Engineering Applications of Artificial Intelligence 25 (2012) 889–897

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Predicting web user behavior using learning-based ant colony optimization

Pablo Loyola*, Pablo E. Román, Juan D. Velásquez

Department of Industrial Engineering, Universidad de Chile, República 701, P.O. Box 8370439, Santiago, Chile

A R T I C L E I N F O

Article history:
Received 15 July 2011
Received in revised form
29 September 2011
Accepted 19 October 2011
Available online 6 December 2011

Keywords:
Ant colony optimization
Web usage mining
Multi-agent simulation
Text preferences

A B S T R A C T

An ant colony optimization-based algorithm to predict web usage patterns is presented. Our methodology incorporates multiple data sources, such as web content and structure, as well as web usage. The model is based on a continuous learning strategy based on previous usage in which artificial ants try to fit their sessions with real usage through the modification of a text preference vector. Subsequently, trained ants are released onto a new web graph and the new artificial sessions are compared with real sessions, previously captured via web log processing. The main results of this work are related to an effective prediction of the aggregated patterns of real usage, reaching approximately 80%. In the second place, this approach allows the obtaining of a quantitative representation of the keywords that influence the navigational sessions.

1. Introduction

Since the beginning of the World Wide Web, one of the aspects that has caught the attention of the emerging researchers was the way in which the users interact with the structure and content of web sites. For that purpose, multiple analyses and models were generated to understand web user behavior in order to display relevant content and maximize traffic. The emergence of e-commerce represented a major change in terms of the valuation of the web user, not only as a consumer of content, but also as a client that has to be seduced into making a purchase. This, together with the rise of the Web 2.0 paradigm, promoted the unification of knowledge and techniques in what is now commonly known as Web usage mining (WUM), a field specializing in the study of web user behavior. One of its main objectives is to achieve web user *personalization*, which means the capability of generating adaptability within the web site, both through link structure transformation and real time suggestions, in relation to the user's preferences and generated paths.

In general, WUM uses traditional behavioral models, operations research and data mining methods to deal with web usage data. However, some modifications are necessary according to their respective application domain. Two families of techniques have been used to analyze the sequential patterns: deterministic and stochastic. Each one has been used depending on the approaches that have been adopted.

Soft computing methodologies have gained a considerable amount of relevance in relation with WUM, due to their flexible implementation and results in the field of recommendation-based systems and adaptive web sites (Lin and Tseng, 2010). Within these fields, special attention has been concentrated on bio-inspired metaheuristics, which are commonly ruled by the concept of *swarm intelligence*, the ability of a group of agents to perform complex tasks through a collaborative process. Instead of trying to mimic human intelligence, the inspiration is taken from the observation of social insects such as ants or bees (Christensen et al., 2007).

Ant colony optimization (ACO) is one of the tools used for these purposes. This metaheuristic is inspired by the way ants optimize their trails for food foraging based on releasing chemical substances into the environment called *pheromones*. This simple idea is applied to the web user trails of visited web pages, also called sessions (Liu, 2007). Artificial ants are trained through a web session clustering method modifying an intrinsic text preference vector which represents the importance given by the users to the set of most important keywords. Furthermore, trained ants are used to predict future browsing behavior.

This paper is organized as follows. Section 2 provides an overview of related work. In Section 3 the proposed model is presented. Then, in Section 4 an application of our work on a real web site is described. Finally, conclusions and future work are presented in Section 5.

2. Related work

ACO is inspired by the behavior of some insect species which is related to the presence of a social component in their

* Corresponding author. Tel./fax: +56 2 978 4834.
E-mail addresses: ployolah@dii.uchile.cl (P. Loyola), proman@ing.uchile.cl (P.E. Román), jvelasqu@dii.uchile.cl (J.D. Velásquez).

0952-1976/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2011.10.008

organizational structure, allowing them to perform complex tasks through a cooperative and coordinated process. The key component behind this is their capability of generating an indirect communication by modifying the environment. This mechanism is called *stigmergy*. Specifically in ants, it is based on the use of chemical *pheromones* which can be deposited on the ground and detected by the colony in order to execute labors such as food foraging, cooperative transport and corpse grouping. The methodology is based on the progressive construction of pheromone trails from the nest to the food source, searching for the minimum path. This is achieved through an auto-catalytic process in which pheromone levels are reinforced in inverse proportion to the time needed to walk through its respective path. As ants choose paths with higher pheromone levels, suboptimal solutions evaporate and the algorithm returns the shortest path. Biological studies have shown that colony-level behavior can be explained using models based on stigmergic communication, in terms of simulating the levels of self-organization (Dorigo and Stutzle, 2004).

Marco Dorigo proposed the first ACO model to solve the TSP[1] in 1992 (Dorigo and Gambardella, 1997). This work stated the principles of the methodology in terms of the use of four components: graph representation, problem restrictions, pheromone trails and solution construction. Most of the calculation is produced by using two variables that manage trails generation. The first is a support measure η_{ij}, also called heuristic information, which contains problem information that is not directly accessible to the ants, but has been introduced externally. This measure can store either costs or utility associated with a decision. The second, a pheromone function, $\tau_{ij(t)}$ is defined as

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}, \qquad (1)$$

where $\Delta\tau_{ij} = Q\eta_{ij}$, with Q a constant and ρ an attenuation factor, which is associated with the pheromone evaporation. Finally, the trail preference $p_{ij}(t)$

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{k=1}^{n} [\tau_{ik}(t)]^{\alpha} * [\eta_{ik}]^{\beta}}, \qquad (2)$$

where α and β are the weights for pheromones and support, respectively.

