Identifying User Sessions from Web Server Logs with Integer Programming

Pablo E. Román [a,1] Robert F. Dell [b,2] Juan D. Velásquez [a,3] Pablo S. Loyola [a,4]

[a] *Department of Industrial Engineering, University of Chile, República 701 - P.O. Box: 8370439 Santiago, Chile*

[b] *Operations Research Department, Naval Postgraduate School, Monterrey, California 93943, USA.*

Abstract. Web usage mining has proven to be an important advance for e-business systems, both by finding web user buying patterns and suggesting ways to improve web user navigation. A primary input for web usage mining is web user sessions that must be constructed from web server logs (called *sessionization*) when such sessions are not otherwise identified. We use bipartite cardinality matching and a more general integer program to construct sessions. We also propose several variations of our integer program to provide additional insights into session characteristics. For testing, we retrieve 15 months of web server logs and corresponding real sessions from an academic web site. We compare real sessions, results obtained by our optimization models, and results from a commonly-used timeout heuristic. We find our optimization models dominate the timeout heuristic using several comparison measures. Solution time for a typical month is seven hours for our integer program, 30 minutes for our bipartite cardinality matching, and about 1 minute for the heuristic. Although solution time is significantly greater for the integer program, its variations contribute additional analysis of web user behavior.

Keywords. Web Server Logs, Sessionization, Web User, Web Usage Mining, Integer Programming, Network Flow

1. Introduction

Capturing user activity at a web site and mining this information is part of the field which has been coined web usage mining [38]. E-business motivates the field with its desire to improve web sites in order to capture more users and sales. The data commonly used for web usage mining are sessions. A session is the sequence of pages visited by a single user at a single web site for a specified length of time. Direct monitoring of a web user's activity (e.g. [7]) provides an accurate session but such tracking can constitute a violation of privacy [11] and can be forbidden by law [27]. When direct monitoring is not possible, sessions must be estimated (*sessionization*).

[1]proman@ing.uchile.cl, Corresponding author.
[2]dell@nps.edu
[3]jvelasqu@dii.uchile.cl
[4]ployola@ing.uchile.cl

The primary sessionization input is the record of anonymous user activity collected by each web site (*web server log*). A web server log [3] is a large text file with each line (*register*) containing the following: the time of document (web page) access, the user's IP address, the *agent* field that identifies the user's browser, and the document retrieved. A web server log by itself does not reflect the sequence of an individual user's document access because, among other reasons, many individual users can share the same IP address (e.g. use of Network Address Translation (NAT) [9]).

This paper addresses the problem of estimating individual user sessions from web logs (sessionization). For testing purposes, we were granted special permission to track the browsing activities of users on an academic web site [1] and record every individual page accessed for 15 months. This was done using time-ordered access registers from cookies. We use these observed sessions for comparing our optimization models for sessionization and a commonly-used sessionization heuristic.

Prior to our work on sessionization [12, 13, 33] using integer programming, different heuristics had been proposed [6, 7, 10, 21, 34, 40]. In this paper, we reintroduce the prior integer program with some modifications and present bipartite cardinality matching as an efficient algorithm for computing accurate sessions. We test the accuracy of the sessions obtained by our optimization models using our 15-month data set. We also propose several variations of our integer program to provide additional insights into session characteristics. Such variations consist of obtaining the most likely session by number of copies, by a given size (number of registers), and by fixing a page in a given order. Such variations provide additional insight into browsing tendencies and are thereby valuable in web site design.

The rest of this paper is organized as follows. Section 2 provides a brief summary of related research. Section 3 presents our optimization models for sessionization. Section 4 outlines our test data and presents results. Section 5 shows variations of the optimization models to explore the likelihood and characteristics of specific sessions. Section 6 provides conclusions.

2. Related work

Sessionization is part of the Knowledge Discovery in Databases (KDD) process [29] for web mining [30]. The KDD process is a sequence of steps for extracting novel patterns from large data repositories. This process is roughly divided into the following stages: data collection, pre-processing, pattern discovery, and pattern analysis. In the case of web usage mining, data collection relates to the extraction of relevant registers from web logs. Pre-processing corresponds to register filtering that is needed because not all registers correspond to user actions (e.g. web search engine robots [14]). It is well known that web usage pattern discovery algorithms (e.g. clustering [18]) are sensitive to the quality of the obtained sessions [32, 19, 5]. This motivates the search for more accurate sessionization algorithms.

Strategies for sessionization can be classified as reactive and proactive [34]. Proactive sessionization strategies capture a rich collection of a user's activity during his/her visit to a site but are invasive, and in some countries forbidden [34], or they are legally regulated to protect user privacy [27]. Examples include cookie-oriented session retrieval

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

[7], where personal data are stored on the user's computer, from which a complete session can be retrieved. Using URL rewriting [15] to store personal information, which is finally kept in logs, is another way to track the user. The most invasive example is web-tracking software (which is close to being spyware) on a user's computer (or browser), which captures the entire session [28].

Reactive sessionization has fewer privacy concerns because it only uses web server log registers that do not include explicit user information [34]. However, a web log only provides an approximate way of retrieving a user's session for several reasons. The same IP address and agent as recorded in the web log often contains the requests of several concurrent users without each user being uniquely identified. Additionally, a user's activation of the back and forward browser buttons is often not recorded in the web log because, in most cases, the browser retrieves the page from its own cache [13]. A proxy server, acting as an internet web page cache to reduce network traffic, can also capture web requests that are not recorded in a web log (e.g. [17]).

Prior methods to estimate sessions from a web server log have been heuristic and most commonly based on limited session duration [34] (timeout heuristic). An information retrieval method with such a characteristic is called unsupervised. These heuristics form one session at a time so that a session does not exceed a maximum duration parameter, usually 30 minutes [10]. Another heuristic approach is to reconstruct sessions that share the same semantic [24].

Several authors have looked at the overall characteristics of sessions. They find that the size n of a web user session can be approximated by a power law ($n^{-\alpha}/\zeta_K(\alpha)$) distribution [20, 37], where $\zeta_K(\alpha) = \sum_{n=1}^{K} n^{-\alpha}$ is the generalized harmonic number [25, 1] function needed for normalization. The *size n* of a session is the total number of registers in the session. The parameter α is the decay exponent. K is the total number of registers and is interpreted as the maximum number of possible sessions. Prior work uses this empirical property as a measure of sessionization quality (e.g. [12]).

A rich literature exists on mining sessions after they are identified. Techniques such as statistical analysis, association rules, clustering, classification, sequential pattern and dependency modeling are used to discover patterns of web user behavior [23, 26, 35].

3. Optimization models for sessionization

We present two optimization models for sessionization. Each optimization model considers a groups of log registers having the same IP address and agent. Each explicitly enforces the link structure of the site in any constructed session. Each also constructs all sessions simultaneously unlike heuristics that construct one session at a time.

The first optimization model is a slight generalization of our sessionization integer program (SIP) that was orginally presented in [12] without any comparison to real sessions. The second optimization model is a novel use of the well-known bipartite cardinality matching (BCM) problem [2]. There are several specialized algorithms available for solving BCM with complexity $O(\sqrt{M}A)$ where M is the number of nodes and A is the number of arcs (e.g. [2]). SIP does not have the same polynomial time guarantee.

