ALLOCATIVE AUCTIONS: AN APPLICATION CONTEXT FOR CSCW AND INTELLIGENT AGENTS

HELEN MENG*, YUK CHI LI, KA KIT LAU AND YUN WING LEE
Human-Computer Communications Laboratory
Department of Systems Engineering and Engineerng Management
The Chinese University of Hong Kong
Shatin, N.T.,
Hong Kong, China
E-mail: hmmeng@se.cuhk.edu.hk*

This paper describes an online auction system (OAS), designed for the allocation of *M* items among *N* parties. We strive to distribute one item per party while maximizing the overall preference of all particpants. The specific application context is in the distribution of student projects among student teams. OAS is intended to support group decision-making among the teams, in distributing their preferences among projects. The system offers several advantages as a solution to our allocation problem. First, preference points are used to elicit preference levels of student teams for the projects. This scheme offers greater flexibility and granularity than conventional rank ordering. Second, users participate actively in bidding, using partial knolwedge of the other teams' preferences. This should help ameliorate situations in which most preferences coalesce into a few popular projects. Third, the final allocation is chosen to maximize the *average normalized score*—a metric which reflects the overall preference level. Finally, OAS is a groupware which facilitates CSCW for time-synchronous but possibly geographically-distributed usage. We have enhanced the system such that a user can freely choose to bid in person, or delegate the task to an intelligent agent by specifying the bidding strategy through a prescribed set of parameters.
Keywords: intelligent agents, CSCW, groupware, online auctions.

1 Introduction

This work tackles the problem of distributing *M* items among *N* parties, where $M>=N$. The objective is to assign a single item per party while maximizing the parties' collective preference. This allocation problem frequently arises in real life, e.g. [1]. In our specific application, there are *M* student projects which have been conceived by professors, and these projects need to be assigned to *N* student teams according to the teams' preferences. Team formation was previously completed in accord with the students' desires. We find that this allocation task arises quite frequently over the course of an academic year. Hence it is desirable to have an effective and efficient solution for the task.

A conventional method of tackling this problem proceeds as follows: Each student team is asked to submit several prioritized choices (e.g. 6 projects, in rank order). The submissions are gathered and allocation is done manually. In the event that the majority of specified preferences coalesce to a few popular projects, allocation essentially reverts to a lottery. Under such circumstances, there is little provision for maximizing the overall preference of the students.

We believe the prevailing procedure can possibly be improved in three ways: First, there can be a better scheme for the elicitation of preferences. Second, students can be more actively involved over the course of the allocation process. Third, we can use a more rigorous evaluation metric in chosing among possible allocation alternatives. With these elements in mind, we propose to use an auction process for our allocation task [2,3].

2 Proposed Solution: The Online Auction

We believe that a possible solution to our allocation task is an *online auction*. Each student team may initialize the auction with a preliminary investment of preference points. The bidding session can then proceed for a prescribed time duration. During the session, each team may observe the distribution of preference points for all the projects, coming from all the teams. Having acquired partial knowledge regarding competition, the teams can alter their preferences dynamically, and these changes are immediately updated and revealed to all. We have chosen to implement a *sealed-bid auction* [4], i.e., the preference points assigned by the student teams for the projects are publicly known, but team identities are not. The online auction offers improvement along three dimensions:

(i) *Enhance the elicitation of preferences*

We replace the rank order of preferences by numeric preference scores. In the rank ordering scheme, student teams are allowed 6 slots for specifying their preferred projects. Each slot can only accommodate a single project. The top ranking slot is the most preferred, while the bottom ranking slot is the least preferred. This scheme cannot easily support equal preferences for multiple projects. Furthermore, due to the discrete nature of rank ordering, it is difficult to express variations in the incremental preference between adjacent slots. These problems can be alleviated by the use of preference scores. Each student team is allotted the same number of total scores / points (e.g. 1000 points). The team can freely invest the points in their preferred projects. Multiple projects can receive the same number of points. The new scheme can also differentiate between high and low preferences with greater granularity. Furthermore, student teams may invest in more than the prescribed number (i.e. 6) of projects, should that be indicative of their preference profiles.

(ii) *Support dynamic changes in preferences*

As mentioned previously, we often encounter situations in which all selections coalesce into a few highly popular projects, despite having a broad selection of projects. We believe that it is desirable for the student teams to be aware of their competitors' selections, as some teams may opt to redistribute their preference points in order to retain greater control of which project they will likely be assigned. During the bidding session, student teams can dynamically revise their preferences point investments. The changes are also updated and sent to all participants in real time.

