Intelligent Agents in E-Commerce

Thomas Tran
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
ttran@site.uottawa.ca

Outline

- Introduction
- BargainFinder and Jango Shopping Agents
- Kasbah Agents
- Recursive Modeling Based Learning Agents
- TAC Agents

Introduction: Agents

- An **agent** is a computer program that can **perceive**, **reason**, **act**, **communicate** and **coordinate**.
  - An agent acts on behalf of a user, carrying out actions that meet the user's goals and preferences.
  - Example: A selling agent knows if any buying agent has requested some good and whether it has that good for sale, then reasons at which price it should sell the good, and communicates with the buying agent to make an offer.
- An agent is said to be **intelligent** if it is able to respond in a timely fashion to environmental changes (**reactivity**), to interact with other agents and possibly humans (**sociality**), and to take the initiative (**pro-activeness**).

Introduction: Agents and Objects

- Agents and objects have some similarities because objects encapsulate some internal states, are able to perform actions (methods), and communicate by message passing.
- Agents and objects also have significant differences:
  - Agents are **autonomous** or **semi-autonomous**.
  - Agents are capable of reactive, pro-active, and social behaviours.
  - Each agent is considered to have its own thread of control.
- See [8] for more detailed info regarding the design and implementation of agents.

Introduction: Multi-Agent Systems

- A multi-agent system is defined as a system (or environment) composed of multiple interacting agents.
- A well-designed multi-agent system is one in which agents can operate effectively and interact with each other productively.
- The system should provide some computational infrastructure to allow agents' interaction to take place:
  - Communication protocols enable agents to exchange and to understand messages.
  - Interaction protocols enable agents to have conversations (e.g., structured exchanges of messages).

BargainFinder and Jango

- BargainFinder [1] was the first shopping agent for price comparison.
- Given a specific product, BargainFinder will query multiple online merchants, and display the price as well as other product information to the customer.
- The customer will make the decision about which merchant to purchase the product from.
- BargainFinder encounters a problem called the **merchant blocking problem**:
  - Some merchants are not interested in competing on price alone because their products are offered with value added services.
  - As the result, these merchants block all price requests originated from BargainFinder.

BargainFinder and Jango (cont.)

- Jango [3] can be viewed as an advanced BargainFinder.
- Jango cleverly gets around the merchant blocking problem by having product requests originate from the customer's Web browser, instead of a central site as in the case of BargainFinder.
- Consequently, product requests from Jango appear to online merchants as requests from *real* customers.

BargainFinder and Jango (cont.)

- BargainFinder and Jango shopping agents provide customers with useful information for merchant comparison. However, they have the following shortcomings:
- They are not fully autonomous: They leave the task of analyzing the resultant information and selecting appropriate merchants completely for customers.
- The algorithms underlying these agents' operations do not capture information on product quality or other value added services, which is important for customers to select merchants.
- These agents are not equipped with any learning capabilities to help customers to improve their decision making in the future.

Kasbah

- Kasbah [2] is a multi-agent system for consumer-to-consumer electronic commerce.
- A user can create a selling (or buying) agent, give it some strategic direction and launch it into the system.
- The user can direct the agent's behavior by setting 3 parameters: (1) the **desired date** to sell (or purchase) the product by, (2) the **desired price**, and (3) the **lowest** (or **highest**) **possible price**, and also by specifying a negotiation function.
- The agent will use this negotiation function to lower (or raise) the asking price.

Kasbah (cont.)

- Kasbah supports 3 types of negotiation functions, namely **linear**, **quadratic**, and **exponential** corresponding to anxious, cool-headed, and frugal negotiation strategies, respectively.
- Once released into the system, the agent will seek out potential buyers (or sellers) in order to make a transaction.
- Kasbah's agents are autonomous in making decisions, therefore freeing the user from having to find and negotiate with buyers and sellers.
- However, Kasbah's agents do not make use of any AI learning techniques to improve their performance over time.

Recursive Modeling Based Learning Agents

- Vidal and Durfee [6, 7] design buying and selling learning agents using a recursive modeling approach.
- They recursively define different levels of agents according to their capabilities of modeling other agents:
- The 0-level agents are the agents that learn from their observations about the environment, and from any environmental rewards they receive.
- The 1-level agents are those agents that model others as 0-level agents, the 2-level agents are those that model others as 1-level agents, and so on.

Recursive Modeling Based Learning Agents (cont.)

- In theory, agents with deeper models of others should do better.
- However, in practice, because of the computational costs associated with maintaining deeper (i.e., more complex) models, there should be a level at which the gains and the costs of having deeper models balance out for each agent.
- Therefore, Vidal and Durfee's work only concentrates on the first three levels of agents.