Proper construction of the bipartite graph ensures SIP binary variables and BCM graph have identical feasible regions. Sessions must follow constraints based on web site topology and time ordering. Sessionization requires identifying each register r from the web log as part of a unique web user's visit. Each constructed session is an ordered list of log registers where each register can only be used once and in only one session. In the same session, register $r1$ can be an immediate predecessor of $r2$ only if: the two registers share the same IP address and agent; a link exists from the page requested by $r1$ to the page requested by $r2$; and the request time for register $r2$ is within an allowable time window from the request time for register $r1$.

SIP permits a general objective function while the only BCM objective function minimizes the number of sessions. An objective function that minimizes the number of sessions is of potential interest because it provides the lower limit on the number of sessions for a given log file. Such an optimal solution also has intuitive appeal because fewer overall sessions from the same web server log should result in fewer, less interesting sessions that consist of only one visited page. Sessions of size one do not reflect a web user's interaction with the web site, so they are not usually considered for web usage studies.

3.1. Sessionization integer program (SIP).

We present the SIP formulation in NPS standard format [8] (sections 3.1.1 to 3.1.6).

Our SIP uses a binary variable X_{ros} that has value "one" if a log register r is assigned as the *oth* position during session s, and zero otherwise. Each index r identifies a unique register, each index s identifies a unique user session, and the index o is the ordered position of a register during a session. The upper limit for a session size ($\sum_{ro} X_{ros}$) is much less than the number of registers, a limit for the o index is introduced as the maximum size of a session.

The SIP objective function expresses the overall reward of all constructed sessions. By altering the objective function coefficients, SIP has the flexibility to reward different session characteristics. This is a significant advantage over the commonly-used timeout heuristic that has no reward mechanism but it does present the challenge of selecting these objective function coefficients in a way that constructs as many real sessions as possible. This paper suggests possible values and prior work found little variability as long as the values of the objective function coefficients increase as function of o [12]. Such an increasing function has the general tendency to reward longer sessions (session with more registers).

3.1.1. Indices

o	Order of a log register visit during a session (*e.g.*, $o = 1, 2, \cdots, 30$). The cardinality defines the maximum size of a session.
p, p'	Web page.
r, r'	Web server log register.
s	Web user session.

3.1.2. Data [units]

Used to produce the index sets below:

$T(r)$	The access time of register r [seconds].
$IP(r)$	The IP address of register r.
$A(r)$	The agent of register r.

$p(r)$	The page of register r.
T^{min}, T^{max}	The minimum, maximum time between accessing pages in a session.
$(p, p') \in E$	If a page p has a link to page p', (p, p') belongs to the set of edges E.

3.1.3. Index Sets

$bpage_r$	$\{ r' \mid T(r') < T(r),\ (p(r'), p(r)) \in E,\ IP(r') = IP(r),\ A(r') = A(r),$ $T^{min} \leq (T(r) - T(r')) \leq T^{max}\}$ The set of registers that can be the register immediately before register r in the same session. Based on: - Pages that have a link to the page of register r. $(p(r'), p(r)) \in E =$ {set of hyperlinks} E = {(p,p')| p,p' are web pages, ∃ a hyperlink from p to p'} - Register r and register r' have the same IP adddress, i.e. $(IP(r) = IP(r'))$. - Register r and register r' have the same agent, i.e. $A(r) = A(r')$ - Time of register r and register r'. - We assume a user-defined minimum (T^{min}) and maximum (T^{max}) time between two consecutive registers in the same session.
$first$	$\{ r \mid \text{bpage}_r = \phi\}$ Set of registers that can be first in a session. This set is auxiliary and helps to reduce the number of variables.

3.1.4. Objective function coefficients

C_{ro}	The objective function coefficient for register r assigned to the *oth* position in a session.

3.1.5. Binary Variables

X_{ros}	1 if log register r is assigned as the *oth* request during session s and zero otherwise.

3.1.6. Formulation

Maximize

$$Z(X) = \sum_{ros} C_{ro} X_{ros} \tag{1}$$

Subject to:

$$\sum_{os} X_{ros} = 1 \qquad \forall r \tag{2}$$

$$\sum_{r} X_{ros} \leq 1 \qquad \forall o, s \tag{3}$$

$$X_{r,o+1,s} \leq \sum_{r' \in \text{ bpage}_r} X_{r',o,s} \qquad \forall r, o, s \tag{4}$$

$$X_{ros} \in \{0, 1\} \qquad \forall r, o, s \tag{5}$$

The objective function, (1), expresses the overall reward of all constructed sessions. Selecting different values for the objective function coefficients, C_{ro}, allows the flexibility to reward different session characteristics. For example, setting $C_{ro} = 1 \;\; \forall r, o = 3$ and $C_{ro} = 0 \;\; \forall r, o \neq 3$ provides an objective function for maximizing the number of sessions of size three. In section 4.4, values for C_{ro} are based on an earlier study [12].

Constraint set (2) ensures that each register is used exactly once. Constraint set (3) restricts each session to having at most one register assigned for each ordered position. Constraint set (4) ensures the proper ordering of registers in the same session. $X_{ros} \in \{0, 1\} \forall r, o, s$ declares variables as binary. To improve solution time, we can fix (eliminate) a subset of these binary variables to zero ($X_{ros} = 0, \forall r \in first, o > 1, s$). After forming the set bpage_r, the set $first$ is easily found ($r \in$ first if $\text{bpage}_r = \phi$).

Other parameters like T^{min} and T^{max} are standard in sessionization and are set to 1 and 300 seconds respectively [10] for all computational results reported in this paper. The implementation must select a maximum session size parameter and larger values can increase solution time. Because there are typically only a few large sessions sizes [20], we use 30 as our default and haven't observed any noticeable improvement when increasing its value.

3.2. Bipartite cardinality matching (BCM)

The second optimization model we present for sessionization is bipartite cardinality matching (*BCM*) (e.g. [2]). The BCM problem consists of finding a matching of maximum cardinality in a bipartite undirected network. A bipartite graph consists of two sets of nodes with edges only connecting nodes in different sets. The matching of maximum cardinality is the set of edges such that each node is connected by at most one edge. We construct the bipartite undirected network from our web server log so that the matching of maximum cardinality is equivalent to minimizing the number of sessions. This is equivalent to SIP with $C_{r,o=1,s} = -1 \; \forall rs$ and $C_{r,o>1,s} = 0 \; \forall rs$.

In our network, each register is represented by two nodes, one on the *from* side (representing an immediate predecessor) and one on the *to* side (representing an immediate successor). Figure 1 shows a six-register example. On each side, we order the nodes (registers) in order of increasing time as recorded in the web server log. An arc exists from a node on the *from* side, $r1$, to a node on the *to* side, $r2$, if the register corresponding to $r1$ could be an immediate predecessor of $r2$. The definition of what is a "predecessor" of r is the same as defined for set bpage_r (see section 3.1.3). For the example in Figure 1, we assume seven arcs exist.

