International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **4**, August **2018**

ICIC International ©2018 ISSN 1349-4198
pp. 1311–1326

ENHANCEMENTS TO GREEDY WEB PROXY CACHING ALGORITHMS USING DATA MINING METHOD AND WEIGHT ASSIGNMENT POLICY

Julian Benadit Pernabas[1] and Sagayaraj Francis Fidele[2]

[1]Department of Computer Science and Engineering
Faculty of Engineering
Christ University, Kengeri Campus
Mysore Road, Kanmanike, Kumbalgodu, Bangalore 560074, India
julian.p@christuniveristy.in

[2]Department of Computer Science and Engineering
Pondicherry Engineering College
Pillaichavadi, Puducherry 605014, India
fsfrancis@pec.edu

Received July 2017; revised February 2018

Abstract. *A Web proxy caching system is an intermediary between the users and servers that tries to alleviate the loads on the servers by caching selective Web objects and behaves as the proxy for the server and service the requests that are made to the servers by the users. In this paper the performance of a proxy system is measured by the number of hits at the proxy. A higher number of hits at the proxy server reflects the effectiveness of the proxy system. The number of hits is determined by the replacement policies chosen by the proxy systems. Traditional replacement policies that are based on time and size are reactive and do not consider the events that will possibly happen in the future. The outcomes of the paper are proactive strategies that augment the traditional replacement policies with data mining techniques. In this paper, the performances of the greedy replacement policies such as GDS, GDSF and GD* are adapted by the data mining method and weight assignment policy. Experiments were conducted on various data sets. Hit ratio and byte hit ratio were chosen as parameters for performance.*
Keywords: Web proxy caching, Classification, Fuzzy bi-clustering, Weight assignment policy

1. **Introduction.** The World Wide Web and its usage are growing at a rapid rate, which has resulted in overloaded Web servers, network congestion, and consequently poor response time. Multitudes of approaches are continuously being made to overcome these challenges. 'Web caching' is one of the approaches that can enhance the performance of the Web. A Web cache is a buffered repository of the Web objects that are most likely to be requested frequently and in the near future. The general architecture of the World Wide Web caching [1] consists of the client users, the proxy server, and the origin server. Whenever the client requests the Web object, it can be retrieved either from the intermediate proxy server immediately, or it can be retrieved from the origin server. Therefore, whenever a user's request is satisfied from the proxy server, it minimizes the response time and it reduces the overload of the Web origin server. Typically, the Web cache may be located at the origin server cache, at the proxy server cache, or at the client cache. The major goals of Web proxy cache are to reduce the user's response time by improving the hit ratio and to reduce the network traffic by improving the byte hit ratio. The overall

DOI: 10.24507/ijicic.14.04.1311

objective of the research work is to improve the performance of the greedy Web proxy caching algorithms such as Greedy Dual Size (GDS), Greedy Dual Size Frequency (GDSF) and Greedy Dual* (GD*) by augmenting the fuzzy bi-clustering based Web proxy caching using weight assignment policy mechanism. In Section 2, we address the related work of the traditional greedy Web proxy caching algorithms and classification based Web caching methods. Section 3 presents the working principle of Web proxy caching based on data mining method. Section 4 details the working flow method of Web proxy caching based on data mining method. Section 5 presents the generic model for Web proxy caching using data mining method. Section 6 details the experimental results of web proxy caching simulation. Section 7 provides the conclusions.

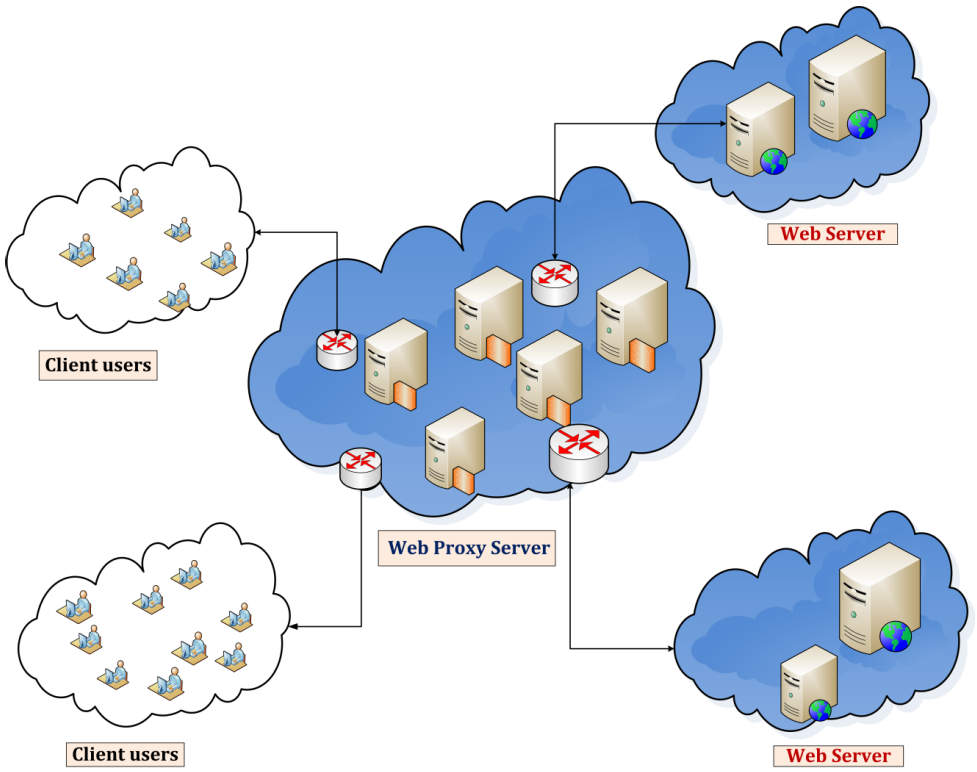

FIGURE 1. World Wide Web architecture

2. **Related Work.** This section summaries the traditional greedy Web proxy caching algorithm [1,2] based on the key parameters.

The algorithm for the traditional greedy replacement algorithm is shown in Figure 2. Several policies have been proposed in literature for augmenting the traditional Web caching methods with data mining techniques. In the following section, the data mining based Web caching methods are discussed.