Recursive Modeling Based Learning Agents (cont.)

- The agents proposed in [6, 7] are autonomous agents with learning capability.
- Drawbacks:
  - Agents at high levels suffer from computational costs.
  - Selling agents may not be able to maximize their profits by only adjusting the price.
  - The challenge of dealing with dishonest agents.

TAC Agents

- TAC (Trading Agent Competition) is a well-known competition that is very well participated by many universities around the world.
- TAC Web site: **http://www.sics.se/tac/**
- In this lecture, we shall introduce this competition and a TAC agent named ATTac [4, 5], which is the first-place finisher in the first TAC.

Competition Overview

- A TAC agent is a simulated travel agent whose task is to organize itineraries for a group of clients who wish to travel from TACTown to Boston and back again during a 5-day period.
- Travel goods, such as airline tickets and hotel reservations, are complementary, and tickets to entertainment events, such as the Boston Red Sox and the Boston Symphony, are substitutable.
- A TAC agent's objective is to win items that best satisfy its clients' preferences as inexpensively as possible.
- An agent's score is the difference between the utilities it earns for its clients and the agent's expenditures.

Market Supply

- The market supply consists of 3 types of goods: *(i)* flights to and from Boston, *(ii)* hotel rooms at 2 competing hotels, namely the Grand Hotel and Le Fleabag Inn, and *(iii)* entertainment tickets for the Boston Red Sox, the Boston Symphony, and Phantom of the Opera.
- There is a separate auction corresponding to every combination of travel good and day, yielding 28 auctions in total:
  - 8 flight auctions (all clients must stay at least one night in Boston, so there are no inbound flights on the 5-th day and there are no outbound flights on the 1-st day),
  - 8 hotel auctions (2 hotel types and 4 nights), and
  - 12 entertainment ticket auctions (3 entertainment event types and 4 nights).
- All 28 auctions are simultaneous (parallel) auctions that run for 15 minutes.

Auction Rules: Flight Auctions

- An infinite supply of flights is sold by the *TAC seller* at continuously clearing auctions in which prices are initialized between $250 and $400, and perturbed every 30 - 40 seconds by a random value uniformly selected in the range [-$10, $10], but confined within the bounds of $150 and $600.
- Flight Auctions close at the end of the game.
- Current ask price is issued immediately in response to new bids.
- Bid withdrawals are allowed, however no resale of flights is permitted.
- Flight auctions are clear continuously. Any buy bids that are at least as high as the current ask price will match immediately. Any buy bids that do not match remain in the auction as standing bids.

Example of Flight Auctions

- TAC seller places a sell bid of ((-64 300)).
- A buy bid of ((5 370)) would match 5 units at $300 each.
- A buy bid of ((3 290)) would not match, and would remain standing in the auction. A subsequent sell bid at $290 (or lower) would match 3 units at that ask price from this buy bid.
- A buy bid of ((2 370) (3 280)) would match 2 units at $300 each. The remaining portion ((3 280)) would remain standing in the auction.

Auction Rules: Hotel Auctions

- The TAC seller also makes available 16 hotel rooms per hotel per night, which are sold at open-cry, ascending, multi-unit, sixteenth-price auctions.
  - The winning bidders are those who bid among the top sixteen and these winning bidders pay at the sixteenth highest price.
  - Transactions clear when the auctions close, which typically occurs at the end of a game instance.
  - Hotel auctions are subject to early closing after random periods of inactivity.
  - No bid withdrawal or resale in hotel auctions is permitted.

Example of Hotel Auctions

- Hotel bid: ((-16 0))
- Agent A: ((8 20))
- Agent B: ((2 40) (6 60))
- Agent C: ((4 80))
- Agent D: ((7 100))
- Agent D wins 7 rooms, Agent C wins 4 rooms, Agent B wins 5 rooms, and Agent A does not win any rooms. The price of the rooms is $60 (ie., the sixteenth price).

Auction Rules: Entertainment Ticket Auctions

- Entertainment tickets are traded among TAC agents in continuous double auctions, where agents can act as either buyers or sellers, and transactions clear continuously.
- Price quotes (the lowest ask or the highest bid) are issued immediately in response to new bids.
- Each agent receives an initial endowment of tickets for each event on each night.
- Ticket resale is permitted.

Example of Entertainment Ticket Auctions

- Consider the following standing bids in an entertainment ticket auction:
  - ((-1 100))
  - ((-4 90) (-2 50))
  - ((-6 60))
  - ((1 40) (3 10))
  - ((1 20))
- The (lowest) ask price is $50 and the (highest) bid price is $40.
- If a new bid of ((6 70)) enters the auction, it would match 2 units at $50 each and 4 units at $60 each. Since the bid ((-6 60)) does not match completely, the remaining portion will remain standing in the auction as ((-2 60)). The (lowest) ask price will become $60.