Given a solution of the BCM problem, we construct the sessions from the matching. A node on the *from* side that is not matched is the last node in a session. A node on the *to* side that is not matched is the first node in a session. A session follows the connected segments in the matching where (to aid in visualizing the sessions) we add directed arcs (from the *to* side to the *from* side) connecting nodes representing the same register. Figure 2 provides a solution to the example of Figure 1. Nodes 4 and 6 end sessions, nodes $1'$ and $4'$ start sessions, which results in two sessions ($1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$ and 4).

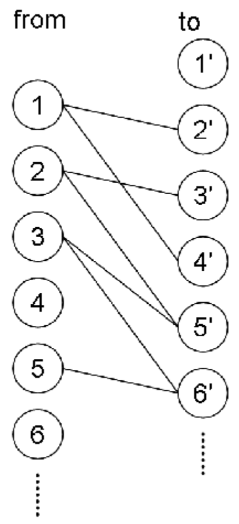

Figure 1. Bipartite maximum cardinality matching. Each register is represented by two nodes. An arc exists if the register on the *from* side can be an immediate predecessor of the node on the *to* side.

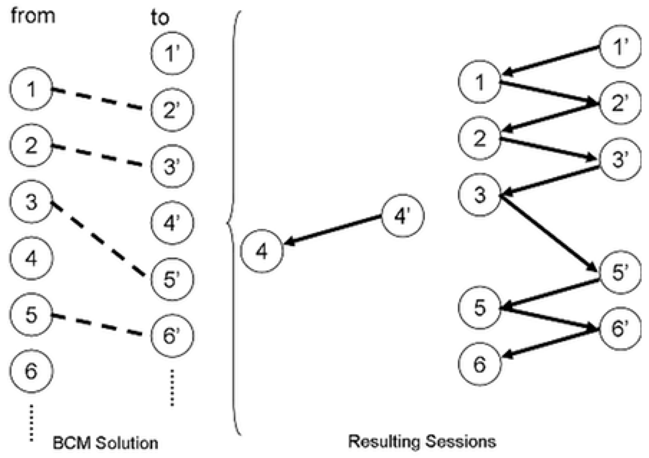

Figure 2. The maximum cardinality matching has four arcs. We construct two sessions from the matching: 1-2-3-5-6 and a session with only register 4.

4. Test data and results

In this section, we compare results from BCM, SIP, a timeout heuristic and real sessions obtained by cookies. We use five sub-sites from the University of Chile Industrial Engineering Department web site (http://www.dii.uchile.cl), where permission was granted for tracking web user activities. These correspond to the main department site, three sub-sites from a master's degree program, and a project web site. These sites consist of nearly $4,000$ web pages. Each site has its own unique content and structure without a unique framework for the format and design of the whole web site. Only one uses a content-management system that standardizes content addition. The others require manual insertion of new content. The main topics on these web sites include: general information about the Industrial Engineering Department; general information about faculty and staff; descriptions of the undergraduate and graduate programs; and news and information about upcoming events and conferences.

These sub-sites have a relatively simple construction consisting only of static HTML pages without relevant flash animation. They contain information about programs of study, academics, projects, and news. In a typical month, about 5% of the links are modified, 2% of the pages are new or deleted, and 30% of the words change their frequency of appearance. Most of the changes are on pages with news stories that are updated weekly (Churn and Scroll updating [31]). Based on these observations, we divide web logs by month and assume a static web structure (links and pages) during a month.

4.1. Web site and web log characteristics

We record the hyperlink structure of the web site by month. The average number of hyperlinks on the web site is $4,058$ with a mean absolute difference per month of 109 (2.7%). We also record the number of pages each month. The average per month is 691 different pages with a mean absolute difference about of 15 pages per month (2.2%).

The distribution of the number of links per page (Figure 3) is an indicator of possible session variability. In this data set, there are on average 16 links per page, providing an estimated 16^L unique sessions of size L.

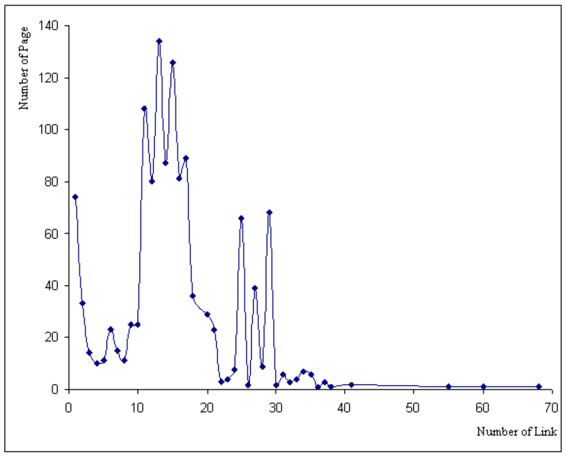

Figure 3. Number of links per web page.

The IP address is a commonly used attribute for separating sessions [10]. We find that only a few IP addresses account for the vast majority of all registers. Over 98 percent of all IP addresses (nearly $150,000$) have less than 50 registers for the period of study. Figure 4 displays the number of registers for the 100 IP addresses that account for the most registers.

We also determine how many pages are visited by users from the same IP address. This provides some evidence of the diversity of navigation patterns from a given IP

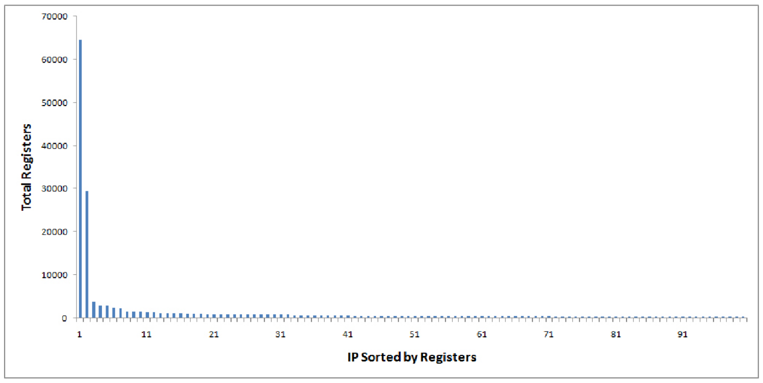

Figure 4. The number of registers for the 100 IP addresses that account for the most registers.

address. Figure 5 shows the number of different pages requested by the first 2,000 IP addresses that account for the greatest number of different pages requested. Of the IP addresses not shown, almost 65 percent visit three or less different pages for an entire month.

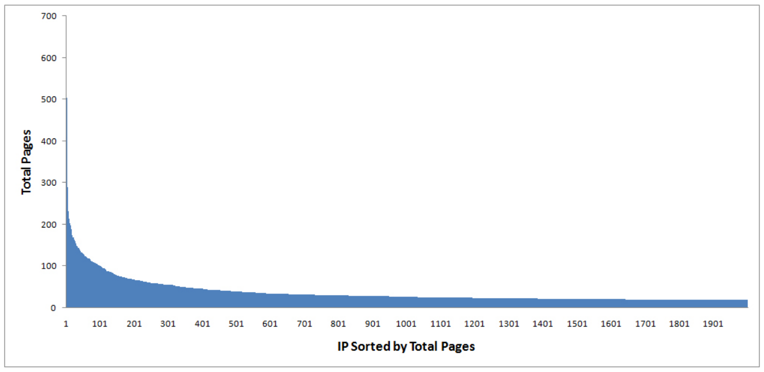

Figure 5. The 2,000 IP addresses that account for the greatest number of unique page requests.

