Proceedings of the 2001 IEEE Congress on Evolutionary Computation Seoul, Korea • May 27-30, 2001

Towards Evolution of Software Agents in Electronic Commerce

Fangming Zhu **Sheng-Uei Guan**
Department of Electrical and Computer Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 119260
{engp9402,eleguans}@nus.edu.sg

Abstract -- With the development of Internet computing and software agent technologies, agent-based electronic commerce (e-commerce) is emerging. Software agents have demonstrated tremendous potential in conducting various tasks in e-commerce. However, when agents are initially created, they have little knowledge and experience with relatively lower capability. They should also strive to adapt themselves to the changing environment. It is advantageous if they have the ability to learn and evolve. This paper addresses evolution of software agents in e-commerce. Agent fitness and life cycle are proposed as evolution mechanisms, and modularized agent structure is introduced to facilitate the evolution process. Genetic Programming (GP) operators are employed to restructure agents in the proposed multi-agent evolution cycle.

1 Introduction

Doing business on the Internet is becoming more and more popular. The use of the Internet to facilitate commerce among companies and customers brings forth many benefits, such as automated transactions, greater access to buyers and sellers, and dramatically reduced costs.

In the recent decade, agent-based e-commerce has emerged and become a focus of the next generation of e-commerce. The motivation of introducing software agents into e-commerce is to overcome the arising barricades which include overload of information, difficulty in searching, lack of negotiation infrastructure, etc. In this new approach, software agents act on behalf of customers to carry out delegated tasks automatically. They have demonstrated tremendous potential in conducting various e-commerce activities, such as comparison-shopping, negotiation, payment, mediation, distribution, auction, sales promotion, etc. [1, 2, 3].

Some research projects have concentrated on employing software agents in specific e-commerce applications. Typical examples are AuctionBot [4], Kasbah [5], etc. Issues such as integrity and security also attract many efforts of research work [7, 8, 9].

As numerous agents are roaming throughout the Internet, they compete for the limited resource to achieve their own goals. For instance, when a limited number of products are available, agents will have to compete against each other to get them. In the end, some of them will succeed, while the others will fail. The successful agents may become more powerful, and the unsuccessful agents may lose some fitness. This is similar to the evolution in natural ecosystems. Furthermore, the Web environment also changes rapidly and continuously, and there may exist some malicious agents and hosts. Therefore, agents should strive to protect and adapt themselves in order to complete their tasks successfully.

However, when agents are initially created, they have little knowledge and experience with relatively lower capability. Although owners may give some basic knowledge or functionality to these agents, it is advantageous if they have the ability to learn and evolve. For example, when agents participate in an auction, they all aim to win the auction with desirable prices. However, young agents are not familiar with the auction regulations, and their bidding strategies are inferior to those of senior agents. Therefore, they may fail to win the auction. But they can learn something from this experience, and may become stronger after some period of time.

Many issues are essential in agent evolution. Firstly, evolution of an agent is closely related with agent structure. Thus, a suitable agent structure is one of basic concerns in agent evolution. Secondly, agents should have their own mechanisms to advance evolution. How to design strong and adaptive evolution mechanisms is another pursuit. Thirdly, in multi-agent system, evolution of individual agent is also related with many social concerns, such as coordination, relationship, topology, communication, etc. Finally, agent owners should closely interact with the evolution procedures of their agents. For instance, they should be informed of the information of the fitness of agents and can interfere with evolution by some tools.

0-7803-6657-3/01/$10.00 © 2001 IEEE

In this paper, we address multi-agent evolution for agents in e-commerce. Agent fitness and life cycle are proposed as evolution mechanisms to control the evolution process. Agent group and modularized structure are introduced to facilitate the evolution process. Genetic Programming (GP) operators, such as reproduction, crossover, and mutation, are main tools to restructure software agents.