Demand

- A TAC game instance pits 8 TAC agents against one another, with each TAC agent representing 8 clients.
- The market demand is determined by the 64 clients' preferences.
- Each client is characterized by a random set of preferences for
  - ideal arrival and departure dates (IAD and IDD, which range over days 1 through 4 and 2 through 5, respectively),
  - a grand hotel room reservation value (HV, which takes integer values between 50 and 150), and
  - reservation values for each of the 3 types of entertainment events (RV, SV, and TV, taking integer value between 0 and 200, for the Red Sox, Symphony, and Theater events, respectively).

Client's Package

- The job of each TAC agent is to assemble packages of goods for its clients.
- A package of goods is characterized by
  (i) arrival and departure dates (AD and DD, ranging over days 1 through 5),
  (ii) a hotel type (H, which takes on value G for Grand Hotel or F for Le Fleabag Inn), and
  (iii) entertainment tickets ($I(j,k)$ indicating whether or not the package includes a ticket on night j to event $k \in \{r,s,t\}$).
- Note: a notation such as R1 is also used to indicate that the package includes a ticket for Boston Red Sox on night 1.

Feasible Packages

- To obtain a positive utility for a client, an agent must construct a *feasible package* for that client; otherwise, the client's utility is zero.
- A feasible package is one in which

  (i) the arrival date is strictly less than the departure date,

  (ii) the same hotel is reserved during all intermediate nights,

  (iii) at most one entertainment event per night is included, and

  (iv) at most one of each type of entertainment ticket is included.

Utility

- Given a feasible package, a client's utility for that package is calculated as follows:

$$utility = 1000 - travel\_penalty + hotel\_bonus + fun\_bonus \tag{1}$$

where

$$travel\_penalty = 100(|IAD - AD| + |IDD - DD|) \tag{2}$$

$$hotel\_bonus = \begin{cases} HV & \text{if } H = G \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$fun\_bonus = \sum_j [I(j,r)RV + I(j,s)SV + I(j,t)TV] \tag{4}$$

Example of Client Preferences

Client	IAD	IDD	HV	RV	SV	TV
1	2	5	73	175	34	24
2	1	3	125	113	124	57
3	4	5	73	157	12	177
4	1	2	102	50	67	49
5	1	3	75	12	135	111
6	2	4	86	197	8	59
7	1	5	90	56	197	162
8	1	3	50	79	92	136

Table 1: Preferences assigned to ATTac's clients during game 3070.

Example of Goods Acquired

Good	Day1	Day2	Day3	Day4	Day5
R	1	1	1	2	-
S	1	1	0	0	-
T	1	0	1	1	-
G	4	1	0	0	-
F	1	2	3	3	-
I	5	2	1	0	-
O	-	4	1	0	3

Table 2: Goods acquired by ATTac in game 3070. R, S, and T denote tickets to the Red Sox, symphony, and theater, respectively; G and F denote the Grand Hotel and Le Fleabag Inn, respectively; I and O denote inbound and outbound flights, respectively.

Example of Allocation

Client	AD	DD	H	Tickets	Utility
1	2	5	F	R4	1175
2	1	2	G	R1	1138
3	3	5	F	T3, R4	1234
4	1	2	G	-	1102
5	1	2	G	S1	1110
6	2	3	G	R2	1183
7	1	5	F	S2, R3, T4	1415
8	1	2	G	T1	1086

Table 3: ATTac's final allocation in game 3070: No goods were left unallocated, and the total utility is 9443. During the game, the agent spent $5364 acquiring the goods and earned $75 in selling entertainment tickets, resulting in a final score of 9443 - 5364 + 75 = 4154.

ATTac's Strategy

- High-Level Strategy Overview
- Bidding for Flights
- Bidding for Hotel Rooms
- Bidding for Entertainment Tickets
- Allocation Strategy

High-Level Strategy Overview

- At every bidding opportunity, ATTac computes the most profitable allocation of goods to its clients (denoted by G^*), given the currently-owned goods and the current prices of hotels and flights.
- Two observations:
  (i) Since airline prices periodically increase or decrease by a random amount chosen from $\{-10, -9, ..., 9, 10\}$ with equal probability, the expected change in price for each airline auction is 0. Thus, ATTac has no incentive to bid before the *witching hour* (ie., the final few moments of the game), to wait for more information about hotel and entertainment ticket acquisitions.

  (ii) Since hotel prices are monotonically increasing as the game proceeds, the hotel prices approach the eventual closing prices.
- ATTac therefore aims to delay most of its purchases, particularly airline purchases, until late in the game.