4.2. Data pre-processing

We retrieve 15 months of sessions using cookies [7]. The web site was modified to track web user navigation and recover sessions without personal information such as usernames or other personal identifiers. Information retrieved by this method was restricted to a session identifier, access time, IP address, the agent field, and the accessed web page.

Each web page in the site included a JavaScript program for tracking each individual's navigation actions (Figure 6). This program uses both client-side and server-side elements to automatically collect the sessions. First, when a user starts navigation, this script sets a session identifier based on a randomly-generated number and stores it on the user's computer using a cookie. This cookie updates the session and stores the collected session on a remote server. A session database collects accessed registers (time, IP address, agent field, web page) including a session identifier.

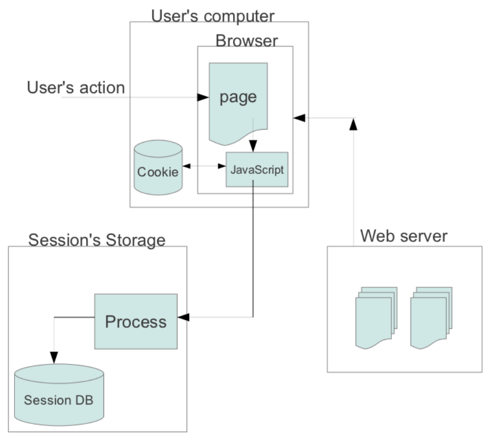

Figure 6. Session retrieval by a cookie method.

The cookie-based sessionization method has remarkable advantages regarding the automatic identification of web user sessions. Its drawbacks, in addition to privacy [11], are related to its dependency on the "onload" and "onbeforeunload" JavaScript event used to identify when a user enters or leaves a page. Different web browsers handle and execute these functions differently resulting in registers with incorrect insertions or missing values. In addition, a cookie should record both the entering event (IN) and leaving event (OUT) of a page but this doesn't always take place. To fix any issues, registers were processed to consider any IN or OUT event on different pages to be different registers.

Data collection started in June 2009 and finished in August 2010. Collected data was stored using the structure shown in Table 1. For the 15 months, $1,224,812$ rows were inserted into the table, containing $382,047$ sessions, $121,968$ different IP addresses and $1,227$ different web pages. Cleaning was necessary to improve data quality. It included deletion of URLs referencing non-web page documents (images or files) and irrelevant frames, as well as deletion of records from robot crawling activity.

After processing, a clean data set was obtained with a total of $708,007$ rows, consisting of $360,748$ sessions from $114,041$ different IP addresses and visits to $1,192$ web pages. Although there was a significant reduction in the total number of records (42.1%), it did not substantially alter a number of key components: only a 5.6% reduction in the number of web user sessions; only a 6.4% reduction in the number of IP addresses; and only a 2.8% reduction in the number of web pages.

Figure 7 shows the number of registers and sessions by month. As a rule of thumb, the number of sessions is on the order of half of the number of registers. There is clearly less activity during the summer months of January and February. The number of sessions varies little from June to November.

Consistent with prior research [20, 37], we find the distribution of session size appears to follow a power law distribution known as the "Web Surfer Law" with little variation from month to month. It has a good linear approximation in logarithmic scale.

Name	Type	Description
id_log_session	int	primary key (autoincrement)
id_session	varchar	randomly generated value that identifies the session
time	int	unix timestamp
ip	varchar	web user IP address
host	varchar	web host of the requested web page
URL	varchar	web url of the requested web page
event	varchar	identification of entering a page (IN) or leaving (OUT)
query	varchar	query parameters passed to the url
agent	varchar	web user's browser

Table 1. Data retrieved from javascript events and cookies.

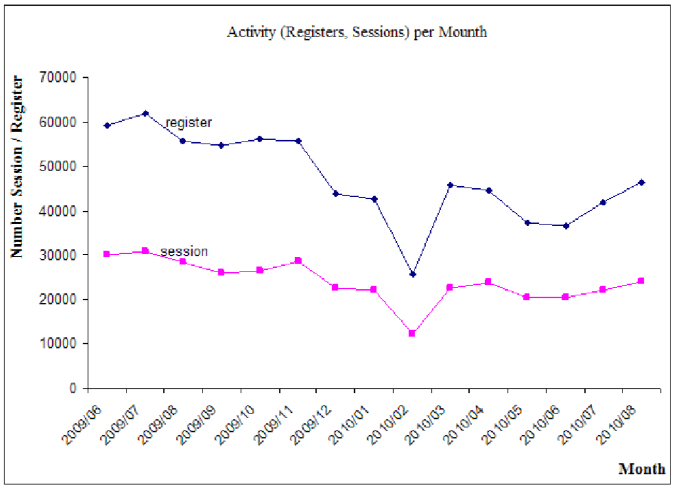

Figure 7. The number of sessions and registers from June 2009 to August 2010.

More precisely, the distribution has a better piecewise linear fit (Figure 8) where shorter and longer sessions show slightly different behavior.

4.3. Performance Measures

We consider the sessions identified by cookie extraction to be the real sessions and propose several measures of comparison. The most aggregate performance measure is the total number of sessions. Of course, it is possible to find the same number of sessions without having identified a single real session so we use several other measures.

We adopt precision and recall as in [36] and [4], with exact matching of real sessions for constructing precision and recall measures [7]. Let C be the set of all real sessions, $\mathcal{M} \subset C$ the set of sessions matched exactly, and $\mathcal{S}$ the set of all sessions constructed by sessionization. In this case, precision and recall are defined by equations 6 and 7.

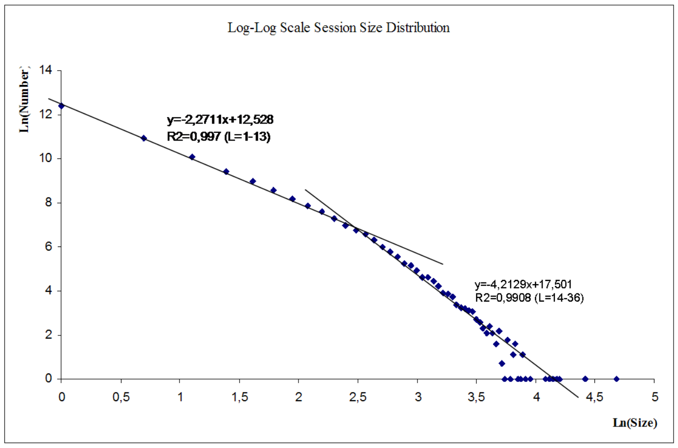

Figure 8. Piecewise linear distribution approximation for session size in Log-Log scale.

$$p = \frac{|\mathcal{M}|}{|\mathcal{S}|} \tag{6}$$

$$r = \frac{|\mathcal{M}|}{|\mathcal{C}|} \tag{7}$$

The harmonic mean $F = \frac{2pr}{p+r}$ is called the F-score [4], taking positive values less than one, and where a higher score is better. The F-score only counts sessions that exactly match without any reward for partial matches.

