IntWEB: An AI-Based Approach for Adaptive Web

Oznur Kirmemis Alkan, Pinar Senkul
Middle East Technical University, Computer Engineering Department, Ankara , Turkey
{oznur.kirmemis, senkul}@ceng.metu.edu.tr

Abstract

The World Wide Web is an endless source of information, which is mainly represented in the form of Web pages. The way that the users browse the Web depends on both user-oriented factors like the information the users are seeking for and site-oriented factors like the attractiveness of the Web sites, the structure or more specifically, the navigational organization. However, the site's design changes through time, due to factors like including more information about items or introducing new items. In addition, the user's needs and preferences also change, which together bring difficulties for building Web sites that best suit users' needs. Therefore, it is a very important and challenging task to adapt the Web sites automatically in order to facilitate users' navigation in the Web site. This paper proposes a solution to adapt Web sites structural organization according to users' navigation patterns. In the proposed solution, the adaptive web problem is formulated as a classical AI search problem, and a novel Hill Climbing based solution is devised. The proposed solution is realized in a framework, namely IntWEB. The technique is applied to a real-life case study and results are discussed in order to evaluate IntWEB's performance.

1 Introduction

There is a huge growth of information sources on the Internet every day and together with this growth, the user base also grows. In such a situation, the necessity of managing this information for large number of users with possibly diverse needs arises. Automatic personalization is the key technique that is utilized for developing solutions in such situations. Different personalization methods have been utilized by different solutions in order to deliver and present relevant information for individual users. For instance, personalization can be needed for developing information systems that act according to user preferences for WAP-enabled devices [Cotter and Smyth, 2000], or adapting web sites for users' interests, which is the problem discussed in this paper.

The World Wide Web, as an endless source of information, is mainly represented in the form of web pages. The way that the users browse the Web depends both on the information that they are searching for and the navigational structure that the Web site provides to its users. Therefore, factors such as the structure of the Web sites, or more specifically, their navigational organization affect users' tendency for Web surfing. On the other hand, users may not be seeking any information, but they may just be examining some product information. In such cases, attractiveness of the Web site and how well it meets user's taste become even more important in order to facilitate the navigation and keep the user in the Web site as long as possible. Therefore, Web sites design should in some way get adapted to its users' needs. Adaptive Web sites research emerged from these requirements.

Adaptive Web site is an attractive and challenging topic and the definition of this problem is provided in [Perkowitz and Etzioni, 1998] as follows: "*adaptive Web sites are the Web sites that automatically improve their organization and presentation by learning from visitor access patterns*" Adapting Web sites is considered as a challenging task since the adaptation process is built on the preferences of users; however, preferences of users are generally highly diverse and demanding. In other words, each of the visitors of a Web site may be searching for something different, and each may have unique needs or concerns. One solution to this problem can be clustering users according to their usage patterns and then creating different user interfaces for each user group. However, handling all these interfaces can be very difficult and may not be preferred by sites' managers. Another solution can be handling as many users' navigational patterns as possible so as to satisfy most of the users' preferences. Although this will dissatisfy some of the users, it can be considered a more feasible solution than creating many interfaces.

Adaptive Web sites have been studied by different disciplines like Machine Learning and Data Mining to be able to provide solutions from the ideas from these fields. In addition, Perkowitz and Etzioni mentioned and discussed in their work [Perkowitz and Etzioni, 1998] that, "*the goal of creating self-improving Web sites requires breakthroughs in different areas of Artificial Intelligence (AI)*", and they discuss several aspects of adaptive web problem and try to show that the problem is in fact an AI challenge. In addition, in [Perkowitz and Etzioni, 2000], they view automatic improvement of a Web site as a search problem in the space of possible Web sites, and they further mention that different approaches to creating adaptive Web sites correspond to different ways of searching the space. They also present a solution that relies on conceptual clustering techniques to solve adaptive Web problem.