Classification based Web caching methods. Khalid and Obaidat [3] proposed a new cache replacement algorithm called KORA (Khalid Obaidat Replacement Algorithm) to enhance the Web cache performance. The KORA algorithm performs well compared to the conventional algorithm by having a lower miss ratio. There has been significant work with respect to data mining and Web caching in the last decade. Tian et al. [4] used an adaptive Web cache access predictor using neural networks as one of the approaches to Web caching. In this approach, a Back-Propagation Neural Network (BPNN) is used to improve the performance of Web caching by predicting the most likely re-accessed objects. Koskela et al. [5] studied Web cache optimization with a non-linear model using object

TABLE 1. Summary of traditional Web proxy caching algorithms

S.no	Algorithms	Parameters	Evictions
1.	GDS	Object size S_p. Object cost C_p. Inflation Value L.	Least valuable objects with a key value. $k_p = L + \frac{C_p}{S_p}$.
2.	GDSF	Object size S_p. Object cost C_p. Inflation Value L. Number of non-aged references F_p.	Least valuable objects with a key value. $k_p = L + \frac{C_p \times F_p}{S_p}$.
3.	GD*	Object size S_p. Object cost C_p. Inflation Value L. Temporal correlation measures β. Number of non-aged references F_p.	Least valuable objects with a key value. $k_p = L + \left(\frac{C_p \times F_p}{S_p}\right)^{\frac{1}{\beta}}$.

/ Traditional Algorithm for GDS/GDSF/GD* replacement*/*
1. *begin*
2. *for* each Web object p requested do
3. *begin*
4. *if* Web object p resides in the proxy cache then */*Cache Hit*/*
   Update the Key Value of the Web object p as
   $$K_p = L + \frac{C_p}{S_p}. \quad \text{(4a) /*GDS*/}$$
   Update the Key Value of the Web object p as
   $$K_p = L + \frac{C_p \times F_p}{S_p}. \quad \text{(4b) /*GDSF*/}$$
   Update the Key Value of the Web object p as
   $$K_p = L + \left(\frac{C_p \times F_p}{S_p}\right)^{\frac{1}{\beta}}. \quad \text{(4c) /*GD*/}$$
5. *else*
6. *while* there is not enough free space in the proxy cache */*Cache Miss*/*
7. do $L \leftarrow \min_{q \in cache} K(q)$ from the proxy cache.
8. *Evict* q such that $K(q) = L$.
9. *end while*;
10. Bring Web object p into proxy cache and update the key value based on Equations (4a)-(4c).
end if;
11. *end*;
12. *end for*;
13. *end*;

FIGURE 2. Algorithm for GDS, GDSF and GD*

features. It utilizes the Multilayer Perceptron (MLP) network for predicting the value of the object based on the syntactic features of the HTML document. The strategy of Cobb and ElAarag [6] is based on Neural Network Proxy Cache Replacement (NNPCR) which integrates the neural network for the Web caching based on the sliding window mechanism. In this method, back propagation adjusts the weight factors in the network. An object is

selected for replacement based on the ratings returned by the Back Propagation Neural Network (BPNN).

Ali et al. [7,8] used intelligent Naïve-Bayes approach for Web proxy caching. In this method, the Naïve Bayes approach is used to classify the Web object, whether it can be re-accessed in the future or not. Benadit and Francis [9] improved the performance of a proxy cache using tree augmented Naïve Bayes approach followed by very fast decision tree algorithm for improving the Web proxy cache and data mining classification performance [9,10]. This method is integrated with traditional replacement algorithms LRU, GDS, GDSF and GD* to form a novel Web caching.

3. **Working Principle of Web Proxy Caching Based on Data Mining Method.** In this method, when a user requests a Web object, the proxy server checks its cache. If the Web object is available, the proxy server sends the Web object to the client and forwards the requests to the Request Processing Module (RPM) in order to predict the future request for the Web object as shown in Figure 3. Similarly, if the Web object is not available, the proxy server will forward the requests to the origin server and send it back to the clients and these requested Web objects are stored in the proxy's disk cache for future requests based on the Data Mining Method (DMM). The DMM is further explained in the following section. Therefore, if the Web object stored in the proxy cache is likely to be re-accessed again in the future, then this Web object will be retained in the proxy cache for future use. In addition, if there is no space in the proxy cache to store the required Web object from the origin server, then the request process module invokes the data mining method for the proxy cache replacement to remove the Web object that will not be re-accessed again in future. Similarly, the remaining Web objects, which is stored in the proxy cache are also updated by the data mining method for obtaining the future request.

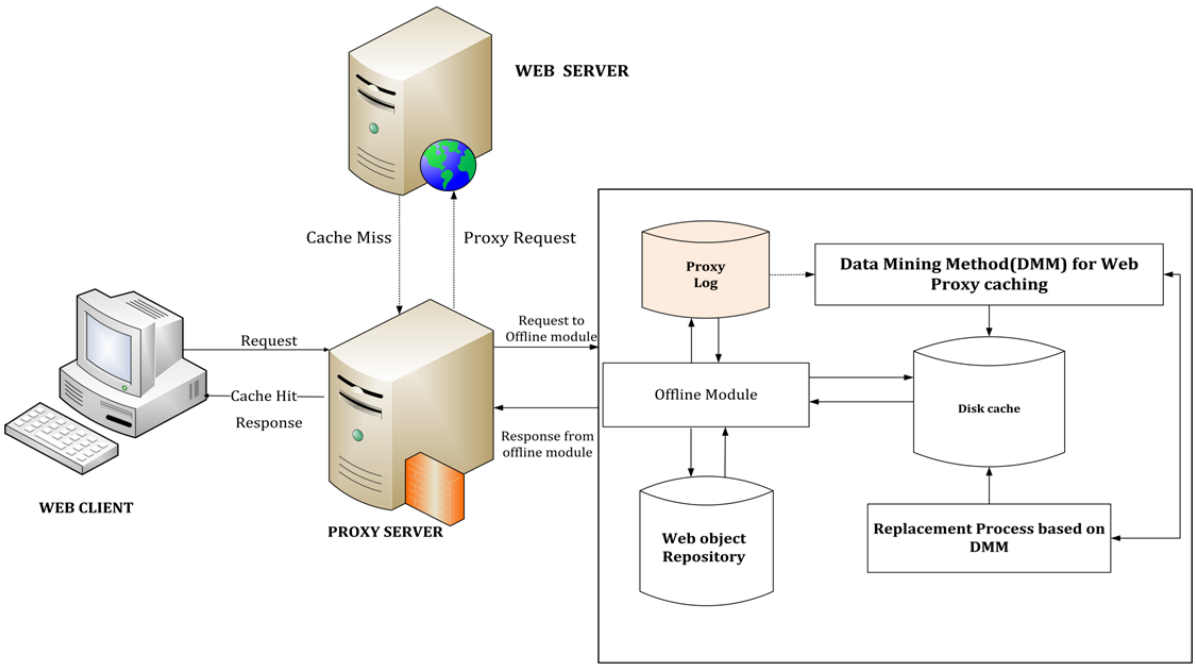

FIGURE 3. Working principle of Web proxy caching based on data mining method

4. **Working Flow Method of Web Proxy Caching Based on Data Mining Method.** Web proxy caching aims at enhancing the performance of the proxy server by increasing the hit and byte hit ratio. One class of strategies augments the traditional replacement policies with data mining technique. The strategy uses a data mining method based on the fuzzy bi-clustering model and weight assignment policy. The overall working flow model consists of different methods as shown in Figure 4. The Web log files for the Web proxy cache simulation are obtained from the National lab for Applied Network Research (NLANR) as shown in Table 2. Each data set represents a proxy server located in a particular location.

