International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:8, 2014

Bin Bloom Filter Using Heuristic Optimization Techniques for Spam Detection

N. Arulanand, K. Premalatha

***Abstract*—**Bloom filter is a probabilistic and memory efficient data structure designed to answer rapidly whether an element is present in a set. It tells that the element is definitely not in the set but its presence is with certain probability. The trade-off to use Bloom filter is a certain configurable risk of false positives. The odds of a false positive can be made very low if the number of hash function is sufficiently large. For spam detection, weight is attached to each set of elements. The spam weight for a word is a measure used to rate the e-mail. Each word is assigned to a Bloom filter based on its weight. The proposed work introduces an enhanced concept in Bloom filter called Bin Bloom Filter (BBF). The performance of BBF over conventional Bloom filter is evaluated under various optimization techniques. Real time data set and synthetic data sets are used for experimental analysis and the results are demonstrated for bin sizes 4, 5, 6 and 7. Finally analyzing the results, it is found that the BBF which uses heuristic techniques performs better than the traditional Bloom filter in spam detection.

***Keywords*—**Cuckoo search algorithm, levy's flight, meta-heuristic, optimal weight.

I. Introduction

INTERNET traffic has been growing at a tremendous pace and various application services are becoming indispensable in day-to-day life. To keep up with the growing demand, the internet must have high performance, sophisticated packet control functionality. One such popular application is the use of e-mail. E-mails are extensively used as a mode of communication. From the beginning of the Internet, sending of junk e-mail or spam e-mail has been prohibited. However, spam e-mails have become indispensable and have steadily grown since the early 1990s. There are various methods developed to detect and filter the unsolicited spam e-mails.

Spamming is the action of sending unsolicited commercial messages in bulk without the explicit permission or desire of the recipients. That is spam is an irrelevant or inappropriate messages sent on the internet to a large number of users. Everyday billions of e-mails are being passed around and that many spam e-mails are also sent. A lot of time is wasted trying to get rid of spam from the computer and spam also takes up desirable memory space. A spam filter works by scanning all incoming e-mails and evaluating them for signs of spam such as specific keywords, illegitimate links, etc. There are different types of spam filters. Some are pre-programmed to trace known spammers and others filter e-mails based on the words that are written in the body of the e-mail. For spam detection, an efficient data structure is needed to test the membership of words in the e-mails. A Bloom filter is a compact data structure designed to store queries that help to establish set membership information.

A Bloom filter is a space-efficient randomized data structure for representing a set in order to support membership queries [1]. Bloom filters allow false positives but it saves space. It balances the drawback when the probability of an error is made adequately low. The bloom filter is very popular in database applications and networking. In this work, a weight is assigned for each word appears in e-mails based on its frequency in both spam and legitimate mails [2]. This weight value indicates its probable belongings to spam or legitimate. In the first step of spam filtering each word is assigned by a weight. Bloom filter treats every word equally and it allocates the same number of hash functions to every word which results in high total membership invalidation cost.

The proposed work introduces an enhanced concept in Bloom filter named as Bin Bloom Filter (BBF) which groups the words into a number of Bloom filters which will have different false positive rates based on the weights of the spam words. Instead of giving equal importance to all the words in a spam word list, different importance is given based on their weights. The words with high weights are stored in a Bloom filter with lowest false positive rate and the words with low weights are stored in a Bloom filter with high false positive rate. For a high cost bin it is expected to allocate more hash functions to decrease the membership invalidation rate, and conversely, a low cost bin allows higher false positive probability, and then the BBF makes the total membership invalidation cost a minimum. In traditional Bloom filter, every word is treated equally and it allocates the same number of hash functions to every word which results in high total membership invalidation cost.

For this process, the number of words to be stored in the individual Bloom filter and the number of Bloom filters are identified to reduce the total membership invalidation cost for a given collection of words with their weights. The number of possible solutions in the search space is so large. So it is an optimization problem to identify the number of words to be stored, false positive rate and number of hash functions of each Bloom filter in the Bin which minimize the total membership invalidation cost. It prevents an exhaustive search for the best answer.