In this work, we propose an AI-based solution for adapting Web sites according to users' preferences. The solution utilizes Hill Climbing search algorithm [Russell and Norvig, 2003] and the problem formulation through providing state description, actions, evaluation function and the adapted Hill Climbing solution is given in detail. The proposed technique is realized in a framework named IntWEB (Intelligent Web). The proposed solution is applied to a real-life case study, more specifically, the department's Web site, in order to evaluate the results of the adaptation system proposed.

The rest of the paper is organized as follows: In Section 2, information about the related work in this area is presented. In Section 3, INTWeb is presented in three parts; problem formulation, system components that are designed to realize the approach and the algorithm. In Section 4, the evaluation of the system is presented and the results are discussed. Finally, concluding remarks and future directions of research are given in Section 5.

2 Related Work

The main goal of any user-adaptive system is to provide users with what they need without asking it to the users explicitly [Mulvenna, Anand and Buchner, 2000]. Therefore, automatic personalization is the main technology in such systems. For the case of adaptation of web sites to user preferences, web servers hold the common and very rich source of knowledge about user navigation patterns and interests. Large amounts of data can be collected from server log files, which include the clickstream (web usage) data. Personalization on the Web requires to take full advantage of the flexibility provided by this data, and to effectively use the discovered models in an automatic personalization system.

The process of personalization can be viewed as an application of data mining through applying steps of a general data mining solution like data collection, pre-processing, pattern discovery and evaluation in an off-line mode, and finally the deployment of the knowledge in real-time to mediate between the user and the Web [Mobasher, 2007]. Data mining techniques have been extensively applied in order to extract usage patterns in several studies [Mobasher, Cooley, and Srivastava, 2000], [Mobasher, Dai, Kuo, and Nakagawa 2001], [Nasraoui and Petenes 2003]. In e-commerce also, researchers of adaptive web are increasingly using clickstream data, which was originally collected for website performance analyses. For instance, clickstream data has been used to find out the behaviors of customers across websites through user-centric scenarios [Goldfarb 2002] and within specific websites through site-centric scenarios [Sismeir and Bucklin, 2004]. For the case of site-centric scenarios, some studies focused on single visits to a given website, whereas some others studied multiple visits.

One example of well known adaptive web solution is the WebWatcher [Joachims, Freitag, and Mitchell, 1997] which suggests links to users based on the online behavior of other users. Initially, the system ask the users to provide the information regarding their reason to enter the Web site, or in other words, what they are seeking for towards using the Web site. In addition to this information, before users leave the Web site, they are asked to provide whether they have found what they were looking for. After that, those users' navigation paths together with their feedback are used in order to create suggestions for future visitors that seek the same content. The resulting suggestions are presented by highlighting the already existing hyperlinks.

Avanti project [Fink, Kobsa and Nill, 1996] is another system that also requires user's explicit feedback so as to adapt Web sites. In that project, the user's final objective as well as his next step is aimed to be discovered. A model for the user is built, partly based on the information that the user provides about him and also from his navigation paths. Using this information, direct links to pages, that are considered to be liked by the users, are presented to them. In addition to the consideration of specific likeliness, hyperlinks that lead to pages of potential interest to each visitor are highlighted. A drawback of both the WebWatcher and the Avanti project is that, they require the users to be in active participation with the system in the adaptation process by asking them to provide information about themselves.

The Footprints system [Wexelblat and Maes, 1999] does not need explicit user feedback, which is an advantage of it over the described systems above. It only uses the navigation paths of the users. The Footprints system does not perform user identification and the most frequent navigation paths are presented to the visitor in the form of maps and also the percentage of people who have followed those maps are displayed next to each link. Possible enhancements to the Web site are presented as a list of suggestions to users.

As mentioned in the previous section, Perkowitz and Etzioni presented Web site adaptation as an AI problem in [Perkowitz and Etzioni, 1997]. They discuss that, many AI advances, both practical and theoretical; can be used as a solution to such challenges. They mention that, the quest to build a chess-playing computer, for example, has led to many advances in search techniques, and similarly, the autonomous land vehicle project at CMU [Thorpe 1990] resulted not only in a highway-cruising vehicle but also in breakthroughs in vision, robotics, and neural networks. They furthermore mention that, they believe the adaptive Web sites challenge will both drive AI advances and yield valuable technology.

