ELSEVIER

Computer Networks 37 (2001) 195–204

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

A genetic agent-based negotiation system

Samuel P.M. Choi *, Jiming Liu, Sheung-Ping Chan

Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong

Abstract

Automated negotiation has become increasingly important since the advent of electronic commerce. Nowadays, goods are no longer necessarily traded at a fixed price, and instead buyers and sellers negotiate among themselves to reach a deal that maximizes the payoffs of both parties. In this paper, a genetic agent-based model for bilateral, multi-issue negotiation is studied. The negotiation agent employs genetic algorithms and attempts to learn its opponent's preferences according to the history of the counter-offers based upon stochastic approximation. We also consider two types of agents: level-0 agents are only concerned with their own interest while level-1 agents consider also their opponents' utility. Our goal is to develop an automated negotiator that guides the negotiation process so as to maximize both parties' payoff.

Keywords: Negotiation; Intelligent agents; Genetic algorithms

1. Introduction

The advent of electronic commerce has revolutionized the way that contemporary business operates. Nowadays, online purchasing in becoming widely adopted and its selling volume is continuously increasing. [1] Not only does electronic commerce reduces the operation cost of a business, but also makes a company's service available to customers virtually 24 h a day. In addition, goods are no longer necessarily traded at a fixed price but based on the market demand (e.g., ebay [2] and priceline [3]). In the near future, buyers and sellers can negotiate among themselves so as to reach a deal that maximizes the payoffs to both parties. It is even more interesting to develop autonomous agents that can strive for the best deal on behalf of the traders. In this paper, we study how such an autonomous negotiation system can be built based upon agent technologies and genetic algorithms.

1.1. Negotiation

Negotiation [11,13] is a process of reaching an agreement on the terms (such as price and quantity) of a transaction for two or more parties. The negotiation process typically goes through a number of iterations, and in each of which one of the parties proposes an offer and sees whether the others accept. If not, other parties can propose their counter-offers and the process repeats until a consensus is reached. For an effective negotiation, it is important that all parties are willing to concede such that the differences among themselves reduce at each round and eventually converge to an

* Corresponding author. Tel.: +852-2339-7840; fax: +852-2339-7892.

E-mail addresses: samchoi@comp.hkbu.edu.hk (S.P.M. Choi), jiming@comp.hkbu.edu.hk (J. Liu), rickychan@icon.com.hk (S.-P. Chan).

[1] Source: Forrester Research, Inc.

[2] http://www.ebay.com

[3] http://www.priceline.com

1389-1286/01/$ - see front matter
PII: S1389-1286(01)00215-8

agreement. On the other hand, negotiation can also be viewed as a search process in which conflicts among different parties are resolved by finding a feasible alternative. However, negotiation is not just a matter of finding an acceptable deal, but an attempt to maximize all parties' payoffs. In order to achieve this goal, one needs to know the others' utility function. However, the utility functions are usually private and sensitive information. In this paper, we consider how this utility function can be estimated from the history of the opponent's offers.

Negotiation is common in conventional commerce, especially when large and complex business transactions are involved. Negotiation on various terms of a deal offers the potential to yield the involved parties the best payoffs. It also allows the terms of the deal (e.g., price) to be set according to the market demand and supply. However, human-based negotiation could be costly and non-optimal. Automated negotiation is therefore particularly useful due to its relatively low cost. Nevertheless, most existing e-commerce sites still employ fixed-pricing models, or allow only one-side negotiation (e.g., auction sites).

While negotiation is often beneficial to the participants, there are impediments to apply it in conventional business. The first concern is the time involved. Negotiation is often a time-consuming process because all parties desire to maximize their own payoff while they may have opposing goals. If some of the parties do not concede, it could take forever to reach an agreement. The second is that negotiation requires skillful tactics, and could be difficult for average dealers to bargain effectively on their own. The third difficulty is that all parties must first get together so as to negotiate. This imposes some restrictions on the customers since e-commerce can be worldwide and often involves people from various time zones.