2 Background and related work

Research in the fields of DAI (Distributed Artificial Intelligence) and MAS (Multi-Agent System) has invested considerable efforts in evolving cooperation strategies in domains where multiple, autonomous agents share goals and resources, and need to use mutually acceptable work-sharing strategies to accomplish common goal. Thomas Haynes et al. [10, 11] proposed a cooperation strategy approach for multi-agent problem solving situation. They aimed to evolve programs to control an autonomous agent capable of learning how to survive in a hostile environment. Namatame [15] provided a model for investigating collective behaviors that emerge from local interaction among self-interested agents. Dworman et al. [13] simulated artificial agents in two coalition games, and they claimed that simple artificial agents could formulate effective strategies for negotiating agreements that approximate those prescribed by the theory of cooperative games.

However, little research effort is dedicated to agent evolution in e-commerce activities. Most of the existing work mainly focuses on evolving strategies for agents, extending the MAS approaches. Gimenez-Funes [12] used possibility-based and case-based decision models to design the bidding strategies for agents in electronic auctions. Ritcher and Sheble [14] developed bidding strategies which were used for electric utilities in the scenario of double auctions.

In our previous work, we have proposed a SAFER (Secure Agent Fabrication, Evolution & Roaming) architecture, which aims to construct a standard, dynamic and evolutionary agent system for e-commerce [6]. SAFER provides a framework for agents in e-commerce and establishes a rich set of mechanisms to manage and secure them. As suggested in its name, we aim to integrate agent fabrication, evolution, and roaming in the architecture. We have elaborated agent fabrication and roaming in [19] and [18] respectively. This paper addresses the evolution part of the SAFER architecture.

In the following sections of the paper, we first propose agent fitness and life cycle as basic evolution mechanisms in section 3. After the paper elaborates agent group and modularized structure in section 4, it covers multi-agent evolution in section 5. Section 6 discusses the integration of agent evolution in SAFER architecture, and section 7 concludes the paper.

3 Agent fitness & life cycle

Agent fitness is an essential metric for agent evolution. It shows the performance of an agent and its ability to survive and adapt to the environment. It is also an indicator of the trend of evolution. Generally, the higher the fitness of an agent is, the stronger it is. If an agent's fitness is found too low, or decrease rapidly, we may suspect that the agent might have been attacked or something may have gone wrong in the agent body. We consider that agent fitness can consist of two parts:

- *Integrity fitness* f_i. It is used to measure the integrity of an agent, as the agent may be compromised during the process of evolution, roaming, or communicating. It may be caused by intentional damages from malicious agents or accidental errors during routine procedures.
- *Achievement fitness* f_a. The achievement fitness is evaluated by the history of an agent, which includes the number of tasks carried out and the quality of tasks completed. Every time an agent completes a task and reports to its owner, the owner will assess the outcome and give a corresponding mark. Through analyzing the trend of agent performance by analyzing its mark history, the achievement fitness can be evaluated.

Thus, the overall agent fitness F can be obtained as $F=\alpha \cdot f_i + \beta \cdot f_a$, and α, β are the fitness weights specified by owners.

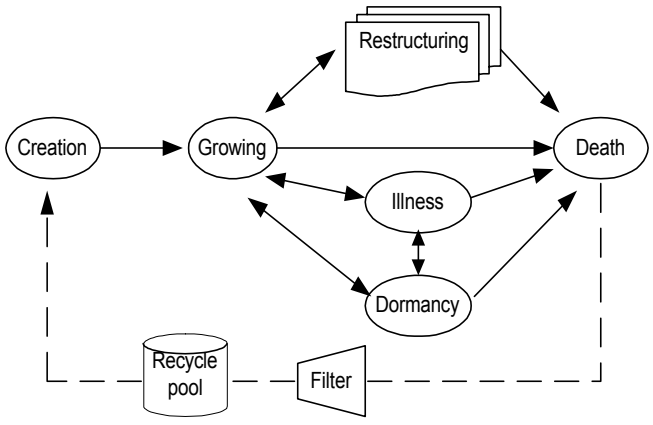

Fig. 1 Agent Life Cycle

In its lifespan from creation to termination, an agent experiences several states. We construct an agent life cycle to demonstrate the transition between these states, as shown in Figure 1. These states are explained as follows.