Following this discussion, same authors proposed a conceptual framework for adaptive Web sites in [Perkowitz and Etzioni, 2000]. The focus is mainly on the semi-automatic creation of index pages which are created through discovering clusters of page visits. The assumption in their solution is that, if a large number of users visit a set of pages frequently, these pages should be related. In order to find out those frequently accessed pages, they have developed two cluster mining algorithms, namely, PageGather and IndexFinder. PageGather relies on statistical methods to discover candidate link sets, whereas the IndexFinder is a conceptual cluster mining algorithm, which finds link sets that are conceptually coherent. The work was evaluated using three Web sites in which the automatically generated pages are placed. They observed the user response to these adapted pages. Although Perkowitz and Etzioni discussed that adaptive web problem can be considered as an AI problem, they did not provide a full formalization of the problem from an AI perspective.

There have not been many studies that propose solutions to adaptive web from the view of AI. In addition, most of the adaptive web based systems developed so far are described as compound systems that consist of many modules such as user profilers, log and web usage miners, content managers, Web site authoring tools and information acquisitioners and searchers [Eirinaki and Vazirgiannis, 2003]. However, in many of these approaches, clear description of an algorithm that performs the adaptation process is not provided. Therefore, one of the main drawbacks of the related work so far is that, in most of the approaches, there is no clear focus on adaptation formalism and algorithm on how the adaptation is performed. In this paper, IntWEB, a full AI based solution to the adaptive web problem is presented where Hill Climbing search method is used and the system is evaluated against a real-life case study. The main motivation of the study presented in this paper is to first formalize the problem as a classical AI search problem through giving all components of a classical search, and implementing a novel adaptation algorithm as well as evaluating the proposed approach with a real-life case-study.

2 IntWEB: Adapting Web Sites by using Hill Climbing

In IntWEB, adapting Web sites to users' preferences is handled as a search problem. The proposed solution is formulated as a classical AI problem and Hill Climbing search method is utilized to provide a solution for the adaptive Web. In this section, firstly, the problem formulation is described below. The components that construct the IntWeb framework are presented next. Finally, details of the algorithm are given.

2.1 Problem Formulation

Any classical AI problem can be formulated with four components, namely; *initial state, possible actions, goal test,* and *path cost*. In the following subsections, the formulation details of the adaptive web problem which forms the basis for IntWEB system is provided.

States

Each state in IntWEB is the entire Web site with the links that are taken to be under consideration for the Web site adaptation task. A state is therefore modelled as follows;

$$\mathbf{S} = < \mathbf{L_1}\langle \text{from\_ } L_1, \text{ to\_ } L_1\rangle, \mathbf{L_2}\langle \text{from\_ } L_1, \text{ to\_ } L_2\rangle, \ldots, \mathbf{L_k}\langle \text{from\_ } L_k, \text{ to\_ } L_k\rangle, \mathbf{Covers}>$$

Here, ***S*** is any configuration of the Web site that includes links *{**L_1, L_2,... L_k**}*. Each link ***L_i*** corresponds to a link from page ***from_ L_1*** to page ***to_ L_1***.

Covers parameter keeps the *goodness* of a state. The goodness of a state is described in the following subsections.

Initial State

Initial state refers to the initial configuration of the Web site. The search starts with the Web site that includes no links. It incrementally adds links to the initial state through state transitions so as to maximize the ***Covers*** value of the state. The search continues until no more improvements can be made to the current state at hand.

Action

Initial state refers to the initial configuration of the Web site. The IntWEB is based on Hill Climbing search which makes state transformations through **add_shortcut** action. The **add_shortcut** action is designed and implemented as a transformation function which takes a state **S** as input and produces a new state **S'** by adding a link from a page to another page such that the newly added link does not exist in **S**.