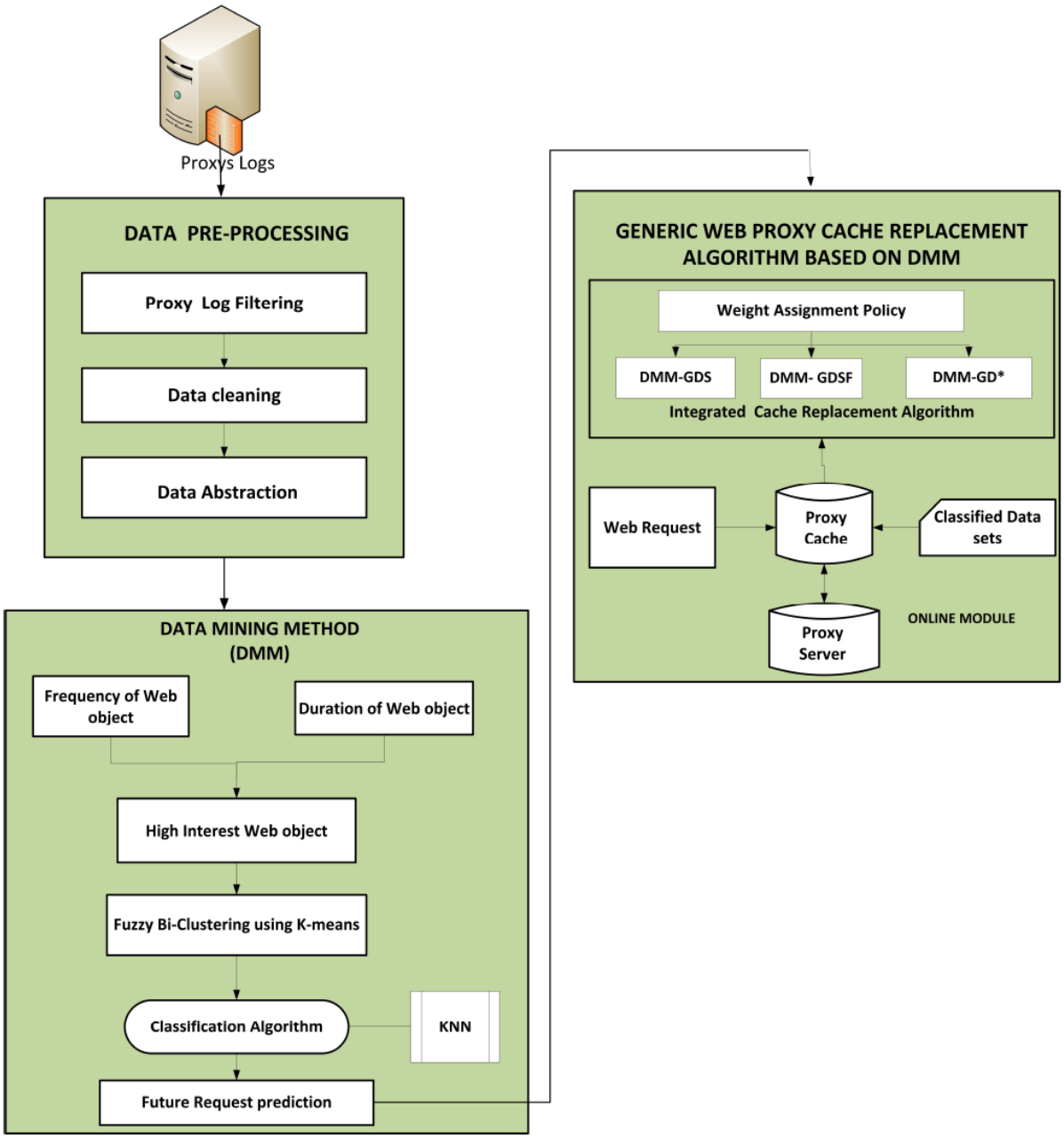

FIGURE 4. Overall working model for the Web proxy caching based on data mining method and weight assignment policy

TABLE 2. Statistics for the trace file

Trace File	Unique Requests	Unique Servers	Mean Object Size (bytes)	Median Object Size (bytes)
BOSTON	125,505	2,580	13,280	2,070
SILICON	935,630	27,674	5,822	2,238
UC	823,649	47,210	41,651	2,814
NY	674,352	59,139	17,440	209
SD	2,406,556	77,092	53,312	652

4.1. **Data pre-processing.** The datasets are pre-processed and converted into a structured format for reducing the time of simulation. The techniques undergo few pre-processing [11] steps to remove irrelevant requests, and to extract useful information. The steps involved in data pre-processing are proxy logs filtration, data cleaning, and data compression.

4.1.1. *Proxy logs filtration.* In this technique, the recorded proxy log files obtained from the NLANR have undergone the basic filtration process in order to reduce the size of the log data sets as well as the running time of the simulation. Thus, only the three proxies log data sets are considered for the filtration process [12] as shown in Figure 5. This filtration module is based on the methods such as a latency based method, SIZE based method, dynamic based method, content-based method, GET and ICP type method. Figure 5 illustrates the steps involved in various filtering methods and its details are described below. After these filtering, the size of the proxy log files is reduced in order to have some unique requests, which have been obtained after the filtering phases, i.e., 823,649 for the UC data sets, 674,352 NY data sets, and 2,406,556 SD data sets as shown in Table 2. From the filtration results obtained, the URL convention and the required HTTP requests are sorted based on the timestamp. Now the input trace file from UC, NY, SD is ready to undergo data pre-processing steps for the simulation of the data mining method.

4.1.2. *Data cleaning.* Data cleaning is the process of removing the irrelevant entries in the proxy log file [11]. Here only the relevant HTML files are considered and all other irrelevant log entries that were recorded by requesting graphics, sound, and other multimedia files are discarded.