There exist several types of negotiation models [8]. In this paper, a bilateral, multi-issue negotiation model is studied. Multi-issue negotiation is concerned with reaching an agreement on a deal with multiple terms. The involved parties typically do not want to reveal their underlying utility function and attempt to strive for the best deal through the negotiation process. An important topic for multi-issue negotiation is the question on how to avoid a sub-optimal deal; namely, how to avoid an agreement in which one party can modify to obtain better payoff without sacrificing the others. This is also known as pareto-inferior agreement or the problem of "leaving money on the table" [12].

1.2. Previous work

There have been several approaches to automated negotiation [19]. For multi-issue negotiation, the search space is typically complex and large, and with little information on its structure apriori. One recent research direction is to address such difficult problems with genetic algorithms (GAs) (e.g., [10,15]). GA [4] is particularly suitable for such tasks due to its efficient searching in a complex and large search space. In Oliver's approach, negotiation strategies are based on simple sequential rules with utility threshold for delimiters. One shortcoming for such representation is the lack of expressiveness. In order to enhance the strategy models, Tu et al. [15] deploy finite state machines for representing simple decision rules. In this paper, we are particularly concerned with learning issues; namely, how to speed up the negotiation by considering the offer history and adaptive mutation rate.

1.3. Agent technologies

Personal software agents [6,18] are continuously running programs that intimately work with their owners. Ideally, software agents should be proactive, intelligent, capable of understanding their owners' requirements, and therefore can perform tasks on behalf of their owners. While the current status is still far from its ideal goal, agent technologies have been successfully applied in various domains such as information filtering and job matching [2]. It is natural to extend the agent technology to automated negotiation tasks. Imagine a picture of our daily life in the near future. Everyone owns at least one agent as a personal assistant. These agents are responsible for various types of personal tasks, ranging from reminding appointments, to handling payments, to scheduling meetings. One task particularly about interesting to electronic business is how agents can

be used to deal with online purchasing. Unlike the way we purchase today, agents may negotiate with the sellers on the price among other terms, and even aligns with other buyer agents to bargain for a better deal. Agent negotiation involves three major steps: initiating a negotiation task by proposing an offer, evaluating the opponent's proposal if the offer is not accepted, and suggesting a new counter-proposal. This process is repeated several rounds until an agreement is reached.

An ideal negotiation system should minimize the involvement of human beings. As noted earlier, software agents are capable of undertaking the task. One reason is that personal agents have the access to their owners' schedule, and thus can infer a deadline for a negotiation. For instance, agents can keep track of the food consumption rate in a refrigerator, and set the negotiation deadline to the date that the food will be exhausted. In addition, agents know their owners' preferences so that the weights can be automatically determined rather than relying on user input. For instance, if the agent finds that most of the electric appliances of its owner are of brand A, it is plausible to assume that brand A is its owner's favorite. When purchasing other appliances, the agent would give higher priority to that brand. While it is impossible for an agent to know everything about its owner, its ability to anticipate can substantially reduce the required user input. A user's role thus becomes to confirm, rather than to initiate or to specify, a task.

1.4. Genetic agent-based negotiation system

We propose a framework of automated negotiation systems on the basis of genetic algorithm and agent technology. Our proposed system can be used in conjunction with the electronic marketplace such as [1]. Fig. 1 illustrates the details of the idea.

In our proposed framework, agents proactively predict their owner's needs as well as their requirements. This feature is particularly important since need identification is often regarded as the most important stage in the consumer buying behavior (CBB) model [5]. Nevertheless, currently only very primitive event-alerting tools are available at some commercial web sites (e.g., Amazon.com will send an e-mail to its customer if there is any new book that may be of interest to them), and little research effort has been paid to this issue. The advent of agent technology now brings hope in addressing this problem. Alternatively, users may manually initiate a negotiation process. In either case, the agent fills out the default specification according to its owner's profile. It should note that users may modify these specifications and the agent also learns from the feedback.