Berendt et al. [7] propose an additional measure (an overlap measure) to count the degree of overlap between real and constructed sessions. As they define it, the degree of overlap between a real and constructed session is the number of registers in common divided by the total number of registers in the real session. Sessions are compared by selecting pairs of maximal similarity. A final overlap measure is the total number of exactly matched registers from all pairs divided by the number of real registers.

Zhang et al. [40] propose the S-measure of similarity. The S-measure between two sessions is the intersection of the pages of the two sessions divided by the union of their pages. As with the overlap measure, sessions are compared by selecting pairs of maximal similarity and the final S-measure is the average over all pairs. One is the best possible value for the F-score, overlap, and the S-measure.

4.4. Results

We construct sessions using SIP (section 3.1), BCM (section 3.2), and a commonly-used timeout heuristic for all 15 months where we have obtained the real sessions. We report results by month, session size, and aggregated. All computation was done using a two core 1.6Ghz PC with 2 Gbs of RAM. We generate BCM and SIP instances using GAMS [16] and solve them using CPLEX version 10.1.0 with default settings [22], controlled by a php script and MySQL 5.0.27 (e.g. [39]) as a data storage engine.

4.4.1. *Reducing SIP solution time*

It is easy to construct instances of SIP that cannot be solved in reasonable time. For example, a web server log of $100,000$ registers, allowing a maximum of $5,000$ sessions, and a maximum session size of 20 produces 10^{10} binary variables and even more constraints.

Fortunately, pre-processing allows us to partition the set of web log register $\mathcal{R} = \{r\}$ into M *chunks* $\mathcal{R} = \biguplus_k^M C_k$. The idea is to ensure that no register in one chunk C_k could ever be part of a session in another. Thus the integer programing problem of finding sessions in $\mathcal{R}$ is reduced to an M equivalent sub-problems, each one restricted to a subset of registers C_k.

This is easily accomplished by restricting each chunks C_k to a unique IP number (ip_k), agent field (a_k), and time restrictions as $C_k = \{\ r \mid IP(r) = ip_k,\ A(r) = a_k,\ T^{max} \geq (T(r) - T(P(r))\}$. The functions $IP(.)$, $A(.)$, and $T(.)$ are as defined in section 3.1.3. The function $P(r)$ returns the previous register sorted by time, except for the first register where $P(r_0) = r_0$. T^{max} is the maximum time allowed between registers in the same session. Such division into chunks is equivalent to the original undivided problem as long as no register in one chuck could ever be part of a session in another chunk. We avoided making a chunk too small because there is a fixed time (overhead) associated with generating and solving each chunk. For our computational work (SIP), we used a minimum chunk size of 50 registers and $T^{max} = 300$, resulting in between $6,000$ to $7,500$ chunks per month.

4.4.2. *SIP processing*

For SIP comparisons in this section, we set $C_{ro} = 3/2Log(o) + (o-3)^2/12o \quad \forall r$ as this was found to work well in previous studies [12]. Results for other SIP objective function coefficient values are presented in the next section (Integer programming extensions).

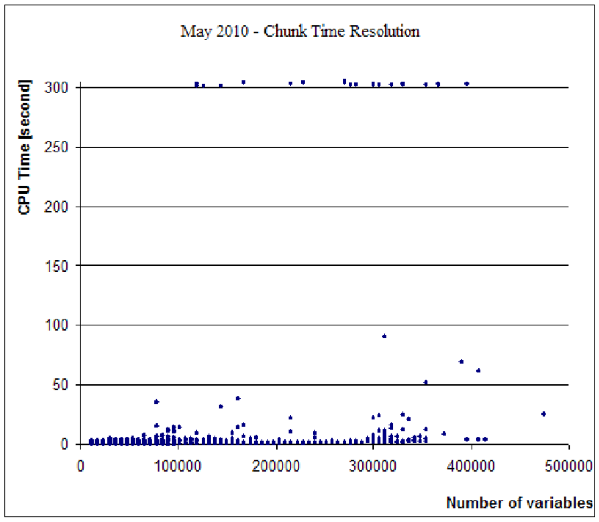

Figure 9. Solution time in seconds vs. number of binary variables for May 2010.

Using GAMS and CPLEX, SIP requires about 7 hours to solve a given month. There is little difference in solution time or instance characteristics from month to month. As an example, for May 2010 there are $76,019$ different SIP instances (chunks) (with $T_{min} = 1$, $T^{max} = 300$, and a maximum session size of 30). These instances range in size from about $12,000$ to $942,000$ binary variables and $11,800$ to over $783,000$ constraints.

For each SIP instance, we set the maximum time limit to 300 seconds and the relative gap to one percent. The 300 second (non-optimal) limit is reached in less than 30 instances (or chunks) each month. Additional solution time for these 30 or so instances had little impact on the prescribed sessions. These few chunks correspond to over 30% of the total computation time. Figure 9 shows the solution time as a function of the number of binary variables (up to 500,000 binary variables) for May 2010.

4.4.3. BCM processing

We solve BCM using the CPLEX linear programming solver. Each month takes, on average, only 33 minutes to solve optimally.

4.4.4. Timeout heuristic

For the timeout heuristic, each month takes, on average, a little more than one minute to solve. The heuristic is straightforward, because it considers filtering by IP and agent fields and separates each session by a timeout condition of 30 minutes [6].

4.5. Comparison of sessionization methods

Figure 10 shows the total number of sessions found per month using the three presented methods and the total number of real sessions obtained by cookies. SIP and BCM underestimate the total number of sessions every month while the timeout heuristic overestimates the number of sessions. BCM and SIP show nearly the same number of session (on average they find only 14% fewer sessions than the total number of real sessions). The timeout heuristic on average overestimates the total number of sessions by 37%.

Table 2 summarizes results for SIP, BCM, and the timeout heuristic. Both SIP and BCM identify nearly 25% more real sessions (precision) than the traditional timeout heuristic, with SIP performing only slightly better. Both SIP and BCM have about a 13% better F-score when compared to the heuristic. The heuristic performs slightly better for the overall recall measure however by session size, we see that both SIP and BCM dominate the heuristic for all session sizes greater than one (Figure 11).

Method	Precision	Recall	F-Score	S measure	Avg. Processing Time
SIP	0.7788	0.6696	0.7201	0.7535	7hrs.
BCM	0.7777	0.6671	0.7182	0.8511	33min.
Timeout	0.5091	0.6996	0.5893	0.6979	1min.

Table 2. Overall comparison of SIP, BCM, and the timeout heuristic (15 months).