This paper compares the optimization of bloom filter in spam filtering using Genetic Algorithm (GA), Particle Swarm Optimization (CSA), Clonal Selection Algorithm (CSA), Cuckoo Search (CS) and Bat algorithm. The organization of this paper is as follows. Section II deals with the Bloom filter.

Arulanand N. is with the PSG College of Technology, Coimbatore, Tamil Nadu, India (e-mail: arulnat@yahoo.com).

Section III presents the Bin Bloom Filter (BBF). Section IV discusses the BBF using heuristic optimization techniques. Performance evaluation of heuristic optimization techniques for the BBF is reported in Section V.

II. Bloom Filter

The base data structure of a Bloom filter is a bit vector of size m. Given a string X the Bloom filter computes k hash functions on it producing k hash values ranging from 1 to m. It then sets k bits in an m-bit long vector at the addresses corresponding to the k hash values. The same procedure is repeated for all the members of the set. This process is called programming of the filter. The query process is similar to programming, where a string whose membership is to be verified is input to the filter. The Bloom filter generates k hash values using the same hash functions it used to program the filter. The bits in the m-bit long vector at the locations corresponding to the k hash values are looked up. If at least one of these k bits is not found in the set then the string is declared to be a nonmember of the set. If all the bits are found to be set then the string is said to belong to the set with a certain probability. This uncertainty in the membership comes from the fact that those k bits in the m-bit vector can be set by any other n-1 members. Thus finding a bit set does not necessarily imply that it was set by the particular string being queried. However, finding a bit not set certainly implies that the string does not belong to the set. Fig. 1 shows the structure of Bloom filter.

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

$$f = (1-e^{-kn/m})^k \quad (1)$$

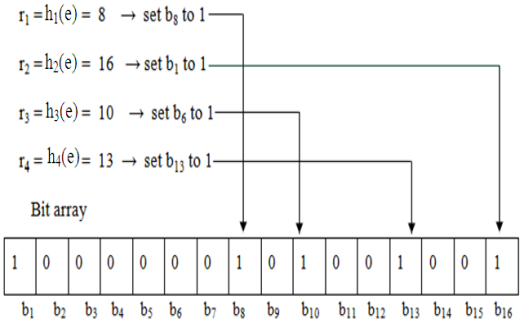

Fig. 1 Bloom Filter

III. Bin Bloom Filter

Consider a set $S=\{s_1,s_2,...,s_n\}$ of n elements. Bloom filter depicts membership Information of S using a bit vector B of length m for the k hash functions, $h_1,h_2,...,h_k$ with $h_i : v \rightarrow \{1..m\}$. A Bin Bloom Filter is a variant of Bloom filter which is used to store the words that have different spam weights. Instead of giving equal importance to all the words in a spam word list, the importance is given based on their weights. The words with high weights are stored in a Bloom filter with lowest false positive rate and the words with low weights are stored with high false positive rate. A Bin consists of a number of Bloom filters and each Bloom filter has different tuple <n,f,w> which causes different membership invalidation cost. Fig. 2 shows BBF with its tuple <n,f,w> configuration where n_i, f_i and w_i respectively represents the number of strings, false positive rate and average weight of strings in Bloom filter i and L represents the number of Bloom filters in a bin.

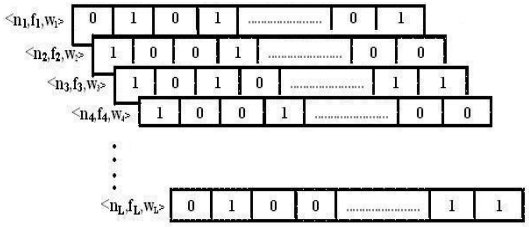

Fig. 2 Bin Bloom Filter

For membership testing the total cost of the set is the sum of the invalidation cost of each subset. The total membership invalidation cost is given as,

$$F = n_1 f_1 w_1 + n_2 f_2 w_2 + ...+ n_i f_i w_i + ...n_L f_L w_L \quad (2)$$

The total membership invalidation cost is

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

to be minimized; where $\sum_{i=1}^{L} n_i = N$ and $n^l \le n_i \le n^u$

N- Total number of strings in a spam set

n^l – Lower limit for number of strings to be stored in a Bloom filter

n^u – Upper limit for number of strings to be stored in a Bloom filter

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

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

Since Bloom Filter allows false positive, the membership invalidation cost is unavoidable [3]. For BBF, the total membership invalidation cost is expressed in (3). Having different weights in different Bloom filters into consideration, the total membership invalidation cost for Bloom filter is then as follows:

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

$$F_{standard}(L) = \sum_{i=1}^{L} n_i w_i \left(\frac{1}{2}\right)^{\ln 2 \left(m \Big/ \left(\sum_{j=1}^{L} n_j \right) \right)} \quad (5)$$

The average false positive rate of the last iteration of BBF is taken as the false positive rate of Bloom filter.

