A Q VALUE LEARNING ALGORITHM FOR E-COMMERCE WEB APPLICATIONS

Rayaguru Akshaya Kumar Das\textsuperscript{1}, Dr. A. B. Khan\textsuperscript{2}, Dr. S. K. Misra\textsuperscript{3} and Dr. Binod Pattanayak\textsuperscript{4}

\textsuperscript{1}Computer science, Institute of Management and Information technology, Cuttack, Odisha, India
rakdas@yahoo.co.uk
\textsuperscript{2}Computer science, Institute of Management and Information technology, Cuttack, Odisha, India
\textsuperscript{3}Management, Institute of Management and Information technology, Cuttack, Odisha, India
\textsuperscript{4}Computer science, ITER, SoA University, Bhubaneswar, Odisha, India

Abstract: The rapid growth of e-commerce brings itself new innovative technologies and algorithm to predict the user behaviour and suggest product accordingly. Increasing the QoS is a major challenge in e-commerce sites. This paper considers the browsing pattern of a user in the current session by analyzing the click stream this helps to determines the section the user interested in and accordingly suggests the products of a particular category. An algorithm using the MDP is suggested which analyzes the click stream to determine the user need thereby contributing improvement of QoS.

Keywords: E-Commerce, QoS, MDP, response time, MDP, Q value

INTRODUCTION

The popularity and success of E-Commerce not only depends on the penetration of ICT rather it brings itself a whole range of challenges in automation and technological architecture. Today, it is possible to seamlessly conduct the E-Commerce transaction both in terms of technology and logistics due to the various level of sophistication of technology. At present the e-commerce is considered one of the useful Internet applications that can be on soft real time which are beneficial to the Web Services [2]. This happens because in complex e-commerce applications, different servers of the company interact with each other. These servers may use different platforms and languages. While doing so, it is very important that the desired level of QoS should be maintained by the e-commerce web applications. Guaranteeing QoS which will help in avoiding dissatisfied customers leave the sites and do not comeback. [3] The QoS in Web-service is a set of non functional parameters such as performance reliability, security and availability. Some of the QoS attributes are:

- Availability: Availability of a web service is calculated as the percentage of time the web service is available for service.
- Confidentiality: It means only the sender and the receiver could be able to understand the content of the message and the confidentiality of the message is maintained for others.
- Cost: It is the cost of service charged by the service provider for the service
- Reputation: It is a measure of user satisfaction
- Response time: Response time is measure of time spent between the request made to the server for a service and service provided.

Each web service has its own QoS attributes, to calculate the QoS of the entire web services, it is essential to calculate the value of attributes. In Table 1 we show s an example of aggregation of the attributes which is adapted from [4].

<table>
<thead>
<tr>
<th>QoS Attributes</th>
<th>[ \prod_{i=1}^{n} availability (w_{si}) ]</th>
<th>[ \sum_{i=1}^{n} Cost (w_{si}) ]</th>
<th>[ \sum_{i=1}^{n} ResponseTime (w_{si}) ]</th>
<th>[ \sum_{i=1}^{n} Confidentiality (w_{si}) \times \frac{1}{n} ]</th>
<th>[ \sum_{i=1}^{n} Reputation (w_{si}) \times \frac{1}{n} ]</th>
</tr>
</thead>
</table>

The action of web service can be described as a sequence of tasks with an distinct initial and final task. Every web has a different QoS attribute even though there may be other service with same or similar functionality. E-commerce systems are usually highly QoS-sensitive and the profits of them are directly dependent on their QoS provision [6]. The QoS provision of an e-commerce system is mainly depend on its e-Commerce web services.
RELATED WORK

Customer behavior modeling in E-Commerce websites is used to analyze the recorded customer history. To predict the future action of a customer is essential to understand online-buying behavior. This is of utmost importance for e-commerce website managers [7]. Extensive studies have been done, using different features and techniques. A simple logistic model, which predicts the status of visitors (active or lapsed) and the usage frequency was proposed in [8]. Probability of a customer leaving the E-commerce site was proposed using session frequency parameters[9]. The paper categorizes the website pages in several categories, then using Logit model, and measured the probability of all pages in each category per current and previous session[9]. Another approach was proposed to predict the number of webpages in specific categories that are visited in a single session based on Hidden Markov Model [10]. Based on statistical methods a predictive model was proposed which considers the sequence of pages visited by users, and their think time on each page [7]. Another model was proposed a customer behavior model based on past customer behavioral variables (recency, frequency, and monetary value)[6]. Detailed click-stream measures and customer demographics[6]. A large scale analysis was performed on the eBay query log data. They considered the customers requested query sequence and their demographic as their model features[11]. Most important features used in the previous studies to model the customer behavior are recent visited pages, the number of visits of each page, and the sequence of visited pages in each session, but none of them has taken into account the quality of services provided to customers. This feature plays an important role in the customer behavior model, because E-commerce websites are usually highly QoS-sensitive.

APPROACH

E-Commerce sites are very complex and consists of multiple layers of webservers, database servers, application servers etc. A request execute is firstly handled by the web servers. In response to the request generally a HTML page is returned to the client which is created dynamically by the web service. This paper suggest an algorithm that uses the browsing pattern of the customer in the different sections of the E-Commerce website and displays the section the section of the E-Commerce site that the customer is interested in a more prominent way thereby increases the response time of the QoS and thus improve the QoS.

Modern E-Commerce sites are very huge and technologically complex and consist of large number of sections. The sections vary from site to site but they are present in almost all generic E-commerce site. To increase the throughput and to target the customers, the E-Commerce sites have come up with different attractive sections including different offers or similar sections or categories. Some sites present few products of each product on the homepage based on the customer profiling. Most of the times the customer has to linearly browse through products. Our algorithm will help to choose the customer by providing more products of the desired category of the customer by instant decision making based on the immediate click stream feedback from the customer. This feature incorporated will display the products of a particular section in prominence compared to the products of other section and thus will improve the QoS.

