Artificial Immune System for Bloom filter Optimization

Arulanand Natarajan
Anna University of Technology
Coimbatore
Tamil Nadu, India

Swathy Priyadharsini P
Bannari Amman Institute of
Technology, Sathyamangalam
Erode, Tamil Nadu, India

Subramanian S
Sri Krishna College of
Engineering and Technology
Coimbatore, Tamil Nadu, India

ABSTRACT

Bloom filter is a probabilistic and space efficient data structure designed to check the membership of an element in a set. The trade-off to use Bloom filter may have configurable risk of false positives. The percentages of a false positive can be made low if the hash bit map is sufficiently massive. Spam is an unsolicited or irrelevant message sent on the internet to an outsized range of users or newsgroup. A spam word may be a list of well-known words that usually appear in spam mails. In the proposed system, Bin Bloom Filter (BBF) groups the words into number of bloom filters that have different false positive rates primarily based on the weights of the spam words. Clonal Selection Algorithm is one of the methods in Artificial Immune System (AIS) involved with computational methods inspired by the process of the biological immune system. This paper demonstrates the CSA algorithm for minimizing the total membership invalidation cost of the BBF which finds the optimal false positive rates and number of elements to be stored in bloom filters of Bin. The experimental results demonstrate the application of CSA in BBF and compare the results with Genetic Algorithm (GA).

General Terms

Genetic Algorithm, E-Mail, Data Structure, Immune System, Optimization, Spam word, Internet, Synthetic Dataset, Membership Query.

Keywords

Clonal Selection Method, Bloom filter, Spam filter, False positive rate, Hash function.

1. INTRODUCTION

Spamming is the action of sending inappropriate messages in bulk without the explicit permission or need of the recipients. That is spam is an unwanted or unsolicited messages sent on the internet to a large number of users. This huge numbers of spam raise the requirement of automated filters to detect or remove these spam e-mails. The spam does cost money. Bayesian filters or other heuristic filters attempt to identify spam through word frequency. The Bayesian filter is based on the occurrence of words in spam mails and legitimate mails. Content based filters can classify the mail into spam or legitimate by examining within the e-mail contents and these mostly seek for spam words.

A Bloom filter [1] is a memory efficient data structure for representing a set in order to support membership queries. Bloom filters enable false positive but it saves space. The bloom filter is a widespread I database applications and networking. In this paper, CSA is used for optimizing the bloom filter in spam filtering. The clonal selection acquires the behavior and capabilities of antibodies in the immune system.

2. BLOOM FILTER

Bloom filter has a bit vector of size m and for any given string X, Bloom filter computes k hash functions on it which produces k hash values ranging from 1 to m. It then sets k bits in an m-bit vector at the addresses corresponding to the k hash values. This similar procedure is repeated for all the members of the set. This process is called programming of the filter. The query process is analogous to programming, where a string whose membership is to be verified is given as input to the filter. Initially,

Bloom filter generates k hash values using those hash functions which is used to program the filter. The bits that are corresponding to the k hash values in the m-bit vector are looked up. If any one of the bits is not set, then the string is asserted to be a nonmember of the set. If all the bits are found to be set, then the string is said to be present in the set with certain probability. This uncertainty in the membership of an string is because of the fact that those k bits in the m-bit vector may be set by any other n-1 members. Thus finding a bit set does not necessarily imply that it had been set by the actual string being queried. However, finding any of the bits not set implies that the string does not belong to the set.

In an Bloom filter B, initially all the bits are set to 0 and k independent hash functions h1, h2, …, hk are chosen. For any element e to be added in the Bloom filter from the set S, the bit positions h1(e), h2(e), …, hk(e) in B are set to 1. A specific bit may be set to 1 more than one time. For testing the membership of an element o the bits at positions h1(o), h2(o), …, hk(o) are checked. If any one of them is 0, then certainly o is not present in the set S. Otherwise, o may be present in the set with a certain probability that it is wrong. This is called as false positive. The parameters k and m should be chosen such that the likelihood of a false positive is acceptable. Figure 1 is an example where m=16, k=4 and e is the element to be stored in the bit array.