This allows the development of the metaheuristic and its further applications in fields from vehicle routing problems to image processing and machine learning (Dorigo and Stutzle, 2004).

The main advantages of ACO are related with its inherent parallelism (which can be useful when working together with an effective multi-threading computing strategy), its flexible implementation (which allows its use in dynamic problems), and the positive feedback (which accounts for rapid discovery of good solutions). As drawbacks of these techniques, it can be mentioned that although convergence was mathematically demonstrated (Dorigo and Stutzle, 2004), convergence time is difficult to predict and a deficient parameter setting can lead to a local optimal solution instead of a global one (Umarani and Selvi, 2010).

The implementation of ACO in web intelligence tasks, together with other bio-inspired heuristics, has gained a considerable amount of attention due to the necessity of finding new ways to explain and simulate web user behavior.

Ling (Lin and Tseng, 2010) proposes an ACO-based method for discovering navigational patterns, by using a support measure based on real transitions between pages. The resulting user navigational preferences, represented by pheromone levels, are directly proportional to the real frequency of usage. Clustering is another area in which ACO has been used widely (Bharne et al., 2011). Abraham and Ramos (2004) propose an approach based on the clustering of web user sessions. The model takes inspiration from the ability of certain kinds of ants to group corpses and generate *ant cemeteries*.

White et al. (2010) use an ACO implementation to progressively find the best correlation between web content and a set of online advertisements in order to maximize the probability of profit. To achieve this, a vector space model is used together with the notion of web user preference vector.

[1] Traveling Salesman Problem.

3. A methodology for simulating web user behavior using ACO

The key aspect of the proposed model is to enable artificial ants to learn from real user behavior in terms of the settings of the intrinsic parameters of an ACO implementation, which have to be adapted to a web environment.

3.1. Web user sessions clustering

The learning algorithms to be used in this paper are based on a continuous comparison between the artificial sessions generated by the ants and the web user behavior represented through real web user sessions. A first approach could be to compare each artificial session with the complete set of real sessions available. This method could take a considerable amount of time due to the large number of sessions generated in the study time intervals. Thus, it is proposed to perform a clustering process of the web user sessions in order to reduce the number of subsequent comparisons.

3.1.1. Similarity measure

To generate a valid similarity measure two main subjects must be considered. In the first place there are the sets of web pages belonging to each session, and in the second place there is the order in which those web pages were visited. Most approaches to this problem have used a similarity measure based on the number of operations needed to transform one session into another, for instance, using the Levenshtein distance (Velásquez and Palade, 2008) or the sequence alignment method (Hay et al., 2004). But those methods do not incorporate the notion of sequential path between the elements, which is fundamental in terms of the existence of a link element which makes the user trail possible. We propose to use a degree of similarity (Spiliopoulou et al., 2003) between sessions based on the calculation of longest common subsequence (LCS) (Gusfield, 1999).

Given r a real session and c an artificial session, the similarity measure is defined by

$$sim(r,c) = \frac{LCS(r,c)}{\max\{\|r\|,\|c\|\}}, \qquad (3)$$

where $\|r\|$ represents the length of a given session r and $LCS(r,s)$ is the longest common subsequence between sessions. This method allows to compare two sessions regardless if they have different lengths.

3.1.2. Clustering method

Web user sessions are basically a categorical representation of the paths followed by users and do not have an intrinsic value. Rather, it is their components, web pages, which give them an inner characteristic. Thus, the notion of a mean value for representing them does not exist (Liu, 2007; Murtagh, 1983; Zhao and Karypis, 2002). We propose to use a hierarchical method for clustering consisting of an agglomerative process that at every step group sessions according to the similarity measure. This method uses only the similarity (or distance) between elements which fit into the categorical nature of web user sessions (Liu, 2007). It also allows visualizing at every step the partial groups being generated. This is very useful in relation with the task of deciding the optimal number of clusters (Salvador and Chan, 2004). Additionally, hierarchical clustering allows the use of several *linkage criteria* to determine the optimal number of clusters as a function of the distance (or similarity) between the observations. Examples of this criteria are the *Simple Linkage method*, in which the distance between clusters is calculated as the minimum distance between elements from both clusters, the *Complete Linkage method*, in which the distance between clusters corresponds to the maximum distance between elements (Blashfield, 1976).

3.1.3. Characteristic value calculation

It is necessary to define a representative session for each cluster, which can be labeled as a characteristic session or *characteristic value*. As we are working on a discrete space of sub-sequences using a hierarchical approach, the common notion of *centroid* present in other methods such as *K*-means, is not available. We propose, for a given cluster, to calculate in each session the accumulative value of the similarity measure and to choose the session which has the maximum. This session will represent the cluster and will be labeled as the characteristic value.

3.2. Web graph modeling

A web site is represented by a directed graph in which the web pages are the nodes and links the edges. As the process of web user session recollection implies constant tracking during a certain period of time, changes may occur in both structure and content within the web graph. For this reason it is necessary to register those modifications. This results in the generation of many graphs, which contain the dynamics of the web site during the time interval under study.