High-Level Strategy Overview (cont.)

- ATTac accomplishes this delay by bidding in 2 different modes:
- *Passive Mode:*
  - lasting until the witching hour
  - designed to keep as many options open as possible
  - computing the average time it takes to compute and place bids, T_b.
- *Active Mode:*
  - switching to active mode when $T_l \leq 2T_b$ where T_l is the time left in the game
  - Buying the airline tickets required by the current G^*
  - Place high bids for the required hotel rooms

Bidding for Flights

- While in passive mode, ATTac does not bid in the airline auctions.
- In active mode, ATTac buys all currently unowned airline tickets needed for the current G^*.

Bidding for Hotel Rooms

- Hotel auctions are ascending, sixteenth-price auctions, subject to early closing after random period of inactivity.
- While in passive mode, ATTac bids either to win hotel rooms cheaply (should the auction close early) or to try to prevent the auctions from early closing.

Bidding for Hotel Rooms (cont.)

- For each hotel room of type i (such as “Grand Hotel, night 3”), let R_i be the number of rooms of type i needed for G^*. Let P_i be the current price of i. ATTac will try to acquire n rooms where

$$n = \begin{cases} 8 & \text{if } P_i = 0 \text{ (only true at the outset of the game)} \\ max\{R_i, 4\} & \text{if } P_i \leq 10 \\ max\{R_i, 2\} & \text{if } P_i \leq 20 \\ max\{R_i, 1\} & \text{if } P_i \leq 50 \end{cases} \tag{5}$$

- The above formula was selected so as to risk wasting up to \$40 - \$50 per room type for the benefit of maintaining flexibility later in the game.

Bidding for Hotel Rooms (cont.)

- If ATTac's current bids would win n rooms then ATTac does nothing; otherwise, it bids for n rooms at $1 above the current ask price.
- In active mode, ATTac bids on hotel rooms based on their *marginal values*:
  - Let $V(G^*)$ be the value of G^* (the utilities of all clients minus the cost of the yet-to-be-acquired goods).
  - Let G_c^* be the optimal allocation if client c fails to get his hotel rooms.
  - ATTac bids for the hotel rooms assigned to client c in G^* at the price of $V(G^*) - V(G_c^*)$.

Bidding for Entertainment Tickets

- On every bid iteration, ATTac places a buy bid for each type of entertainment ticket, and a sell bid for each type of entertainment ticket that it currently owns.
- To avoid overbidding (underbidding), ATTac gradually decreases (increases) its prices, depending on the time left in the game (T_l).
- For each owned entertainment ticket E, if E is assigned in the G^*, let $V(E)$ be the value of E to the client to whom it is assigned in G^*. ATTac offers to sell E for $min\{200, V(E) + \delta\}$ where δ decreases linearly from 100 to 20 based on T_l.
- ATTac bids to buy each type of entertainment ticket E (including those that it is also offering to sell) based on the increased value of G^* that would be obtained by owning E. Again, a sliding price strategy is used, this time with the purchase price increasing as the game proceeds.

Allocation Strategy

- ATTac relies heavily on computing the most profitable allocation of goods to its clients, G^*.
- G^* is computed using an integer linear programming approach.
- This approach works by defining a set of variables, constraints on these variables, and an objective function:
  - An assignment of values to the variables represents an allocation of goods to the clients.
  - The constraints ensure that the allocation is legal (e.g., no client gets more than one travel package, each client can only use one entertainment ticket per day, each client can only use each type of entertainment ticket once etc.)
  - The objective function is utility minus cost, which must be maximized, subject to the constraints.
- Details can be found in [5].

References

[1] BargainFinder. http://bf.cstar.ac.com/bf

[2] A. Chavez and P. Maes. Kasbah: An Agent Marketplace for Buying and Selling Goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996.

[3] R. B. Doorenbos, O. Etzioni, and D. Weld. A Scalable Comparison-Shopping Agent for the World Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.

[4] P. Stone and A. Greenwald, Editors. The First International Trading Agent Competition: Autonomous Bidding Agents. In *Electronic Commerce Research*, 5(1), 2005.

[5] P. Stone, M. Littman, S. Singh, and M. Kearns. ATTac-2000: An Adaptive Autonomous Bidding Agent. In *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.

[6] J. M. Vidal and E. H. Durfee. The Impact of Nested Agent Models in an Information Economy. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 377-384, 1996.

[7] J. M. Vidal. Computational Agents That Learn About Agents: Algorithms for Their

Design and A Predictive Theory of Their Behavior. PhD Thesis, Department of Computer Science & Engineering, University of Michigan, 1998.

[8] G. Weiss, editor. Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, 2000.