Mobile Agents in E-Commerce: A quantitative evaluation

Rahul Jha, Srinath Perur, Vikram Jamwal and Sridhar Iyer
K R School of Information Technology, IIT Bombay.
{*rahul, srinath, vikram, sri*}*@it.iitb.ernet.in*

Abstract

Mobile Agents (MA) are emerging as a promising paradigm for the design and implementation of e-commerce applications. While MAs have generated considerable excitement in the research community, they have not translated into a significant number of real-world applications. One of the main reasons for this is the lack of work that quantitatively evaluates the effectiveness of MAs versus traditional approaches. This paper contributes towards such an evaluation.

We first classify the existing MA applications in e-commerce and identify the underlying mobility patterns. We discuss possible Client-Server (CS) and Mobile Agent based implementation strategies for each of these patterns. We have performed experiments to quantitatively evaluate these different strategies using the Voyager framework to implement MAs. We present our observations and discuss their implications

Keywords : mobile agents, client-server, e-commerce, quantitative evaluation, Voyager framework, implementation issues.

1 Introduction

The emergence of e-commerce applications has resulted in new net-centric business models. This has created a need for new ways of structuring applications to provide cost-effective and scalable models.

Mobile Agent (MA) systems have for some time been seen as a promising paradigm for the design and implementation of distributed applications including e-commerce. A mobile agent is a program that can *autonomously migrate* between various nodes of a network and perform computations on behalf of a user. Some of the benefits provided by MAs for creating distributed systems include reduction in network load, overcoming network latency, and disconnected operations [9]. MAs are also useful in applications requiring distributed information retrieval since they move the location of execution closer to the data to be processed.

While MAs have generated considerable excitement among the research community, they have not translated into a significant number of real applications. One of the main reasons for this is the lack of work that tries to quantitatively compare the performance of MA implementations of specific applications with other implementations [9].

In this paper, we first classify existing MA applications in the domain of e-commerce and identify patterns of mobility that underly these applications. We discuss several strategies for implementing each of these patterns using the traditional Client-Server (CS) and the MA paradigms. We also identify various application parameters that influence performance, such as, size of CS messages, size of MA, number of remote information sources, etc., and study their effect on application performance. We have chosen an application representative of the domain of e-commerce and have implemented it using both CS and MA paradigms. These implementations consist of CS applications using Java and MA applications using the Voyager framework [7]. We present our observations and conclusions regarding the choice of appropriate implementation strategies for different application characteristics.

Section 2 provides a discussion on MA applications in e-commerce. Section 3 provides a classification of identified mobility patterns, and section 4 discusses various implementation strategies for these mobility patterns. Section 5 describes our experiments and presents our results. Section 6 concludes with a discussion on our inferences and their implications.

While there exist some qualitative studies of MAs in e-commerce [2], [6], [13] and some quantitative studies of MAs in other application domains [8], [12], to the best of our knowledge, there is no literature on the quantitative study of MA applications in the domain of e-commerce.

2 Mobile Agents in E-Commerce

The number of people buying, selling, and performing transactions on the Internet is expected to increase at a phenomenal rate. The application of MAs to e-commerce provides a new way to conduct business-to-business, business-to-consumer, and consumer-to-consumer transactions [10]. We classify existing MA applications in e-commerce into three categories, viz., shopping agents, salesman agents, and auction agents.

2.1 Shopping agents

These MAs make purchases in e-marketplaces on behalf of their owner according to user-defined specifications. This model of e-commerce uses a *customer-driven market place*. A typical shopping agent may compare features of different products by visiting several online stores and report the best choice to its owner. The MA carries the set of features to be considered and their ideal values as specified by its owner. It is given one or more sites to visit and may dynamically visit other sites based on subsequent information. Since the MA moves to the source of information, the overhead of repeatedly transferring potentially large amounts of information over a network is eliminated. One example of a system that implements shopping agents is MAgNET, where agents deal with procurement of the many components needed to manufacture a complex product [11].

2.2 Salesman agents

These MAs behave like a traveling salesman who visits customers to sell his wares. This model of e-commerce uses a *supplier driven marketplace* and is particularly attractive for products with a short shelf-life. A supplier creates and dispatches an MA to potential buyers by giving it a list of sites to visit. The MA carries with it information about available stock and price of the product. Since the MA moves to the destination, the network and processing latencies that contribute to delays in servicing orders may be reduced. A system implementing salesman agents is discussed in [10].