Creation: A new agent is created for some specific purpose. As in the SAFER architecture, agent factory is responsible for fabricating new agents according to

customizations from owners and fabrication formalities [19].

Growing: An agent gradually grows up, as it learns from experience, becomes more adapted to the environment, and fulfills assigned tasks successfully. Normally agent fitness will also increase along with growth.

Illness: When an agent is damaged or tampered with by malicious agents or hosts, it will fall back into the illness state. After recovery, it can be treated as a growing agent again.

Dormancy: An agent becomes inert in this state, as it may finish its task and be waiting for new instructions. Under this state, all the evolution and communication activities are also suspended, until it receives some reactivating commands.

Restructuring: An agent can undergo restructuring which is one of the important steps of evolution. We will discuss it in detail in section 5.

Death: As an agent reaches its expiry time prescribed by its owner, or it cannot be recovered from illness, it will be brought to death. Besides these, it is owner's privilege to bring agents residing in any states to death.

As an agent is being decomposed, we may want to preserve something useful for reuse. The Filter and Recycle Pool in the life cycle shown in Figure 1 are for this purpose. An agent owner can customize the filtering criteria. For instance, the basic criteria may include the rules like that a recycled agent should have never been attacked or tampered with, or/and the fitness of the recycled agent is above a certain level. The recycling process has several steps. First, the recycled agent is decomposed into several modules. Then, these modules are evaluated and the potentially useful ones are saved in the recycle pool for future reuse.

4 Agent group & modularized structure

4.1 Agent group for evolution

As shown in Figure 2, we organize agents into groups for evolution. Agents in an agent group can come from same or different owners, and each owner can organize many agent groups. The criteria for group formation are varied. For instance, agents in one group can be homogeneous or heterogeneous, which is determined by their purposes or functions. According to the variability of membership of group, there are two types of groups, i.e. closed group and open group. For a closed group like Group A in Figure 2, the group members are fixed. While in open groups such as Group B or C in Figure 2, new agents can join or leave the group as the membership can be changing all the time.

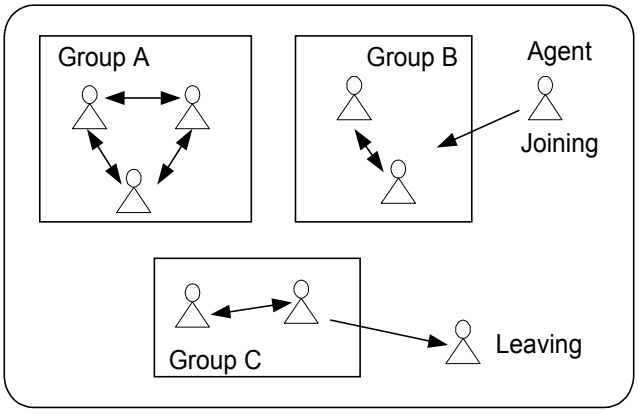

Fig. 2 Agent Groups for Evolution

Fitness of an agent group is also a useful value, which can act as a hint for better organization of the agent group. Representation of the fitness of an agent group can be largely different according to the owner's objectives. For instance, an owner may choose the average fitness of all the agents in the group as the fitness of group, while others may choose the minimum or maximum fitness. For the two different types of group discussed above, the fitness of a group may have different properties. For a closed group, the average fitness is supposed to rise steadily, while the average fitness of an open group is fluctuating. A newborn agent may bring down the average fitness, while a senior agent can boost average fitness. The dynamically joining and leaving agents also affect the average group fitness.