4.1.3. *Data abstraction.* Data abstraction is the process of abstracting the log entries based on the session identification. The goal of session identification is to divide each user into different segments based on the user's pattern, which is called session. Session identification approaches [11] identify the user's session by a maximum time limit. Based on the empirical findings, the maximum time limit has been set to be 30 minutes [11] for our simulation. After the proxy datasets are pre-processed, it is further analyzed to obtain the common user features for Web user clustering.

4.2. **High-interest Web object set.** Once the data set has been preprocessed, it is segmented to find out the high-interest Web object, based on these two measures, i.e., *Frequency* and *Duration* [13]. Let P be the set of a Web object $P = \{P_1, P_2, P_3, \ldots, P_n\}$ accessed by the users in the proxy server logs. The parameters involved for obtaining the high-interest Web object is shown in Table 3. The frequency of the Web object p is calculated by the number of times the Web object p is accessed.

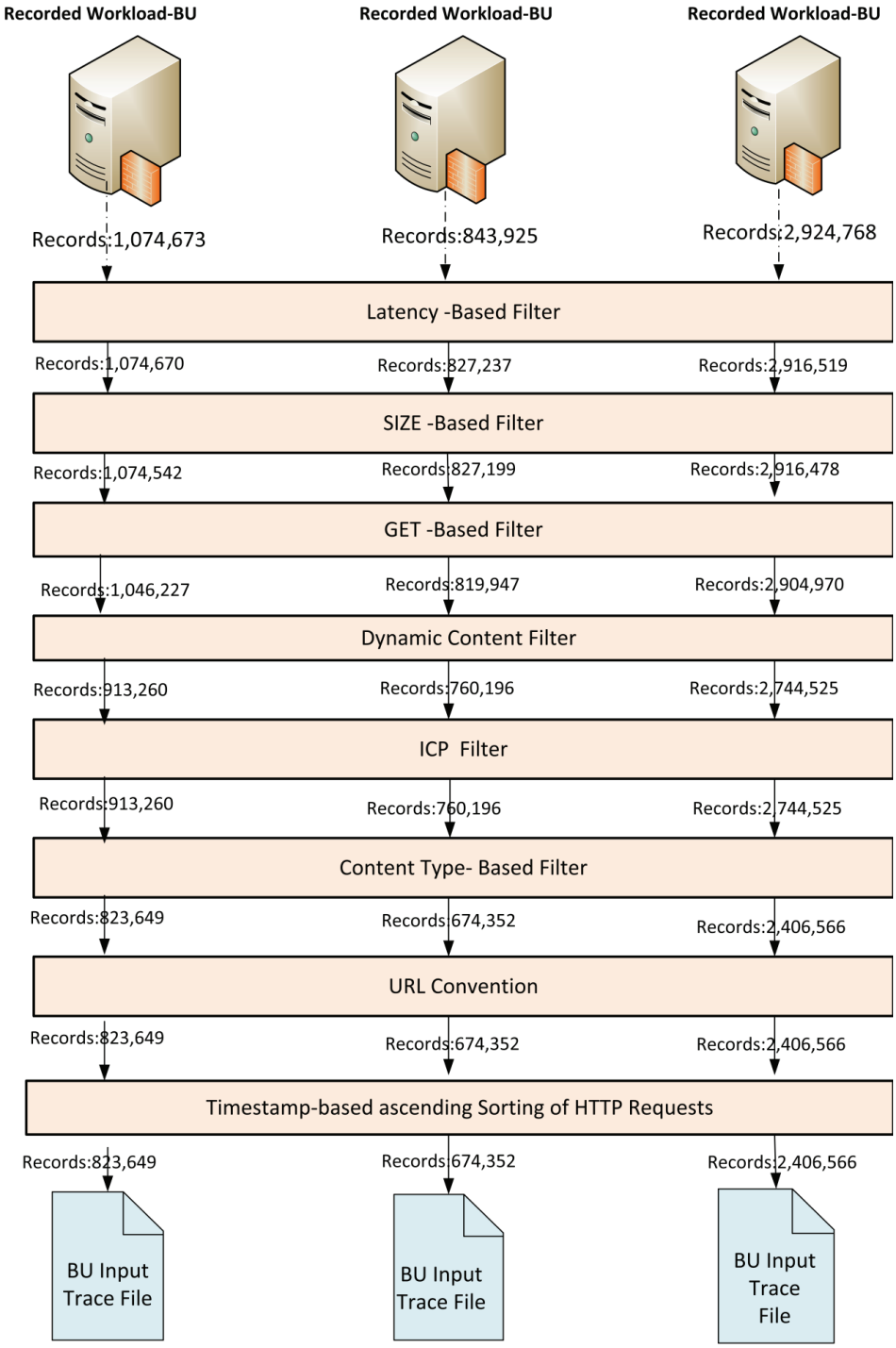

FIGURE 5. Filtering phases for the Web proxy datasets

TABLE 3. Parameters for high-interest Web object

Symbols	Parameters for High-Interest Web Object
p	Web object.
α	Frequency of the Web object p.
β	Duration of the Web object p.
$Size$	Size of the Web object p.
λ	High Interest of the Web object p.

The formula for the *Frequency* of the Web object p is given in Equation (1):

$$\alpha(p) = \frac{\text{Number of times}(p) \text{ accessed}}{\sum_{\text{WebPage} \in \text{Visited Web Pages}} (\text{Number of times}(p) \text{ accessed})} \tag{1}$$

Similarly, the *Duration* of the Web object p is given by Equation (2) as shown below:

$$\beta(p) = \frac{\text{TotalDuration of } (p)/\text{Size}(p)}{\sum_{\text{Max WebPage} \in \text{Visited Web Pages}} \text{TotalDuration}(p)/\text{Size}(p)} \tag{2}$$

From these two measures, we can obtain the high-interest Web object set and this high-interest object set value is normalized to 0 or 1 based on Equation (3) given below:

$$\chi(p) = \frac{2 \times \alpha(p) \times \beta(p)}{\alpha(p) + \beta(p)} \tag{3}$$

Once the high-interest Web object p is obtained from the URL, $P = \{URL_1, URL_2, \ldots, URL_m\}$ a navigation pattern profile is generated from a set of the individual user, which is given by $U = \{U_1, U_2, \ldots, U_n\}$ and these given URLs are segmented based on the user interest object set.

4.3. **Fuzzy bi-clustering model.** Once the data pre-processing step is completed, a Web usage mining technique has been incorporated. In our system we focus on the fuzzy bi-clustering algorithm based on the work of Koutsonikola and Vakali [14]. The authors used spectral clustering methods over traditional methods as its underlying implementation is simpler, and used well-defined linear algebra techniques. The advantage of adopting a fuzzy bi-clustering algorithm is that users grouped in the same users' cluster may be related to more than one web pages' cluster, which allows us to quantify users' interest to different web pages' clusters. Once the users have been clustered, a navigation pattern representative result is obtained. The parameters involved for fuzzy bi-clustering method are listed in Table 4.