2.3 Auction agents

These MAs can bid for and sell items in an online auction on behalf of their owners. Each MA carries along with it information about its owners bidding range, time within which the item is to be procured, bidding pattern, and other relevant attributes. In the presence of multiple auction houses, MAs can be used for collecting information across them. An agent can make a decision to migrate to one of them dynamically, depending on the amount of information transmitted, latency, etc. Some advantages of using MAs include allowing disconnected operation of auction agents, reducing network traffic, and facilitating quicker response during auction. One example of a system that implements mobile auction agents is Nomad[14].

From the above classification, it may be observed that mobility in MAs can be characterized by the set of destinations that an MA visits, and the order in which it visits them.

3 Mobility Patterns

We have identified the following parameters to characterize the mobility of an MA:

1. **Itinerary:** the set of sites that an MA has to visit. This could either be statically fixed at the time of agent initialization, or dynamically determined by the MA.
2. **Order:** the order in which an MA visits sites in its itinerary. This may also be determined statically or dynamically.

Based on these parameters, we distinguish MA applications in e-commerce as possessing one of the following *mobility patterns*:

- **Static Itinerary (SI)**
  The itinerary of the MA used in the application is known *a priori* and does not change. We further distinguish such applications based on order as:
  - **Static Itinerary Static Order (SISO)**
    The order in which an MA completes its itinerary is static and known *a priori*. An example application is an auction MA which is required to visit a set of auction houses in a specified order.
  - **Static Itinerary Dynamic Order (SIDO)**
    The order in which an MA completes its itinerary is decided dynamically by the MA. An example application is a shopping MA which finds the minimum price for a product from a set of on-line shops. The order in which the shops are visited may be irrelevant and could be dynamically determined by the MA.
- **DI (Dynamic Itinerary)**
  The itinerary of the MA used in the application is determined dynamically by the agent itself. However, at least the first site in the itinerary should be known *a priori*. An example application is a shopping MA that is required to find a particular product. A shop that does not contain the product may recommend an alternative shop, and this recommended shop is included in the MAs itinerary dynamically.
  It may be noted that dynamic itinerary implies dynamic order and the distinction between static order and dynamic order is not meaningful in this case.

4 Implementation Issues

4.1 Implementation Strategies

We have identified four implementation strategies that may be adopted by e-commerce applications:

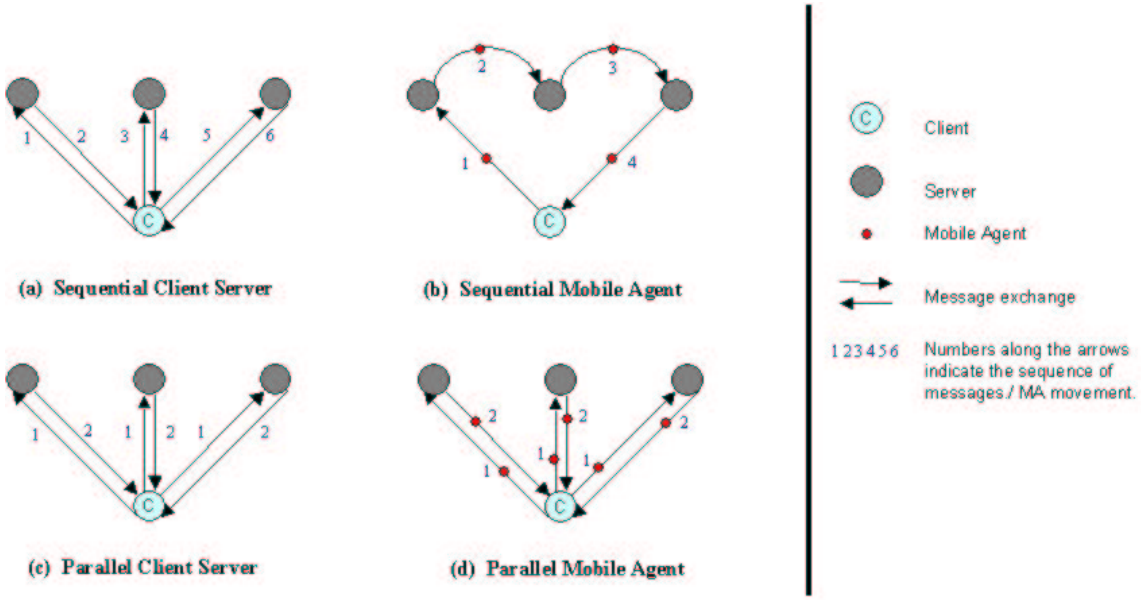

Figure 1: Implementation strategies

1. **Sequential CS**
   This is based on the traditional client-server paradigm. The client makes a request to the first server and after processing the reply, makes a request to the second server and so on, till the list of servers to be visited is exhausted. This strategy is illustrated in figure1(a).

2. **Sequential MA**
   In this case a single MA moves from its source of origin (client) to the first site (server) in its itinerary. It then moves to the next site and so on, till it has visited all the sites in its itinerary. This strategy is illustrated in figure 1(b).

3. **Parallel CS**
   This also based on the client-server paradigm. However, instead of sequential requests, the client initiates *parallel* threads of execution where each thread concurrently makes a request to one of the servers and processes the reply. This strategy is illustrated in figure 1(c).

4. **Parallel MA**
   In this case the client initiates multiple MAs, each of which visits a subset of the servers in the itinerary. The MAs then return to the client and collate their results to complete the task. This strategy is illustrated in figure 1(d).

It is also possible to use combinations of the above strategies. In our experiments, we restrict ourselves to these four strategies only.

4.2 Implementation for different mobility patterns

The feasible implementation strategies for different mobility patterns identified in section 3 are as follows:

1. **SISO**: Since the order of visit is fixed statically, the possible implementation strategies in this case are:

   - Sequential CS
   - Sequential MA

   Parallel CS and parallel MA strategies cannot be used for SISO applications since the order in which the MA visits servers may be important to the application being implemented.

2. **SIDO:** Since the order of visit is determined dynamically, all the strategies outlined in section 4.1 are possible, namely:

   - Sequential CS
   - Sequential MA
   - Parallel CS
   - Parallel MA

3. **DI:** Since the itinerary is determined dynamically, the possible implementation strategies in this case are:

   - Sequential CS
   - Sequential MA

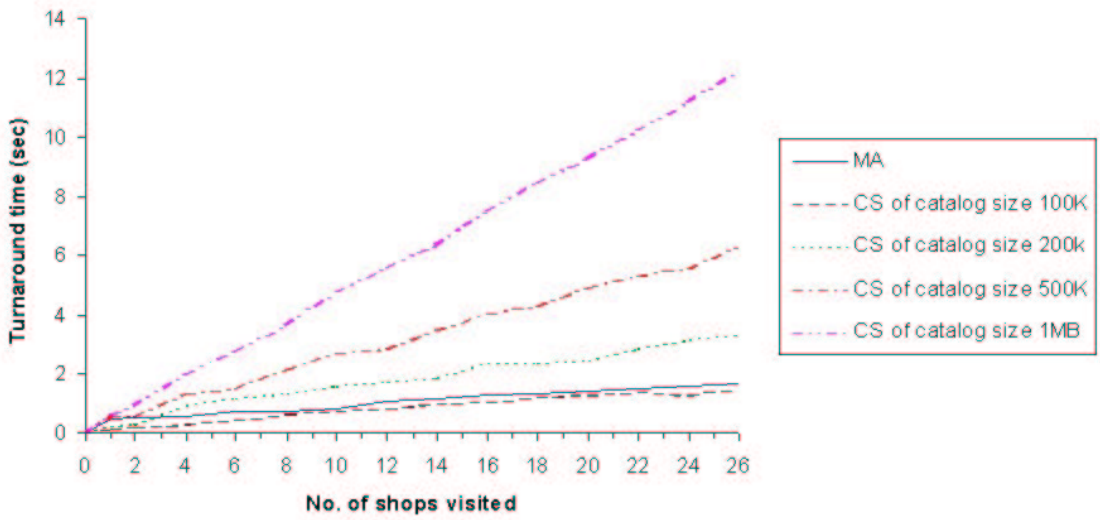

Figure 2: Effect of catalog size on turnaround time for sequential MA & sequential CS

Parallel CS and parallel MA strategies cannot be used for DI applications since information about the servers to be visited is not known *a priori*.