3.2.1. Nodes

In the first place it is necessary to extract the most important keywords present in the corresponding web page. As the TF-IDF has shown good results in terms of representing the relevance of keywords within a corpus (Velásquez and Palade, 2008; Velásquez and Jain, 2010; Hiemstra, 2000), we propose a method based on that statistical measure consisting of the following steps:

Given a web graph G,

- All keywords present must be extracted and stored in a container, the *KeywordsGraph*.
- For each keyword *k* in the *KeywordsGraph*, its TF-IDF value is calculated in relation to every page *p* in the web graph. If *k* is not present on a determined page, its TF-IDF value is 0.
- For each keyword *k*, the maximum value of the TF-IDF is selected. Formally, for a given keyword *k*, $tfidf_k = \max\{tfidf_{k1}, tfidf_{k2}, \ldots, tfidf_{kN}\}$.
- $tfidf_{min}$ and $tfidf_{max}$ limits are defined in relation to the values obtained with the set of keywords and must be set in order to avoid extreme elements.
- All keywords whose TF-IDF representative value is between $tfidf_{min}$ and $tfidf_{max}$ are stored in a *MIK*, a container for the *Most Important Keywords*. Formally, $MIK = \{k_i \in KeywordsGraph \backslash tfidf_{min} \le tfidf_i \le tfidf_{max}\}$.
- Finally this set can be converted into a vector the components of which must be ordered decreasingly with respect to the TF-IDF values.

In the second place, we must generate a vector for each page *p* which represents in a quantitative fashion the correspondence between its own keywords and the set *MIK*.

For each page *p* in the web graph, all keywords must be extracted and stored in a set called the $KeywordsPage_p$. Then a L_p vector is generated, the length of which has to be exactly the same as the *MIK* and the value in a determined index must be associated with the keyword present in that index in the vector *MIK*. Values in L_p will be set according to the following rule:

$$L_p i = \begin{cases} TF\text{-}IDF_{ip} & \text{if } MIK_i \in KeywordsPage_p, \\ 0 & \text{otherwise.} \end{cases}$$

Each keyword *i* in the *MIK* must be searched within the $KeywordsPage_p$. If the keyword *i* is found, the value that will be set in the index *i* of vector L_p will be equal to the TF-IDF value of the keyword related to the page *p*. If not, the value will be 0. Thus, vector L_p will have non-zero values proportional to the fact that the keywords in page *p* are contained in the group of the most important keywords. The idea behind this method is the construction of a same-length vector for all pages in the web graph. This standardization will be necessary in the subsequent comparison algorithms.

3.2.2. Links

Following the common representation, edges will be used to simulate the links in the web graph. Initial pheromone trails must be placed in terms of the probability of transition of a random surfer (Brin and Page, 1998) defined by

$$p_{ij} = \frac{p * g_{ij}}{c_j} + \frac{1-p}{n}, \tag{4}$$

where *n* is the total number of nodes, *p* is an arbitrary value for the probability of continuing surfing within the web site (following the 80–20 rule). The adjacency matrix is represented by g_{ij}. This expression takes the value of 1 if a link between nodes *i* and *j* exists and 0 if otherwise. c_j corresponds to the number of links present in node *i* (*node degree*).

3.3. Artificial ants

In order to adapt web user behavior to ACO it is necessary to model each ant with elements that allow the management of data extracted from a web site.

3.3.1. Text preference vector μ

For each ant, a text preference vector μ is defined as a measure for the relative importance that a particular agent gives to a set of keywords (Román, 2011; Román and Velásquez, 2010). Each component of every vector is initially an artificial representation of the relevance given to the determined component of the *MIK* vector. For example, in the μ_i text preference vector (belonging to ant *i*) the value of the component *j*, μ_{ij}, will be a quantitative expression for the importance the ant *i* gives to the keyword present in the *j* index in the *MIK* vector.

Possible values are between 0, which means no interest in the keyword, and 1, which represents full interest.

3.3.2. Ant utility

The main goal for every ant is to travel through the graph and collect information in order to satisfy its need for content. Every time an ant *i* arrives at a specific page *p*, a comparison between the μ_i text preference vector and the corresponding L_p vector is performed. As a result of this comparison, a utility is computed proportional to the level of text similarity between both vectors. As the ant passes from node to node, it accumulates the utility until a predefined limit is reached. In that instant the ant finishes the session.

3.3.3. Ant initialization

Each ant is assigned a μ vector, which will define its initial text preference. Additionally, a web user session cluster is associated with the ant. This is the core idea behind the learning algorithm that each ant will modify its preference vector in order to model the sessions belonging to the cluster it was associated with.

Initially, all μ vectors are constructed using a weighted random generation based on the following distribution:

- *High preference*: 20% of the vector components. Values between 0.8 and 1.
- *Medium preference*: 60% of the vector components. Values between 0.2 and 0.8.
- *Low preference*: 20% of the vector components. Values between 0.0 and 0.2.

This distribution was conceived based on the supposed fact that for a given set of keywords, only a small group will concentrate a high value of preference.

3.3.4. First node election

While in a standard implementation of ACO the starting and ending points are well defined (for example, in a vehicle routing problem), in a web-usage approach ants can start and end their sessions in any node.

As each ant is associated with a cluster of web user sessions, we propose to generate a set of possible first-visited pages for that ant, based on the pages that appear in first place within the constituent sessions of the associated cluster. The selection of the first page for the artificial ants is performed by using the frequencies of the first pages present in the corresponding cluster. These values are used as a probability in a Monte Carlo approach.

3.4. Learning algorithm

The main idea behind the ant learning process is the progressive modification of their respective μ text preference vector in order to generate artificial sessions which have an adequate similarity to the ones present in their corresponding session cluster.

3.4.1. μ text preference vector modification for improving generated sessions

For a given ant, before beginning to travel through the web graph, a fixed increment in one of the components of its μ vector is performed. This can be understood as a guided increase in the importance that the ant gives to a determined keyword. Each of these changes will influence the behavior of the ants and the sessions they generate. Thus, it is necessary to find the modification which results in sessions most similar to the real sessions in the respective cluster.