2. GA (Genetic agent) model

Our basic GA negotiation model is based on the one proposed by [7]. Initially, a negotiating agent

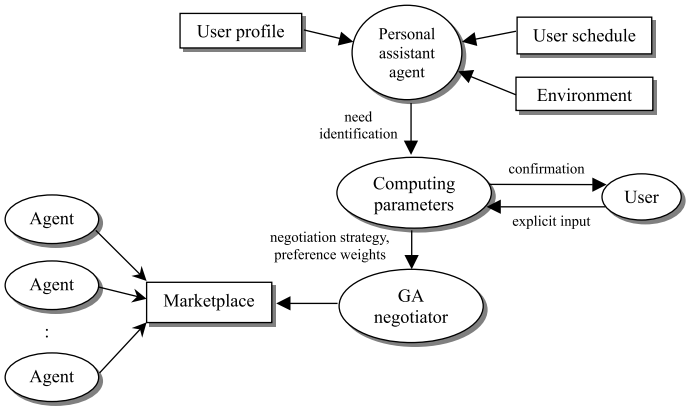

Fig. 1. Genetic agent-based automated negotiation system.

identifies itself as either a buyer or a seller and receives an offer from its opponent. If the offer is not satisfactory, the agent generates a counter-offer by the following procedure. The agent first sets up a population of chromosomes, each of which represents a candidate offer for the current round of negotiation. (In our experiments, this population contains 90 randomly generated chromosomes and 10 heuristic ones.) The heuristic chromosomes include the counter-offer proposed by the opponent as well as nine best offers suggested by the agent in the previous round. The inclusions of the counter-offer guides the negotiation process gradually converging to an offer that is acceptable by both parties; the nine selected offers retain the computation effort from the previous round and thus speed up the whole negotiation process. Nonetheless, this generating process is not sufficient to guarantee efficient searching. To improve the model, one needs to consider two additional constraints for the new population – retraction and non-negative constraints. The first constraint insures that the generated chromosomes cannot be better than their most recent offer while the second ensures that all chromosomes give positive payoff. These constraints are mandatory or otherwise the negotiation process may get into an infinite loop. Now according to the prior concessionary behavior of the opponent, the appropriate value of λ can be computed. The value of λ is a degree of penalization towards the opponent and forms the evaluation function for the genetic algorithm (see [7] for the details). Next, each chromosome is evaluated based on the objective function described earlier, and those with larger fitness values are selected to produce offspring through the genetic operators (i.e., crossover and mutation). Finally, the new population replaces the original one and the process continues until the negotiation agents reach a consensus.

2.1. Objective function

The above GA model is used for proposing counter-offers to the opponent. In particular, a user first initiates and assigns a strategy to a negotiation agent. The agent then starts the negotiation process and bargains with its opponent based on the history of proposed offers. The opponent's utility function is assumed of the following product form:

$$u = \prod_i f_i^{w_i},$$

where f_i and w_i corresponds to the utility value and weight (importance) of issue i. Note that this objective function differs from the linear combination formulation of the multi-attribute utility theory (MAUT) (see, e.g., [17]) that most existing GA negotiation systems employ. We find that our objective function is more general and exhibits some desirable characteristics. For instance, when the weight w_i is 0 (i.e., don't care), $f_i^{w_i}$ is equal to 1. The utility value of f_i thus has no effect on the overall utility u. On the other hand, if a particular issue is very important, a low value on that issue will force the overall utility to be small. Hence, this objective function consists of the joint requirement characteristics and is sensible in many real-world cases.

We also study two levels of negotiation agents. For level-0 agents, opponent's feedbacks are ignored. In other words, the values of w_i are fixed. For level-1 agents, opponent's feedbacks are used to adjust the weights in order to reveal its underlying preferences. This can be achieved by comparing the changes of consecutive offers, namely:

$$R_i = \left| \frac{f_i(t) - f_i(t-1)}{f_i'(t) - f_i'(t-1)} \right|,$$

where $f_i(t)$ and $f_i'(t)$ denotes respectively the user's and the opponent's proposed values for issue i at round t. Note that this ratio reveals the opponent's relative preference compared with the user upon issue i. A large ratio suggests that issue i is more important to the opponent than the user in a quantitative measure.

It is now plausible to divide the weight into three regions:

Categories	Weight range
Very important	0.7–0.9
Quite mind	0.4–0.6
Don't care	0.1–0.3

The main advantage of using GA for automated negotiation is its capability of efficient searching for a feasible solution in a complex and high-dimensional space.