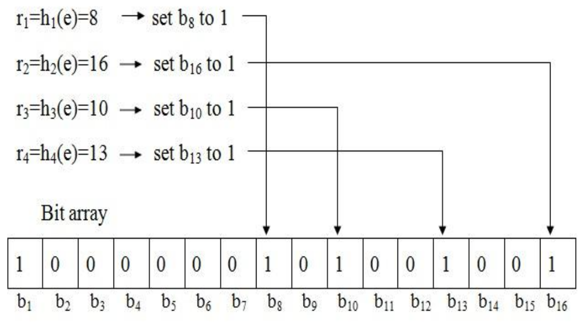

$r_1=h_1(e)=8$ → set b_8 to 1
$r_2=h_2(e)=16$ → set b_{16} to 1
$r_3=h_3(e)=10$ → set b_{10} to 1
$r_4=h_4(e)=13$ → set b_{13} to 1
Bit array
1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16}

Fig 1: Bloom Filter

The false positive rate of Bloom filter is given in equation (1)

$$f = (1 - e - kn/m)k \qquad (1)$$

For given m and n, the optimal value of k is obtained from the equation (2)

$$k = (m/n)\ln(2) \qquad (2)$$

The BF has been widely used in many database applications ([2] [3]). It is applied in networking literature [4]. A BF can be used as a summarizing technique to aid global collaboration in peer-to-peer networks ([5] [6] [7] [8]). It supports probabilistic algorithms for routing and locating resources ([9] [10] [11] [12]) and share Web cache information ([13]). BFs have great potential for representing a set in main memory [14] in stand-alone applications. BFs have been used to provide a probabilistic approach for explicit state model checking of finite-state transition systems [14]. It is used to summarize the contents of stream data in memory ([15] [16]), to store the states of flows in the on-chip memory at networking devices [17], and to store the statistical values of tokens to speed up the statistical-based Bayesian filters [18]. The variations of BFs are compressed Bloom filters [19], counting Bloom filters [13], distance-sensitive Bloom filters [20], Bloom filters with two hash functions [21], spacecode Bloom filters [22], spectral Bloom filters [23], generalized Bloom filters [24], Bloomier filters [25], Bloom filters based on partitioned hashing [26], fully pipelined bloom filter architecture [27] and dynamic Bloom filters [28].

3. IMMUNE SYSTEM

The immune system is a network of cells, tissues, and organs that work together to defend the body against attacks by foreign invaders. The primary role of the immune system is detection and elimination of pathogen where as pathogen is the agent of disease. Antigen is a foreign molecule that triggers the production of antibody. An antigen can be a microbe such as a virus, or a part of a microbe such as a molecule. The organs of the immune system are positioned throughout the body. White blood cells are the key players in the immune system.

White blood cells originate in the bone marrow however migrate to parts of the lymphatic system such as the lymph nodes, spleen, and thymus. There are two main types of lymphatic cells, T cells and B cells. The B cell creates new cell types called plasma cells and B memory cells during cloning. The plasma cell is specialized in producing a specific protein, called an antibody that will respond to the same antigen that matched the B cell receptor. When the antibody finds a matching antigen, it attaches and demolishes the antigen. The memory cells are another cell type produced by the division of B cells. These cells have a prolonged life span and remember specific intruders.

3.1 Clonal Selection Algorithm

The clonal expansion of B-cells will increase the average antibody affinity for the antigen which triggered the clonal expansion. This is termed as affinity maturation. Affinity maturation is caused by somatic hypermutation and selection mechanism which occurs during the clonal expansion of B-cells. The two important features of affinity maturation in B-cells are

- The proliferation of B-cells is proportional to the affinity of the antigen that binds it, thus the higher the affinity, the more clones are produced.
- The mutations suffered by the antibody of a B-cell are inversely proportional to the affinity of the antigen it binds.

Somatic hypermutation alters the specificity of antibodies by introducing random changes to the genes that encode for them. In proliferation it is differentiated into memory cells and plasma cells. The memory cells are used to fight with the same pathogen if the host is infected again. The plasma cells are used to secrete lots of the recognizing antibody into the blood. Figure 2 shows the Clonal selection method.