The selection of the "ideal" implementation strategy from those feasible for a given application could be based on several criteria such as ease of implementation, performance, availability of technology, etc. The following section describes the details of implementing an application using different strategies.

5 Experimentation and Results

5.1 Experimentation

We have chosen a typical e-commerce application, viz., that of a single client searching for information about a particular product from the catalogs of several on-line stores. We assume that the client requires a highly customized search which the on-line store does not support. This would require the client to fetch a relevant subset of the catalog and implement a search at its end. We have implemented such an application using all four strategies mentioned in section 4.1.

We have used the Voyager Framework for MA implementations. Voyager is an ORB (Object Request Broker) implemented in Java and provides support for mobile objects and autonomous MAs [7]. The CS implementation consists of a server that sends a catalog on request and a multi-threaded client that requests a catalog from one or more servers. The client and the server have been implemented in Java.

The experiments were carried out on P-III, 450 MHz workstations connected through a 10 Mbps LAN with typical student load. We have considered the following parameters for comparing the performance of these implementations:

1. number of stores (varies from 1 to 26);
2. size of catalog (varies from 20 KB to 1 MB);
3. size of client-server messages (varies proportionately with catalog size);
4. size of an MA (fixed at 4.6 KB);
5. processing time for servicing each request (varies from 10 ms to 1000 ms);
6. network latencies on different links (assumed constant since all workstations were on the same LAN);

Our performance metric is the user *turnaround time*, which is the time elapsed between a user initiating a request and receiving the results. This includes the time taken for agent creation, time taken to visit/collect catalogs and the processing time to extract the required information.

5.2 Results

The results of our experiments are shown in the graphs of figures 2, 3, 4, and 5. Some of our observations are:

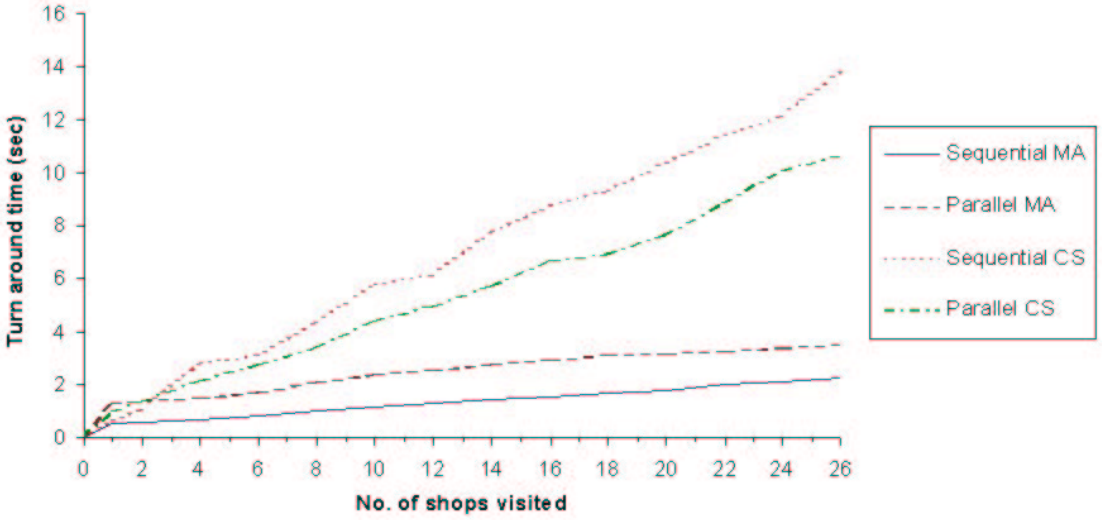

Figure 3: Turnaround time for different implementation strategies for a processing time of 20 ms (catalog size of 1 MB).