(iii) Utilize rigorous evaluation for possible allocations

A rigorous criterion can be used to evaluate allocation alternaties. We conceive of two possible metrics to reflect overall preference level. The *average rank metric* (R_{ave}) is defined as:

$$R_{ave} = \frac{\sum_{i=1}^{n} R_i}{n}$$

where n is the total number of student groups, and R_i is the rank of the project assigned to group i according to the group's preference list. If all the groups are allocated with their first choice, then the average rank equals 1, which is the best-case scenario. If we give a rank of 7 to projects which cannot make the preference list , then an average rank of 7 may be taken as the absolute worst-case scenario (a loose lower bound).

The *average normalized score* (S_{ave}) is defined as:

$$S_{ave} = \frac{\sum_{i=1}^{n} \frac{S_i}{Max_i}}{n}$$

where S_i is the number of preference points invested by group i on its assigned project, and Max_i is the maximum number of preference points invested by group i on a single project. If all the groups are allocated with their top-choice projects, then S_{ave} = 1, which is the best-case scenario. On the other hand, if all groups are assigned projects which are they have not invested an points, then the S_{ave} = 0, which is the absolute worst-case scenario (again, a loose lower bound). We favor S_{ave} over R_{ave} as it offers greater granularity in capturing preference specifications.

As we can see, the online auction facilitates a *group decision process*. The preference profiles at the end of the bidding session are used for computing S_{ave}. A search algorithm has been implemented to find the “optimal” allocation among allocation alternatives.

3 Online Auction System: Groupware Implementation

We have implemented an online auction system (OAS) to facilitate this group decision process, which strives to obtain an "optimal" allocation of projects to student teams. Therefore OAS is a groupware which supports CSCW (Computer-Supported Collaborative/Competitive Work) [5].

The system architecture of OAS is shown in Figure 1. The client/server

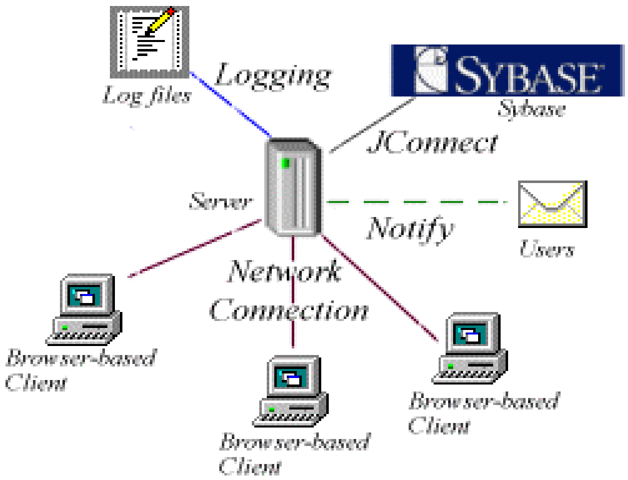

Figure 1. The client/server architecture of the Online Auction Ssystem (OAS).

architecture allows multiple clients connect via TCP/IP to the server. Clients are browser-based and utilize Java applets for socket connections. This facilities clients to run from anywhere but can still connect to the server easily. Two server sockets are used for every client connection – one for input, and the other for output. Multiple-client concurrency is handled by forking the server (on the Unix platform). With this setup, our OAS groupware supports CSCW of the *synchronous-distributed* category in the time-space matrix [5], i.e. all student teams need to be logged in at the same time for a bidding session, but they can log in from different locations. User (student team) information and preference profiles are stored in the Sybase database, using JConnect as our middleware, and Transact-SQL for communication.

OAS supports a number of features and functionalities:

- *User login:* OAS requires each student team (user) to register for an account prior to the auction. Users receive account confirmation via email notification (see Figure 1). The account is protected by a password, which is used for user authentication at the auction.
- *Timer*: OAS monitors the time elapsed for a bidding session. A counter program is launched at the beginning of the auction, at which time all student teams may begin bidding. A checker program monitors the time elapsed and freezes all actions when bidding time is up. Hence the timing for the bidding session is controlled automatically.