2.2. Data structure

The implementation details are described as follows. We define an array structure to represent the population, where each structure contains six fields: chromosome, payoff, fitness, crossover point and two fields for pointing to the parents. A chromosome represents the agent's potential offer and is stored into an integer array. As the names suggested, the payoff field records the payoff of the agent while the fitness field keeps the strength of the chromosome. Both fields are dedicated for providing direct access to these frequently referred values so as to reduce computation time. The crossover point field indicates the chromosome break point for the crossover operator. The parent fields are also maintained for the purpose of efficient chromosome manipulation and for error checking.

3. Experiments

3.1. Three concession-matching tactics

We study three types of commonly used concession-matching tactics suggested by [7]; namely, reciprocal tactic, exploiting tactic and cooperative tactic. Reciprocal tactics is the negotiation strategy that imitates the opponent's concessionary behavior. In other words, if the opponent concedes more, the agent concedes more too. In the exploiting tactic, an agent concedes less when its opponent is cooperative. In the cooperative tactic, an agent concedes in order to reach a consensus quickly.

From the implementation perspective, different tactics could result in a different range of λ. We therefore define the concession-matching rate as the measure of the concessionary behavior of the opponent against the agent itself. Specifically, the agent x computes the concession-matching rate (c_{xyi}) against the agent y at ith round by the following formula:

$$c_{xyi} = \frac{Y_{i-1} - Y_i}{X_{i-1} - X_i}.$$

The value of c_{xyi} specifies the degree that the opponent concedes. When c_{xyi} is equal to zero, it indicates that the opponent has not made any concession. In other words, the agent y is purely competitive orientation. Likewise, when c_{xyi} is equal to or greater than 1, it indicates that the opponent has fully or more than reciprocated concessions. In this case, the agent y has a purely cooperation orientation.

The relationship between the concession-matching behavior of the opponent and λ is defined in the pseudo-code (Fig. 2).

As shown in Fig. 2, an agent's tactic determines the value of λ, which is obtained from c_{xyi}. When an agent uses a reciprocal tactic, the range of λ is between 0 and 1. In the exploiting tactic, the range of λ is from 0 to 0.5. For the cooperative tactic, the value of λ ranges from 0.5 to 1. It should also be noted that the concession-matching rate is inversely proportional to λ.

3.2. Levels of goal difficulty

Intuitively, the level of goal difficulty indicates how ambitious the agent is. If the agent is not satisfied with the current offer, the payoff of that offer will be set to zero. The fitness value of that offer will therefore be low compared with the offers that fulfill the agent's goal. In Fig. 3, the detailed calculation for the payoff of an agent is shown.

In Fig. 3, the variables $f1$, $f2$, $f3$ and $f4$, respectively, represent price, quantity, processing day and the features of the goods, and $p1$–$p4$ denotes the payoff of each field. The function then checks if each field fulfills the agent's minimum requirement. If all fields are satisfied, the agent's payoff will be computed based on the formula shown above.

3.3. Predicting opponent's preferences

As discussed earlier, predicting opponent's preferences is important for negotiation. Possessing

```
double CNeg_ModelDlg::Calculate_Lamda(double Cxy, int index)
{       double match, lambda;
        if (Cxy >= 1)
                match = 0;
        else if (Cxy >=0)
                match = 1.0 - Cxy;
        else printf("wrong Cxy");
        if (buyertactics == 0)                  //reciprocate
                lambda = match;
        else if (buyertactics == 1)  //exploit
                lambda = match/2.0;
        else if (buyertactics == 2)  //cooperate
                lambda = 0.5 + (match/2.0);
        return lambda;
}
```

Fig. 2. Pseudo-code for calculating lambda.

the knowledge of the opponent not only speeds up the whole negotiation process, but also maximizes what one could possibly get from the negotiation. The main idea for predicting opponent's preference is based on the value fluctuation of the opponent's offers. The formula is quite similar to that of the concession-matching tactics and is shown below

$$R_i = \left| \frac{f_i(t) - f_i(t-1)}{f_i'(t) - f_i'(t-1)} \right|.$$

The weight for a particular field can now be computed as shown in Fig. 4.

