MultiCraft

International Journal of Engineering, Science and Technology
Vol. 16, No. 2, 2024 pp. 48-67

INTERNATIONAL JOURNAL OF ENGINEERING, SCIENCE AND TECHNOLOGY

www.ijest1-ng.com
www.ajol.info/index.php/ijest

A technical research survey on bio-inspired intelligent optimization grouping algorithms for finite state automata in intrusion detection system

S. Prithi[1], S. Sumathi[2*]

[1]*Department of Computer Science and Engineering, Sri Rajalakshmi Engineering College, Chennai. Tamil Nadu, INDIA*
[2]*Department of Electrical and Electronics Engineering, PSG College of Technology, Coimbatore, Tamil Nadu, India*
**Corresponding Author: e-mail: ssi.eee@psgtech.ac.in, Tel. No.: +99947 59330*
ORCID iDs: http:/orcid.org/0000-0001-5165-0212 (Sumathi)

Abstract

Network Security plays an essential role in the modern world. Current network services mainly rely on processing of payload in packets. Deep Packet Inspection (DPI) is a key factor in examining the packet payload which uses the signatures to identify the packet that carries any viruses, worms, malicious traffic, unauthorized access and attacks. DPI uses regular expression matching as a core operator to examine the packet payload. Finite State Automata (FSA) are natural representations for regular expression. FSA is usually too large to be constructed or deployed and has a huge overhead. Finite State Automata frequently leads to state explosion problem which require more storage space, high bandwidth and more computational time. To overcome this problem, Intelligent Optimization Grouping Algorithms (IOGA) can be used to distribute the regular expressions into various groups and for each group the Deterministic Finite Automata (DFA) are built independently. Grouping the regular expression efficiently solves the state explosion problem by achieving large-scale best tradeoff among the memory utilization and computational time. This paper reviews the various Intelligent Optimization Grouping Algorithms like Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization, Bacterial Foraging Optimization, Artificial Bee Colony Algorithm, Biogeography Based Optimization, Cuckoo Search, Firefly Algorithm, Bat Algorithm and Flower Plant Optimization. The discussions states that by effectively using these grouping algorithms along with finite state automata can reduce the number of states by solving the state explosion blow up problem, providing a balance between the memory consumption, number of groups and provide faster convergence.

Keywords: Intelligent Optimization Grouping Algorithms, Finite State Automata, Regular Expression, Deep Packet Inspection, Network Intrusion Detection System.

DOI: http://dx.doi.org/10.4314/ijest.v16i2.6

Cite this article as:
Prithi S., Sumathi S. 2024. A technical research survey on bio-inspired intelligent optimization grouping algorithms for finite state automata in intrusion detection system. *International Journal of Engineering, Science and Technology*, Vol. 16, No. 2, pp. 48-67. doi: 10.4314/ijest.v16i2.6

Received: June 16, 2023; Accepted: August 26, 2023; Final acceptance in revised form: September 4, 2023

1. Introduction

In recent years network security has acquired a tremendous interest due to the anxiety in mounting the security in today's network. An extensive variety of algorithms have been proposed which can detect and battle with these security threats. Amongst these proposals, Network Intrusion Detection System (NIDS) have been a commercial success and have seen a tremendous adoption (Paxson, 1998). The events that occur in a computer system are captured by NIDS and it analyses the signs of possible events in the data packets such as computer security policy violation and standard security policy violation. Network devices have a stipulate growing demand and have the capability of analyzing the data packet contents so as to facilitate network security and to

offer application specific services. NIDS makes use of Deep Packet Inspection (DPI) that allows the network holder to analyze internet traffic, throughout the network, in real-time and to differentiate them according to their packet payload (Nitin et al., 2012).

Network services use signatures recognized from the packet payload to perform intrusion detection system. DPI matches the set of predefined patterns with the content of the payload byte by byte and examines whether the patterns are identified in the payload content. DPI uses regular expressions to represent complex string patterns as attack signatures. DPI examines whether a packets payload matches any of a set of predefined regular expressions. Regular expressions are highly preferred because of its compactness, flexibility and expressive power to specify attack signatures. Finite state automata are abstract model that recognizes the same language that is expressed by the regular expression. FSA plays a major role in matching regular expression (Hopcroft et al., 2001). Finite state automata comprises of a set of states, set of state transitions, input alphabet, start state and set of accept states. Each final state corresponds to a pattern or a signature. Regular expression matching proceeds from the start state and reads the payloads first byte. For each byte read, the state transition occurs based on the current state and the payload byte read. At a course of time accept state is reached and the payload is matched with the equivalent signature. Each byte is processed and each byte requires one or many main memory accesses therefore regular expression matching in DPI is time consuming and depends on the efficiency of the matching algorithm.