The objective of the paper is to improve the QoS of web services by increasing the reputation which is a measure of customer satisfaction. The customer satisfaction increases when a web application can predict the product or even the category in which the desired product is placed which the customer is searching in the current session. This current session search of a product may or may not correspond to the previous session browsing history. This paper only considers the browsing of the current session and based on the click stream and the browsing pattern different sections are presented to the customer so that it minimizes the number of clicks to find the product. The lesser the number of clicks the more is the satisfaction and hence increases the reputation which ultimately increases the QoS. This will also improve the throughput by lowering runtime overhead which will ultimately improve the efficiency of the entire system.

Web services contain a huge number of configurable parameters. Out of all the parameters, only some are performance relevant. Therefore to improve the performance of the system, the critical configuration parameters on which the performance depends has to be identified first. Including all the configuration parameters will unnecessarily increase the overhead which will cause substantial time delay. This may lead to an unstable system. From the performance perspective, it is a challenge to select the appropriate configuration parameters. In [18], authors used parameter dependency graph to find out the performance of relevant parameters and their relationship. The essential parameters crucial for searching can be MaxClients, Keepalive timeout, MaxSpareServers, MinSpareServers etc (which some key performance parameter in Apache).

In this paper we propose an agent to improve the set of objectives. The proposed agent consists of three key parts: (I) monitor, (II) decision maker, and (III) controller. The monitor is meant to measures the performance of the System. In regular intervals the monitor measures the web performance and it sends the information thus collected to the decision maker. The information generally related to performance of the application such as response time or throughput etc. Based on the information sent by the monitor, the decision maker produces a state action table, called Q-value table. In this context a state is defined as the status of configured parameters. The possible actions include increasing, decreasing their values of the parameters or keeping unchanged. Based on the dynamically updated Q table, the configuration controller generates the rules which can reconfigure the entire system if needed.

DECISION MAKING METHODOLOGY

The web applications can be configured or programmed to learn the behavior of the customer through interactions with an external environment The configuration problem can be as a finite “Markov Decision process(MDP)”. The MDP consists of states and actions for each state. During the transition of state, the agent receives a reward defined by a reward function:

\[ R = E[r_{t+1}|s = s, a = a, s_{t+1} = s'] \]

The aim of the agent is to develop a policy \( \pi : S \rightarrow A \) to maximize the collected cumulative rewards based on iterative trial-and error interactions [13].

In our case, the configuration problem is considered as a MDP, by defining state space \( S \), action set \( A \), and immediate reward function \( r(s, a) \).
State Space: In this case, a state is defined as possible system configurations. If there are n performance critical parameters, a state can be represented in the form:

\[ s_t = (\text{Para}_1, \text{Para}_2, \cdots, \text{Para}_n) \]

Action Set: Three basic action sets associated with parameters can be defined in this scenario: increase, decrease, and keep the parameter. A vector \( a_t \) used to represent the action taken on each parameter \( i \). Each parameter has three element vectors which take values 1 or 0 indicating taken or not taken for each of the three actions: increase, decrease and keep. The following notation can be used to represent an increase action on parameter \( i \):

\[ a_{\text{increase}}^i = (\cdots, 1, 0, 0, \cdots, 0) \]

Reward: The reward should accurately reflect the system performance. The immediate reward \( r \) at time interval \( t \) can be defined as

\[ r_t = \text{SLA} - \text{perf} \]

where, \( \text{SLA} \) is Service Level Agreement predefined time, and \( \text{perf} \) is measured response time. For a given \( \text{SLA} \), a lower response time returns a positive reward else negative is returned.

**Q VALUE LEARNING**

The temporal difference (TD) is most suitable here. It has two advantages: (i) It needs no model of the environment (II) it updates Q-values at each time step. Using this, the average Q-value of a particular action \( a \) on particular state \( s \), denoted by \( Q(s, a) \), can be filtered after each reward \( r \) is received:

\[ Q(s, a) = Q(s, a) + \alpha \times (r_t + \gamma Q(s_{t+1}, a) - Q(s, a)) \]

where, \( \alpha \) is a learning rate parameter that facilitates convergence to the true Q-values in the presence stochastic rewards and different state transitions, and \( \gamma \) is the discount rate to guarantee the accumulated reward convergence in continuing task. Algorithm 1 presents the pseudo code of our Q-value learning algorithm.

Text of manuscript should be arranged in the following order: Title, Abstract, Introduction, Body Text, Results and Discussion, Conclusion, Acknowledgements and References.

**Q-value Learning Algorithm.**

Initialize Q table

Initialize state \( s \)

error = 0

Do

for each state \( s \) do

\[ a = \text{get action}(s) \text{ using } \text{greedy policy} \]

for \( (i = 1; i < 0; i < \text{limit}; i++) \)

observe \( r \text{ and } s_{t+1} \)

\[ Q_t = Q_t + \alpha \times (r + \gamma Q_{t+1} - Q_t) \]

error = MAX(error, \( |Q_t - Q_{\text{previous}}| \) )

\[ s_{t+1}, a_{t+1} = \text{get action}(s_t), a = a_{t+1} \]

end for

end while error < 0

A practical problem with the basic algorithm is that the number of Q values that need to be explored increases exponentially with the number of the attributes used the state representations.

**EVALUATION**

To evaluate the working of this algorithm, we deployed in a experimental two tired website. A client machine was used to emulate the customer, all the experiments were carried out in same local network. The two tired websites were deployed on two VMs with Apache web server in the first one and the Tomcat application server and MYSQL database server in the other.

**REFERENCES**