In the first place, copies of the μ vectors must be generated. Then, for each original μ vector, its copy is constantly modified. In each modification, the ant will generate a determined session. Finally, the original μ vector is modified according to the index in which its copy was modified and that generated the most similar session in relation with the associated cluster.

3.4.2. Ant arrival to the web graph

Once an ant has been set with a modified copy of its original μ vector, it is placed on the web graph according to the method described in Section 3.3.4. When arriving at the first node, the ant must compare its μ copy with the corresponding L text vector associated with the node. This step aims to simulate what happens when a user arrives at a web site, he/she evaluates what the first page offers in terms of content, comparing it with his/her own preferences. Then, he/she decides whether or not to continue through the link structure.

We propose to use the cosine similarity measure to compare user preference vector μ and the corresponding text vector L_p from a given page p (Román and Velásquez, 2010). The resulting value will be taken as the utility obtained by the user for visiting page p

$$utility = \frac{\langle \mu \bullet L_p \rangle}{\|\mu\| \|L_p\|}. \tag{5}$$

This parameter will accumulate as the ant travels through the graph until a limit is reached.

3.4.3. Method for selecting next page

After the calculation of the utility given on the first page, the ant must decide which page to follow. Each of the links to the neighbors contains a pheromone level which was initially calculated as a transition probability. This probability will be used in the decision process as is pointed out in Algorithm 3.1.

Algorithm 3.1. Next node selection.

```
    Data Actual node $i$
1   Initialize NeighbourNodes=GetNeighbourNodes($i$);
2   Initialize NeighbourPheromoneLevels
      =GetNeighbourPheromoneLevels($i$);
3   Initialize sum =0;
4   Initialize r =GenerateRandom(0,1);
5   foreach node j $\in$ NeighbourNodes do
6       $\tau_{ij}$ = NeighbourPheromoneLevels($i$,$j$);
7       sum = sum + $\tau_{ij}$;
8       if sum $\geq$ r then
9           return NeighbourNodes($j$);
10  return −1;
```

3.4.4. Pheromone levels update

The process of pheromone levels update is divided into two steps. The first is related to the modification in the link which was chosen by the ant. In the second place, a general decrease of all levels must be performed. This process is called *evaporation* (Dorigo and Gambardella, 1997).

Given an actual node i, the process of selection of the next node to follow results in the choice of the neighboring node j. Thus, the pheromone levels in the link τ_{ij} will be updated. The increment that will be added is directly proportional to the probability value given by the Logit model for discrete choices. This model has been widely used to simulate the way in which users search for and decide in relation to a discrete set of possibilities (Román, 2011; Román and Velásquez, 2010)

$$\tau_{ij} = \tau_{ij} + \frac{e^{V_{ij}}}{\sum_{k \in Neightbours_i} e^{V_{ik}}} * \varepsilon, \tag{6}$$

where V_{ij} is the utility obtained by the ant when passing from node i to j. ε is an arbitrary value ($\varepsilon > 1$) to help tweak the magnitudes obtained.

This modification will result in the sum of all levels of pheromone contained in the links that come out of node i being greater than 1. Therefore a normalization process must be performed.

Additionally, with the purpose of generating a stronger fit between the newly generated sessions and the set of sessions from the respective cluster, we propose to modify expression (6) with a boost component. If the transition between nodes i and j exists within the sessions that belong to the respective cluster, the modification of τ_{ij} will be given by

$$\tau_{ij} = \tau_{ij} + \frac{e^{V_{ij}}}{\sum_{k \in Neightbours_i} e^{V_{ik}}} * \varepsilon + boost. \tag{7}$$

The next step is to evaporate the pheromone level in all links in the web graph. This process is normal in every ACO implementation and has as a main purpose to progressively eliminate pheromone levels from unused links.

3.4.5. *End conditions*

Ants travel through the web graph until one of the following conditions is reached:

- The utility limit is reached: Each ant is set with a limit for the amount of utility to obtain by traveling through the web graph. If the limit is reached, it can be understood that the need for information has been filled.
- The ant has fallen into a loop: If the ant begins to move in a circular way, it is perceived as a non-genuine web user behavior and the session is stopped. To identify a feasible loop, each time a node is added to the current session, its previous additions are inspected and if the complete sequences are repeated twice, the session is stopped.
- Persistent visits to a determined page: If the ant visits, within a given session, a unique page during more than four consecutive times, the session is stopped.

3.4.6. *Generated sessions storage*

When one of the previous conditions is met, the session is stopped and its components are associated to a unique ID. Finally the session is stored together with the index in which the copy of the μ vector was modified.

3.4.7. *Modification of μ vector*

As was mentioned before, for every modification made to the copy of the μ vector, the ant generates an artificial session. Each modification consists of an increment of one of the components of the vector which represents the user preference for an associated keyword that belongs to the set of most important keywords in the web graph. The next step is to choose one of the modifications made in the copy and reproduce it in the original vector. The method for finding it is to compare each artificial session with the central session of the respective cluster. The modification which produced the session which is most similar to the central session will be selected. Finally, the original μ vector is modified.

3.5. *Model validation*

The main idea behind the model validation process is to collect structure, content and usage from a next interval of time. Thus, both training data and validation data must belong to the same web site, but from disjoined periods. Then, trained ant sessions are generated and compared with real web user behavior.

3.5.1. *Data preparation*

Both content and structure data must be collected and processed in the same way used with the training data. Usage data must be represented through web sessions and the hierarchical clustering process must be performed. Thus, both training and validation sources will have an equivalent structure which will allow the implementation of coherent comparisons.