3.4. Learning opponent's preferences

We employ stochastic approximation as a tool for learning opponent's preferences. Stochastic approximation [16] is an online estimation method widely used in machine learning community [9]. It estimates the desired function by obtaining its empirical values over time and gradually adjusts its estimation until convergence

$$V_i = \alpha V_{i-1} + (1-\alpha)(R_i/2).$$

And the weights can now be adjusted by a direct mapping from the prediction value as in the original model, as shown in the following equation:

$$w_i = \beta w_{i-1} + (1-\beta)T_i.$$

For the purpose of normalization, the prediction ratio (R_i) is deliberately set between 0 and 2. Note that α and β are the learning rates for the stochastic approximations.

3.5. Generation

The population generator does not deviate from the standard one. Our experiments are based on the genetic parameter settings similar to the one described in [3]. Specifically, we ran the simulator for 40 generations, with a population size of 50 each, and the length of chromosome was set to 20 bits. The crossover rate is 0.7 and the mutation rate (described below) changes over time according to the responses of the opponent.

```
double CNeg_ModelDlg::Payoff(int chrom[], int index)
{       double f1, f2, f3, f4, p1,p2,p3,p4;
        f1 = decode(chrom, 5);
        f2 = decode(chrom+5, 5);
        f3 = decode(chrom+10, 5);
        f4 = decode(chrom+15, 5);
        if (index==0)       //seller
        {       p1 = f1 - m_spgoal;
                p2 = m_squatity - f2;
                p3 = f3 - m_spday;
                p4 = m_sfeatures - f4;
        }
        else if (index==1)//buyer
        {       p1 = m_bpgoal - f1;
                p2 = f2 - m_bquatity;
                p3 = m_bpday - f3;
                p4 = f4 - m_features;
        }
        if (p1<0 || p2<0 || p3<0 || p4<0)
                return 0;
        else
                return  (pow(f1,p_weight[index])*pow(f2,q_weight[index])*
                        pow(f3,d_weight[index])*pow(f4,f_weight[index]));
}
```

Fig. 3. Pseudo-code for calculating payoff.

```
case 0:  //for "very important"
         d_weight[index] = 0.7 + (ratio[2]*0.2);
         break;
case 1:  // for "quite mind"
         d_weight[index] = 0.4 + (ratio[2]*0.2);
         break;
case 2:  //for "don't care"
         d_weight[index] = 0.1 + (ratio[2]*0.2);
         break;
```

Fig. 4. Calculating weight for a particular field.

3.6. Mutation rate

Unlike the conventional GA models, our agents adapt to their opponents by dynamically adjusting their mutation rate. The rate changes according to the number of generations and the length of the chromosomes. In particular, the mutation rate μ is computed by

$$\mu = \frac{\log(p)}{g\sqrt{l}},$$

where p, g and l are respectively the population, goodness and length of the chromosome. This formula suggests that mutation rate should be directly proportional to the log population while inverse proportional to the goodness value. In other words, the higher the goodness value one obtains, the lower mutation rate one should use. This is sensible as mutation is likely to take the current solution away from the local maxima and hence might take a longer time to converge. This

problem is known as the exploitation and exploration dilemma, and has been studied by different communities [14]. It seems that our proposed formula works in a number of different settings and empirical results verify that this adaptive setting substantially improves the performance of the negotiation agents.

4. Empirical results

In our experiments, we consider four different issues for negotiation: price, quantity, processing days and features. In the first experiment, we first investigate the effects of combining different concession-matching strategies on the average negotiation time and the obtained payoff. We set all four issues as "quite mind" for the testing purpose. As suggested from Table 1, the exploit strategy in general performs the best in terms of the payoff but takes up the most negotiation time. This strategy is suitable for the negotiators who have plenty of time and are eager to strive for maximal payoff. On the contrary, the cooperative strategy can reach an agreement quickly, but obtains the worst payoff. This strategy is appropriate for the negotiation whose time is tight, and thus must sacrifice payoff in order to get the deal in time. Reciprocate strategy tries to make a balance between the two and is good for the conservative negotiators. Nonetheless, the performance of reciprocate strategy largely depends on the strategy of its opponent.

In our second experiment, we verify the effects of various tactics on the price, and the rest of the issues are set to the same weight. Table 2 shows that if one ranks an issue as important, the negotiation time will take significantly longer. This phenomenon becomes worse as more issues are ranked as important. It is therefore important for the negotiation agent to consider not only the strategy, but also the issues to strive for. It is generally desirable to set the issue weights as needed, in order to attain an agreement faster.