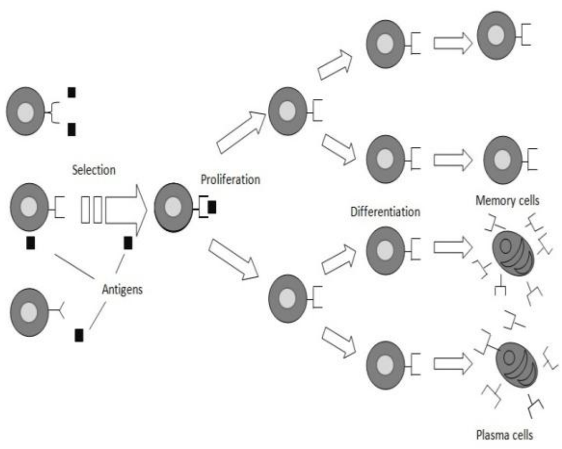


Fig 2: Clonal selection method ([29])

In CSA, antigen typically refers the problem and its constraints. The antibodies are selected which is based on the affinity either by matching against an antigen pattern or via evaluation of pattern by a cost function. Selected antibodies are subjected to cloning proportional to affinity. The hypermutation of clones are inversely proportional to clone affinity. The resultant clone set competes with the existent antibody population for membership in the next generation. In addition low-affinity population members are replaced by randomly generated antibody population for the membership in next generation.

The main immune aspects taken into account are maintenance of the memory cells functionally disconnected from the repertoire, selection and cloning of the most stimulated cells, death of non-stimulated cells, affinity maturation and reselection of the clones with higher affinity, generation and maintenance of diversity, hypermutation proportional to the cell affinity.

Algorithm: Pseudo code for CSA

1. Generate a set (P) of candidate solutions composed of the subset of memory cells (M) added to the remaining (Pr) population (P = Pr + M).
2. Select the n best individuals of the population based on a fitness function.
3. Clone these n best individuals of the population, giving rise to a temporary population of clones (C).
4. Submit the population of clones to a hypermutation scheme, where the hypermutation is proportional to the fitness function of the antibody. A maturated antibody population is generated (C*).
5. Re-select the improved individuals from C* to compose the memory set M. Some members of P can be replaced by other improved members of C*

Replace d antibodies by novel ones. The lower affinity cells have higher probabilities of being replaced.

4. OPTIMIZATION OF BBF USING CSA

A Bin Bloom Filter is a variant of Bloom filter where it has number of Bloom filters as bins. Each Bloom filter has different <n,k,f> which causes dissimilar membership invalidation cost. The proposed system applies the concept of BBF for spam filtering. In spam filtering each word is assigned by a weight. BBF provides each bin with different false positive rates based on the weights of the strings. The false positive rate and number of words to be stored in each bloom filter is identified through optimization technique which minimize the total membership invalidation cost. Figure 3 shows Bin BF with its tuple <n,f,w> configuration.

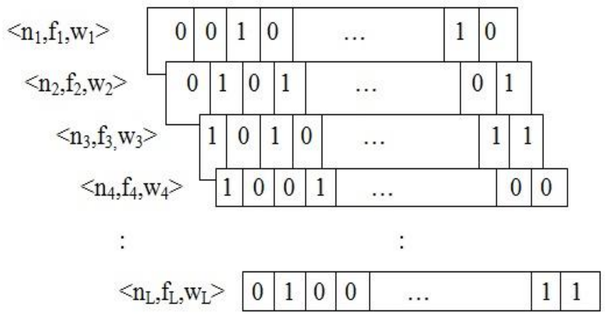


Fig 3: Bin Bloom Filter

where ni, fi and wi respectively represents number of strings, false positive rate and average weight of strings in Bloom filter i.