- *Bidding status update:* OAS supports continuous, real-time updates of the bidding status. The top *K* bids (where *K* is a parameter set by the user) for the projects of interest (again, user-specified) are continuously passed from the server to the entire group of users. The information is displayed graphically by the clients' applet as bar charts.
- *Preference points revisions:* OAS allows users to revise their bids at any instant during the bidding session. Changes from the user are immediately entered into the database, and the bidding status is updated for the *entire* group of users. We have carefully avoided these two concurrent processes from fighting for the same resource by means of the Peterson's algorithm for critical sections [6]. The client's applet also performs summation to ensure that the total preference points lies within the maximum allotted.
- *Auction log:* OAS will time-stamp and log every action during the bidding session into an ASCII file (see Figure 1). The log preserves valuable information for us to analyze bidding behaviors / strategies after the auction.
- *Graphical user interface:* OAS has a GUI with many features. The interface supports all the functionalities mentioned above, and is specially designed to provide sufficient feedback to the user at all times. There are a number of adjustable parameters for the screen layout, e.g. for zooming, and for selecting the number of bar charts to display. There are also many pop-up windows for notifications, reminders and errors. A screen shot of the bar chart display is shown in Figure 2.

4 Intelligent Bidding Agents

While our online auction calls for all student teams to participate simultaneously for the entire bidding duration, the task of bidding can be delegated to an intelligent software agent. Users can freely switch between bidding in person and bidding via agents at any time during the auction. We have incorporated software agents as a key feature in the OAS system. These agents serve to monitor the auction status, and bid for the projects according its user's specifications. In OAS, the software agents are implemented in Java. They have the following characterisitcs [7,8]:

(i) *Autonomy*: Once an agent is launched by a user, it can bid autonomously without the user's intervention.

(ii) *Continuity*: The agent is a continuously running process, which stops only upon user termination.

(iii) *Goal-oriented:* The agent aims to maximize the user's preference points in seizing the available project, and hence would try to outbid any competitiors.

(iv) *Reactivity*: The agent continuously monitors the auction status, and react accordingly to accomplish its goal.

(v) *Initiative*: The agent will also take initiative in bidding, which is part of its goal-directed behavior.