IV. Heuristic Optimization Techniques for BBF

The proposed work applies various heuristic optimization techniques in BBF for identifying the minimal total membership invalidation cost for different bin sizes and the results are compared with Bloom filter.

A. Genetic Algorithm

The GA begins with an initial population of the problem [4]. This algorithm encodes a potential solution to a specific problem on a simple chromosome and it applies crossover and mutation operators to these structures. Each one of these solutions must be evaluated by means of a fitness function. The goodness of a solution is typically defined with respect to the current population.GA mimics the survival-of-the-fittest principle of nature to make a Search process. Therefore, GA is naturally suitable for solving maximization problems. Minimization problems are usually transformed into maximization problems by suitable transformation. For minimization problems, to generate non-negative values in all the cases and to reflect the relative fitness of individual string, it is necessary to map the underlying natural objective function to fitness function.

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

$$f(L) = \begin{cases} C_{max} - F(L) & \text{if } F(L) < C_{max} \\ 0 & \text{if } F(L) \geq C_{max} \end{cases} \quad (6)$$

$X_i =$ | 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. 3 Encoding Solution for BBF

The transformation from minimization to maximization does not alter the location, but it converts a minimization problem into an equivalent maximization problem. In the perspective of BBF, a chromosome represents a number of Bloom filters with its words, false positive rate and average spam weight of the words. Explicitly, each chromosome X_i, is constructed is shown in Fig. 3 where n_{ij}, f_{ij} and w_{ij} refer respectively the number of words, false positive rate and the weight of the j^{th} Bloom filter of i^{th} chromosome. A set of 3 genes <n,f,w> encodes a protein – a trait, that is a single Bloom filter of a bin.

B. Clonal Selection Algorithm

In CSA antigen refers to the problem and its constraints [5], [6]. The algorithm performs the selection of antibodies based on affinity either by matching against an antigen pattern or via evaluation of a pattern by a cost functions. In clonal selection n, the 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 (7):

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

where N_C is the total amount of clones generated, β is a multiplying factor which decides number of clones to be produced, P is the total amount of antibodies and round() is the operator that rounds its argument towards the closest integer. The Cauchy mutation operator is applied for mutation. In the perspective of BBF, an antibody represents number of Bloom filters with number of words, false positive rate and average spam weight of the words [7]. The representation of each antibody X_i is shown in Fig. 3. An antibody in the population represents one possible solution for assigning the triples <n, f, w> for L Bloom filters. Initially each antibody randomly chooses different <n, f, w> for L bins. The fitness function of particle is calculated using (6)

C. Particle Swarm Optimization

PSO algorithm is an optimization technique, using stochastic search guided by the principles of collective behaviour and self-organization of flock of birds [8], [9]. In PSO, each single solution is like a 'bird' in the search space, which is called particle. It is initialized with a population of random solutions. All particles have fitness values which are evaluated by the fitness function to be optimized. Each particle in PSO is associated with a velocity. Particles fly through the search space with certain velocities which are dynamically adjusted according to their historical behavior [10].

In the framework of BBF, an individual particle representation is shown in Fig. 3. One particle in the swarm represents one possible solution of BBF. Therefore, a swarm represents a number of candidate solutions for the current data vectors. At the initial stage, each particle randomly chooses different <n, f, w> for L Bins. The fitness function of particle is calculated using (4). The BBF is optimized using PSO with three types of velocity equations by incorporating static inertia weight, deterministic inertia weight and constriction factor [11].