Evaluation Function

Evaluation function measures the *goodness of a state* according to the problem at hand. Here, the goodness of a state is measured in terms of the *number of sessions that the state's link configuration* covers. The assumption used in the evaluation function is that, *the pages that accessed together in a session should be related and they should have a link to each other*. Therefore, the evaluation function assigns higher scores to states which have better link configuration. Here, as more paths that the user follows in sessions are covered by a state, the link configuration gets improved. Moreover, as the link configuration of a state improves, higher score is assigned to that state by the evaluation function.

Stopping Condition

Since Hill Climbing algorithm is used, IntWEB will stop when no more improvements can be made.

2.2 IntWEB: System Components

In order to realize the solution described above, IntWEB is established on three components, namely, *Pre-processor*, *Web Crawler*, and *Web Adapter modules*. The description of each module is provided in the following subsections.

Pre-processor

The Pre-processor module takes log files, which contain accesses to pages for different users and from different sessions as input, and produces sessions, which are in fact sets of pages that are accessed together, as output. The module does its job in three phases. In the first phase, it cleans the log data so as to remove irrelevant access information like robot accesses, multimedia accesses, etc. In the second phase, it does user identification on this cleaned log file. In order to identify users, it will use the IP address of the users and the browser information. This is a commonly applied technique in literature [Srivastava, Cooley, Deshpande and Tan, 2000]. After users are identified, the log file together with user information is re-processed in the third phase, to identify sessions. In session identification, two subsequent accesses are assigned to the same session if they come from the same visitor and if the time interval between them does not exceed a certain threshold value. Therefore, pre-processor module creates the session list as the output, which is used by the evaluation function.

Web Crawler

The Web Crawler module processes the Web pages of the Web site in order to form the states. Hence, Web crawler extracts the web site's topology automatically and the states are formed accordingly.

Web Adapter

The Web Adapter module runs the proposed algorithm for the whole state space. It aims to maximize the *Covers* value of each state using the evaluation function, which utilizes the session list produced by the Pre-processor module. The Web Adapter presents all discovered adaptations as a list, where this output list contains suggestions of inserting new links for the web pages under consideration to the user. More specifically, adaptation list contains lists of suggestions where each suggestion contains three values, which can be represented as <*link_from, link_ to, Covers*>. Here, a shortcut is suggested from *link_from* page to *link_to* page and such a shortcut exists in *Covers* number of sessions. Sample output of the Web Adapter module is given in Figure 5.

2.4 IntWEB: The Algorithm

For the solution, Hill Climbing Search is adapted to the Web adaptation problem. Hill Climbing is a technique in AI that can be utilized to solve problems with many possible solutions, where all these solutions make up the whole search space. The pseudo code of the algorithm is provided in Figure 1. In general terms, in Hill Climbing Search, the search space contains different solutions that have different evaluation values. Hill Climbing starts with an initial state, or an initial solution which is generally a poor solution, and it iteratively makes transitions to possible states through applying actions. At each step, it chooses the next best state according to an evaluation function. When the current solution can no longer be improved, it terminates.

The algorithm contains two basic parts. The generation of possible next states is done by ***GENERATE_NEIGHBORS()*** function and the evaluation is performed by ***EVAL()*** function, as shown in Figure 1.

```
currentNode = startNode;
loop do
  L = GENERATE_NEIGHBORS(currentNode);
  nextCovers = -INF;
  nextNode = NULL;
  for all x in L
    covers =  EVAL(x);
    if (covers > nextCovers)
          nextNode = x;
          nextCovers = covers;
    else if nextCovers <= EVAL(currentNode)
   //Return current node since no better neighbors exists
        return currentNode;
  currentNode = nextNode;
```