4.1 Problem Definition

Consider a standard supervised learning problem with a set of training data D = {<Y1,Z1 >,..., <Yi, Zi>, … ,< Yr ,Zr >} , where Yi is an instance represent as a single feature vector, Zi = C(Yi) is the target value of Yi , where C is the target function. Where Y1, Y2, … , Yr set of text document collection, C is a class label to classify into spam or legitimate (non-spam).

The text documents are preprocessed and features (words) are identified. The spam weights for words are calculated from the set. This weight value indicates its probable belongings to spam or legitimate. The weight values are discretized and assigned for different Bins. The tuple to describe the Bin BF is, {{n1, n2,, …, nL}, {w1, w2,…, wL}, m, {k1, k2, …, kL}, {f1, f2, …, fL}}. It is an optimization problem to find the value of n and f that to minimize the total membership invalidation cost. For membership testing the total cost of the set is the sum of the invalidation cost of each subset. The total membership invalidation cost [30] is given as,

$$F = n_1 f_1 w_1 + n_2 f_2 w_2 + ... + n_L f_L w_L$$

The total membership invalidation cost

$$F(L) = \sum_{i=1}^{L} n_i f_i w_i \qquad (3)$$

to be minimized. Where $\sum_{i=1}^{L} n_i = N$

N- Total number of Strings in a spam set.

$$f_i = \left(\frac{1}{2}\right)^{\ln 2 \times \left(r_i m \Big/ \sum_{j=1}^{i} n_j r_j \right)} \qquad (4)$$

$$r_i = \ln(f_i) \quad (1 \le i \le L)$$

The objective function f(L) taken as standard for the problem of minimization is

$$f(L) = \begin{cases} C_{max} - F(L) & \text{if } F(L) < C_{max} \\ 0 & \text{if } F(L) \ge C_{max} \end{cases} \qquad (5)$$

where C_{max} is a large constant.

4.2 Weight Assignment

The first step for assigning weight to spam words is estimating the word probability that depends on word frequency. Word frequency is measured by the number of occurrences of a specific word in the document. Estimating probabilities is achieved using Bayes conditional probability theorem. The probability of a word given that the message is spam can be estimated as follows:

$$P_s = \frac{\dfrac{f_s}{N_s}}{\dfrac{f_{ns}}{N_{ns}} + \dfrac{f_s}{N_s}}$$

$$P_{ns} = \frac{\dfrac{f_{ns}}{N_{ns}}}{\dfrac{f_{ns}}{N_{ns}} + \dfrac{f_s}{N_s}}$$

P_s is the probability of a word given the mail is spam,

P_{ns} is the probability of a word given the mail is legitimate,

f_s is the frequency of word in the spam documents,

f_{ns} is frequency of words in the legitimate documents,

N_s is the total spam documents,

N_{ns} is the total legitimate documents.

The next step is calculating word weights. Estimating a weight for each word is based on its frequency and its probability in spam mail documents and non-spam mail documents. The weight of every word is estimated using the formula:

$$Weight_{word} = \frac{P_s}{P_{ns}} \qquad (6)$$

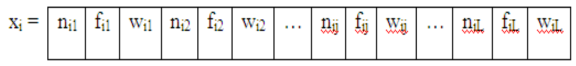
This weight value is based on text collection containing spam messages and non-spam messages. The word weights are estimated from spam list during the training process and stored in a separate text document.

4.3 Antibody representation

The three main features of clonal selection are

- Proliferation and differentiation on stimulation of cells with antigens
- Generation of new random genetic changes, subsequently expressed as diverse antibody patterns by somatic mutation
- Elimination of newly differentiated lymphocytes carrying low affinity antigenic receptors.

In the perspective of BBF, an antibody represents number of bloom filters with number of words, false positive rate and weight. That is, each antibody X_i, is constructed as follows:



$x_i =$	n_{i1}	f_{i1}	w_{i1}	n_{i2}	f_{i2}	w_{i2}	…	n_{ij}	f_{ij}	w_{ij}	…	n_{iL}	f_{iL}	w_{iL}

Fig 4: Antibody representation for Bin Bloom Filter

where n_{ij}, f_{ij} and w_{ij} refer respectively the number of words, false positive rate and weight of the jth Bloom filter of ith antibody in a bin. The false positive rate f_{ij} can be obtained from equation (1).