D. Cuckoo Search

CS is an optimization technique developed by Yang and Deb based on the obligate brood parasitism of cuckoo species by laying their eggs in the nests of other host birds [12]. If a host bird discovers the eggs which are not its own, it will either throw these foreign eggs away or simply abandon its nest and build a new nest elsewhere. Each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The better new solution (cuckoo) is replaced with a solution which is not so good in the nest. In the simplest form, each nest has one egg [13], [14].

When generating new solutions for a cuckoo i, a Levy flight is performed. Basically Levy flights provide a random walk while their random steps are drawn from a Levy distribution for large steps given in (8):

$$\text{Levy} \sim u = t^{-\lambda} \quad (8)$$

which has an infinite variance with an infinite mean.

E. Enhanced Cuckoo Search

The Enhanced Cuckoo Search (ECS) is an extended version of CS with different types of host nest with multiple eggs. The cuckoo selects three types of nest for laying their eggs [15], [16].

1. Common cuckoo selects a group of host species with similar nest sites and egg characteristics to her own. There is a significant level of similarity between cuckoo eggs and typical eggs of the host species.
2. Some hosts do not exhibit egg rejection behaviour and the cuckoo eggs look very dissimilar from the host eggs.
3. Other species of cuckoo lay cryptic eggs, which are dark in color when the eggs of their hosts are light. This is a trick to hide the egg from the host that parasitizes hosts with dark nests.

In the environment of BBF, an egg represents a number of bloom filters with its number of words to be stored, false positive rate and average spam weights of the words. The fitness function for CS and ECS is calculated using (4). The representation egg is shown in Fig. 3. In ECS each nest contains 4 eggs. Three different types of eggs are produced by applying Levy flight, Uniform mutation and Random mutation on current solution.

F. Bat Algorithm

Yang proposed bat algorithm for optimization. Bats use echolocation to locate and catch their prey [17]. When bats fly, they produce a constant stream of high-pitched sounds that only bats are able to hear. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. The loudness also varies from the loudest to a quieter base when searching for prey and homing towards prey respectively.

The velocity and position of the bat is given by

$$f_i = f_{min} + (f_{max} - f_{min}) \times \beta \tag{9}$$

$$V_i(t) = V_i(t-1) + (X_i(t) - X^*) \times f_i \tag{10}$$

$$X_i(t) = X_i(t-1) + V_i(t) \tag{11}$$

where $\beta \in [0,1]$ is a random drawn from a uniform distribution. X^* is the current global best location among n bat solutions. For the local search, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk.

$$X_{new} = X_{old} + \varepsilon A^t \tag{12}$$

where ε is a random number $[-1, 1]$ and A^t is the average loudness of all bats at the time step t. The loudness A_i and the rate r_i of pulse emission have to be updated accordingly as the iterations proceed.

$$A_i(t+1) = \alpha A_i(t) \tag{13}$$

$$r_i(t+1) = r_i(0)(1-\exp(-\gamma t)) \tag{14}$$

where α and γ are constants , where $0 < \alpha < 1$ and $\gamma > 0$. The representation of bat is shown in Fig. 3 and the fitness function of each bat is evaluated using (4).

V. Experimental Analysis

The heuristic optimization techniques are applied for synthetic data sets of 250, 500 and 1000 strings and real time data set of 1000 strings. Table I show the parameters and their values of BBF. The number of Bloom filters taken for bin size is 4, 5, 6 and 7 and each Bloom filter size is 1024. For synthetic data set, the weights of the words are generated randomly from 0.0005 to 5. In real time data set, the spam weights of the words are calculated from 1000 spam e-mail and 1000 non spam e-mails. The spam weight of the words ranges from 0.0094 to 5. For the implementation of the proposed work Java Developer's Kit 1.6.0 is used in Windows environment. Table II show the parameters and their values of optimization techniques respectively.

TABLE I
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 – 250 words	16	128	0.0005	5	1024	50
Synthetic data set – 500 words	32	256				
Synthetic data set – 1000 words	32	512				
Real time data set – 1000 words	32	512	0.0094	5		

Figs. 4-7 show the cost obtained from BBF for synthetic data sets of 250 strings, 500 strings, 1000 strings and real time data set of 1000 strings using GA, CSA, PSO with static inertia weight (PSO1), PSO with deterministic inertia weight (PSO2), PSO with constriction factor (PSO3), CS, ECS and Bat algorithm respectively.