Table 4. Parameters for fuzzy bi-clustering

Symbol	Description
n	Denotes the number of users
m	Denotes the number of objects
U	Denotes the users set $U = \{u_1, \ldots, u_m\}$
P	Objects set $P = \{p_1, \ldots, p_n\}$
$CP(C_x, P_y)$	Denotes the probability with which the users of cluster C_x visit the P_y page.
V	$n \times m$ users pattern matrix
PV	$n \times m$ probability distribution matrix
$f(C_x, C'_y)$	Denotes the relation between the obtained users and pages clusters
f	Function of relation degree

The algorithm for the clustering process is clearly depicted in Figures 6(a) and 6(b). The clustering process consists of three steps as follows.

1) The patterns of users' access are stored in the form of a probability distribution matrix. We have 2 vectors, $U = \{u_1, u_2, \ldots, u_n\}$ and $P = \{p_1, p_2, p_3, \ldots, p_m\}$ where 'U' denotes the set of n users and 'P' denotes the set of m pages that have been recorded. We generate a user visiting pattern for each user, which is a vector $V(u_i, :) = V\{(u_i, p_1), (u_i, p_2), \ldots, (u_i, p_m)\}$ where each element denotes the number of

Input: Given the set of 'n' users denoted as $U = \{u_1, \ldots, u_m\}$. Let K be the number of users clusters
Output: The sets $C = \{C_1, C_2, \ldots, C_K\}$ of K users' clusters
1. V = Create Visiting Patterns (U, P);
2. PV = Create Probability Distribution (V);
3. CS = Find Users Similarity (PV); // refer Equation (4).
4. $D(i,i) = \sum_{j=1}^{n} CS(u_i, u_j)$
5. L = Calculate Laplacian (D, CS);
6. EV = Eigen Vectors (L, K);
7. C = K-means (EV, K);

(a)

Input: Given the set of 'm' pages denoted as $P = \{p_1, \ldots, p_m\}$ and let K' be the number of page clusters respectively.
Output: The sets $C' = \{C'_1, C'_2, \ldots, C'_K\}$ of K' page clusters
1. CP = Cluster2 Pages (C, V);
2. SP = Find Pages Similarity (CP); //Refer Equation (5)
3. $D'(i,i) = \sum_{j=1}^{n} SP(p_i, p_j)$
4. L' = Calculate Laplacian (D', SP);
5. EV' = Eigen Vectors (L', K');
6. C' = K-means (EV', K');
7. $f(C_x, C_y)$ = Calculate Relation Coefficients (C, C', V);

(b)

FIGURE 6. (a) Fuzzy clustering algorithm for Web users; (b) fuzzy clustering algorithm for Web pages

times the user u_i has visited the particular page p_y where y is the index number of the element. The vectors are arranged in a 2-dimensional $u \times p$ users' pattern matrix. We obtain the probability distribution matrix by simply dividing each element in the row, with the sum of elements in that row, i.e., normalizing the row vector. The resultant matrix is now used to extract k users' clusters, by grouping together users with similar behavior. This is performed using cosine coefficient as shown in Equation (4).

$$CS(u_i, u_j) = \frac{PV(u_i, :) \cdot PV(u_j, :)}{|PV(u_i, :)| \cdot |PV(u_j, :)|} = \frac{\sum_{l=1}^{m} PV(u_i, p_1) \cdot PV(u_j, p_l)}{\sqrt{\sum_{l=1}^{m} PV(u_i, p_1)^2 \cdot \sum_{l=1}^{m} PV(u_j, p_l)^2}} \tag{4}$$

A weighted undirected graph that denotes the similarities between the n users is created. The degree and Laplacian matrix of the graph are computed, in addition to the eigenvectors of the Laplacian. Applying k-means clustering on the eigenvectors gives us the clusters.

2) The users' clusters are now used to perform the clustering of web pages. A $k \times p$ matrix is generated, with each element given by Equation (5)

$$CP(C_x, P_y) = \frac{\sum_{u_i \in C_x} V(u_i, p_y)}{\sum_{u \in C_x} \sum_{j=1}^{m} V(u_i, p_j)} \tag{5}$$

Each element specifies the probability of users in the particular will visit the page p. The generation of clusters is similar to the graphical method used in step 1.

3) Using the clusters obtained, we now develop the relation coefficients between the clusters using the relation function as shown in Equation (6).

$$f\left(C_x, C'_y\right) = \frac{\sum_{u_i \in C_x} \sum_{p_j \in C_y} V(u_i, p_j)}{\sum_{u_i \in C_x} \sum_{l=1} V(u_i, p_l)} \tag{6}$$

These coefficients give frequency of the web pages belonging to a particular cluster of a page, observed for each cluster of users.

4.4. **Clustering results.** The K-means clustering algorithm generates the common user's requests and groups these 100 users in the different Clusters Number (**C-N**) (Varying 1 to 9) based on the fuzzy bi-clustering model. After obtaining the Web user's requests, the individual clusters are evaluated based on the performance metrics as shown in Figure 7.

According to the clustered results obtained, the precision is defined as the ratio of hits to the number of URLs that are cached; similarly, recall is the ratio of hits to the number of URLs that are requested. Figure 7 signifies the clustering results based on the performance metrics.