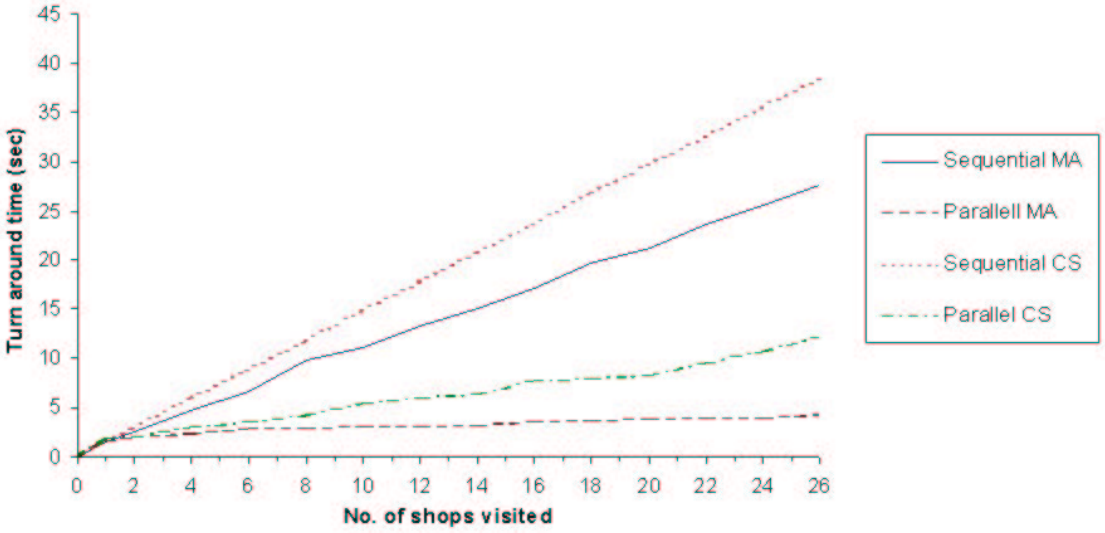

Figure 4: Turnaround time for different implementation strategies for a processing time of 1000 ms (catalog size of 1 MB).

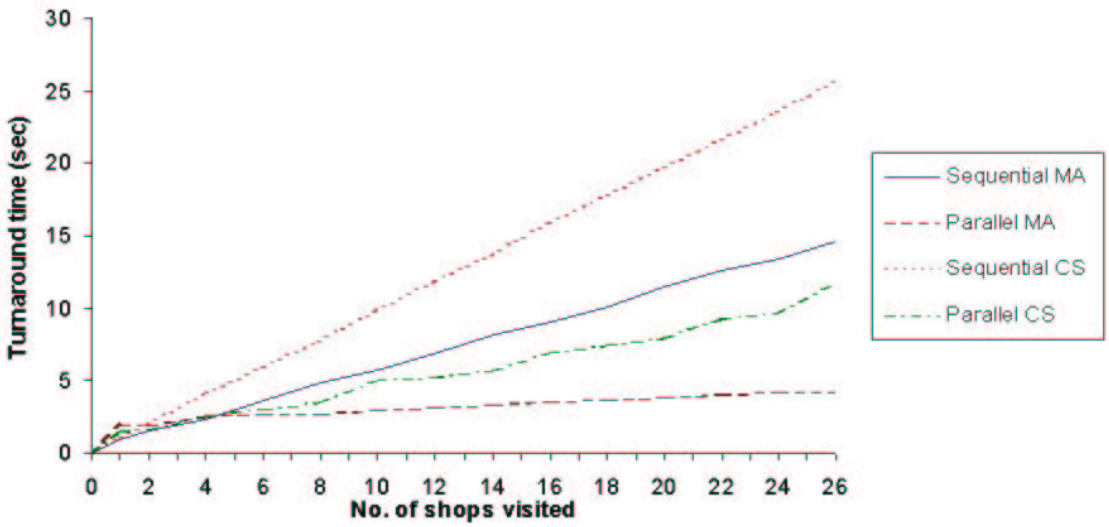

Figure 5: Turnaround time for different implementation strategies for a processing time of 500 ms (catalog size of 1 MB).

- The performance of sequential MA remains the same for different catalog sizes while performance of sequential CS degrades with increase in catalog size (Fig.2).
- Sequential CS implementation in our case performs better than sequential MA for a catalog size less than 100 KB (Fig.2).
- With a catalog size of 200 KB, sequential MA starts to perform better than sequential CS when the number of shops to visit is 4 (Fig.2).
- For small processing delays (20 ms), sequential MA performs better than all other strategies (Fig.3).
- For higher processing delays (1000 ms), parallel implementations show better performance than sequential implementations. Also, parallel MA performs better than parallel CS in this case (Fig.4).
- With a processing time of 500 ms, parallel MA and parallel CS implementations begin to perform better than sequential MA when the number of shops to visit is 6 (Fig.5).