4.2 Modularized agent structure

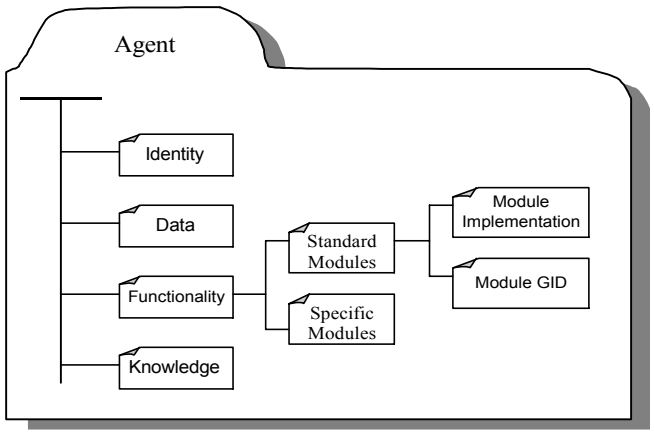

Fig. 3 Modularized Agent Structure

Modularization is a basic thread which runs through the whole process of agent fabrication in the SAFER architecture. Various modules for different types of agents are stored in agent factories. When a new agent is created, necessary modules are combined according to the

custimization from its owner and guidelines for fabrication [19].

A general agent strucutre is shown in Figure 3. An agent is composed of four kinds of modules, i.e., identity, data, knowledge, and functionality. The identity module contains basic elements of the identity of an agent, such as agent ID, certificate, timestamp, agent-digest, etc. The data modules are to store information collected from hosts, parameters used in functionality modules, as well as logs of the agent activities. The knowledge modules store the agent knowledge which is to support analysis and decision-making. The most important part of an agent is the functionality modules. They comprise specific and standard modules. The specific modules, such as negotiation modules, are one of the variable and important components of an agent. For the standard modules, SAFER provides two choices, direct module implementation or virtual module with Global ID (GID) [19].

The modularized agent structure can facilitate agent evolution, as we aim to enhance agent capability by acquring or exchanging modules with GP operators.

5 Multi-agent evolution

The evolution process in agent groups can be incessant and cycling. Figure 4 shows the multi-agent evolution cycle. Typically it operates in one agent group and is supervised by an agent owner. There are three main stages, namely, restructuring, selection, and growing. In the stages of selection and growing, the agent owner uses agent fitness and agent life cycle as evolution tools, which have been discussed in section 3. This evolution cycle is similar to the GP algorithm cycle, while the stopping criteria of multi-agent evolution are determined by the agent owner.

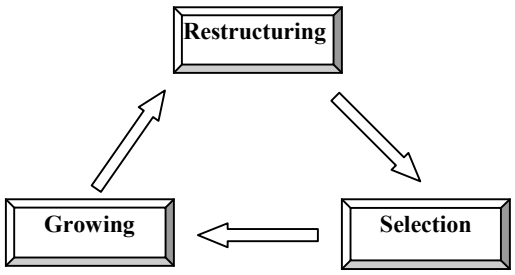

Fig. 4 Multi-agent Evolution Cycle

GP is a technique which enables computers to solve problems without being explicitly programmed. GP first creates the initial generation of software randomly, then employs main operators: crossover and mutation to generate new individuals. After evaluating the fitness of population, GP uses certain selection criteria to form the new population. This cycle will last until the optimal result is found or time-out occurs [16,17].

We use GP operators to restructure agents. The main operators are crossover, mutation, and reproduction. Crossover for agents has two types, i.e. inter-agent crossover and intra-agent crossover. With the inter-agent crossover, as shown in Figure 5, an agent exchanges a module with another agent. It should be ensured that exchanged sub-tree is compatible with the original structure. Otherwise, the resulting agent may be in a mess and cannot work. Figure 6 illustrates the intra-agent crossover, which exchanges some parameters between modules in the same agent. For example, a bidding agent may have many decision-algorithm modules which are used to decide the next bid depending on the last increment of price and the approaching of deadline. Intra-agent crossover can alter some parameters in the decision algorithms to form new algorithms. Mutation is not so popular, but it can still be useful in some circumstances. We can specify the allowed ranges for some parameters, and let them mutate in a reasonable way, as shown in Figure 7. Reproduction is a very straightforward operator which produces a copy of agent components or the whole body directly. (In Fig. 5, 6, 7, "M" denotes a module, and "D" denotes a piece of data.)