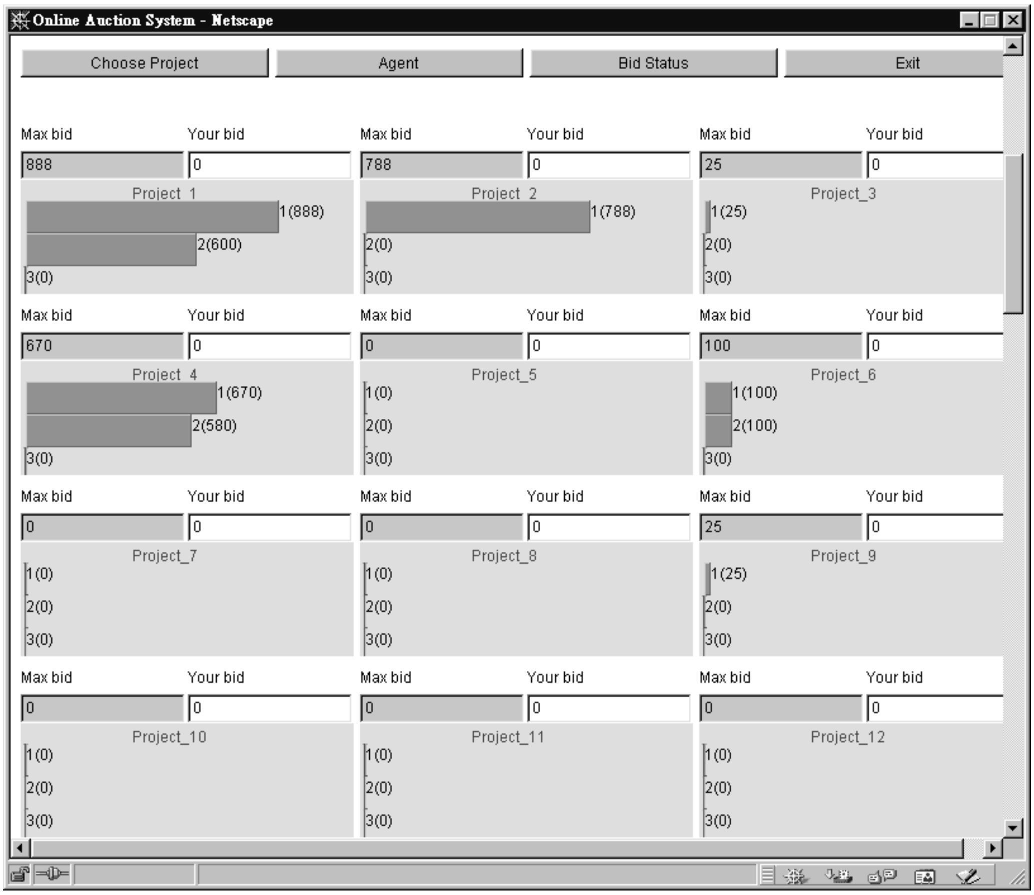

Figure 2. A screen shot of the bar chart display as part of the GUI for OAS.

4.1 Agent Parameters

Users are able to configure their agents to bid on their behalf, and in a manner (or with a strategy) they desire, by setting four parameters. [1]

(i) *Project priority*: The user needs to rank order its preferred projects. The agent will focus on the available projects as prioritized.

[1] A special GUI was also designed for these agent settings.

(ii) *Maximum project points*: The user also needs to assign the maximum number of points for each project. As such the agent's investment for that particular project cannot exceed the limit. The maximum for each project should not exceed the total number of points (e.g. 1000 points) allotted to the student team / user.

(iii) *Bid reallocation strategy*: At a given instant, the agent is focused on the *available* project of highest preference. It will try to retain a firm hold on the project by outbidding any competitors. Should the agent need to divert additional points to outbid a competitor, it may either choose to divert points from the lowest ranking projects; or from others which have similar priorities as the one in focus.[2]

(iv) *Bid increments:* Bid increments for the agent are also user-specified, and depends on the time available until the finish line. The default is set to 1 point. However, it is conceivable that towards the end of the auction, the agent should take an increasingly aggressive attitude in outbidding any competitors, e.g. 10-point increments with 10 minutes left, 50 with 5 minutes left, etc.[3] The timer is implemented in JavaScript, and communicates with the client's applet by means of Netscape's "live-connect" technique.

4.2 An Example

As an example, consider the following parameter settings for an agent (which can reallocate its bids from other projects not in focus):

Project	Priority (Rank)	Maximum Project Points
1	1	800
2	2	700
3	3	600

Also consider the current auction status:

Project	Highest Bid (other teams)	Project Bid (agent's team)
1	900	800
2	300	0
3	400	0

For the agent's team, Project 1 has the highest priority among the three projects. However, there exists another team which has placed 900 points for Project 1, and this exceeds the maximum (800 points) allowed for the agent to invest in Project 1.

[2] For this latter case, the diversion of resources targets the projects from high to low priorities.

[3] Again, a special GUI is designed for setting bid increments.

Consequently, the agent shifts its focus to win the project of second highest priority, i.e. Project 2. In order to outbid another team with 300 points on Project 2, the agent needs to invest its 200 points available, as well as divert more than 100 points from elsewhere. With the default increment set at 1, the agent can reallocate its preference points, so that Project 1 is left with 699 points, and Project 2 receives 301 points – enough to win over Project 2. However, as long as the priority of Project 1 stays above that of Project 2, the agent remains vigilant about the running bid of Project 1. Should it decrease below the maximum, the agent will immediately divert resources in an attempt to seize Project 1.

5 The Final Allocation

OAS supports the entire auction process, with *multiple* users and *multiple* agents bidding for the projects. Preference points at the end of the bidding session are used for allocation. We may compute the S_{ave} for all allocation alternatives, and select the one with the highest S_{ave}. However, this is impractical due to the large size of the search space. For example, in a typical scenario of (*M=57)* projects and (*N=26)* student teams, ${}_{M}P_{N}=N!/(N-K)! = 4.9x10^{42}$! Therefore we have implemented a depth-first search algorithm, to find the "optimal" allocation, according to the S_{ave}.

6 Experiments

We have run three simulations using OAS, scaling up our experiments gradually. Furthermore, we have also used OAS in a real project assignment task .

6.1 The First Simulation

Our first simulation involved 5 parties (3 agents and 2 users), bidding for 8 projects. The auction was 15 minutes in duration, and the experimental parameters and results are tabulated in Table 1.