Up to now, we have been testing level-0 agents. In the next experiment, we would like to test the

Table 1
Average negotiation time with different concession-matching strategies

Buyer	Seller		
	Reciprocate	Exploit	Cooperative
Reciprocate	Rounds: 42.7 Payoff: 100.5	Rounds: 52.3 Payoff: 123.7	Rounds: 40.6 Payoff: 85.3
Exploit	Rounds: 54.9 Payoff: 126.8	Rounds: 68.5 Payoff: 108.5	Rounds: 40.3 Payoff: 153.3
Cooperative	Rounds: 38.5 Payoff: 92.4	Rounds: 47.2 Payoff: 143.5	Rounds: 34.2 Payoff: 105.7

Table 2
Result of experiment 2

Buyer's price	Seller's price		
	Very important	Quite mind	Don't care
Very important	Rounds: 90.5 Price: 20.6	Rounds: 47.8 Price: 7.9	Rounds: 44.6 Price: 3.5
Quite mind	Rounds: 63.3 Price: 21.1	Rounds: 36.3 Price: 19.7	Rounds: 46.3 Price: 12.1
Don't care	Rounds: 55.6 Price: 29.3	Rounds: 29.2 Price: 26.1	Rounds: 18.5 Price: 3.6

Table 3
Result of experiment 3

Buyer's price	Seller's price		
	Very important	Quite mind	Don't care
Very important	Rounds: 83.2 Payoff: 105.5	Rounds: 51.6 Payoff: 131.6	Rounds: 51.7 Payoff: 120.3
Quite mind	Rounds: 57.3 Payoff: 128.8	Rounds: 43.1 Payoff: 110.3	Rounds: 54.7 Payoff: 157.0
Don't care	Rounds: 62.5 Payoff: 90.3	Rounds: 69.4 Payoff: 123.5	Rounds: 51.8 Payoff: 106.9

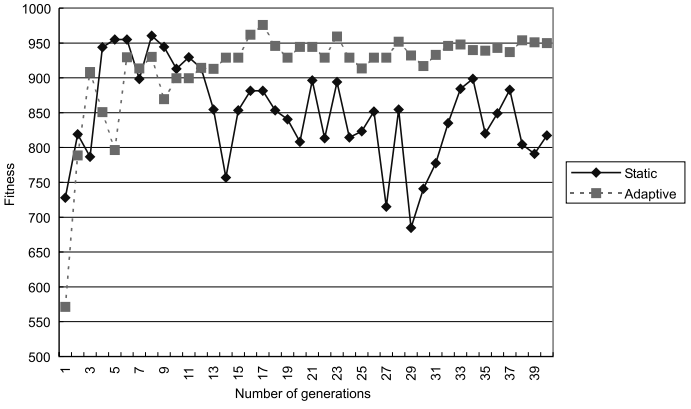

Fig. 5. Static mutation rate vs adaptive mutation rate.

learning effect of level-1 agents. We use the same experimental setting described above. We notice that level-1 agent yields better payoff and achieve the goal more quickly (see Table 3).

In our last experiment, we verify the effectiveness of the adaptive mutation rate. Fig. 5 shows the performance of two different mutation methods. The x-axis is the best fitness value obtained and the y-axis is the number of generations that the GA model goes through. After the first few generations, it is clear that the adaptive approach exhibits more stable and better fitness values. This method significantly speeds up the process for generating counter-offers.

5. Conclusion

Automated negotiation has become increasingly important since the advent of electronic commerce. In this paper, we propose a genetic-agent-based automated negotiation system for electronic business. Unlike other negotiation systems, our proposed system is able to proactively anticipate the user needs and initiate a purchase process. According to the user's profile and schedule, the agent suggests a set of default parameters and asks its owner for confirmation. This approach thus minimizes the required user input. Given the deadline and the preferences, the negotiation agent is able to select an appropriate negotiation strategy. We describe the implementation details of a GA negotiation agent. Unlike other GA negotiation agents, our agents attempt to learn the opponent's preference by observing the counter-offers and adapt to the environment by dynamically modifying its mutation rate. While some parts of our system are still under implementation, we believe that our initial effort has laid a fundamental framework for developing an online automated negotiation system.