FSA can be broadly classified into Deterministic Finite Automata (DFA) and Non-deterministic Finite Automata (NFA). The expressive power of NFA and DFA are same but the processing is different (Hopcroft et al., 2001). DFA have only one state transition for each character while NFA can have multiple state transitions for a character. Thus a DFA have only one state active at a time whereas NFA have multiple states active. Thus the processing cost for each step for DFA is O(1) whereas it is $O(n^2m)$ for NFA, where m denotes number of regular expression and n denotes the average length of regular expression and to convert NFA to DFA it is as large as $O(2^{nm})$ which leads to state explosion problem (Yu et al., 2006). NFA and DFA have a conflicting feature in the consumption of memory and bandwidth. As there is a huge increase in the network applications, the number of signatures that is to be matched with the patterns simultaneously becomes very difficult. Thus the storage and scalability of regular expression becomes a big challenge. A majority of ongoing research are searching for a substitute for storage and performance. Thus the main objective is to perform the matching as fast as DFA at the same time keep the storage as small as NFA for handling a large set of regular expressions.

In the recent years, to improve the overall performance of DPI a significant effort has been done to optimize the automaton based on pattern matching. To implement the DFA bases regular expression matching the primary obstacle is the blow up of states. To avoid state explosion, rule grouping method (Rohrer et al., 2009; Liu et al., 2014; Yu et al., 2006) was used which groups the rules into several groups and for each group a DFA is generated. Each combination of patterns has varying magnitude of interactions therefore the objective of rule grouping is to split the patterns to different groups based on the interaction and to leave the patterns with least interaction without splitting. Yu's grouping algorithm (Yu et al., 2006) tactically compiled a set of regular expressions into several groups that showed an increase in the regular expression matching speed without much rise in the usage of memory. Rohrer et al. (2009) enumerated the method of rule grouping based on Yu's algorithm (Yu et al., 2006) and also computed the measure of how the patterns interact. The enumeration resulted in an optimal grouping. But Yu's and Roherer's algorithm consumed more computational time and memory utilization. Liu et al. (2014) proposed an algorithm called DFA size estimator that estimated the size of the DFA for the given set of regular expression without building the actual DFA. DFA size estimator is orders of magnitude faster, has better grouping results and more efficient than the Yu's algorithm (Yu et al., 2006).

Based on the performance of these existing rule grouping method, grouping multiple regular expressions can be transformed into an optimization problem to obtain a better result. The key goal of the optimization problem is to determine the objective functions maximum or minimum value (Konar, 2005). It literally means finding the best possible or desirable solution. Intelligent optimization grouping algorithms such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Bacterial Foraging Optimization (BFO), Artificial Bee Colony (ABC), Biogeography Based Optimization (BBO), Cuckoo Search, Firefly Algorithm, Bat Algorithm and Flower Plant Optimization can be used efficiently to group the regular expression and IOGA plays a major role in improving the overall performance of DPI. In this paper, a detailed study on IOGA is done to build memory efficient and compact automaton based regular expression matching.

The rest of the paper is structured as follows. Section 2 illustrates about the regular expression its common features and how it works in DPI, the finite state automaton, its basic types namely NFA and DFA and the state of the art system of evolutionary strategies that are implied to finite state automata. Section 3 discusses and analyses about the various Intelligent Optimization Grouping Algorithms that are used in the last three decades and a basic idea of how it can be used with finite state automata and section 4 discusses about the proposed approach for FSA in DPI which is used to group the regular expression efficiently using Intelligent Optimization Grouping Algorithms thereby to compress the memory space and improve the convergence speed and section 5 delivers the concluding remarks.

2. Finite automata based on regular expression matching

The techniques that are used for designing automaton based regular expression matching are discussed in this section. The regular expression plays a major part in pattern matching so a detailed background about the regular expression and its importance in popular DPI systems and how it works in DPI is illustrated. The next core part of matching regular expression is the relationship between regular expression and finite state automaton, which is explained in this section. The two traditional FSA namely NFA

and DFA are discussed in detail. Finally the automaton based DPI is illustrated which can obtain high throughput, low implementation cost, reduced memory utilization, high inspection speed and fast convergence.

2.1 Regular Expression

A regular expression is a formula that was first developed by Kleene, 1956 (Kleene, 1951) for identifying the basic classes of strings. A string represents a sequence of symbols. Each symbol in a regular expression is either a regular ASCII character or a meta-character that has a special meaning. Regular expression can be described by a formal language known as regular language. Regular language is one of the basic classes of the Chomsky Formal Language (Harrison, 1978). The regular language over alphabet Σ is defined as:

1. Φ is a regular language.
2. For any a, a ∈ Σ, {a} is a regular language.
3. If R_1 and R_2 are regular language, then so are:
   a. $R_1.R_2 = [ab \mid a \in R_1, b \in R_2]$, the concatenation of R_1 and R_2
   b. R_1 U R_2, then union of R_1 and R_2
   c. R_1 *, Kleene closure of R_1

The languages that satisfy the above properties are called the regular language. Regular expression has been widely used in programming languages, compiler design, text editors, text processing, network security etc. In network applications, regular expression matching is mainly focused on application protocol identification and NIDS, where packet payloads is inspected at line rate over large sets of complex patterns. As regular expressions are widely adopted for packet content scanning, it is essential that they handle packet header processing. A regular expression pattern may represent the unique characteristics of an application level protocol, a virus, a spam and a malware. The expressive power and flexible characteristics of regular expression has the ability to be widely used in open source DPI and commercial DPI applications.

2.2 Functions of Regular Expression in DPI

The set of strings which are unspecified explicitly are represented using regular expression. Table 1 lists the frequent features of regular expression patterns (Yu et al., 2006) that are widely used in packet payload scanning.