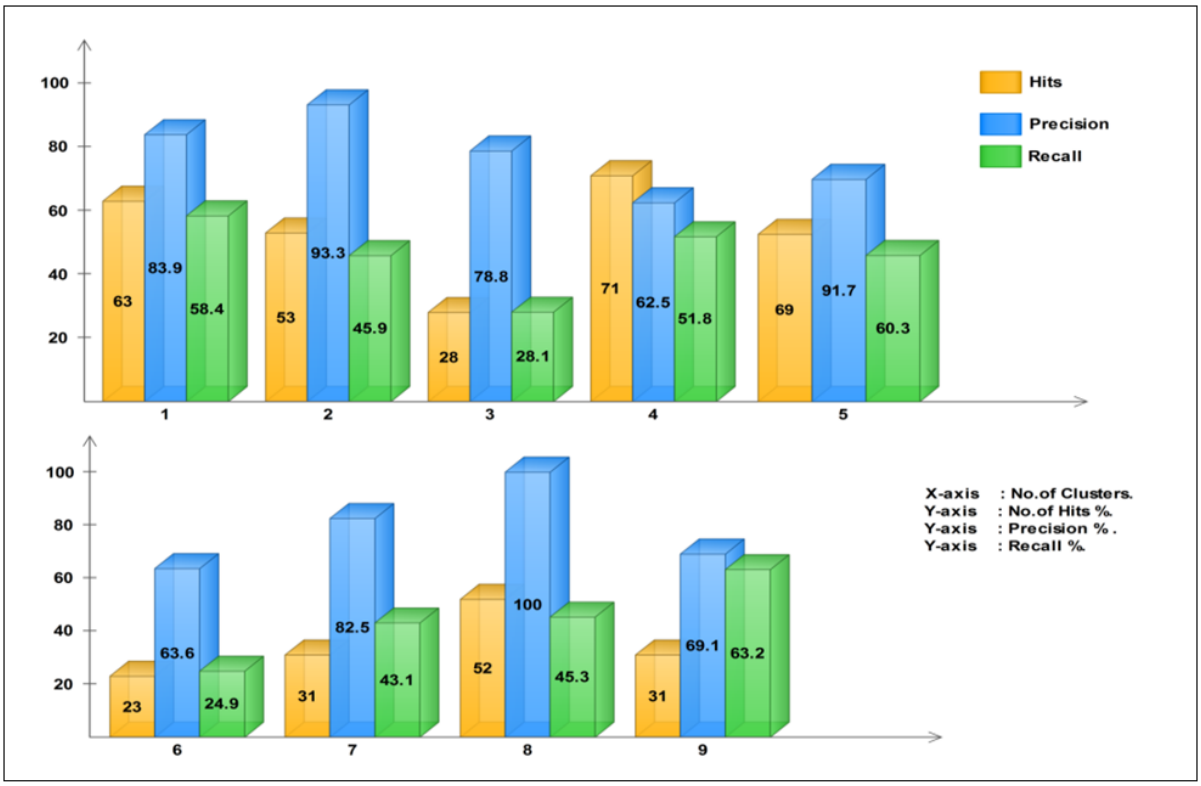

FIGURE 7. Results of precision, recall, and hits in clusters

4.5. **Data mining classification.** K-nearest neighbor classifier [13] is one of the supervised machine learning algorithms used in a variety of application. In this work, the K-nearest neighbor classifier works under the principle of distance measures. The KNN algorithm trains not only the data set but also the classification for each training example. This indicates that in the KNN algorithm, the training samples are used to build classification models.

4.5.1. *Classifier results.* The data mining classification (KNN) aims to evaluate the experimental results of classification. The evaluation of the classification is based on whether the Web page belongs to the class of common user navigation pattern profile. This navigation pattern profile is generated based on the high-interest page, the frequency of access, and duration of the Web page.

The different classifiers train these values whether the requested Web object belongs to a common user navigation pattern profile. So, if the requested Web object belongs to the common user profile pattern it is assigned as class one otherwise zero. Therefore, if the class value of the Web object 1, it indicates that Web object may request in future and this type of Web object is considered cacheable requests. In this section, different classifiers like Support Vector Machines (SVM), Decision Tree (J48), Naïve Bayes (NB) classifier and K-Nearest classifier (KNN), are used for classification purpose which is shown in Figure 8. So among the different classifiers, the KNN classifiers have better classification accuracy.

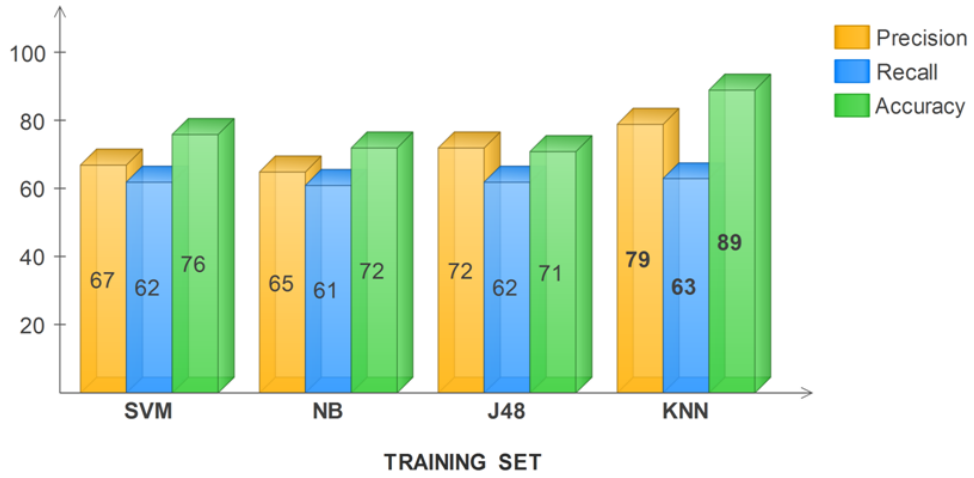

FIGURE 8. Comparisons of accuracy for data mining classifier in training sets

5. **Generic Model for Web Proxy Caching Using Data Mining Method.** In this section, a generic model for Web proxy caching strategies integrated with data mining method [15] was introduced. The algorithm for generic Web proxy caching algorithm integrated with data mining method is shown in Figure 9.

In (line 1), an initial data mining method is built on history Weblogs. For each Web object p requested from the proxy cache t that contains the Web object p and then the Web object p is returned to the Web client. Concerning this performance, measures (lines 7-11); in this case, it is considered as *Cache Hit*. In addition, the number of bytes transfer back to the client is counted for the weighted hit rate measure. Once the data are transferred back to the client, the proxy cache is updated (line 12) by the data mining method based on the weight assignment policy, (Class value of the Web object, i.e., whether the Web object p that can be revisited again in future or not). On the contrary, if the requested

```
Procedure Data Mining Methods (DMM)
Proxy cache entry t, t_fresh; int Hits = 0, Byte.Hits = 0;
int   Cache Max_Size N
1.  begin
2.     DMM.Build ();
3.        begin
4.           loop forever
5.              begin
6.                 do
7.                    Get request the Web object p from the proxy cache t.
8.                       if (Proxycache(t).Contains_fresh_copy (Web page p))
9.                          begin
10.                            Hits = Hits++.    /*Cache Hit*/
11.                            Byte.Hits = Byte.Hits + t. Bytes-Retrieved-to-client.
12.                            Cache.Update (t, (DMM))
13.                         end;
14.                      else
15.                         begin
16.                            Cache.Delete(t).    /*Cache Miss*/
17.                            Retrieve_Fresh Copy of Web object from origin server S.
18.                            Cache.Push (t, DMM)
19.                            While (Proxy cache (t).Size > Max_size (N))
20.                            Cache.Pop(t); */Remove the Web object with lowest key*/
21.                               Switch ()
22.                               Case 1: "DMM-GDS".
23.                               Case 2: "DMM-GDSF".
24.                               Case 3: "DMM-GD*".
25.                         end;
27.                   While (Condition);
28.             end;
29.    DMM.Update model();
30.  end;
```