Figure 1. Hill Climbing Search

GENERATE_NEIGHBORS() uses **add_shortcut** action described in Section 2.1. This action produces all possible next states by considering all possible links from one page to another page. Once a link is chosen to be added by using the evaluation function, call it a link from page **A** to **B**, then the next possible states will be generated such that, only the possible links from B to all other possible Web pages will be considered. Once no more improvements can be made, we will end up with a list of links that will traverse the Web site starting from page or request **A** to the last request found at the end. Then, the search will restart again by the same empty initial state, however this time considering only possible links that starts with a possible addition of a link such that, the link will not have **A** as the source. Here *source* refers to any link L_i that appears in the $from\_L_i$ field of that link.

```
Covers = 0;
for each L_k<from_ L_k, to_ L_k> in current state S
 for each session s in Session_List
   if((s contains path from  from_ L_k to to_ L_k)  and
     (length_of_path(from_ L_k to  to_L_k)  in s notexceeds
           path_length_threshold))
      Covers=Covers+length_of_path(from_L_k,to_L_k);
 return Covers;
```

Figure 2. The pseudocode for *EVAL (S)* where state S is $S = <L_1<from\_ L_1, to\_ L_1>, L_2<from\_ L_1, to\_ L_2>, L_k<from\_ L_k, to\_ L_k>, Covers>$ using threshold *path_length_threshold*

As it is mentioned in Section 2.1, the evaluation function, ***EVAL(),*** measures the *goodness of a state* according to the problem at hand. The pseudo code of ***EVAL()*** is presented in Figure 2. Here, the main aim is to make it easier for the users to traverse the Web site, therefore, if a path such as, access to page **C** occurs after pages **A** then **B**, exists in many sessions, then it will be beneficial for the users to add a direct link from **A** to **C**. Here, there exist two considerations. First of all, the degree of goodness of adding a shortcut depends on the length of the path that is shortened. With path, we mean, the number of pages that you have to traverse in order to get to the desired page. For instance, in the above scenario, path from **A** to **C** needs getting to **B** from **A** then **C** from **B,** which is of length 2**.** Therefore, if system proposes to add a direct link between such two pages, and if the actual path between them is long, then this is more valuable compared to shortening an already short path via adding a link. Accordingly, the EVAL() function increases the covers value as much as the length of the path that is being shortened, as shown in Figure 2.

However, one more thing that should be considered is that, if such a path is too long, then it is highly probable that these two pages, in our case pages **A** and **C**, are not really related. For instance, if the user accesses **C** after accessing many pages from **A**, then these pages may not really be related and adding a direct link between them may not be meaningful. In order to control this, a parameter is kept in the system, namely ***path_length_threshold.*** With this parameter, EVAL() examines only the paths whose length does not exceed *path_length_threshold* parameter.

As a result, ***EVAL()*** processes each session for a state, and considering the paths that exist in the state, it calculates and returns the **Covers** value of that state considering the above discussions.

Assume that we have three web pages, p1(5), p2(4), p3(3). The numbers in parenthesis show the number of sessions that these pages are included. Firstly, the function chooses p1 as the source, and generates all possible states that have p1 as *from* field and each pi as *to* field. EVAL() function calculates the coverage value of the state. Next, all of the states that have p2 as the source and all of the possible states except p1 as the destination state are generated. This continues until all pages are covered as source.

3 Experimental Evaluation

The described IntWEB framework is evaluated by using the log data taken from a live Web site. In the evaluation phase, **path_length_threshold** parameter is set to different values so as to determine the effect of it on the results. In addition, the resulting adaptation list under an optimal path_length_threshold parameter is presented and discussed. Before presenting the evaluation phase, following subsection gives details about the dataset, and the results of the data pre-processing step.

3.1 Dataset

The designed system is evaluated by using the log files taken from the Web site of Middle East Technical University Computer Engineer Department [1]. The web server log files include an entry for each access to the server from a user. Each entry in the log file includes information regarding the IP address, authentication name, date-time of the access, HTTP request, response status, size of the requested resource, referrer URL and browser identification. The log data used in the experimental evaluation spans one week during the semester, more specifically, within the exams period. However, when logs are examined for different periods, it is observed that the patterns do not change much during the semester.