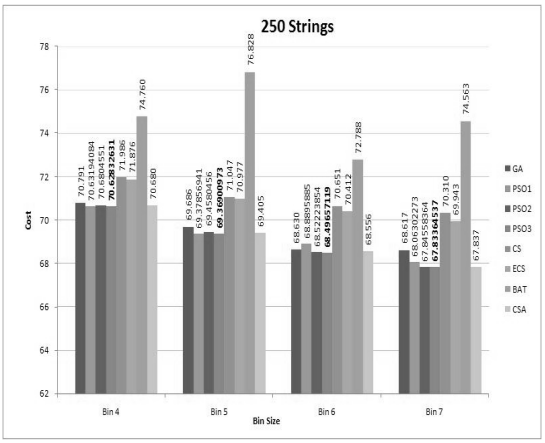

Fig. 4 Cost of BBF for synthetic dataset of 250 strings

TABLE II
PARAMETERS AND THEIR VALUES FOR DIFFERENT OPTIMIZATION TECHNIQUES

Parameter	Value
Genetic Algorithm	
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
Clonal Selection Algorithm	
Population size	10
No. of iterations	10
Cloning rate β	0.5
d	20% of population size
Particle Swarm Optimization	
Population size	10
Number of Iterations	50
c1	2.1
c2	2.1
ω	0.9
ωmax	0.9
ωmin	0.4
kf	0.729
Cuckoo Search	
Number of nests	10
Number of iterations	50
pa	0.3
α	1
δ	1.5
Bat Algorithm	
Number of bats	10
Number of iterations	50
α	rand(0,1)
β	rand(0,1)
γ	0.5
A	0.25
R	0.5