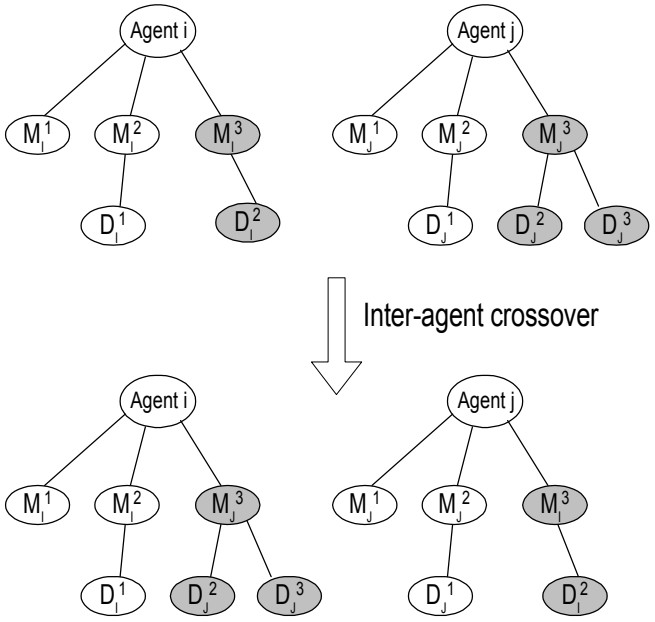

Fig. 5 Inter-agent Crossover between Two Agents

There are some tags in agents to facilitate the operator of crossover and mutation. Some of them indicate the allowed crossover point and the attributes of the sub-tree under this point, and others show the type of nodes and their corresponding ranges.

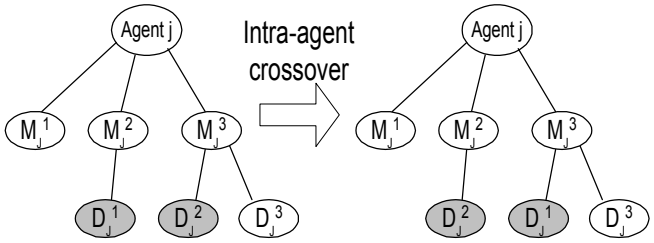

Fig. 6 Intra-agent Crossover in an Agent

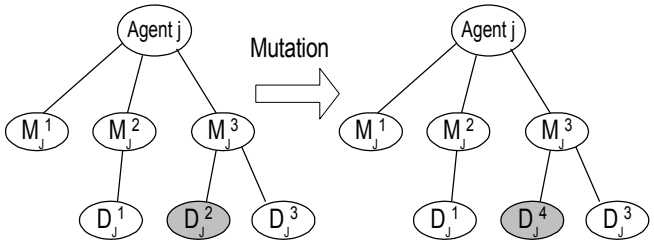

Fig. 7 Mutation in an Agent

Other types of restructuring in multi-agent evolution include replication, combination, and split, which are illustrated in Figure 8. These types of restructuring involve the combination of the basic GP operators. For example, when an agent is split into two agents, reproduction and mutation are employed in the process.

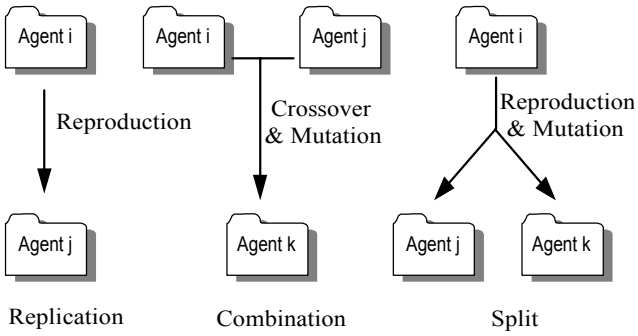

Fig. 8 Other Types of Agent Restructuring

6 Integration in the SAFER architecture

Agent evolution is an essential part of the SAFER architecture. We have implemented a typical agent community that includes one agent factory, one community administration center, and several agent owners. With these facilities, agents can be successfully fabricated according to the formalities prescribed and customizations from owners, although these agents are simple with little intelligence [19]. After modularized agents are fabricated by the agent factory, they will be dispatched to their owners. Some of them are organized as groups to evolve.