Table 1. Features of regular expression (Yu et al., 2006)

Syntax	Meaning	Example
.	A single character wildcard	
*	A quantifier that refers to zero or more	*S** means an arbitrary number of *S*s.
?	A quantifier that refers to one or less	*pq?r* represents *pr and pqr*
+	A quantifier that refers to one or more	p+q represents pq, ppq, pppq, etc
|	OR relationship	P|Q represents P or Q
^	Patterns that are matched at the start of the input	^PQ represents the input that begins with PQ. A pattern without "^", e.g., PQ, can be matched anywhere in the input.
[]	A class of characters	[*pqr*] denotes a letter *p* or *q* or *r*.
[^]	Anything but	[*^pq*] denotes any character other than *p* or *q*
{}	Repeat	*P{100} denotes 100 Ps.*

2.3 Finite State Automaton

Finite State Automata are computing devices that recognizes the regular languages. The operations of finite state automata are simulated by a simple basic computer program. A finite state automaton consists of a finite set of states which are classed to be either accepting or not accepting. A string of input symbols are taken in by the finite state automata. After processing all the input if the current state is an accepting state then the input is accepted or else the input is rejected. The regular expression matching problem determines whether a given string belongs to the member of the language defined by a particular regular expression. For an automaton based regular expression matching in DPI the given string is considered as the packet payload.

2.4 Deterministic Finite Automata

In deterministic finite automata from each and every state there is only one transition that takes place on each input symbol of the alphabet (Σ) i.e., δ(s,a) is unique (Koza et al., 1994). This implies that for an input w, the execution is predictable.

A DFA is defined as a 5–tuple (S, Σ, δ, s_0, A) where,

- S is a set of finite states.
- Σ is the set of finite input symbols.
- δ is the transition function where δ: S × Σ → S e.g., δ (s,a) =p gives the transition from state s on the input a.
- q_0 Є S is the initial state.
- A ⊆ S is the set of accepting states.

Let M be a DFA such that M = (S, Σ, δ, q_0, A). String w is acceptable by M if δ *(q_0, w) = p for some p in A.

A primary characteristic of DFA is that only one state can be active at any point of time. However this characteristic becomes infeasible for regular expressions that are present in the repeatedly used rule - sets. Particularly, when the regular expression

contains repeated wildcards it becomes difficult to construct a DFA with a reasonable number of states (Hopcroft, 1971). It takes only one main memory accesses per byte. A worst case analysis (Hopcroft et al., 2001) illustrates that a an NFA which contains a single regular expression of size n has O(n) states. The same expression when transformed into a DFA generates $O(\sum^n)$ states. In a DFA the processing complexity for every character in the input is O(1) and however in a NFA it is $O(n^2)$ when all n states are active at the same time.

2.5 Nondeterministic Finite Automata

Nondeterministic Finite Automata (NFA) has multiple or null state transitions for each input symbol. Non determinism therefore implies that there may not be a unique execution trace as in DFAs. There are multiple paths of possible executions. For NFAs a string is accepted if there exists at least one execution path that ends in a final state, whereas a string is rejected if all possible execution paths end in a non accept state. NFAs also allow for ε–transitions i.e. transitions which do not read a character of the input string. For every NFA there is an equivalent DFA which accepts exactly the same language L. It is therefore possible to convert any NFA to a DFA. NFAs are necessary as it is easier to construct whereas DFAs can become very complex. DFAs on the other hand are easier to implement and interpret on a computer.

An NFA is defined as a 5–tuple $(S, \Sigma, \delta, s_0, A)$ where,

- S is the set of finite states.
- Σ is the set of finite input alphabet.
- δ is the transition function where $\delta: S\times (\Sigma \cup \varepsilon) \rightarrow P(S)$ where P(S) is the power set of S and ε is the empty string.
- $s_0 \in S$ is the start state.
- $A \subseteq S$ is the set of accepting states.

Therefore, Sidhu and Prasanna (2001) proposed a NFA-based automaton which improves the memory utilization problem. All the states in NFA can be active at the same time which needs a prohibitive amount of memory bandwidth. To match a regular expression of size m, the memory required by a serial machine is $O(2^m)$ and requires the time complexity of O(1) per input character. However, the authors proposed a method that requires the $O(m^2)$ space and a text character can be processed in O(1) time. Furthermore, they obtained a simple and fast algorithm that rapidly constructs the NFA for the given regular expression. To construct an NFA rapidly is crucial because the NFA structure depends upon the regular expression, which is known only during runtime.

2.6 Automaton based DPI

The packets are organized in a Network Intrusion Detection System with the help of a predefined rule set and the malicious packets are identified by scanning the packet payloads for any signature in the rule set. Automaton based approaches such as HFA, H-FA, XFA, D^2FA, δFA etc discussed in Prithi et al. (2016) are used extensively for pattern matching or regular expression matching. These approaches can be used along with the IOGA to obtain an optimal solution. To accomplish high throughput in regular expression matching and to reduce memory access frequency is critical for the overall intrusion detection performance based on the metrics such as detection rate, intrusion detection capability, false alarm rate, base rate etc. The classical automaton based Aho-Corasick algorithm (Aho and Corasick, 1975), Commentz-Walter algorithm (Commentz-Walter, 1979) and Wu-Manber algorithm (Wu and Manber, 1994) for string matching have been proposed to reduce memory requirement but these algorithms slow down the matching speed for large pattern set. To solve these problems automaton based DPI approach has been analyzed and discussed to obtain high processing throughput, low implementation cost, reduced memory consumption, high inspection speed, high intrusion detection capability, better detection rate, better false alarm rate and fast convergence.