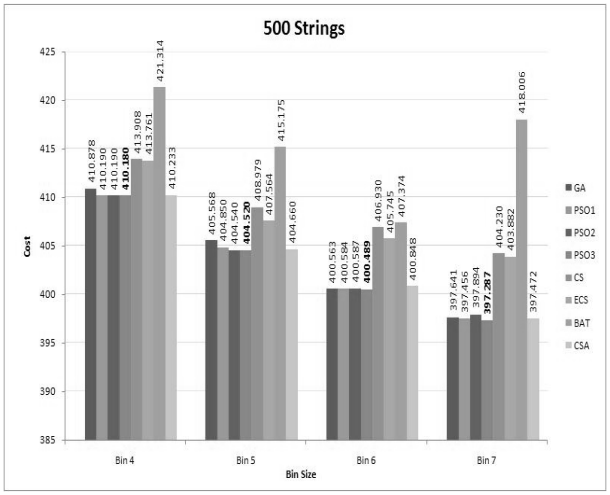

Fig. 5 Cost of BBF for synthetic dataset of 500 strings

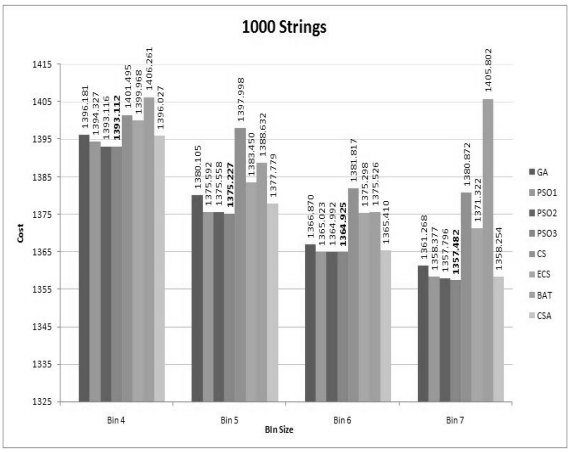

Fig. 6 Cost of BBF for synthetic dataset of 1000 strings

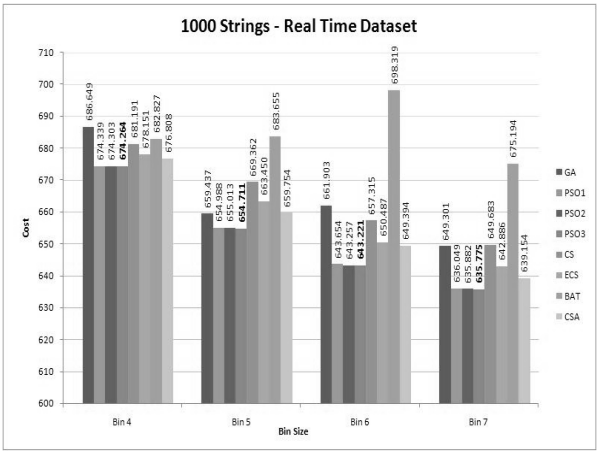

Fig. 7 Cost of BBF for real time dataset of 1000 strings

GA solves the problem through evolution. In GA, the parameters play an important role in getting better optimization results. It is computationally intensive and has premature convergence. Most of the time, it gives better results than CS, ECS and Bat algorithm.

CSA is a population based stochastic method. CSA operators namely Clonal Selection and Somatic Hypermutation lead to better results compared to GA. CSA has quicker convergence compared to GA due to the population that always contains higher affinity measures. It has only a few parameters to fine tune.

CS, ECS and Bat Algorithm are nature-inspired heuristic algorithms. CS is less sensitive to variation in tuning parameters. A modification of Cuckoo Search, named as Enhanced Cuckoo Search is presented in this paper by considering multiple eggs in the nest and dealing with different types of cuckoo eggs. Experimental results show that ECS gives better performance than CS. In Bat Algorithm, various parameters like loudness, velocity, frequency, pulse emission and wavelength are used. It is difficult to fine tune various parameters to get near optimal solution. CS and ECS give better performance than Bat algorithm.

TABLE III
PERCENTAGE OF IMPROVEMENT IN BBF OVER BF

Data set	No. of Strings	Bin size	GA	CSA	PSO1	PSO2	PSO3	CS	ECS	Bat
Synthetic	250	4	17.875	17.978	18.033	17.977	18.038	16.462	16.589	13.243
		5	19.156	19.458	19.488	19.396	19.499	17.552	17.633	10.843
		6	20.456	20.443	20.055	20.482	20.512	18.011	18.289	15.532
		7	20.420	21.277	21.015	21.267	21.281	18.408	18.833	13.472
	500	4	13.835	13.970	13.979	13.979	13.982	13.200	13.231	11.647
		5	15.041	15.139	15.099	15.164	15.168	14.233	14.530	12.934
		6	16.009	15.939	15.994	15.993	16.014	14.663	14.911	14.570
		7	16.619	16.647	16.650	16.558	16.685	15.229	15.302	12.340
	1000	4	10.100	10.106	10.215	10.293	10.294	9.754	9.852	9.447
		5	11.142	11.281	11.422	11.422	11.445	9.979	10.916	10.582
		6	12.000	12.077	12.102	12.102	12.109	11.021	11.441	11.426
		7	12.377	12.538	12.530	12.530	12.588	11.082	11.697	9.476
Real Time	1000	4	18.385	19.555	19.849	19.853	19.858	19.034	19.396	18.840
		5	21.620	21.582	22.149	22.146	22.182	20.440	21.143	18.741
		6	21.327	22.814	23.496	23.543	23.547	21.872	22.684	16.998
		7	22.825	24.031	24.400	24.420	24.432	22.779	23.587	19.747

PSO is a popular swarm intelligence technique. PSO is easy to implement when compared to other methods, and there are a few parameters to adjust. PSO has faster convergence in finding near global optimal solutions. Experimental results show that PSO gives better performance compared to other optimization techniques due to stochastic and deterministic component present in the particle. Different variants of PSO, namely static inertia weight, deterministic inertia weight and constriction factor are experimented in this paper. PSO with constriction factor gives better results for all the configurations compared to the other PSO variants. Table III shows the percentage of improvement in BBF over BF using various optimizations for synthetic data sets and real time data set.

VI. CONCLUSION

In this work an optimal configuration of BBF with spam word weight is designed. In BBF different false positive rates are assigned to the Bloom filters depending on the spam weight of the words. Instead traditional Bloom filter uses same false positive rate. From the results it is seen that BBF always outperforms the traditional Bloom filter.

ECS gives better performance than CS and Bat algorithm. The parameters p_a and α in CS and ECS remain constant. Due to this the efficiency and performance of these algorithms are decreased. The performance of the Bat Algorithm is determined by the parameters loudness, velocity, frequency, wave length and pulse emission. It is difficult to fine tune the parameters to get near optimal solution.