FIGURE 9. Generic model for Web proxy caching replacement algorithms using DMM

Web object is not available in the proxy cache or it is stale, a cache miss occurs, i.e., the Web object is deleted from the cache (line 16), in this case, the proxy server forwards the request to the origin server (line 17), and a fresh copy of the Web object is retrieved from the origin server and pushed into the proxy cache (line 18). The push method consists of assigning the class value of the Web object by the data mining method based on the weight assignment policy. In addition, if the cache space t exceeds the maximum cache size N (line 19), the Web object q from the cache is popped out from the cache (line 20), based on the class assigned and lowest key value based on the weight assignment policy by the data mining methods. Such an approach is known as weight assignment cache replacement policy (lines 21-24), i.e., each time when the cache gets overflows. Finally, the data mining method periodically updates the key value of the remaining Web object's stored in the proxy cache t. This process continues iteratively when the cache performance decreases. Also, notice that update of the data mining method (line 28), is dissociated from the online caching of the Web object, and it can be performed in parallel.

5.1. **Weight assignment policy for cache replacement.** The weight assignment policy of the Web object p in the proxy cache t is expressed in Equation (7) and the parameters are shown in Table 5. From the above strategy, the key value used in the caching system is applied to the greedy family replacement algorithm (GDS, GDSF, GD*) and the key factor of the replacement algorithm is modified according to Equation (7) as shown below.

$$K_n(p) = L + \frac{F(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)} \tag{7}$$

Therefore, whenever the cache replacement occurs, the replacement algorithm replaces the Web object based on the key value used by the weight assignment policy.

TABLE 5. Parameters for weight assignment policy

Parameters	Description
L	Inflation factor to avoid cache pollution in the proxy cache t.
$f(p)$	Previous frequency access of Web object p in the proxy cache t.
$F(p)$	Current frequency access of Web object p in the proxy cache t.
$K_{n-1}(p)$	Previous key value of the Web object p in the proxy cache t.
$\Delta T_t(p)$	Difference in time between the current requests and previous requests for the Web object p in the proxy cache t.
$C_t(p)$	Current reference time of the Web object p in the proxy cache t.
$L_t(p)$	Last reference time of the Web object p in the proxy cache t.
$S(p)$	Size of the Web object p.
$K_n(p)$	Current key value of the Web object p.

5.2. **Integration of GDS/GDSF/GD* replacement with the DMM.** The respective policy is adapted with data mining method when there is a need for cache replacement. GDS considers variability in cost and size of a Web object p by choosing the victim based on the ratio between the cost and size of documents. GDSF considers variability in cost, size and additionally, the frequency of a Web object p by choosing the victim based on the ratio between the cost and size of documents. GD* captures both popularity and temporal correlation in a Web object. The chosen policy associates a key value $K(p)$ with each Web object p in the cache. When a Web object p is requested in the proxy cache t and it is already available in the proxy cache t then *cache hit* occurs and the Web object p is pushed onto the top of the cache, also the key value is updated based on the DMM and weight assignment policy of the caching system, $K(p)$, i.e., is respectively set to the equations below. Equation (8) corresponds to GDS, Equation (9) to GDSF, and Equation (10) to GD*.

$$K_n(p) = L + \frac{F(p) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)} \tag{8}$$

$$K_n(p) = L + \frac{(F(p) + f(p)) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)} \tag{9}$$

$$K_n(p) = L + \left[\frac{(F(p) + f(p)) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)}\right]^{\frac{1}{\beta}} \tag{10}$$

Similarly, if a *cache miss* occurs and the Web object p is retrieved from the origin server S and if there is no space in the proxy cache the Web object q has to be replaced based on the DMM. The lowest key value assigned by the weight assignment policy of the caching system, i.e., Web object q with minimum key value $\min_{q \in cache}\{k(q)|q\}$ in the cache is chosen among the other Web objects resident in the proxy cache t. Subsequently, the values are reduced by $k_{\min}$ and the key value of the Web object p is updated as shown in the policy's corresponding equation and it is pushed to the top of the cache, and also the data mining classifier model updates the remaining Web objects. The algorithm for the GDS, GDSF and GD* replacement based on weight assignment policy and DMM is given in Figure 10.

6. **Experimental Results for the Web Proxy Cache Simulation.** For the simulation of the Web proxy cache algorithm, the Windows-based cache simulators [16] are

1. *begin*
2. *for* each Web object p requested by user do
3. *begin*
4. *if* Web object p resides in the proxy cache t then /**Cache hit**/.
5. *begin*
6. Update information of p by DMM.
7. Class of p = DMM.
8. *if* Class of Web object $p = 1$ then
9. *Push* the Web object p into the top of cache and update the key value for GDS, GDSF, and GD*

$$K_n(p) = L + \frac{F(p) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)} \tag{9a}$$

$$K_n(p) = L + \frac{(F(p) + f(p)) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)} \tag{9b}$$

$$K_n(p) = L + \left[\frac{(F(p) + f(p)) \times C(p) + K_{n-1}(p) \times \left(\frac{\Delta T_t(p)}{C_t(p) - L_t(p)}\right)}{S(p)}\right]^{\frac{1}{\beta}} \tag{9c}$$

10. *end*;
11. *else*
12. *begin*
13. *if* Web object p is not available in the proxy cache t
14. Bring the Web object p into proxy cache t from the origin server S.
15. *While* there is not enough free space in the proxy cache t.
16. Remove $\min\{k(q)|q\}$ from the proxy cache t && Class of $q = 0$.
17. *end While*;
18. Class of p = DMM
19. *if* Class of Web object $p = 1$.
20. *Push* the Web object p in to the top of cache and update the key Value for GDS, GDSF, GD* based on Equations (9a)-(9c).
21. *end*;
22. *end*;
23. *end*;