Figure 12 shows the F-score by month. SIP and BCM clearly and consistently outperform the heuristic. SIP is usually slightly better than BCM, but at the cost of significantly more computation time. The F-score for both SIP and BCM is better than the heuristic over a wide range of session sizes (Figure 13), from a 16% improvement in F-score for sessions of size 2 to 12% for sessions of size 16. For sessions larger than

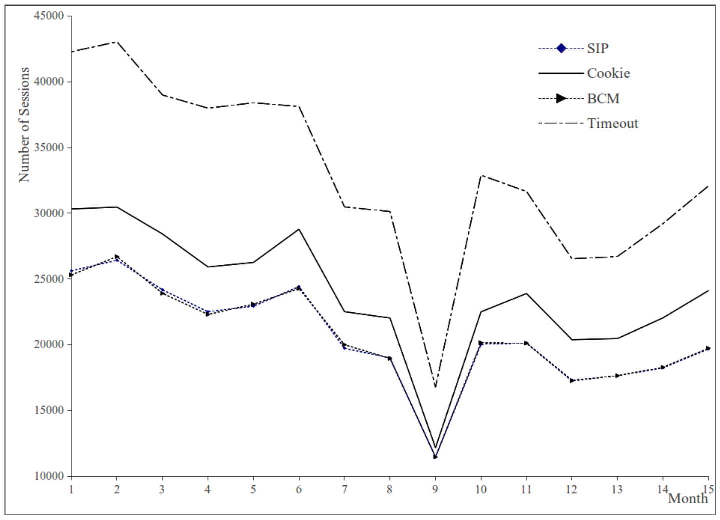

Figure 10. The number of sessions by month obtained by different methods.

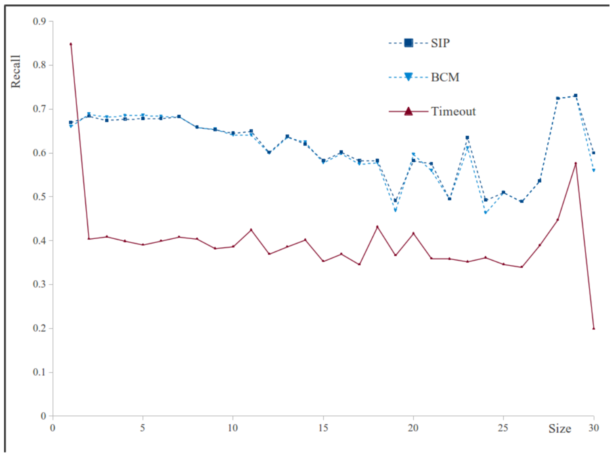

Figure 11. Recall by session size.

16, the optimization models do not always dominate, but performance is typically better. Furthermore, SIP has slightly better performance for the more numerous shorter sessions (less than 10), while BCM does better for larger sessions.

The S-measure for BCM of 0.8511 (Table 3) is better than the SIP S-measure of 0.7515 and significantly better than the timeout heuristic of 0.6979. Not only does BCM have a better total S-measure but it is better for every session size greater than 1 (Figure 14).

BCM and SIP have almost identical overlap measures [7] by month (Figure 15). Both SIP and BCM have nearly 20% more matched registers than the timeout heuristic.

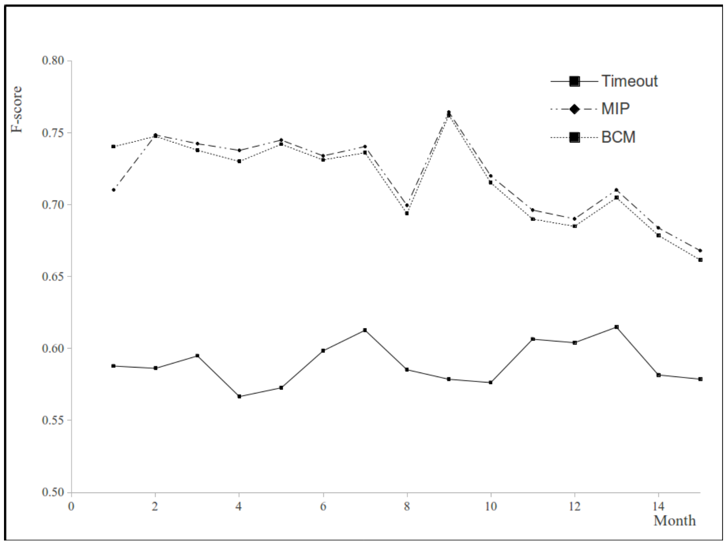

Figure 12. F-score by month.

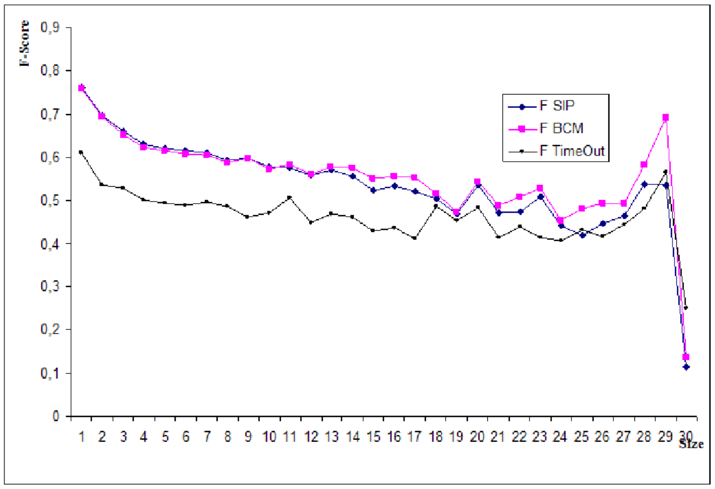

Figure 13. F-score by session size.

5. Integer programming extensions

We develop variations of our optimization models to further explore the likelihood of specific sessions and characteristics of sessions. Specifically, we find the maximum number of copies of a given session, the maximum number of sessions of a given size, and maximum number of sessions with a given web page requested in a given position for each session. All results presented in this section are using the data retrieved for May 2010.

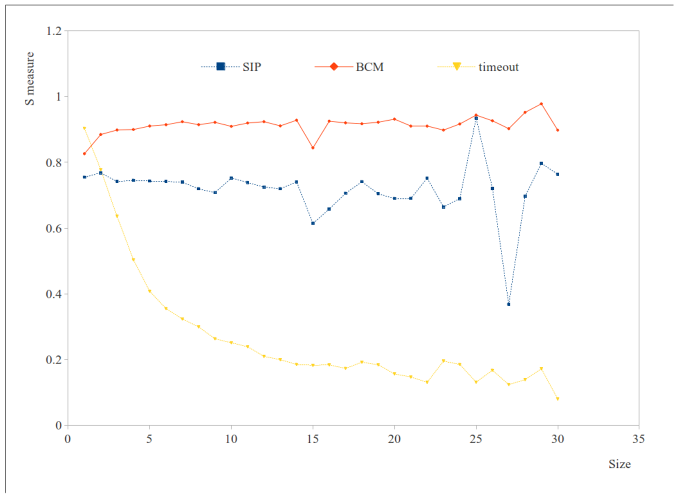

Figure 14. S-measure by session size.

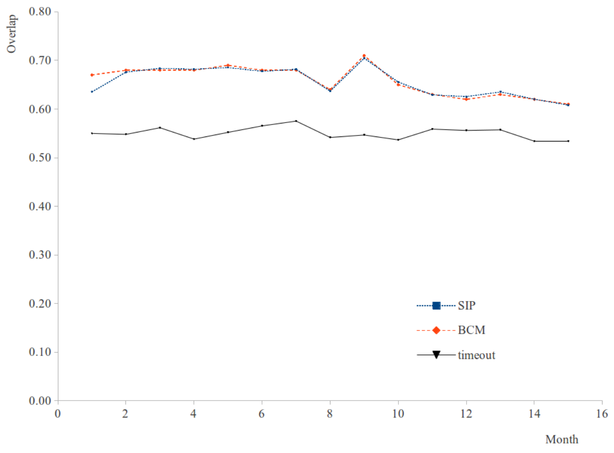

Figure 15. Overlap measure by month.