The log file is processed in order to extract the sessions, where each session is a set of page references from a user during one logical period. One important thing that needs to be mentioned here is that, IntWEB is not designed for any specific domain. In other words, in this paper, the results are presented for an academic department's Web site; however, the framework can work on any other log data from any other domain.

The log data initially contains 214010 accesses. After removing the noisy and irrelevant data (multimedia and robot accesses, erroneous accesses, accesses to newsgroup, etc), we left out with 13979 accesses. From these accesses, 211 different sessions are identified. In Figure 3, the number of requests per each session group is presented in order to give a general idea about the dataset. Here, sessions are grouped according to the number of requests they contain. In the figure, x-axis corresponds to the number of sessions, and y-axis corresponds to the number of requests per those sessions. As it can be seen from this figure, the number of session groups that have requests more than 10 are less than 4, whereas the number of sessions that have requests less than 10 comprises about 78% of the dataset.

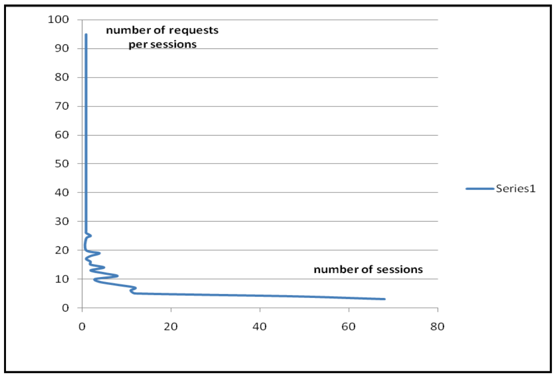

Figure 3. Number of Requests per Session

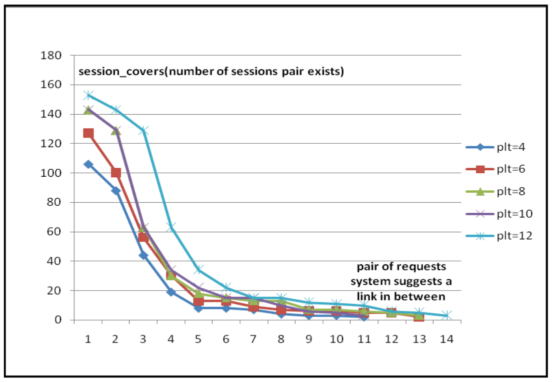

Figure 4. Covers value vs. path_length_threshold

[1] http://www.ceng.metu.edu.tr

From Page	To Page	Covers
/Course/?semester=2009&course=ceng242&cedit=0	/GradingTools/grades_histogram.php	136
/Course/?semester=2009&course=ceng140&cedit=0	/GradingTools/grades_histogram.php	127
/Student/homeworks.php	/GradingTools/grades_histogram.php	100
/Course/?semester=009&course=ceng334&cedit=0	/Student/homeworks.php?hid=1258	56
/GradingTools/grades_histogram.php	/Student/homeworks.php?hid=1165	30
/GradingTools/grades_histogram.php	/Course/?semester=2009&course=ceng280&cedit=0	13
/GradingTools/grades_histogram.php	/Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng280	13
/Student/homeworks.php?hid=1258	/Student/homeworks.php?hid=1195	9
/Student/homeworks.php?hid=1165	/Student/homeworks.php?hid=1201	7
/Student/homeworks.php?hid=1201	/Student/homeworks.php	6
/Student/homeworks.php	Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng382	6
/Course/?semester=2009&course=ceng280&cedit=0	/Course/?semester=2009&course=ceng232&cedit=0	5
/Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng280	Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng382	5
/Student/homeworks.php?hid=1195	Student/homework's.php?hid=1162	2

Figure 5. Sample Adaptation Results under path_length_threshold 6

2.2 Results

In order to evaluate the system's resulting adaptation list for different path_length_threshold parameter settings, the parameter is set to 4,6,8,10 and 12. Results can be seen in Figure 4. In this figure, x-axis corresponds to each pair of requests where the system suggests creating a link between, and y-axis represents the number of sessions that this pair exists, that is, the **session_covers** value. From the figure, we can say that, as the path_length_threshold parameter increases, the **session_covers** value for different pairs increases, since the possibility of pairs' existence in a session increase. Similarly, the length of the adaptation list also increases when path_length_threshold parameter increases, which is as expected, since, the number of possible link pairs increases as we permit longer sequences of requests.