4.4 Initial Population

An antibody in the population represents one possible solution for assigning the triples <n, f, w> for L Bloom filters. Therefore P number of candidate solutions is generated for the bin. Initially each antibody randomly chooses different <n, f, w> for L bins.

The fitness function for each individual can be calculated based on the equation (5).

4.5 Clonal selection

In clonal selection n highest fitness antibodies are selected for cloning with the rate of β. The amount of clones to be generated for all these n selected antibodies is given in equation (7).

$$N_C = \sum_{i=1}^{n} \operatorname{round}\left(\frac{\beta P}{i}\right) \qquad (7)$$

where N_C is the total amount of clones generated, β is a multiplying factor, P is the total amount of antibodies and round() is the operator that rounds its argument towards the closest integer. Each term of this sum corresponds to the clone size of each selected antibody, e.g., for P = 10 and β = 1, the highest affinity antibody (i = 1) will produce 10 clones, while the second highest affinity antibody produces 5 clones, and so on.

4.6 Somatic hypermutation

A speedy accumulation of mutations is necessary for a fast maturation of the immune response. The selection mechanism provides a means by which the regulation of the hypermutation process is made dependent on receptor affinity. Cells with low affinity receptors may be further mutated and die if they do not improve their clone size or antigenic affinity. In cells with high-affinity antibody receptors the hypermutation become inactive in a gradual manner [31].

For mutation the Cauchy mutation operator is applied. The one dimensional Cauchy density function centered at the origin is defined as follows:

$$f(x) = \frac{1}{\pi}\frac{t}{t^2 + x^2} \qquad -\infty < x < \infty \qquad (8)$$

where t>0 is a scale parameter.

The adaptive mutation rate P_m depends on the fitness values of the Antibodies. The adaptation allows the individuals having fitness values of over-average to maintain their value and the individuals with below average fitness values to disturb. The mutation rate adaptation rule is given in equation (9).

$$P_m = \begin{cases} k_1 \times \dfrac{F_{max} - F}{F_{max} - F_{avg}} & F \geq F_{avg} \\ k_2 & F < F_{avg} \end{cases} \qquad (9)$$

In this equation, F denotes the fitness value of the individual, F_{max} denotes the best fitness value of the current generation, and F_{avg} denotes the average fitness value of the current generation. The constants k_1 and k_2 are chosen as $1.75/(n \times N^{1/2})$, where n and N represents number of antibodies and length of antibody [32].

5. EXPERIMENTAL ANALYSIS

The optimization of bloom filter is experimented on synthetic data sets using both genetic algorithm and CSA. For synthetic data sets the total number of strings taken for testing is 500. The string weights are generated from 0.0005 to 5. The minimum and maximum number of words stored in a Bloom filter for 500 words is 32 and 256. The size of each Bloom filter that presents inside the bin is 512. This experimental setup is applied for bins from 4 to 7 for 50 iterations. Table 1 lists the parameters and their values used in BBF.

Table1. Parameter values of BBF

Data set	Number of words stored		Word Weight		Bloom filter size	No. of iterations
	Min	Max	Min	Max		
Synthetic data-set-500 words	32	256	0.0005	5	1024	50

Since Bloom Filter allows false positive, the membership invalidation cost is unavoidable. For BBF, the total membership invalidation cost is expressed in equation (4). Different weights in different Bloom filters into consideration, the total membership invalidation cost for Bloom filter is then as follows:

$$F_{standard} = (n_1 f_1 w_1 + n_2 f_2 w_2 + \ldots\ldots + n_L f_L w_L)\, f$$

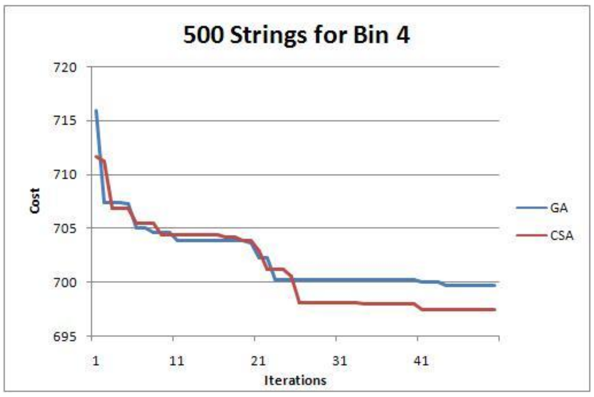
$$F_{standard}(L) = f \sum_{i=1}^{L} n_i w_i$$