FIGURE 10. GDS, GDSF, GD* replacement algorithms based on DMM

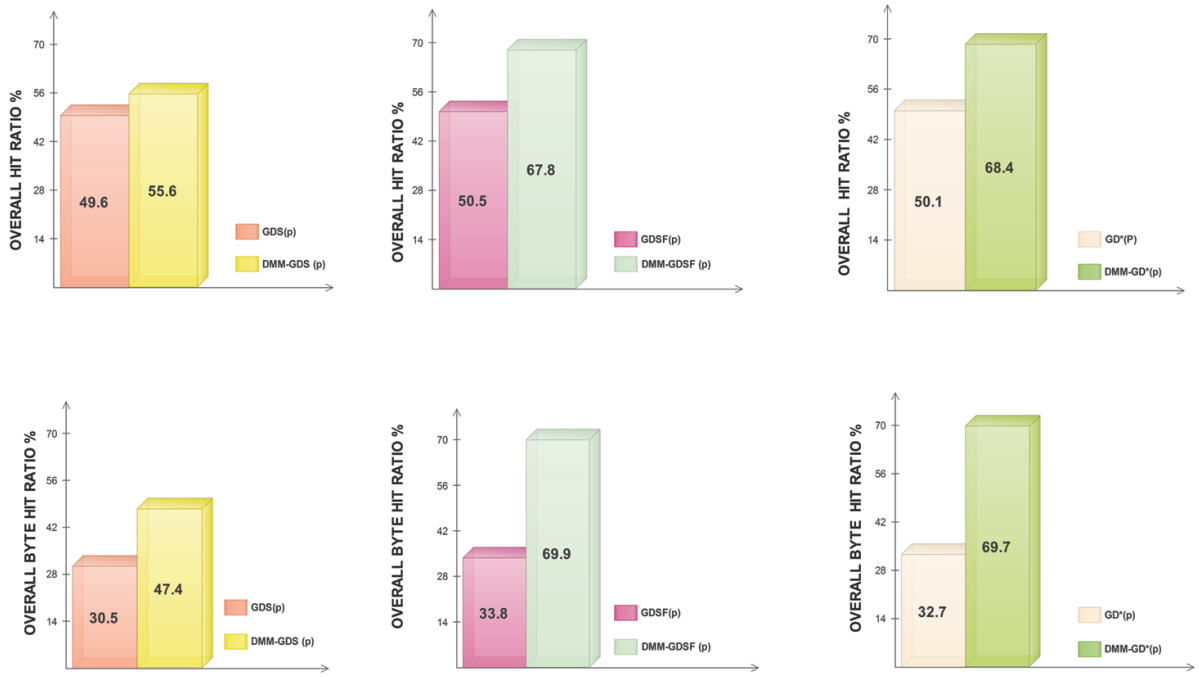

FIGURE 11. Comparison of hit ratio and byte hit ratio of GDS, GDSF and GD* using DMM and weight assignment policy

modified for the integration of data mining classifier model. Results obtained from the classifier are taken as input to the Web proxy cache simulator.

A Windows-based cache simulator was used to run the experiments. The experimental setup is carried out based on parameters like the trace file name, cache size, replacement scheme and the content type used. The trace file name includes the following attributes timestamp, URL-ID, object size, etc. The cache size used in these experiments may vary in size from 5% to 45% (Maximal volume of the cached content) and the replacement scheme used, GDS, GDSF, GD*. From the experimental results it is shown that the performance of hit and byte hit ratio for the DMM based Web proxy caching algorithms outperforms the traditional caching algorithms in all aspects. Here the experimental results are graphically illustrated and the results are given below.

7. **Conclusion.** The various working modules of the overall working flow such as the data pre-processing, fuzzy bi-clustering model, data mining method and generic model for Web proxy caching algorithms based on weight assignment policy were described in detail. The data mining method outperforms by improving the classification accuracy by adapting fuzzy bi-clustering model and also improves the performance of the greedy Web proxy cache algorithm based on weight assignment policy.

REFERENCES

[1] A. Balamash and M. Krunz, An overview of Web caching replacement algorithms, *IEEE Communications Surveys and Tutorials*, vol.6, no.12, pp.44-56, 2004.

[2] S. Jin and A. Bestavros, Greedy-dual* Web caching algorithm, *International Journal of Computer Communication*, vol.24, pp.174-183, 2001.

[3] H. Khalid and M. Obaidat, KORA: A new cache replacement scheme, *Computers and Electrical Engineering*, vol.26, no.3, pp.187-206, 2000.

[4] W. Tian, B. Choi and V. Phoba, An adaptive Web cache access predictor using network, *Developments in Applied Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol.2358, pp.450-459, 2002.

[5] T. Koskela, J. Heikkonen and K. Kaski, Web cache optimization with non-linear model using networks object features, *Computers Networks*, vol.43, no.4, pp.805-817, 2003.
[6] J. Cobb and H. ElAarag, Web proxy cache replacement scheme based on back-propagation neural network, *Journal of System Software*, vol.6, no.3, pp.805-817, 2003.
[7] W. Ali and S. M. Shamsuddin, Neuro-fuzzy system in partitioned client-side Web cache, *Expert Systems with Applications*, vol.38, no.12, pp.14715-14725, 2011.
[8] W. Ali, S. M. Shamsuddin and A. S. Ismail, Intelligent Naïve Bayes-based approaches for Web proxy caching, *Knowledge-Based System*, vol.31, pp.162-175, 2012.
[9] P. J. Benadit and F. S. Francis, Improving the performance of a proxy cache using very fast decision tree classifier, *Procedia Computer Science*, vol.48, pp.304-312, 2015.
[10] P. J. Benadit, F. S. Francis and U. Muruganantham, Enhancement of Web proxy caching using discriminative multinomial Naïve Bayes classifier, *International Journal of Information and Communication Technology*, vol.11, pp.369-381, 2017.
[11] R. Cooley, B. Mobasher and J. Srivastava, Data preparation for mining World Wide Web browsing patterns, *Knowledge and Information Systems*, vol.1, no.1, pp.5-32, 1999.
[12] G. Kastaniotis, E. Maragos, C. Douligeris and D. K. Despotis, Using data envelopment analysis to evaluate the efficiency of Web caching object replacement strategies, *Journal of Network and Computer Applications*, vol.35, no.2, pp.803-817, 2012.
[13] H. Liu and V. Keselej, Combined mining of Web server logs and Web contents for classifying user navigation patterns and predicting users' future requests, *Data and Knowledge Engineering*, vol.61, pp.304-330, 2007.
[14] V. A. Koutsonikola and A. Vakali, A fuzzy bi-clustering approach to correlate Web users and pages, *International Journal of Knowledge and Web Intelligence*, vol.1, pp.3-23, 2010.
[15] F. Bonchi et al., Web log data ware housing and mining for intelligent Web caching, *Data and Knowledge Engineering*, vol.39, pp.165-189, 2001.
[16] F. J. Gonzalez-Cante, E. Casilari and A. Trivino-cabrera, A Windows based Web cache simulator tool, *Proc. of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, pp.1-5, 2008.