We have begun to implement agent evolution in the SAFER architecture. As the first step, we have implemented the "exchange module" function in the agent owner. With this function, agent owner can choose and exchange modules between two agents, which we regard as the typical operation of the crossover operator. Figure 9 shows that an agent owner is choosing the first agent and its module which aims to exchange with another agent. We also have implemented a mutation feature in the "check agent" interface, which enables the owner to change some parameters of modules within the allowed ranges.

Java is chosen as the implementation language, because it has a list of important features including robustness, security, and portability. Moreover, Java provides a three-layered security model and many mechanisms to enhance security protection. The interoperability feature of Java between various platforms is also a motivation for us to choose it as a prototyping language.

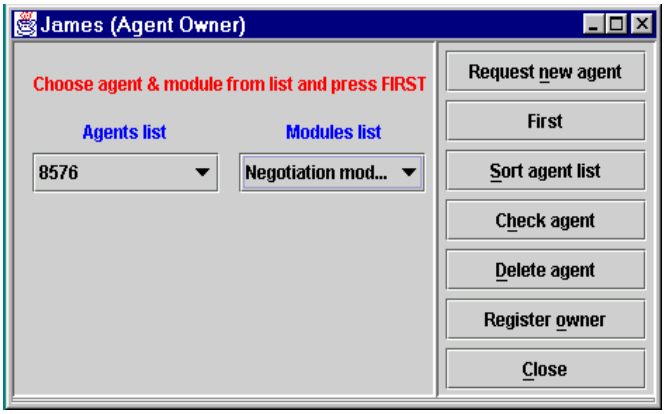

Fig. 9 A Screenshot of Exchanging Modules between Agents

7 Conclusions

In this paper, we present a new approach for evolving software agents in e-commerce. Agent fitness and life cycle are proposed to facilitate and control the process of agent evolution. With agent group and modularized agent structure, we construct multi-agent evolution cycle, which includes stages of restructuring, selection, and growing. We use GP operators to restructure software agents. Inter-agent and intra-agent crossovers as well as other types of restructuring operators are addressed. Finally, we discuss the integration of agent evolution into the SAFER architecture, and the primary phase of implementation is introduced.

We are improving our approach and implementation in two aspects. Firstly, a virtual marketplace is being constructed as an arena for agents, where they can accomplish their tasks through interaction. This marketplace can help us to test an agent's capability and adaptability in certain e-commerce activities, such as bidding and comparison-shopping. Secondly, we aim to observe the

fitness pattern of agent groups in the evolution course to find what kind of group model can boost agent fitness more effectively.

Bibliography

[1] Guttman, R.H. and Maes, P. (1999). Agent-mediated negotiation for retail electronic commerce. In Agent-Mediated Electronic Commerce: First International Workshop on Agent Mediated Electronic Trading, (Noriega, P., and Sierra, C. ed.), Berlin: Springer, 70-90.

[2] Krishna, V. and Ramesh, V.C. (1998). Intelligent agents for negotiation in market games, part1: model. IEEE Transactions on Power Systems, 13(3), 1103-1108.

[3] Wang, T.H., Guan, S.U., and Ong, S.H. (2000). An agent-based auction service for electronic commerce. In Proceedings of International ICSC Symposium on Interactive and Collaborative Computing, Australia.

[4] Wurman, P.R., Wellman, M.P., and Walsh, W.E. (1998). The Michigan Internet AuctionBot: a configurable auction server for human and software agents. In Proceedings of the Second International Conference on Autonomous Agents, Minneapolis, USA, 301-308.