The average false positive rate of last iteration of BBF is considered as false positive rate of Bloom filter. Table 2 shows the parameter and its value in CSA and GA.

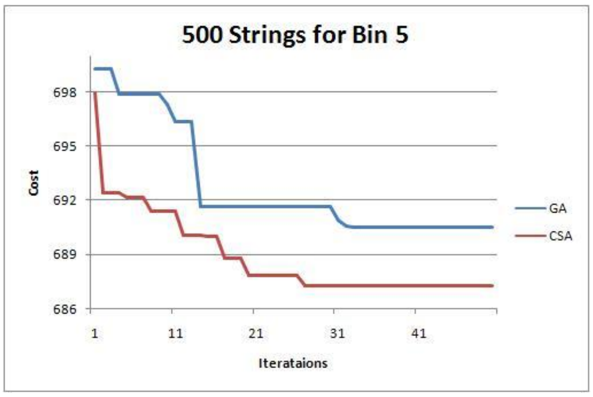
Table 2. Parameter and its value in GA and CSA

Parameter	Value
CSA	
Population size	10
No. of iterations	50
Cloning rate β	0.5
d	20% of population size
GA	
Type of selection	Roulette wheel selection
Type of crossover	One point crossover
Type of mutation	Uniform mutation
Type of evaluation	Elitist selection
Population size	10
Selection rate	0.5
Crossover rate	0.8
Mutation rate	1/3L

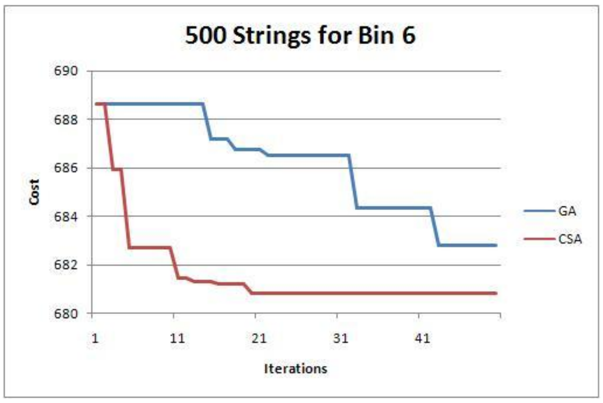
Figures5(a) to 5(d) show the cost obtained for 500 strings from synthetic data set for Bin 4, Bin 5, Bin 6 and Bin 7 respectively. In figure 5(a), for bin size 4 the cost of BBF for GA and CSA remains similar till 22 iterations. Then for next 28 iterations the cost of BBF is reduced for CSA. In figures 5(b) to 5(d), for bin sizes 5,6 and 7 the cost of the BBF gets reduced gradually when number of iterations is increased.



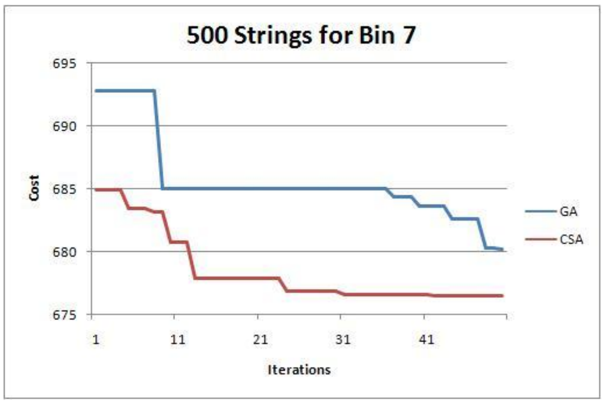
(a)



(b)



(c)



(d)

Fig 5: Comparison of GA and CSA for 500 strings

Figure 6 shows the cost obtained from GA and CSA. The cost and bin size are negatively correlated; when the bin size increases the cost decreases. For many configurations CSA gives better output than GA.