3. Intelligent optimization grouping algorithms

Intelligent Optimization Grouping Algorithms are used to solve optimization problems (Fu et al., 2014). A variety of intelligent optimization grouping algorithms can be used to allocate the multiple regular expressions into groups and for each group the algorithm constructs DFAs independently to efficiently solve the state explosion blowup by obtaining most favorable global solution between memory consumption and computational time. Typical IOGAs include Genetic Algorithm (GA), Ant Colony Optimization (ACO), Artificial Bee Colony Algorithm (ABC), Bacterial Foraging Optimization (BFO), Particle Swarm Optimization (PSO), Biogeography Based Optimization (BBO), Firefly Algorithm (FA), Bat Algorithm (BA), Cuckoo Search (CK) and Flower Plant Optimization (FPO). In this section the various intelligent optimization grouping algorithms are studied and brief idea of how to apply these algorithms along with finite state automaton are discussed.

3.1 Genetic Algorithm(GA)

GA is an evolutionary based stochastic optimization algorithm that was proposed by Holland in 1975 (Koza, 1992; Holland, 1975). They go behind the doctrines of Charles Darwin Theory of survival of the fittest (Darwin, 2007). In GA, the population is defined to be the group of individuals. A chromosome is a collection of genes which corresponds to individual population. Each individual chromosome finds a feasible solution for the optimization problem. The algorithm starts by initializing a population for the solution. Then for each chromosome the fitness is evaluated by means of a proper fitness function appropriate for the problem.

Based on this, the best chromosomes are selected, where they go through crossover and mutation thus giving new set of solutions which are also known as offspring.

The most important genetic operators in GA are the selection, crossover, and mutation. The selection procedure selects the good individuals from the current population and generates next population based on the assigned fitness by implementing the survival-of-the-fittest and natural selection principle. Crossover also known as the recombination operator replaces parents and merges these parts to obtain new individuals known as children by means of crossover probability. Mutation modifies some portion of individuals to generate perturbed solutions. It operates only on a single individual while crossover operates on two or more individuals.

The pseudo code to implement genetic algorithm is as follows:

1. Select initial population
2. Estimate the individual fitness
3. Repeat
4. Choose the best ranking individuals to reproduce
5. Obtain new generation through crossover and mutation and reproduce children
6. Evaluate the individual fitness of the children population
7. Restore the best ranking individuals
8. Until terminating condition is met.

In case of complex search space or the search space is not clearly defined and when mathematical investigation is not available GA provides an efficient result (Mabu et al., 2007). GAs has the ability to locate good solutions faster for the complicated search space but converges towards the local optima when the fitness function is not defined properly. The main problems of GAs are it does not operate on dynamic data and selecting encoding and fitness function is very difficult. Certain simple optimization algorithms obtain a better solution for certain optimization problems than GA when the same amount of computation time is given. Finally GAs does not have the ability to solve constraint optimization problems.

3.1.1 Genetic Algorithm and Finite State Automata

Genetic algorithm can be applied in Finite State Automata by considering the population as the collection of individuals. Each individual consists of one regular expression and for every individual; fitness function is calculated based on the total number of states. Each possible path in the search space is encoded in the form of a binary string which represents the chromosome c. This chromosome represents the distribution of the present individual i.e. distribution of regular expression. The chromosomes length is denoted as N. Each genes value represents the serial number of the corresponding regular expression that is distributed into the groups.

After initializing the individuals, each regular expression is distributed randomly to allow space for the entire range of the search space. The fitness of the population is calculated based on the number of states. The individuals are selected by the standard selection schemes such as proportionate selection, linear ranking selection, tournament selection and genitor selection (Goldberg and Deb, 1991). Select the distinct chromosome individuals by probabilistic method and perform crossover and mutation. The operations of crossover and mutation guarantee that this approach does not fall into local optimum. Since the method can find global optimal solution in distributing the regular expression, the system reduces memory consumption, improves compression rate, increases convergence speed and reduces number of groups.

3.2 Ant colony optimization

Ant Colony Optimization (ACO) is a population based search technique that takes inspiration from the manners of real ants proposed by the Italian scholar Dorigo et al. (1991). The inspirational behavior for ACO is the way ants discover the shortest paths between the nest and the food source. This behavior is known as ant's foraging behavior. Initially ants travel around the place neighboring their nest in an indiscriminate way in search of food. Ants depart a substance known as pheromone on the ground. When searching the way the ants select the path that contains strong pheromone concentration. Once the source of the food is found it calculates the quality and quantity of food and takes some back to their nest. The pheromone trails leads a way for the ants to find the food source. Ants coordinate their activities by means of stigmergy which is an indirect communication intervened by the changes in the environment in which they move.