5.1. Finding the maximum number of copies of a given session

The set of sessions can be ranked by its maximum number of copies. The interpretation for this ranking is to have the highest number of most-likely sessions that can be obtained from the web log. Such most-likely sessions have interest for web designers because these web user sessions can be used for generating shortcuts that are more likely to be used.

It is well known that web designers want to minimize the number of clicks to reach desired information. The maximum number of sessions of a given size can reveal how likely it is for users to reach information within a fixed number of steps.

The most likely set of sessions that have a common page in a given position has an important e-commerce application. The common page could be set as a buying action or intention. Using such information, web sites can be improved so that this page is reached in a smaller number of steps. This result could be implemented by offering a shortcut hyperlink to users that follows a given sequence of web pages obtained by this method.

To find the maximum possible number of copies of a given session from the registers of the web server log, we have two cases. When each page in the session is visited only once, this can be modeled as a maximum flow problem (e.g. [2]). The maximum flow problem seeks a solution that sends the maximum flow through a network from a source node to a sink node. We construct the network with a node for each register that corresponds to a page of the session, a source node and a sink node. An arc exists: from the source node to each node that corresponds to a potential first page in the session; between any two nodes where one can be an immediate predecessor of the other for the session; from each node that corresponds to the last page in the session to the sink; and from the sink to the source. The arc from the sink to the source has unlimited capacity, while all other arcs have an upper capacity of one.

We have a network with side constraints when one or more pages in the session repeat. The network is much like the network where each page is visited only once; we have a source node and a sink node, but we must now also keep track of the node's order in the session. For each position in the session, we have a node for each register that corresponds to the page occurring in that order for the session. Therefore, we replicate a node corresponding to the same register the exact number of times its page repeats in the session. For each page that repeats in the session, there is a constraint to restrict the total flow out of all nodes corresponding to the same register, to be set at less-than-or-equal-to one.

$Session$	BCM	SIP	max
1	41	41	186
2	5	3	43
3	4	5	39
4	7	4	34
5	4	4	34
6	1	0	22
7	1	0	22
8	7	0	19
9	1	0	16
10	1	0	16

Table 3. The top 10 most-likely sessions based on finding the maximum possible number of a specific session compared with the number of sessions found by BCM and SIP.

By maximizing the number of copies of a given session, we find the maximum number of times the session could occur (Figure 16 and Table 3). This provides us some additional guidance on how likely the session is to occur. In Table 3, we provide these values in the "max" column. For sessions of size four, we see there are only a few sessions that can possibly occur very frequently. Specifically, only seven sessions could occur 20 or more times for the entire month. Only 17 sessions could occur more than 10 times for the entire month. We see that both BCM and SIP find session 1 to be most frequent (41 times), but after that differences occur. Session 18 (not shown) is the next most frequent session for SIP (with six sessions) but it is only the seventh most frequent session for BCM (with three sessions). Session 6 doesn't occur at all in SIP and only one time in BCM.

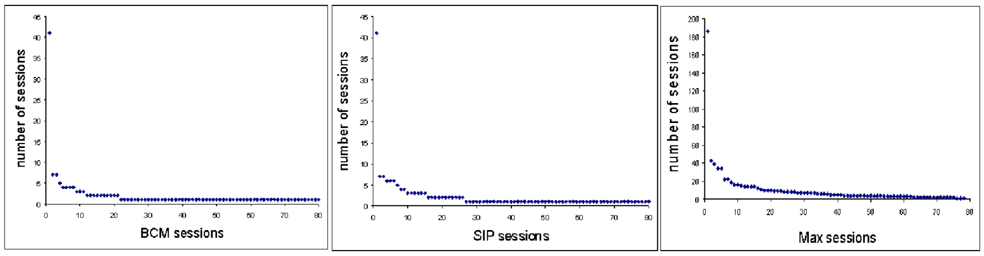

Figure 16. Number of sessions of size four resulting from BCM, SIP and minimizing the number of a given session.

It takes on average about one second to solve for a given session and chunk. 403 chunks per 80 sessions takes a total of approximately 9 hours. All instances are solved to optimality.

5.2. Maximum number of sessions of a given size

We find the maximum number of sessions of a given size. Specifically, for a specific *size* session, $C_{ro} = 0\ \forall r,\ o \neq size$, and $C_{ro} = 1\ \forall r,\ o = size$. Results for size two to ten are shown in Table 4. We see that relatively few sessions of size six (or higher) are possible. Note that the maximum number of sessions of size one is the number of registers.

Size	Num. Sessions
2	3,000
3	1,509
4	755
5	435
6	257
7	181
8	135
9	98
10	82

Table 4. The maximum number of possible sessions of a given size.

5.3. Maximum number of sessions with a page in a fixed position

"What are the most likely second or third pages in a session?" is a question Web site designers often ask. Table 5 and Figure 17 show results obtained by the SIP variation, along with BCM and SIP (with $C_{ro} = 3/2Log(o) + (o-3)^2/12o$) results for the third position. There are only four pages that could have been the third page visited in 100 or more sessions and only 19 pages that could have been the third page visited in 30 or more sessions. The BCM and SIP sessions have only some minor differences when compared to each other. Solution time is about 80 hours, an average solution time of under 5 seconds per instance. No instance reached the 300-second time limit.

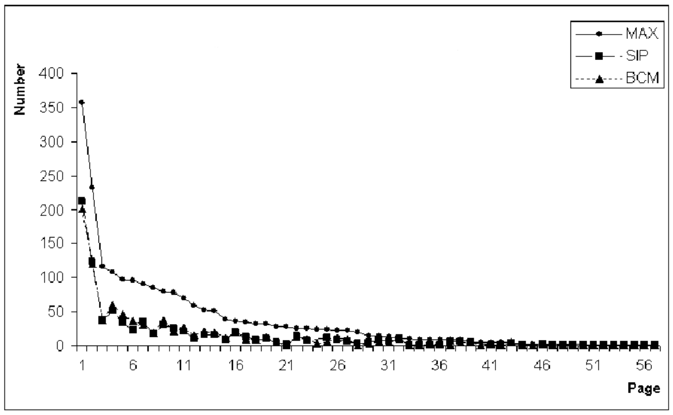

Figure 17. Maximum number of sessions with a page in the 3rd position, compared with SIP and BCM.

Page	*BCM*	*SIP*	*max*
1	201	212	356
2	121	124	232
7	38	38	116
13	60	51	108
5	46	34	97
8	37	23	96
4	31	35	91
9	19	18	86
6	38	31	80
10	22	26	78

Table 5. The top 10 pages that could be in the 3rd position compared with BCM and SIP results.

6. Conclusion

We present a new approach for sessionization using bipartite cardinality matching (BCM) and integer programming (SIP). We test our approach using real sessions retrieved from an academic web site over 15 months. We compare real sessions, results obtained by our optimization models, and results from a commonly-used timeout heuristic. We find our optimization models dominate the timeout heuristic using several comparison measures. For example, SIP has precision of 77.9%, BCM has nearly the same precision 77.8%, and the timeout heuristic is a distant third with only 50.9%.