3.5.2. *Ant initialization*

The main output from the learning process is a set of trained text preference vectors, each one taken as a representation of the real preferences from a determined web session cluster that belongs to the training set. For each of these vectors, a group of ants will be generated. Every member of the group will incorporate the respective vector as its own preference. Additionally, each group of ants will have its own web graph.

3.5.3. *Ant behavior*

For each μ vector, its entire group of ants is released on its respective web graph at the same time. As all ants have the same preference, the same information will be searched. This represents a standard implementation of ACO, where all agents cooperate indirectly in the search for food, updating pheromone levels at the edges of the graph. Thus, this algorithm is similar to the one presented in the training process, but without the learning component.

For a given ant that belongs to the group H_j (associated with text preference vector μ_j) a schematic representation of the algorithm will be:

Algorithm 3.2. Ant behavior.

```
1  Initialize $SessionsContainer(), utility = 0, MAXIterations$;
2  for $i \leftarrow 0$ to $MAXIterations$ do
3      $session()$;
4      $CurrentNode \leftarrow getCurrentNode()$;
5      $session.add(CurrentNode)$;
6      $L \leftarrow getVectorL(CurrentNode)$;
7      $utility \leftarrow utility + CosineSim(\mu, L)$;
8      while utility $< maxAND!InvalidLoopsAND!InvalidRep$ do
9          $Neighbours \leftarrow getNeighbours(CurrentNode)$;
10         $DestinationNode \leftarrow getDestinationNode(Neighbours)$;
11         $session.add(DestinationNode)$;
12         $L \leftarrow getVectorL(DestinationNode)$;
13         $utility \leftarrow utility + CosineSim(\mu, L)$;
14         $\tau_{CurrentNode \to DestinationNode} \leftarrow \tau_{CurrentNode \to DestinationNode} + LogitProb(CurrentNode, DestinationNode)$;
15         $evaporateGraph()$;
16         $CurrentNode \leftarrow DestinationNode$;
17     $SessionsContainer.add(session, i)$;
18     $session.clear()$;
```

Algorithm 3.2 allows each group of ants to fill its respective container with artificial sessions. Each ant travels until an ending condition is reached. The generated session is collected and the ant starts again. Progressively, the changes made in the pheromone levels will shape certain routes within the graph that will be gradually reinforced by the respective group.

3.5.4. *Generated sessions characterization*

For each container of generated sessions, a convergence analysis must be performed. As the process of session storage is sequential, heterogeneity is expected at the beginning because the levels of pheromone have not been modified in order to influence the ants to follow a defined pattern. Homogeneity is expected near the end, when the collaborative work has reinforced some trails. Therefore, it is necessary to find a convergence threshold which allows the selection of the subset of generated sessions that can be understood as the representative behavior of a respective group of ants associated with a certain trained μ preference vector. Two methods are proposed:

- Select a proportion of sessions from the last inserted in the container. For instance, 10% of the last sessions can be extracted and labeled as homogeneous behavior. This is under the assumption of convergence inherent in the ACO algorithm.
- Inspect the consecutive similarity between stored sessions. Using Eq. (3) it is possible to calculate the similarity between sessions. If a tendency toward high values of session similarity is perceived, it can be assumed that convergence has been reached.

Both methods require expert assistance because the parameters will depend on problem intrinsic nature.

For every group of ants, after their respective subset of artificial sessions has been selected, it is necessary to choose a characteristic session as a representative value, which can be performed using the method described in Section 3.1.3. There will therefore be one session for each group of ants.

3.5.5. *Comparison between sessions*

The final step in the validation process is the comparison between the following sets of sessions:

- Real web sessions corresponding to the web usage in the validation period. These sessions have been grouped into C_n clusters. Each cluster has a characteristic value c_i $(i = 1 \ldots n)$.
- Artificial sessions generated through a standard ACO algorithm using trained text preference vectors. These sessions are organized in S_m sets, each one from a different group of trained ants. Each set has a representative session s_j $(j = 1 \ldots m)$.

It must be noted that probably $n \neq m$. In fact, m is associated with the number of μ text preference vectors, which, in turn, comes from the clusters of user sessions from the training period. Thus, both n and m correspond to the number of user-session clusters in different time periods, therefore they do not necessarily have to be the same.

Each S_j must be compared with every C_i using s_j and c_j and the similarity measure defined in Section 3.1.1. A similarity threshold must be defined, which if reached, represents that the artificial session is sufficiently similar to the real session to explain it.

4. Practical application

The presented model was implemented on the web site of the Industrial Engineering Department of the University of Chile.[2] This site represents the main source of information about the institution and the activities related to it. In terms of content, it includes news, academic events, undergraduate and postgraduate information, faculty and administrative staff. It also serves as a transition platform for several sub-sites belonging to postgraduate programs and research centers. Both training and validation periods were set to 1 month, thus data was extracted from September to October in 2010.

4.1. *Structure and content processing*

To collect web site data, a web crawler approach was used. It allowed simultaneous storage of content and structure and set the relationship between them in terms of the corresponding identifiers. Content data were pre-processed based on the application of standard methodologies for cleaning, such as non-html element removal, extraction of stop-words and a stemming process. This step has enormous relevance because it will generate the input for the learning process. Therefore it is necessary to ensure the quality of the results although it may represent a huge computational cost. The experiments performed for this work showed that the whole process of cleaning and filter of the website took an average of 25 min. Although a high level of automation was achieved through the use of regular expressions to identify useless elements, some manual cleaning was necessary. Crawler capabilities enabled the identification of changes through time which were explicitly registered in order to obtain a more realistic source. Finally, collected data were transformed into a vector space model representation.