As a general rule, there are three principle operations that must be repeated until a feasible solution is obtained or a terminating condition is satisfied.

1. BuildSolutions. Each ant travels around the graph in a definite way searching for food. Based ont eh heuristic and pheromone value of the edge the ant decides the subsequent edge to visit. After selecting the edge, it attaches to its path and continues to move to the next node. Generally, it keeps on searching the graph to check whether an entire solution to the problem has been constructed.
2. PheromonesUpdate: On all the edges of the graph the pheromone values are updated. If the edge is been travelled by the ant the pheromone value is incremented or else the pheromone value is decremented. The fitness function is used to calculate the amount of pheromone that each ant drops on the graph edges.
3. DaemonActions: Individual ants cannot perform actions that are executed by certain procedure. Local optimization is one such procedure.

The pseudo code for ACO algorithm is as follows:

1. Start
2. Initialize the parameters and pheromone trails
3. Generate population of k solutions;
4. Calculate fitness(n) for each individual ant n ε k
5. For each ant find out its best position
6. Determine the best global ant
7. Revise the pheromone trail
8. Until terminating condition is satisfied
9. End

ACO can be used in dynamic applications such as circuit switched networks, network routing applications, scheduling problems, etc. It has optimistic feedback which directs to fast innovation of good solutions and the dispersed computations avoid early convergence. It performs better against the genetic algorithm. Convergence is certain, but time to convergence is uncertain. Coding is not simple and theoretical analysis is difficult.

3.2.1 Ant Colony Optimization and Finite State Automata

Initially, one regular expression is randomly selected by each of m ants. Then one of the ungrouped regular expressions is randomly selected by each ant. The probability of including that to the current group is calculated. Check whether the group condition is met. If so add the regular expression into the current group and the pheromone value between each of the current group and regular expression is increased. Else the previous step is repeated until at least one regular expression is added. If none of the regular expression meets the criteria, one regular expression from the ungrouped expression is selected randomly and considered as the start of the next round grouping. When the destination is achieved the best solution is recorded, the pheromones are evaporated and all m ants are placed at the starting point. The algorithm ends after deterministic iterations. The final solution is found to be optimum and is superior to the solutions recorded in round iteration (Janakiriman and Vasudevan, 2009).

The advantage of grouping regular expressions with probability value can avoid the convergence to a local optimum solution, and the pheromones introduction can make the positive feedback easy and can increase the speed of convergence, improve computational time and reduce the number of groups.

3.3 Particle swarm optimization

Particle Swarm Optimization (PSO) is a biologically inspired algorithm evolved by Kennedy et al in 1995 that simulates the behavior of bird gathering, fish schooling or swarming of bee communication in search for food. PSO is broadly implemented in a variety of fields for optimization and design applications (Yang, 2010). PSO is proved to be better than Genetic Algorithm (GA) (Robinson and Rahmat-Samii, 2004) with a very simple concept of calculation and design pattern of only some computational codes. The most important advantages of PSO is that it has a very few parameters to regulate and able to compute a solution as GA. Further PSOs are more appropriate for online parameter tuning (Hopcroft et al., 2001). The primary idea of PSO is not just only on fish/bee/bird swarming behavior, it is also related to reproduce human social behavior which is the main factor of abstractness.

The particles position is inclined by velocity. Let $x_i(t)$ represent the position of particle in the search space at time t step; where t represents discrete time steps. The position of the particle is changed by including velocity $v_i(t)$, to the current position:

$$x_i(t+1) = x_i(t) + \underline{v_i}\,(t+1) \tag{1}$$

where velocity is given by:

$$v_i(t) = v_i(t\text{-}1) + \underline{c_1}\,\underline{r_1}\,(\text{local best}\,(t) - x_i(t\text{-}1)) + \underline{c_2}\,\underline{r_2}\,(\text{global best}\,(t) - x_i(t\text{-}1)) \tag{2}$$

With acceleration coefficients c_1 and c_2, and random vectors r_1 and r_2.

PSO algorithm determines its candidate solutions by making use of the objective function and it functions upon the resulting fitness value. The position, fitness evaluated, candidate solution and velocity is maintained by each particle and it remembers the individual best fitness and individual best candidate solution. It also maintains the global best fitness and the candidate solution that attained this fitness is called as global best candidate solution.

The following steps are continued in a PSO algorithm until the terminating condition is satisfied.

1. Initialize the population, particle position and velocity.
2. Assess the localbest location i.e. the fitness of the individual particle.
3. Keep track of the individual highest fitness i.e. the globalbest location.
4. Adjust velocity based on localbest and globalbest location.
5. Revise the velocity and particle position.
6. End if condition is satisfied else go to step 3.

One of the prime advantages of basic particle swarm optimization algorithm is its intelligence (Bai, 2010). It is used in both scientific research and engineering purpose. PSO does not have overlapping and mutation calculation. The process takes place by considering the particles speed. The exploration speed is very fast. In PSO, all the calculations that are done are very simple and can be completed easily.

The limitations of the basic particle swarm optimization algorithm are the methods effortlessly endure from the partial optimism, which makes the regulation of its speed and the direction less exact. Spreading cannot be done and the method cannot effort out the troubles of non-coordinate system, such as the result to the energy field and the moving rules of the particles in the energy field.