References

[1] S.P.M. Choi, J. Liu, A dynamic mechanism for time-constrained trading, in: Proceedings of Fifth International Conference on Autonomous Agents (Agents 2001), Montreal, Canada, May 2001, pp. 568–575.

[2] B. Crabtree, N.R. Jennings, (Eds.), The Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, 1996.

[3] K. DeJong, Adaptive systems design: a genetic approach, IEEE Trans. Syst., Man Cybernetics SMC-10 (1980) 566–574.

[4] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[5] R.H. Guttman, A.G. Moukas, P. Maes, Agent-mediated electronic commerce: a survey, Knowledge Eng. Rev. J. 42 (3) (1998).

[6] M.N. Huhns, M.P. Singh, Readings in Agents, Morgan Kaufmann, Los Altos, CA, 1998.

[7] R. Krovi, A.C. Graesser, Agent behaviors in virtual negotiation environments, IEEE Trans. Syst., Man, Cybernetics (Part C: Applications and Reviews) 29 (1) (1999).

[8] R.J. Lewicki, J.A. Litterer, Negotiation. Readings, Exercises, and Cases, Irwin, Homewood, IL, 1985.

[9] T. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.

[10] J.R. Oliver, On artificial agents for negotiation in electronic commerce, Ph.D. thesis, The Wharton School, University of Pennsylvania, 1996.

[11] D.G. Pruitt, Negotiation Behavior, Academic Press, New York, 1981.

[12] H. Raiffa, The Art and Science of Negotiation, Harvard University Press, Cambridge, MA, 1982.

[13] J. Rosenschein, G. Zlotkin, Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, MIT Press, Cambridge, MA, 1994.

[14] S. Thrun, The role of exploration in learning control, in: D. White, D. Sofge (Eds.), Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches, Van Nostrand Reinhold, New York, 1992.

[15] M.T. Tu, E. Wolff, W. Lamersdorf, Genetic algorithms for automated negotiations: a FSM-based application approach, in: Proceedings of 11th International Conference on Database and Expert Systems (DEXA 2000), 2000.

[16] M.T. Wasan, Stochastic Approximation, Cambridge University Press, Cambridge, 1969.

[17] D.v. Winterfeldt, W. Edwards, Decision Analysis and Behavioral Research, Cambridge University Press, Cambridge, 1986.

[18] M.J. Wooldridge, N. Jennings, Agent theories, architectures and languages: a survey, Knowledge Eng. Rev. 10 (2) (1995) 115–152.

[19] D. Zeng, K. Sycara, Benefits of learning in negotiation, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence, 1997, pp. 36–41.

Samuel Choi received both his Bachelor and Master degrees in computer science from University of Manitoba (Canada), and his Ph.D. degree in computer science from Hong Kong University of Science and Technology. He taught at Hong Kong University of Science and Technology and is currently a post-doctoral teaching fellow in the department of Computer Science at Hong Kong Baptist University. His primary research interests are electronic commerce, intelligent agents and machine learning.

Jiming Liu is an Associate Professor of Computer Science at Hong Kong Baptist University (HKBU), a Senior Member of the IEEE, and a Member of the ACM. His areas of expertise are artificial intelligence, multi-agent systems, learning, adaptation and artificial life in software and systems, and autonomy-oriented computation (AOC). Dr. Liu earned the Master-of-Arts degree in Educational Technology from Concordia University, and the Master-of-Engineering and the Doctor-of-Philosophy degrees both in Electrical Engineering from McGill University in Montreal, Canada. He worked for some time as Software Engineer, Research Associate, and Senior Research Agent at R&D firms and government labs in Canada before joining the Computer Science Department of HKBU in 1993. In 1999, he was a Visiting Scholar in the Computer Science Department at Stanford University.

Ricky Chan obtained the Bachelor of Science (Hons.) in Computer Science (Computer System) from Hong Kong Baptist University (HKBU) in 1997. He is currently Internet Application Programmer in Interactive Communication Online Networks, Limited (ICON), mainly working on designing and implementing portal tools with servlets and JSP. His research interest is artificial intelligence, especially in machine learning.