We also provide variations of our optimization models to further explore the likelihood of specific sessions and characteristics of sessions. Specifically, we find the maximum number of copies of a given session, the maximum number of sessions of a given size, and maximum number of sessions with a given web page requested in a given position for each session.

Acknowledgment

This work has been partially supported by the Chilean Millennium Institute of Complex Engineering Systems (ICM: P-05-004-F, CONICYT: FBO16) and its associated Engineering System Doctoral program, also FONDEF project D10I- 1198 entitled, "WHALE: Web Hypermedia Analysis Latent Environment". We thank Naval Postgraduate School Professor Matt Carlyle for sharing his network modeling insights.

References

[1] S. Ahlgren, P. Paule, and C. Schneider. Computer proofs of a new family of harmonic number identities. *Advances in Applied Mathematics*, 31:359–378, 2003.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[3] C. Aulds. *Linux Apache Web Server Administration*. Sybex, 2002.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[5] M. A. Bayir, I. H. Toroslu, and A. Cosar. Performance comparison of pattern discovery methods on web log data. In *Proc. of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, pages 445–451, Dubai/Sharjah, UAE, 2006.

[6] B. Berendt, A. Hotho, and G. Stumme. Towards semantic web mining. In *LNCS 2342, Proc. of the Semantic Web International Conference (ISWC)*, pages 264–278, Sardinia, Italy, 2002.

[7] B. Berendt, B. Mobasher, M. Spiliopoulou, and J. Wiltshire. Measuring the accuracy of sessionizers for web usage analysis. In *Proc. of the Workshop on Web Mining, First SIAM International Conference on Data Mining (ICDM)*, pages 7–14, Chicago, USA, 2001.

[8] G. Brown and R. F. Dell. Formulating integer linear programs: A rogues' gallery. *Informs Transaction on Education*, 7(2):1–13, 2007.

[9] Y. Chen and W. Jia. Challenge and solutions of nat traversal for ubiquitous and pervasive applications on the internet. *Journal of Systems and Software*, 82:1620–1626, October 2009.

[10] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Journal of Knowlegde and Information Systems*, 1(1):5–32, 1999.

[11] S. E. Coull, M. P. Collins, C. V. Wright, F. Monrose, and M. K. Reiter. On web browsing privacy in anonymized netflows. In *Proc. of 16th USENIX Security Symposium (SS'07)*, pages 1–14, Berkeley, USA, 2007.

[12] R. F. Dell, P. E. Román, and J. D. Velásquez. Web user session reconstruction using integer programming. In *Procs. of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 385–388, Sydney, Australia, 2008.

[13] R. F. Dell, P. E. Román, and J. D. Velásquez. "user session reconstruction with back button browsing". In *LNAI 5711, of the Procs. Of the 13th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES)*, pages 326–332, Santiago, Chile, 2009.

[14] O. Duskin and D. G. Feitelson. Distinguishing humans from robots in web search logs: preliminary results using query rates and intervals. In *Procs. of the workshop on Web Search Click Data (WSCD)*, pages 15–19, New York, USA, 2009.

[15] F. Facca and P. Lanzi. Recent developments in web usage mining research. In *Procs. of the Data Warehousing and Knowledge Discovery International Conference (DaWaK)*, pages 140–150, Prague, Czech Republic, 2003.

[16] GAMS Development Corporation. General algebraic modeling system (gams). http://www.gams.com, 2012.

[17] S. Glassman. A caching relay for the world wide web. *Computer Networks and ISDN Systems*, 27(2):165–173, 1994.

[18] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[19] H. Hao, J. Dan, and H. Jianqing. Separating interleaved user sessions from web log. In *Procs. of the International Conference on Network Computing and Information Security (NCIS)*, pages 152 –156, Guilin, China, 2011.

[20] B. Huberman, P. Pirolli, J. Pitkow, and R. Lukose. Strong regularities in world wide web surfing. *Science*, 280(5360):95–97, 1998.

[21] T. Hussain, S. Asghar, and N. Masood. Web usage mining: A survey on preprocessing of web log file. In *Information and Emerging Technologies (ICIET), 2010 International Conference on*, pages 1–6, june 2010.

[22] ILOG. Cplex. *www.ilog.com/products/cplex*, 2012.

[23] A. Joshi and R. Krishnapuram. On mining web access logs. In *Proc. of the ACM SIGMOD Workshop on Research Issue in Data Mining and knowledge Discovery (DMKD)*, pages 63–69, Dallas, USA, 2000.

[24] J. Jung and G. Jo. Semantic outlier analysis for sessionizing web logs. In *Procs. of the European Web Mining Forum (EWMF) in the 14th European Conference on Machine Learning and 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 13–25, Dubrovnik, Croatia, 2003.

[25] D. E. Knuth. *The art of computer programming, volume 1 (3rd ed.): fundamental algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[26] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD Explorations: Newsletters of the special Interest Group (SIG) on Knowledge Discovery and Data Mining*, 1(2):1–15, 2000.

[27] D. Langford. *Internet Ethics*. MacMillan Press Ltd, 2000.

[28] V. Mayer-Schonberger. Nutzliches vergessen. In *Goodbye Privacy Grundrechte in der digitalen Welt (Ars Electronica)*, pages 253–265, 2008.

[29] S. Mitra, S. K. Pal, and P. Mitra. Data mining in soft computing framework: A survey. *IEEE Transactions on Neural Networks*, 13:3–14, 2001.

[30] B. Mobasher. Web usage mining. In B. Liu, editor, *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*, chapter 12. Springer Berlin-Heidelberg, 2006.

[31] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. In *Procs. of the ACM international conference on World Wide Web (WWW)*, pages 437–446, New York, USA., 2008.

[32] J. E. Pitkow. In search of reliable usage data on the www. *Computer Networks*, 29(8-13):1343–1355, 1997.

[33] P. E. Román. *Web User Behavior Analysis*. PhD thesis, University of Chile, January 2011.

[34] M. Spiliopoulou, B. Mobasher, B. Berendt, and M. Nakagawa. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. *INFORMS Journal on Computing*, 15(2):171–190, 2003.

[35] J. Srivastava, R. Cooley, M. Deshpande, and P. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 2(1):12–23, 2000.

[36] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehouse and Mining*, 3(3):1–13, 2007.

[37] A. Vazquez, J. Oliveira, Z. Dezso, K. Goh, I. Kondor, and A. Barabasi. Modeling bursts and heavy tails in human dynamics. *Physical Rewiew E*, 73(3), 2006.

[38] J. D. Velásquez and V. Palade. *Adaptive Web Sites: A Knowledge Extraction from Web Data Approach*. IOS Press, Amsterdam, NL, 2008.

[39] J. Zawodny and D. Balling. *High Performance MySQL*. O'Reilly, 2004.

[40] J. Zhang and A. A. Ghorbani. The reconstruction of user sessions from a server log using improved time-oriented heuristics. *Communication Networks and Services Research, Annual Conference on*, 0:315–322, 2004.