3.3.1 Particle Swarm Optimization and Finite State Automata

The population or the search space represents the set of regular expressions. Each particle is a regular expression and the group of particle is called as swarm. At each step of the PSO algorithm, the swarm size is fixed or varied depending upon the convergence speed. The algorithm works by randomly choosing a regular expression from the search space and the fitness function is computed for each particle and uses this function to evaluate the candidate solution (Wang et al., 2009). The position, fitness value and velocity of each particle are maintained. The best fitness and the candidate solution are also remembered among all the particles in the swarm. PSO parameters such as the velocity and position are adjusted and another particle or regular expression is chosen. The same process is continued till the convergence conditions are reached.

The main advantage of grouping regular expression through PSO is that all the particles are likely to come together to the best solution quickly and information sharing is done only one way through globalbest. Thus it will provide global optimal solution with increased convergence speed, high throughput, improved time complexity and reduced number of groups.

3.4 Bacterial foraging optimization algorithm

Bacterial Foraging Optimization Algorithm (BFOA) is a bio-inspired optimization algorithm that was proposed by Passino (2002). The key idea of this algorithm is the group foraging strategy of a swarm of Escherichia Coli bacteria in multi modal function optimization. Each bacterium searches for food in order to maximize the energy E per unit time T. Foraging is an occurrence of a colony of bacteria instead of individual behavior. The foraging strategy of the E-coli is administered by four core operations.

- **Chemo tactics**

The movement of E-coli is simulated through swimming and tumbling. The bacterium either swims or tumbles in the same direction or changes the direction. Let S represents total number of bacterium, information of the ith bacterium is represented by $\theta_i = (\theta_1^i, \theta_1^i, ..., \theta_D^i)$ where i =1,2,…,S. $\theta^i(j,k,l)$ represents the ith bacterium at jth chemo tactic, kth reproductive and lth elimination and dispersal step.

- **Reproduction**

After the chemo taxis process is completed each bacterium's fitness is estimated. The sum of cost function is calculated as $J_{fitness}^i = \sum_{j=1}^{N_c} P^{i,j,k,l}$ where N_c represents the total number of steps in complete chemo taxis process. The bacterium which has the least value dies and the healthier bacteria asexually splits the bacterium into two which are then placed in same location. This process maintains the swarm size constant.

- **Elimination and Dispersal Operation**

Elimination operation occurs when the bacterium gets fascinated into local optima and it improves the ability of global search. The dispersion operation occurs when the bacteria gets randomly positioned in the environment.

- **The Swarming**

As the bacteria move it sends a signal to other bacteria to swarm towards it. Each bacterium signals the bacteria to maintain a safe distance between each other.

The pseudo code for the BFOA algorithm is given below:

1.Initialize the population and parameters.
2.For each variable in elimination dispersal loop
3.For each variable in reproduction loop
4.For each variable in chemo tactic loop
    a. Compute the fitness function
    b. Generate a random vector and apply tumbling
    c. Move in the direction of the tumble
    d. Compute the fitness function
    e. Swim and again calculate the fitness function
5.If chemo tactic loops terminating condition is not satisfied go to step 4.
6.Reproduction
    a. Calculate heath cost for each bacterium

b. Sort bacteria and chemo tactic parameters in the increasing order of health cost.
c. The highest health cost value is died and the remaining bacteria with best value are splitted.

7. Repeat till the reproductions loop terminating condition is satisfied else go to step 3
8. Elimination dispersal
   a. Eliminate and disperse each bacterium
9. Repeat till the elimination dispersals terminating condition is satisfied.

The main advantage of BFOA is that it is not affected by size and non-linearity of the problem. It has less computational burden, global convergence, less computational time is needed and can handle more number of objective function. The disadvantages of the algorithms are that the elimination step makes the bacterium which obtained optimal solution to escape which slows down the convergence speed.

3.4.1 Bacterial Foraging Optimization Algorithm and Finite State Automata

In BFO the population is represented by the set of regular expression. The fitness function is calculated and the set of regular expression is grouped based on the tumble move. Again the fitness value is estimated and the regular expression is grouped based on the swim move. The health cost for each group is calculated and the highest health cost value of the group is eliminated and regrouped. The process is repeated until the optimum group is formed. FSA using BFO algorithm will reduce the memory size and will provide a high throughput. It also will provide a better computational and regular expression grouping time. It does not improve the convergence speed as the optimal group that is formed is escaped in elimination process.

3.5 Artificial bee colony algorithm

Artificial Bee Colony (ABC) algorithm was proposed by Karaboga D in 2005 in favor of the numerical optimization problems (Karaboga, 2005). It is a swarm based meta-heuristic optimization algorithm developed by (Tereshko and Loengarov, 2005; Karaboga and Akay, 2009) based on the intelligent foraging manners of honey bee swarm. There are three essential components of forage selection (Basturk and Karaboga, 2006). The first component is the food source. The food sources values depends on the factors such as its proximity to the nest, its energy, and the easy extraction of this energy. The second one is the employed foragers which are related with a scrupulous food source. The information concerning the particular source, its distance and direction from the nest, the profitability of the source are transmitted and this information is distributed with a firm probability. The last part is the unemployed foragers which constantly search for a food source to exploit. Communication between bees related to the quality of food sources acquires place in the dancing area which is called as waggle dance.