[5] Chavez, A. and Maes, P. (1998). Kasbah: an agent marketplace for buying and selling goods. In Proceedings of First International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology, London, 75-90.

[6] Zhu, F.M., Guan, S.U., and Yang, Y. (2000). SAFER E-Commerce: Secure Agent Fabrication, Evolution & Roaming for E-Commerce. Internet Commerce and Software Agents: Cases, Technologies and Opportunities, (Rahman, S.M. and Bignall, R.J. ed.). Idea Group, PA, 190-206.

[7] Corradi, A., Montanari, R., and Stefanelli, C. (1999). Mobile agents integrity in e-commerce applications. In Proceedings of 19th IEEE International Conference on Distributed Computing Systems, 59-64.

[8] Greenberg, M.S., Byington, J.C., and Harper, D.G. (1998). Mobile agents and security. IEEE Communications Magazine, 36(7), 76-85.

[9] Marques, P.J., Silva, L.M., and Silva, J.G. (1999). Security mechanisms for using mobile agents in electronic commerce. In Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems, 378-383.

[10] Haynes, T. and Wainwright, R. (1995). A simulation of adaptive agents in a hostile environment. In Proceedings of the 1995 ACM Symposium on Applied Computing, ACM Press, 318-323.

[11] Haynes, T., Wainwright, R., and Sen, S. (1994). Evolving cooperation strategies. Department of Mathematical Computer Sciences, The University of Tulsa Technical Report No. UTULSA-MCS-94-10.

[12] Gimenez-Funes, E., Lgodo, L., Rodriquez-Aguilar, J.A., and Garcia-Calves, P. (1998). Designing bidding strategies for trading agents in electronic auctions. In Proceedings of International Conference on Multi Agent Systems, 136-143.

[13] Dworman, G., Kimbrough, S., and Laing, J. (1996). Bargaining by artificial agents in two coalition games: a study in genetic programming for electronic commerce, in Proceedings of the AAAI Genetic Programming Conference, Stanford, USA.

[14] Richter, C.W., Sheble, G.B., and Ashlock, D. (1999). Comprehensive bidding strategies with genetic programming/finite state automata. IEEE Transactions on Power systems, 14(4).

[15] Namatame, A. and Sasaki, T., (1998). Competitive evolution in a society of self-interested agents. In Proceedings of IEEE World Congress on Computational Intelligence.

[16] Koza, J.R. (1992). Genetic programming: on the programming of computers by natural selection, MIT press, Cambridge, MA, USA.

[17] Langdon, W.B. (1998). Genetic programming and data structures: genetic programming + data structures = automatic programming! Kluwer Academic Publishers, Boston.

[18] Guan, S.U. and Yang, Y. (1999). SAFE: secure roaming agent for e-commerce, In Proceedings the 26th International Conference on Computers & Industrial Engineering, Melbourne, Australia, 33-37.

[19] Guan, S.U., Zhu, F.M., and Ko, C.C. (2000). Agent fabrication and authorization in agent-based electronic commerce, in Proceedings of International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce, Wollongong, Australia, 528-534.

[20] Calderoni, S., Marcenac, P., and Courdier, R. (1998). Genetic encoding of agent behavioral strategy. In Proceedings of International Conference on Multi Agent Systems.

[21] Poh, T.K. and Guan, S.U. (2000). Internet-enabled smart card agent environment and applications. Electronic Commerce: Opportunities and Challenges, (Rahman, S.M. and Raisinghani, M. ed.), Idea Group, PA, 246-260.

[22] Hua, F. and Guan, S.U. (2000). Agent and payment systems in e-commerce, Internet Commerce and Software Agents: Cases, Technologies and Opportunities, (Rahman, S.M. and Bignall, R.J. ed.). Idea Group, PA, 317-330.

[23] Yang, Y. and Guan, S.U. (2000). Intelligent mobile agents for e-commerce: security issues and agent transport. In Electronic Commerce: Opportunities and Challenges, (Rahman, S.M. and Raisinghani, M. ed.). Idea Group, PA, 321-336.