GA solves problem with multiple solutions. Most of the time, it gives better result than CS, ECS and Bat algorithm. In GA it is difficult to optimize the parameters involved and it is computationally intensive. It has premature convergence.

CSA has quicker convergence due to the population that always contains higher affinity measures. It has only a few parameters to fine tune.

PSO is easy to implement when compared to other methods, and there are a few parameters to adjust. PSO has very faster convergence in finding near global optimal solutions.

REFERENCES

[1] B.H. Bloom, "Space/time tradeoffs in hash coding with allowable errors," *Commun. ACM.,* vol. 13, no. 7, pp. 422–426, July, 1970.
[2] M. Abdoh, M. Musa and N. Salman, "Detecting Spam by Weighting Message Words," *J. Arts Sci.,* vol. 1, no. 1, pp. 1-14, Aug. 2009.
[3] K. Xie, Y. Min, D. Zhang, G. Xie and J. Wen, "Basket Bloom Filters for Membership Queries," *in Proc. IEEE Tencon'05*, Melbourne, Qld, pp. 1-6, 2005.
[4] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, 2009.
[5] L.N. De Castro and F.J. Von Zuben, "Learning and Optimization using the Clonal Selection Principle," *IEEE Trans. Evol. Comput.,* vol. 6, no. 3, pp. 239-251, Aug. 2002.
[6] J. Timmis and L.N. de Castro, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, London, 2002.
[7] N. Arulanand, P. Swathy Priyadharsini and S. Subramanian "Artificial Immune System for Bloom filter Optimization," *Int. J. Computer App.*, vol. 41, no. 8, pp. 26-32, March 2012.
[8] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," *in Proc. IEEE Int. Confer. Neural. Networks.,* Perth, WA, Australia, pp. 1942-1948, 1995
[9] R.C. Eberhart and Y. Shi, "Particle swarm optimization, developments, applications and resources", *in. Proc Cong. Evol. Comput.,* Seoul, Korea. Piscataway, pp.445-457, 2001.
[10] N. Arulanand, S. Subramanian and K. Premalatha "Optimized Bin Bloom Filter for Spam filtering using Particle Swarm Optimization," *European J. Scientific Research*, vol. 68, no. 2, pp. 199-213, July 2012.
[11] M. Clerc and J. Kennedy, "The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space," *IEEE Trans. Evol. Comput.,* vol. 6, pp. 58-73, Feb. 2002.
[12] N. Arulanand, S. Subramanian and K. Premalatha "An Enhanced Cuckoo Search for Optimization of Bloom Filter in Spam Filtering," *Global J. comp. Scie. Tech.,* vol. 12, no. 1, Jan. 2012
[13] N. Arulanand , S. Subramanian and K. Premalatha "A Comparison study of cuckoo-bat search for Optimization of Bloom Filter in Spam Filtering," Int. *J. Bio-Inspired Comput.,* vol. 4, no. 2, pp.89-99, June 2012.
[14] X.S. Yang, and S. Deb, "Engineering optimisation by Cuckoo search" *Int. J. Math. Modeil. Numer. optim.*, vol. 1, no. 4, pp. 330-343, Dec. 2010.
[15] C. Moskat, and M. Honza "European Cuckoo Cuculus Canorus Parasitism and Host's Rejection Behaviour in a Heavily Parasitized Great Reed Warbler Acrocephalus Arundinaceus Population," *Int. J. Avian. Scie.*, vol. 144, no. 4, pp. 614-622, Sep. 2002

[16] A. Moksnes and E. Roskaft "Egg-Morphs and Host Preference in the Common Cuckoo (Cuculus Canorus): An Analysis of Cuckoo and Host Eggs form European Museums and Collections," *J. Zool.,* vol. 236, no. 4, pp. 625-648, Mar. 1995.
[17] X.S. Yang, "A New Metaheuristic Bat-Inspired Algorithm", Nature Inspired Cooperative Strategies for Optimization (NISCO 2010), Studies in Computational Intelligence, Springer Berlin, Springer, vol. 284, pp.65-74, April, 2010.