The ABC search cycle comprises of three rules (Chan and Tiwari, 2007). The first one is to send the employed bees to a food source, to evaluate the nectar quality, secondly onlookers choose the food source from the employed bees to calculate the nectar quality and finally to formulate the scout bees and to distribute them to possible food sources.

The following are the main steps of the ABC algorithm.

1. Initialize population, optimization problem parameters and Food Source Memory (FSM)
2. Repeat
3. Position the employed bees on their food sources
4. Position the onlooker bees on the food sources based on their nectar amounts
5. Drive the scouts to the search area for determining new food sources
6. Remember the best food source that are found so far
7. Until terminating conditions are satisfied.

Each cycle of the search comprises of three steps. The first step is to send the employed bees onto their food sources and the amount of nectars are calculated and secondly the nectar information of food sources are shared and after that the food source regions are selected by the onlookers and the nectar amount of the food sources are evaluated and the last step is to determine the scout bees and are sent randomly onto promising new food sources. Repeat these steps throughout a predetermined number of cycles known as Maximum Cycle Number (Bahriye and Dervis, 2012) or until a terminating condition are satisfied.

Some of the advantages of ABC algorithm are strong robustness, convergence is fast, highly flexible, fewer setting parameters (Civicioglu and Besdok, 2011), ease of hybridization with other optimization techniques (Karaboga and Akay, 2009) such as GA, PSO, Differential Evolution, Evolutionary Strategies and easily implemented with common mathematical and logical operators. Few disadvantages of ABC are in later search period there are chances of premature convergence and it sometimes cannot satisfy the requirements of the accuracy of optimal value.

ABC remains an interesting and promising algorithm, which are comprehensively used by researchers athwart various fields. Its main advantage is that it can be easily hybridized with various meta-heuristic algorithms and is robustly feasible for continued utilization and improvements are possible.

3.5.1 Artificial Bee Colony Algorithm and Finite State Automata

In ABC, the bees are considered as the set of regular expression and the values are randomly assigned. The parameters of the optimization problem and the food source memory are initialized. The set of regular expression is divided into two groups and initialized to employed bees and onlooker bees. For each regular expression the fitness value is calculated and based on the fitness value the probability to form the group is determined. Based on the probability the count of onlooker bees that are sent to the sources of food of employed bees is calculated. The fitness for each onlooker bee is calculated and based on this value the best

onlooker bee is found and replaced with respective employed bee and the best feasible onlooker is found and replaced with best solution. The iteration is repeated until optimal groups are formed.

When compared to other optimization algorithms ABC is fast and uses less time to group the regular expression. Because of its fast convergence and fewer setting parameters, FSA using ABC will provide limited memory utilization and have faster convergence speed.

3.6 Biogeography based optimization (BBO)

Biogeography based optimization is an optimization technique which was proposed by Simon (2008). It is based on the geography concept of biological organisms. The algorithm works on the principle of migration and mutation. The two main processes in migration are immigration and emigration. Immigration and emigration are affected by the factors such as distance of an island to the nearest neighbor, size of the island, habitat suitability index (HSI) etc. HSI involves various factors such as rainfall, vegetation, climate etc. These factors favor the existence of species in a habitat. Mutation is the sudden drastic change made to the HSI of any habitat due to certain cataclysmic events. Mutation increases the diversity among the population.

The pseudo code of the BBO algorithm is given as:

1. Define the migration probability and mutation probability
2. Initialize the population.
3. Determine the immigration and emigration rate of each candidate in the population.
4. Select the island to be modified based on the immigration rate.
5. Using roulette wheel selection, select the island from which the SIV is to be emigrated.
6. Randomly select an SIV from the selected island to be emigrated.
7. Perform mutation based on the mutation probability of each island.
8. Estimate the fitness of each individual island. If the fitness criterion is not satisfied go to step3.

Immigration rate Ri can be defined as

$$Ri = I\,(1-F(s)/n)$$

Emigration rate Re can be defined as

$$Re = E\,(F(s)/n)$$

where I is the maximum immigration rate, E is the maximum emigration rate, F(s) is the fitness rank of solution s and n is the number of candidate solutions in the population.

BBO algorithm does not take unnecessary computational time. The solutions do not die at the end of each generation like other optimization algorithms. The exploration capability of BBO makes it attractive for solving many complex problems in various fields but the exploiting capability is very poor. It does not have provision for selecting the best members from each generation. In some cases infeasible solutions are generated.

3.6.1 Biogeography Based Optimization Algorithm and Finite State Automata

In BBO, the population is considered as the set of regular expression and the values are randomly assigned. The probability of migration and mutation are defined. For each regular expression the immigration rate and emigration is calculated and based on the values the probability to form the group is determined. For each regular expression the SIV is calculated and the random value is selected to perform emigration and based on the SIV the group is formed. The fitness value for each group is calculated and the iteration is repeated until optimal groups are formed.

