach is an enabling technology to automate the dynamic COD through voting mechanism and negotiation. This paper focuses on the architecture, mechanisms of preference aggregation, and agent negotiation. This architecture provides several advantages:

(1) uncoupling the implementation of discrete or competing concerns;

(2) using relatively simple agents to combinatorially generate complex emergent behavior through their interactions;

(3) encapsulating the representation of a single human principal's preferences and goals into a single agent; and we can simulate the formal interactions amongst human principals—and produce similar results without requiring those interactions to take place.

The use of the STV-like protocol permitted us to implement a single agent that served as a representative for the entire student body, communicating with the individual SA agents that represent their specific human principals. The effectiveness of the approach has been demonstrated via the experiments. Real life implementation of these methods in undergraduate and graduate programs of educational institutions can be expected in the near future.

ACKNOWLEDGMENT

We thank Natural Science and Engineering Research Council (NSERC) of Canada and Athabasca University of Canada for the financial support to the research. Also, we thank anonymous reviewers for their constructive comments.

REFERENCES

[1] V. Conitzer, "Making decisions based on the preferences of multiple agents." Communications of ACM, Vol. 53, No.3, 2010. pp. 84-94.

[2] N. R. Jennings, "An agent-based approach for building complex software systems." Communications of ACM, Vol. 44, No.4, 2001. pp. 35-41

[3] A. Graesser, P. Chipman, B. Haynes, and A. Olney, "AutoTutor: an intelligent tutoring system with mixed-initiative dialogue." IEEE Transactions on Education, Vol. 48, No.4, 2005, pp. 612–618.
[4] A. Mitrovic and S. Ohlsson, "Evaluation of a constraint-based tutor for a database language." International Journal on Artificial Intelligence in Education, 10, 1999. pp. 238-256.
[5] D. C. Wilson, S. Leland, K. Godwin, A. Baxter, A. Levy, J. Smart, N. Najjar, and J. Andaparambil, "SmartChoice: An Online Recommender System to Support Low-Income Families in Public School Choice." AI Magazine, Vol. 30, No.2, 2009, pp. 46-58
[6] F. Lin, Kinshuk, R. McGreal, S. Leung, D. Wen, F. Zhang, Q. Li, and X. Liang, "e-Advisor: A Web-based Intelligent System for Academic Advising." International Transactions on Systems Science and Applications, 4(1), March 2008, pp. 89-98.
[7] M. Oprea, "MAS UP-UCT: A multi-agent system for university course timetable scheduling." International Journal of Computers, Communications & Control, II(1), 2007, pp. 1024–1020.
[8] J. Vassileva, G. McCalla, and J. Greer, “Multi-Agent Multi-User Modeling In I-Help,” User Modeling and User-Adapted Interaction. Volume 13, Issue 1-2, February -May 2003, pp. 179 - 210
[9] M. S. Hamdi, "MASACAD: A Multiagent-Based Approach to Information Customization." IEEE Intelligent Systems, Vol. 21, No. 1, 2006, pp. 60-67.
[10] M. P. Tariq, M. W. Mirza, and R. Akbar, "Multi-agent Based University Time Table Scheduling System (MUTSS). International Journal of Multidisciplinary Science and Engineering, Vol. 1, No. 1, September 2010, pp. 33-39
[11] A. Vainio, and K. Salmenjoki, "Improving Study Planning with an Agent-based System." Informatica, 29, 2005, pp. 453–459.
[12] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets: A tool for representing and reasoning with conditional *ceteris paribus* preference statements", Journal of Artificial Intelligence Research (JAIR) 21, 2004, pp. 135–191.
[13] J. J Bartholdi and James B Orlin, Single Transferable Vote Resists Strategic Voting." Social Choice and Welfare, 8, 1991, pp. 341-354
[14] J. Lang, "Vote and aggregation in combinatorial domains with structured preferences." In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), Rajeev Sangal, Harish Mehta, and R. K. Bagga (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 1366-1371.
[15] T. Walsh, "An Empirical Study of the Manipulability of Single Transferable Voting," Proceedings of the 2010 conference on ECAI, IOS Press. 2010, pp. 257-262
[16] J. S. Rosenschein and G. Zlotkin, Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, 1994, MIT Press.