TF-IDF values from the training data set were calculated and normalized. Following the method described in Section 3.2.1, the set of most important keywords was generated using the keywords whose TF-IDF values were between 0.6 and 0.9, consisting of 165 elements. Using this set, L vectors were calculated for each page within the web graph.

4.2. *Web user session extraction*

The extraction and processing of the user behavior was performed using a proactive strategy consisting of the intensive use of a javascript file which had been inserted previously into all web pages that belong to the web site under study. This javascript file is the key element for tracking and registering visited web pages. This method is based on the utilization of a cookie as a temporary storage object for the navigational data with the purpose of completely identifying web user sessions, which are later sent to a dedicated server for final storage. When a user visits a page, the method collects the IP address, user agent, time stamp, URL, host, and a sequence indicator (entering or leaving the page). Additionally, this script allows the assignment of a unique identifier to the current session.

One of the advantages of this method is the automatic generation of the sessionization process, which represents a considerable savings of time due to the tasks involved. On the other hand, it has some issues related to its dependency on the *onload* and *onbeforeunload* javascript functions, which are used to identify when a user enters or leaves a page. Those functions are handled in different ways depending on which web browser is being used.

4.2.1. *Web user session clustering*

The first step was to remove all sessions with length equal to one because of the lack of the notion of sequential path. Before performing the clustering algorithm it is necessary to define a criterion for selecting the optimal number of clusters. We propose to use the ratio between extra similarity and intra similarity (Ganguly et al., 2009), selecting the number of clusters that minimize it. Additionally, after preliminary tests, the *Complete Linkage* criteria was chosen for determining the clustering generation. The main reason was that this criteria did not produce excessive cluster chaining as *Simple Linkage* and is less computationally expensive than *UPGMA* or *WPGMA* giving similar results.

For the training data set, consisting of 6842 sessions, 806 clusters were found. For the validation data set, consisting of 5701 sessions, 733 clusters were found. Although the number may seem high, it must be noted that the main purpose for using clustering was to reduce the dimensionality of data due to the considerable number of comparisons that must be done. Thus, due to the high exigency of the similarity measure in use, which not only compares between the elements within sessions, but also the sequence in which they appear, it is expected that the algorithm would behave reticently in terms of merging clusters. Finally, using the method described in Section 3.1.3, for each cluster a characteristic session is obtained.

[2] http://www.dii.uchile.cl

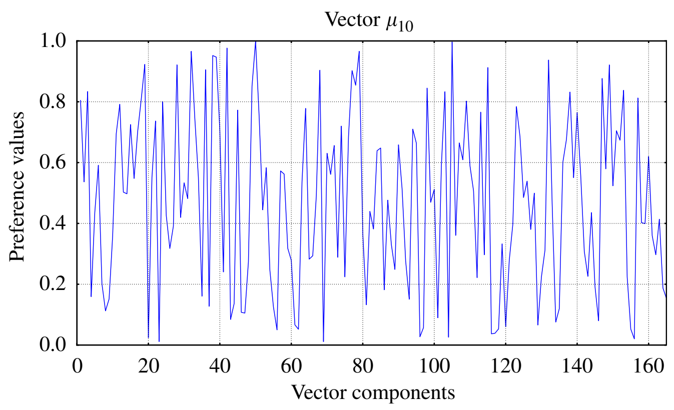

Fig. 1. Initial text preference values for vector μ_{10}.

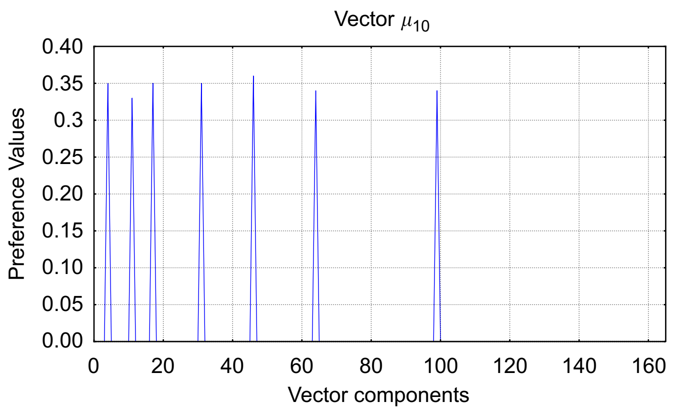

Fig. 2. Text-preference component values for vector μ_{10} are influenced by only seven terms of 165.

4.3. Training algorithm

For each cluster found in the training set, an artificial ant was associated and initialized with an artificially generated μ text preference vector. Several tests were performed in order to obtain model parameters:

- ε: 2.2,
- evaporation factor: 0.8,
- ant utility limit: 0.2,
- boost: 0.2.

Each of the 806 artificial ants was set to iterate 1000 times in their respective web graphs. This process was configured to be executed in parallel using the MASON[3] framework for Java, which is designed to manage multi-agent simulations (Luke et al., 2005). It was executed using an Intel Core 2 Duo 2.1 GHz computer, using 3 GB RAM and Windows 7 Professional Edition. The entire learning process took approximately 4 h.

The output of the learning algorithm is the set of trained μ text preference vectors. Results showed that the initial vectors were significantly modified. It must be noted that initial μ vectors are generated randomly using a rule described in Section 3.3.1. The algorithm gradually modifies the components and most of them decrease considerably while a few increase their values. For instance, Fig. 1 represents the initial and highly heterogeneous values for a given vector μ, in this case the one associated with cluster 10. Fig. 2 shows the final distribution of values after 1000 iterations of the learning algorithm. The subset of keywords that have the higher values are interpreted as the text preference of the users that generated sessions contained in the respective cluster.