In Figure 5, the resulting adaptation list under path_length_threshold 6 is presented. The resulting adaptation list proposes 14 suggestions, which can be summarized as follows:

1. Adding shortcuts from course web pages to grade web pages
2. Adding shortcuts from course web pages to the homeworks
3. Adding shortcuts from homework list to grades and grades to homework
4. Adding shortcuts between different homeworks (same semester, same class)
5.

Here, when the suggestions are examined, all of them appear to be logical and useful when the domain at hand is considered. For instance, from the resulting adaptation list we can conclude that, users enter web pages of homework one after another, therefore; shortcuts between homeworks of a specific years students (like 4th year students), and for a specific semester can facilitate users' navigation.

IntWEB presents the generated list, as shown in Figure 5, to Web site administrator as adaptation suggestions. S/he can then take into account the propositions that s/he thinks best suits the Web site, and then the accepted adaptation suggestions are applied. Afterwards, it is possible to check whether users get satisfied with the changes. This process can continue within the lifecycle of adaptation of the Web site. In order to test the user's satisfaction, Web site admin-istrator can again use the system. For instance, after s/he applies the proposed adaptations, s/he will again collect log files for a specific time period and check the resulting usage patterns in order to find out whether the added links are followed and facilitated user's navigation on the Web site.

From the results presented and discussed above, we can conclude that the system produces feasible adaptation suggestions. One important point to mention here is that, the Web site used in the evaluation does not contain many number of Web pages. Therefore, the log file does not include many different user access patterns. We believe that, with larger Web sites, such as e-commerce sites including information on various types of products, the log file would be richer and the resulting adaptation list could contain more number of suggestions.

4 Conclusion and Future Work

In this paper, adaptive Web problem is formulated as a Hill climbing algorithm through defining all the necessary components of a search problem and presenting a novel algorithm for the adaptation of Web pages from server access log files. The experimental results show that the proposed approach discovers useful adaptation suggestions.

The basic idea is to promote shortcuts that are proposed to shorten long paths, however, these paths should also be related. If the path is too long, then probably the user did not aim to traverse the whole path. This is controlled by cutoff threshold. In addition to adaptation through adding shortcuts, deleting unnecessary shortcuts is also supported. The search starts as if there is no connected Website, but only the Web pages. Then the generated results propose adding shortcuts. Assume that proposed shortcut set is P, and the set of current shortcuts existing in website is W. Then the Web administrator can consider to remove the shortcuts in the set W-P or to include shortcuts in P.

Concerning the notion of adaptive Web systems, in addition to statistical evaluation, human evaluation of the results is also needed. Therefore, as the future work, it is planned to present the adaptation list to the Web site administrator and let him/her evaluate the results. In addition, some sort of questionnaire-based evaluations will be very useful to gather the user's ratings and comments towards the produced adaptations. In addition, it is ex-pected that IntWEB will perform better if a richer Web site with various types of accesses is used in the evaluation. Therefore, as a future work, the system is planned to be tested and evaluated in different environments.

References

[Perkowitz and Etzioni, 1997] M. Perkowitz and O. Etzioni. Adaptive Web sites: an AI challenge. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1997.

[Perkowitz and Etzioni, 1998] Mike Perkowitz and Oren Etzioni. Adaptive Web Sites: Automatically Synthesizing Web Pages. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd ed.)*. Prentice Hall, pages 111–114, ISBN 0-13-790395-2, Upper Saddle River, New Jersey, 2003.