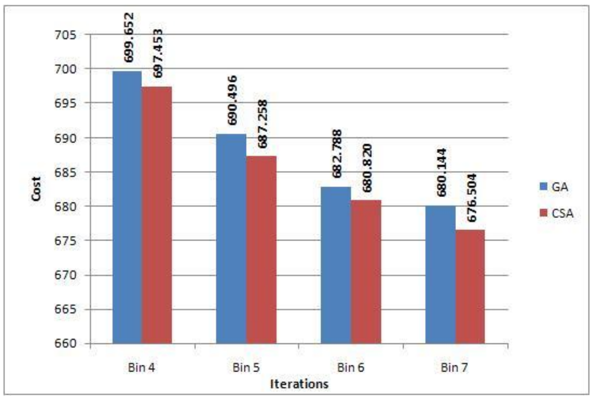


Fig 6: Cost obtained from GA and CSA

5. CONCLUSION

The BBF is an efficient form of data structure which treats words in a set differently depending on the word weights and allocates words into bins with different false positive rates. It has much lower membership invalidation cost compared to the Bloom filter for the methods GA and CSA. GA has premature convergence owing to some high rated individuals quickly attain to dominate the population, constraining it to converge into a local optimum. The GA depends on parameter values rather CSA finds the global optimum independent of initial parameter values. Simulation results show the CSA outperforms GA.

6. REFERENCES

[1] Bloom B, “Space/time tradeoffs in hash coding with allowable errors”, Communications of the ACM, 13, 1970, pp. 422–426.

[2] Mullin J.K, “Optimal Semijoins for Distributed Database Systems”, IEEE Trans. Software Eng., 16, 1990, pp. 558-560.

[3] Mackert L.F. and Lohman G.M., “Optimizer Validation and Performance Evaluation for Distributed Queries”, Proc. 12th Int’l Conf. Very Large Data Bases (VLDB), 1986, 149-159.

[4] Broder A and Mitzenmacher M. “Network Applications of Bloom Filters: A Survey”, Internet Math., 1(4), 2005, pp. 485-509.

[5] Kubiatowicz J Bindel D, Chen, Y Czerwinski S, Eaton P, and Geels D, “Oceanstore: An Architecture for Global-Scale Persistent Storage,” ACM SIGPLAN Notices, 35(11), 2000, pp. 190-201.

[6] Li J, Taylor J, Serban L, and Seltzer M, “Self-Organization in Peer-to-Peer System”, Proc. ACM SIGOPS, 2002.

[7] Cuena-Acuna F.M, Peery C,Martin R.P, and Nguyen T.D, PlantP: “Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities”, Proc. 12th IEEE Int’lSymp. High Performance Distributed Computing, 2003, pp. 236-249.

[8] Chen, H, Jin, H, Chen, L, Liu, Y and Ni, L., “Optimizing Bloom Filter Settings in Peer-to-Peer Multi-keyword Searching”, IEEE Transactions on Knowledge and Data Engineering, 2011, Vol. PP. No.99, pp. 1 – 1.

[9] Rhea S.C and Kubiatowicz J, “Probabilistic Location and Routing”, Proc. IEEE INFOCOM, 2004, 1248-1257.

[10] Hodes T.D, Czerwinski S.E, and Zhao B.Y, “An Architecture for Secure Wide Area Service Discovery”, Wireless Networks, vol. 8, nos. 2/3, 2002, pp. 213-230.

[11] Reynolds P and Vahdat A, “Efficient Peer-to-Peer Keyword Searching”, Proc. ACM Int’l Middleware Conf., 2003, pp. 21-40.

[12] Bauer D, Hurley P, Pletka R, and Waldvogel M, “Bringing Efficient Advanced Queries to Distributed Hash Tables”, Proc. IEEE Conf. Local Computer Networks, 2004, pp. 6-14.