4.4. Model validation

As was mentioned in Section 3.5, the validation consists of two main processes. First, using content and structure data from a different period, a web graph structure is generated. Within this, groups of ants, each one associated with one of the trained μ vectors, generate a set of artificial sessions. The second part is related with the aggregated comparison between the mentioned group of artificial sessions and the clusters of sessions from real web usage during the validation period.

For each of the 806 trained μ vectors, a group of 200 ants was associated. Each group was released on a respective web graph and the generated sessions were inserted sequentially in a database. When a session is inserted, a similarity measure is calculated between the current session and the one that had been stored immediately before

$$h(t) = sim(session(t), session(t-1)). \tag{8}$$

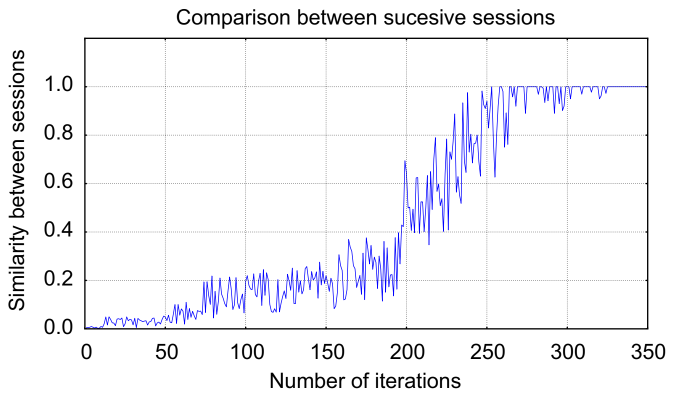

Fig. 3. Sequential comparison between generated sessions for μ_{10}.

Fig. 3 shows the evolution of $h(t)$ for the sessions generated by the group of ants associated with μ_{10}. In the beginning, the similarity between inserted sessions was almost zero. This is because the levels of pheromone in the web graph had not been sufficiently modified to generate defined trails that force the ants to follow a determined pattern. As time passes, similarity progressively increases until approximately iteration number 350, where a value of 1 is reached for the first time, which means equal sessions.

With the purpose of collecting a subset of sessions that have a coherent level of homogeneity, it is proposed to extract the last 50 inserted sessions for each group of ants.

The next step is to compare artificial sessions with real sessions and calculate what proportion of the behavior can be explained by the model. For each cluster of real sessions, the characteristic value was calculated. For each set of artificial sessions, a representative value was selected. Both processes were performed following Section 3.5.5. The similarity threshold was set at $t=0.6$, due to the fact that this value statistically represents at least two matching elements between sessions, including sequence order. If this threshold is reached, a real session is explained by an artificial session.

Each representative artificial session was compared with every characteristic value. Results show that 611 of 806 of them have a

[3] http://cs.gmu.edu/eclab/projects/mason/

similarity greater than or equal to 0.6 with at least one of the 733 characteristic values. If $t=0.7$, that number decreases to 484, and if it is set to $t=0.5$, the number increases to 639. Then, performing the comparison from the characteristic values, when $t=0.6$, 598 characteristic values can be explained by the set of representative artificial sessions. With 0.7, 452 characteristic values are obtained and with $t=0.5$, 601 (81%).

It must be noted that the results are not equivalent when performing the comparisons in both directions. The reason for this is that there is a group of representative artificial sessions that are associated with the same central value, i.e., a particular usage behavior, represented by the characteristic value, and therefore by a cluster. This is explained by different groups of ants, which are associated with different text preference vectors μ.

With respect to the comparison of this method with other techniques used for predicting web usage patterns, the literature is almost vague. But taking clustering analysis into account (Velásquez et al., 2004), the results show similar performance to k-means and SOM-based clustering for discovering web usage patterns, although the time necessary to achieve the results is reduced with the ACO approach. This is primarily due to the parallelism of the metaheuristic that allows the performance of multiple tasks at every step.

Another interesting result is related with the correlation between the most important keywords present in the pages that receive the majority of visitors, and the keywords that have the highest values in the resulting text preference vectors. The stemmed keywords, from the most visited pages, that have the highest value for TF-IDF are:

- *manag* (management), *ingenieri* (engineering), *pregrad* (undergradute), *empres* (enterprise).
- *mgpp* (abreviation for a Master's program), *descripcion* (description), *Patrici* (Patricio).
- *Capit* (capital), *susten* (sustainability), *merc* (market).

All the above keywords are present in the *MIK* set calculated in Section 3.2.1 and they appear with high values (with an average value of 0.71) in 67% of the resulting text preference vectors. Although it is a preliminary result, it can be interpreted as the training process generating text preference vectors that have a coherent resemblance with real user preference. This is due to the fact that keywords that mostly characterize the artificial text preference vectors are also present in the most-viewed web pages.

5. Conclusions

We conclude that the model presented in this paper is a plausible way to integrate the ACO metaheuristic with web mining methodologies. Its multi-variable approach, which uses both content, structure and usage data from web sources, allows the building of a framework for web user simulation based on the construction of text preference vectors, whose fully parameterized structure allows the detection and incorporation of any change in web environment. It must be noted that the simplicity of problem formulation which is intrinsic to ACO cannot be applied directly to web-based problems because of the complexity of data and its time dependency. Additional functionalities are required to handle the interaction of data from different dimensions. For example, as in standard ACO implementations such as vehicle routing, in which both starting and ending points are well defined, web user sessions present a multi-starting–ending nature, which is a challenge to ACO in terms of ensuring algorithm convergence.