Participants	Project Rankings	Bid Increments t>10, 10>t>5, t<5	Final Allocation (Winning Bid)
Agent 1	1>2>3>4>5>6	2, 3, 5	1 (785)
Agent 2	8>7>6>5>4>3	9, 20, 120	8 (7)
Agent 3	6>8>4>3>1>2	4, 8, 12	6 (100)
User 1	1>3	--	3 (220)
User 2	5>4>6	--	5 (801)

Table 1. Results from the first simulation experiment

The bid increments were partitioned into three categories. (t>10), (10>t>5) and (t<5) represent the times when there are more than 10 minutes, between 5 to 10 minutes, and fewer than 5 minutes until the end of the auction. Project rankings in

Table 1 do not show severe collisions. Four of the five bidders should be able to get their first choice, while the remaining one should be able to get their second choice. Agent 1 managed to outbid User 1 to win over Project 3. Hence the agent's performance is rather satisfactory. Overall the allocation achieved an S_{ave} of 0.6814, which corresponds to an R_{ave} of 1.2. We also learnt from this experiment that it is important for the agent to take initiative in bidding. Otherwise it will not start bidding unless some other party has an interest in Project 1 (top choice of Agent 1).

6.2 Other Experiments

The second simulation experiment had 10 participants involving 8 agents and 2 users bidding on 15 projects. Auction duration was 30 minutes. We injected greater competition in the project rankings this time. Projects 1, 3 and 6 each had two top-choice bidders, and Project 9 had three top-choice bidders. Despite the keen competition, OAS achieved an S_{ave} value of 0.6502, which corresponds to an R_{ave} value of 2.0.

The third simulation experiment was scaled up even further. It involved 15 particpants (12 agents and 3 users), bidding on 25 projects for 30 minutes. The final allocation achieved an S_{ave} value of 0.7091 which corresponds to an R_{ave} of 2.07. Analysis of the auction logs suggests that the agent may become more aggressive in bidding, should it lose out on its top-choice project. This suggests that the bid increment may depend on parameters other than time-to-completion.

Finally we adopted the use of OAS in a real task of assigning 53 projects to 17 student teams. All student teams chose to bid in person. Server loading was distributed among 5 machines to expedite system turnaround time. The session lasted 30 minutes, and attained an S_{ave} value of 0.74. Since the project rankings were not specified, we cannot compute a reference value of R_{ave}. However, comparison with the third simulation experiment suggests that the allocation should be satisfactory. The entire allocation process was greatly stream-lined when compared to the conventional manual effort.

7 Summary and Future Work

This paper presents our work in developing an Online Auction System (OAS) as a solution for allocation *M* student projects among *N* student teams, one project per team. The allocation strives to maximize the overall satisfaction among the students. The solution should be applicable to the general class of problems of assigning *M* items to *N* parties.

OAS is a groupware which supports the CSCW of group decision-making. Participants can be actively involved in the distributing their preferences to minimize collisions. OAS is also equipped with intelligent software agents, to whom users can delegate the bidding task. Allocation is also based on an objective preference measure. Our simulation experiments as well as real utilization demonstrates that OAS is a viable and effective solution for allocation. The system also provides a platform for future experimentation with different bidding strategies. In addition, we plan to enhance OAS with facilities to support negotiation among participants (student teams) during the auction, as well as the incorporation of external preferences (e.g. professors' preferences) in the allocation process.

8 References

1. Boyes, W. J. and S. K. Happel, "Auctions as an allocation mechanism in academia: The case of faculty offices." *JEP*, Vol. 3-3, pp. 37-40.
2. Nautz, D., "Optimal bidding in multi-unit auctions with many bidders," Elsevier *Science*, B.V, pp. 301-306, 1995.
3. Alsemgeest, P., C. Noussair and M. Olson, "Experimental comparisons of auctions under single- and multi-unit demand," *Economic Enquiry*, Vol. XXXVI, pp. 87-97, 1998.
4. Noussair, C., "Equilibria in a multi-object uniform price sealed bid auction with multi-unit demands," *Economic Theory*, pp. 337-351, 1995.
5. Shneiderman, B., *Designing the User Interface, Stategies for Effective Human-Computer Interaction*, 3rd edition, Addision-Wesley, 1998.
6. Burns, A. and A. Wellings, *Real-Time Systems and Programming Languages,* 2nd edition, Addison-Wesley, 1999.
7. Chavez, A. and P. Maes, "Kasbah: An Agent Marketplace for Buying and Selling Goods," *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, 1996.
8. Bradshaw, J., ed., *Software Agents*, MIT Press, 1997.