[Perkowitz and Etzioni, 2000] Mike Perkowitz, and Oren Etzioni. Towards adaptive Web sites: Conceptual framework and case study. Artificial Intelligence 118(1-2): 245-275, 2000.

[Joachims, Freitag, and Mitchell, 1997] Thorsten Joachims, Dayne Freitag and Tom. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 770-775, Nagoya, Japan, 1997.

[Wexelblat, and Maes, 1999] Alan Daniel Wexelblat, and Pattie MaesFootprints. History-Rich Tools for Information Foraging. In *Proc. of Proceedings of Human Factors in Computing Systems (CHI)*, pages 270-277, Pittsburgh, Pennsylvania, United States, 1999.

[Fink, Kobsa, and Nill, 1996] Josef Fink, Alfred Kobsa and Andreas Nill. A. User-Oriented Adaptivity and Adaptability in the AVANTI project. In *Proc. of Conference "Designing for the Web: Empirical Studies"*, Microsoft, Redmond, WA, 1996.

[Thorpe, 1990] Charles E. Thorpe. *Vision and Navigation: The Carnegie Mellon Navlab*. Kluwer Academic, Boston, MA,1990.

[Eirinaki and Vazirgiannis, 2003] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization, *ACM Transactions on Internet Technology (TOIT),* v.3 n.1, pages 1-27, February 2003.

[Srivastava, Cooley, Deshpande, and Tan, 2000] Jaideep Srivastava , Robert Cooley , Mukund Deshpande , and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from Web data, *ACM SIGKDD Explorations Newsletter*, v.1 n.2, January 2000.

[Cotter and Smyth, 2000] Paul Cotter and Barry Smyth. WAP-ing the Web: Content personalization for WAP-enabled devices. In: Brusilovsky, P., Stock, O., Strapparava, C. (eds.) Proc. of Adaptive Hypermedia and Adaptive Web-based systens. Lecture Notes in Computer Science, Vol. 1892. 98-108. Springer-Verlag, 2000.

[Mulvenna, Anand and Buchner, 2000] Maurice D. Mulvenna, Sarabjot S. Anand, Alex G. Büchner. Personalization on the Net using Web Mining. Communications of the ACM, Vol. 43, No. 8:23-125, 2000.

[Mobasher 2007] Bamshad Mobasher. Data Mining for Web Personalization. In The Adaptive Web: Methods and Strategies of Web Personalization, Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). Lecture Notes in Computer Science, Vol. 4321, PP. 90-135, Springer, Berlin-Heidelberg, 2007.

[Goldfarb 2002] Avi Goldfarb. Analyzing Website Choice Using Clickstream Data, 209-230, Elsevier Science Ltd., 2002.

[Sismeiro and Bucklin, 2004] Catarina Sismeir and Randolph E. Bucklin. Modeling Purchase Behavior at an E-Commerce Web Site: A Task-Completion Approach. Journal of Marketing Research. XLI.306-323, 2004.

[Mobasher, Cooley, and Srivastava, 2000] Bamshad Mobasher,Robert Cooley, and Jaideep Srivastava. Automatic Personalization based on web usage Mining. Communications of the ACM, Vol. 43, No.8, pp. 142-151, 2000.

[Mobasher, Dai, Kuo, and Nakagawa 2001] Bamshad Mobasher, Honghua Dai, Tao Kuo, and Miki Nakagawa. Effective Personalization Based on Association Rule Discovery from Web Usage Data. In Proceedings of the 3rd ACM Workshop on Web Information and Data Management (WIDM01), in conjunction with the International Conference on Information and Knowledge Management (CIKM 2001). Atlanta, Georgia, 2001.

[Nasraoui and Petenes 2003] Olfa Nasraoui and Christopher Petenes. Combining Web Usage Mining and Fuzzy Inference for Website Personalization. In Proc. of WebKDD 2003, KDD Workshop on Web mining as a Premise to Effective and Intelligent Web Applications. Washington DC. p. 37, 2003.