As the learning algorithm depends basically on a text preferences analysis, the lack of structure of the Web makes mandatory to perform a good pre-processing of the data. If the access to the data is not ensured or the quality is poor, then the results showed in this work may not be replicable. Thus, while the deep dependency on the input data could be seen as an advantage in terms of that the results will be specific for each web site, it also could represent a drawback due to the existence of web sites on which will be infeasible to gather the minimum amount of data necessary for a good training. The importance of the acceptance of web standards along with the rising of new technologies such as HTML5, which incorporates a semantic approach and native support for most multimedia formats, could help to make website content more accessible.

Another fact to be pointed out is that, as this model is based on a collaborative multi-agent construction of solutions, results have to be understood in an aggregated fashion. Thus, this work allows the obtaining of a global estimation of web usage. Specifically, 81% of explanation in relation to the coherent matching between artificial and real sessions based on a similarity measure was achieved.

Additionally, the initial results related with the characterization of text preference vectors and their correlation with the keywords present in the most visited pages could represent a starting point for a new methodology for estimating in an aggregated way the relevant topics for a given web user. This could be compared with existing techniques such as Latent Dirichlet Allocation (LDA). Future work is related to the refinement of the text preference model in order to incorporate different utilities when collecting information, which could adapt in a better way to real user behavior. Changes in the collaborative process could also be made to obtain a faster solution, including comparison with other metaheuristics.

Acknowledgments

This work was partially supported by the FONDEF project D10I1198 and the Millennium Institute on Complex Engineering Systems (ICM: P-05-004-F, CONICYT: FBO16).

References

Abraham, A., Ramos, V., 2004. Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming. CoRR abs/cs/0412071.
Christensen, A.L., O'Grady, R., Dorigo, M., 2007. Morphogenesis: shaping swarms of intelligent robots. In: Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07), Vancouver, Canada.
Bharne, P., Gulhane, V., Yewale, S., April 2011. Data clustering algorithms based on swarm intelligence. In: Third International Conference on Electronics Computer Technology (ICECT), vol. 4, 2011, pp. 407–411.
Blashfield, K.R., 1976. Mixture model tests of cluster analysis: accuracy of four agglomerative hierarchical methods. Psychol. Bull. 83 (3), 377–388.
Brin, S., Page, L., 1998. The anatomy of a large-scale hypertextual web search engine. In: Computer Networks and ISDN Systems. , Elsevier Science Publishers, pp. 107–117.
Dorigo, M., Gambardella, L.M., 1997. Ant colonies for the travelling salesman problem. Biosystems 43 (2), 73–81.
Dorigo, M., Stutzle, T., 2004. Ant Colony Optimization. MIT Press.
Ganguly, D., Mukherjee, S., Naskar, S., Mukherjee, P., 2009. A novel approach for determination of optimal number of cluster. International Conference on Computer and Automation Engineering 0, 113–117.
Gusfield, D., 1999. Algorithms on String, Trees and Sequences. Computer Science and Computational Biology. Cambridge University Press.
Hay, B., Wets, G., Vanhoof, K., 2004. Mining navigation patterns using a sequence alignment method. Knowl. Inf. Syst. 6 (March), 150–163.
Hiemstra, D., 2000. A probabilistic justification for using TFIDF term weighting in information retrieval. Int. J. Digit. Libr. 3, 131–139. doi:10.1007/s007999900025.
Lin, C.-C., Tseng, L.-C., 2010. Website reorganization using an ant colony system. Expert Syst. Appl. 37 (12), 7598–7605.
Liu, B., 2007. Web Data Mining: Exploring Hyperlinks, Contents and Usage Data. Springer-Verlag, Berlin, Heidelberg.
Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G., 2005. MASON: a multiagent simulation environment. Simulation 81, 517–527.

Murtagh, F., 1983. A survey of recent advances in hierarchical clustering algorithms. Comput. J. 26 (4), 354–359.

Román, P.E., 2011. Web User Behavior Analysis. Ph.D. Thesis, Universidad de Chile.

Román, P.E., Velásquez, J.D., 2010. Stochastic simulation of web users. 2010 IEEE/WIC/ACM International Conference, vol. 1., IEEE Computer Society Press, pp. 30–33.

Salvador, S., Chan, P., 2004. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. IEEE International Conference on Tools with Artificial Intelligence, pp. 576–584.

Spiliopoulou, M., Mobasher, B., Nakagawa, B.B.M., 2003. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. J. Comput. 15 (2), 171–190.

Umarani, R., Selvi, V., 2010. Article: comparative analysis of ant colony and particle swarm optimization techniques. Int. J. Comput. Appl. 5 (4), 1–6 (published By Foundation of Computer Science).

Velásquez, J.D., Jain, L.C., 2010. Advanced Techniques in Web Intelligence. Springer-Verlag, Berlin, Heidelberg.

Velásquez, J.D., Palade, V., 2008. Adaptive Web Sites: A Knowledge Extraction from Web Data Approach. IOS Press.

Velásquez, J.D., Yasuda, H., Aoki, T., Weber, R., February 2004. A new similarity measure to understand visitor behavior in a web site. IEICE Trans. Inf. Syst. (2).

White, T., Salehi-Abari, A., Box, B., May 2010. On how ants put advertisements on the web. In: IEA-AIE 2010: Proceedings of the 23rd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, Cordoba, España.

Zhao, Y., Karypis, G., 2002. Evaluation of hierarchical clustering algorithms for document datasets. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management. CIKM '02. , ACM, New York, NY, USA, pp. 515–524.