[13] Fan L, Cao P, Almeida J, and Broder A, “Summary Cache: A Scalable Wide Area Web Cache Sharing Protocol”, IEEE/ACM Trans. Networking, Vol.8 No.3, 2000, pp. 281-293.

[14] Peter C.D and Panagiotis M, “Bloom Filters in Probabilistic Verification”, Proc. Fifth Int’l Conf. Formal Methods in Computer- Aided Design, 2004, pp. 367-381.

[15] Jin C, Qian W, and Zhou A, Analysis and Management of Streaming Data: A Survey, J. Software, 15(8), 2004, 1172-1181.

[16] Deng F and Rafiei D, “Approximately Detecting Duplicates for Streaming Data Using Stable Bloom Filters”, Proc. 25th ACMSIGMOD, 2006, pp. 25-36.

[17] Bonomi F, Mitzenmacher M, Panigrahy R, Singh S, andVarghese G, “Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines”, Proc. ACM SIGCOMM, 2006 , pp. 315-326.

[18] Li K and Zhong Z, “Fast Statistical Spam Filter by Approximate Classifications”, Proc. Joint Int’l Conf. Measurement and Modeling of Computer Systems, SIGMETRICS/Performance, 2006, pp. 347-358.

[19] Mitzenmacher M, “Compressed Bloom Filters”, IEEE/ACM Trans.Networking, 10(5) 2002, pp. 604-612.

[20] Kirsch A and Mitzenmacher M, “Distance-Sensitive Bloom Filters”, Proc. Eighth Workshop Algorithm Eng. and Experiments (ALENEX ’06), 2006.

[21] Kirsch A and Mitzenmacher M, “Building a Better Bloom Filter, Technical Report” tr-02-05.pdf, Dept. of Computer Science, Harvard Univ,2006.

[22] Kumar A, Xu J, Wang J, Spatschek O, and Li L, “Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement”, Proc. 23rd IEEE INFOCOM, 2004, pp. 1762-1773.

[23] Cohen S and Matias Y, “Spectral Bloom Filters”, Proc. 22nd ACM SIGMOD, 2003, pp. 241-252.

[24] Laufer R.P, Velloso P.B, and Duarte O.C.M.B, “Generalized Bloom Filters”, Technical Report Research Report GTA-05-43, Univ. of California, Los Angeles (UCLA), 2005.

[25] Chazelle B, Kilian J, Rubinfeld R, and Tal A, “The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Tables”, Proc. Fifth Ann. ACM-SIAM Symp. Discrete Algorithms (SODA), 2004, pp. 30-39.

[26] Hao F, Kodialam M, and Lakshman T.V, “Building High Accuracy Bloom Filters Using Partitioned Hashing”, Proc. SIGMETRICS/Performance, 2007, pp. 277-287.

[27] Paynter, M and Kocak, T, “Fully pipelined bloom filter architecture”, Communications Letters, IEEE, 2008, Vol.12, No. 11, pp. 855 - 857

[28] Deke Guo, Jie Wu, Honghui Chen, Ye Yuan and Xueshan Luo, “The Dynamic Bloom Filters”, IEEE Transactions on Knowledge and Data Engineering, 2010,Vol. 22 No. 1, pp.120 – 123.

[29] L. N. De Castro and F. J. Von Zuben (2002), “Learning and Optimization Using the Clonal Selection Principle”, IEEE Transactions on Evolutionary Computation , 6(3) 239 – 251.

[30] Xie K., Min Y., Zhang D., Wen J., Xie G. & Wen J, “Basket Bloom Filters for Membership Queries”, Proceedings of IEEE Tencon’05,2005, pp. 1-6.

[31] Berek, C. and Ziegner, M. “The Maturation of the Immune Response”, Immunology Today, Vol. 14, No. 8, pp. 400-402, 1993.

[32] Back, T. “Self-Adaptation in Genetic Algorithms”, Proceedings of the First European Conference on Artificial Life, Paris, France, December 11-13, pp. 263–271, 1991.