6 Conclusions

We have classified existing MA applications in e-commerce, identified underlying mobility patterns for these applications and discussed possible implementation strategies for these patterns using the client-server and mobile agent paradigms. We have performed experiments to quantitatively evaluate these different strategies using the Voyager framework for MA implementations and Java for CS implementations.

Sequential CS implementations are most suitable for applications where a small amount of information has to be retrieved from few remote information sources (servers), and the degree of processing required is low (our experiments provide a good quantitative indication of these parameters). However, these conditions do not hold for most real-world e-commerce applications. MAs scale effectively across the above parameters, and with scalability being one of the needs of net-centric computing, we find that MAs are an appropriate technology for implementing e-commerce applications. Parallel implementations are effective when processing information contributes significantly to the turnaround time.

We are in the process of carrying out further experiments towards identifying the "ideal" implementation strategy given an application's characteristics. Efforts to conduct these performance comparison experiments on an Internet-wide scale are underway. We are also investigating the extent to which latencies introduced by MA frameworks influence performance.

References

[1] Jonathan Bredin, David Kotz and Daniela Rus, "Market-based Resource Control for Mobile Agents", *Proceedings of Autonomous Agents*, May 1998.

[2] Antonio Carzaniga, Gian Pietro Picco and Giaovanni Vigna, "Designing Distributed Appli-

cations with Mobile Code Paradigms", *Proceedings of the 19th International Conference on Software Engineering(ICSE '97)*, pp. 22 - 32, ACM Press, 1997.

[3] P. Dasgupta, N. Narasimhan, L.E. Moser and P.M. Melliar-Smith, "A Supplier Driven Electronic Marketplace Using Mobile Agents", *Proceedings of the First International Conference on Telecommunications and E-Commerce, Nashville, TN*, Nov. 1998.

[4] P. Dasgupta, N. Narasimhan, L.E. Moser and P.M. Melliar-Smith, "MAgNET: Mobile Agents for Networked Electronic Trading", *IEEE Transactions on Knowledge and Data Engineering, Special Issue on Web Applications*, July-August 1999.

[5] Alfonso Fuggetta, Gian Pietro Picco and Giovanni Vigna , "Understanding Code Mobility", *IEEE Transactions on Software Engineering*, vol. 24(5), 1998.

[6] Carlo Ghezzi and Giovanni Vigna, "Mobile code paradigms and technologies: A case study", *Proceedings of the First International Workshop on Mobile Agents, Berlin, Germany*, vol. 1219 of Lecture Notes on Computer Science, Springer, April 1997.

[7] G. Glass, "ObjectSpace Voyager Core Package Technical Overview", *Mobility: process, computers and agents*, Addison-Wesley, Feb. 1999.

[8] Dag Johansen, "Mobile Agent Applicability", *Proceedings of the Mobile Agents 1998, Berlin*, Springer-Verlag, Lecture notes in computer science; vol. 1477, ISBN 3-540-64959-X, (1998), pp. 9-11 September, 1998.

[9] Danny B. Lange and Mitsuru Oshima, "Seven Good Reasons for Mobile Agents", *Communications of ACM*, vol. 42, no. 3, March 1999.

[10] Pattie Maes, Robert H. Guttman and Alexandros G. Moukas, "Agents That Buy and Sell", *Communications of ACM*, vol. 42, no. 3, pp. 81 - 91, March 1999.

[11] Todd Papaioannou, "Mobile Information Agents for CyberSpace - State of the Art and Visions", *To appear in Proc. of Cooperative Information Agents (CIA - 2000)*, 2000.

[12] Stavros Papastavrou, George Samaras and Evaggelia Pitoura, "Mobile Agents for WWW Distributed Database Access", *Proceedings of IEEE International Conference on Data Engineering (ICDE99)*, 1999.

[13] Gian Pietro Picco and Mario Baldi, "Evaluating Tradeoffs of Mobile Code Design Paradigms in Network Management Applications", *Proceedings of 20th International Conference on Software Engineering, ICSE98, Kyoto, Japan, IEEE CS Press*, 1998.

[14] Thomas Sandholm and Qianbo Huai, "Nomad: Mobile Agent System for an Internet-Based Auction House", *IEEE Internet Computing*, vol. 4, no.2, March-April 2000.