When compared to other optimization algorithms BBO has very less computational time and also can group the regular expression in very less time. The exploration capability of BBO can make the FSA to be designed easily and complex regular expression can be easily grouped.

3.7 Cuckoo search

Cuckoo Search is a meta heuristic bio-inspired optimization algorithm deployed by Yang and Deb S. (2009). Cuckoo Search is inspired by obligate blood parasitism behavior of cuckoo's by laying their eggs into nest of host birds and by characteristics of Levy Flights. The Cuckoo Search algorithm is implemented by three basic characteristics.

a. A Cuckoo randomly selects a nest to lay the egg and it lays egg at a time.
b. The very good quality eggs nest is considered as best nest and it is carried to the next generation.
c. The available host nests value is fixed and the probability of a host bird to discover cuckoo's egg is $P_a \varepsilon [0,1]$. In this scenario the host bird destructs the egg or the nest completely and goes for a new nest in new location.

These three characteristics are represented in simple form by representing each egg in a nest by a solution and each cuckoo signifies a new solution. The objective is to make use of the new and most likely superior solution to alternate a least solution in the nests. Pseudo code for cuckoo search is given below:

1. Begin
2. Define objective function f(X), where $X = \{x_1, x_2 \ldots, x_d\}$
3. Generate initial population of host nests x_i, where i=1,2,3,…,n.
4. While terminating condition is not met
5. Generate a cuckoo selection randomly using Levy Flight.
6. Evaluate the fitness function, F_N

7. Select a random nest j among n
8. If(F_N >Fj) Replace j by new solutions
9. Discard worst nest with a fraction of P_a and generate new nests.
10. Maintain the best solutions
11. Grade the solutions and find the current nest
12. End While
13. Post process results and visualization.

The main advantage of Cuckoo Search is that it deals with multi criteria optimization. It is simple and easy to implement. It aims to speed up the convergence and it can be hybridized with other swarm based optimization algorithms.

3.7.1 Cuckoo Search and Finite State Automata

The population of the cuckoo search is represented by the set of regular expression. The cuckoo is selected randomly using Levy Flight from the set of regular expression. Calculate the fitness function and the nest is selected randomly. The worst nest is discarded and the new nest is generated and based on this the regular expression is grouped. Maintain the best solution and find the current group. Repeat the process till optimal regular expression groups are formed. It is easy and simple to design FSA using Cuckoo Search. FSA with Cuckoo search will increase the convergence speed and it can be easily hybridized with other optimization algorithms to provide improved memory consumption, throughput and better inspection speed

3.8 Firefly algorithm

Firefly algorithm that was projected by Yang (2009) is an unconventional swarm based heuristic algorithm for constrained optimization problems inspired by the flashing behavior of fireflies. The algorithm is represented by a population-based iterative process with various fireflies simultaneously taking care of a considered optimization problem. Agents correspond to each other through bioluminescent sparkling which allows the cost function space to be investigated more efficiently than in standard distributed random search. The firefly algorithm comprises of three idealized rules (Farahani et al., 2011; Yang, 2010) which depends on the basic flashing distinctiveness of real fireflies. The first rule is unisex behavior of all fireflies that moves toward more attractive and brighter ones. Secondly, the attractiveness of firefly is relative to its brightness. Lastly, the light intensity of the firefly is obtained by the objective function of the specified problem.

The two main issues of firefly algorithm are the formulation distinction of light intensities, I and attractiveness, β. The attractiveness is relative that varies by distance among firefly i and firefly j. Light is also absorbed by medium and diminishes by increasing distance, therefore the attractiveness also varies with a factor of absorption.

The light intensity I varying with distance r is expressed by the following equation:

$$I(r) = I_0 e^{-\gamma r^2} \tag{3}$$

whereas I_0 denotes intensity of the light at the source, and γ is a coefficient of fixed light absorption. The attractiveness β is defined as:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{4}$$

With β_o is the attractiveness when r is 0 and is a defirepresents the light absorption coefficient.

The distance between two fireflies i and j is represented by the Euclidian distance

$$r_{ij} = | S_i - S_j | = \sqrt{\sum_{k=1}^{n} S_{ik} - S_{jk}} \tag{5}$$

=

where S_{ik} is the position of the k[th] element of the i[th] firefly within the search space. Each firefly i moves to the more attractive firefly j, as follows:

$$S_i = S_i + \beta_0 e^{-\gamma r_{ij}^2} (S_j - S_i) + \alpha \varepsilon_i \tag{6}$$

The first term in eq. (6) specifies the i[th] firefly's position. The next term represents the i[th] firefly's attractiveness and the last term refers the randomized move with the randomized argument α and the random number $\varepsilon_i \in (0, 1)$.

The firefly algorithm is given as

2.Initialize general counter, attractiveness and best solution
3.Initialize the population of firefly
4.While t is less than or equal to MAX_GEN do
5. Find out new value of α
6. Evaluate $S_i^{(t)}$ according to $f(S_i)$
7. Sort $P_i^{(t)}$ according to $f(S_i)$
8. Determine best solution S^*

9. Vary attractiveness according to S_i
10. End While
11. Post process.

Firefly algorithm can deal with highly non-linear multi-modal optimization problems proficiently. The speed of convergence of FA is very high in terms of finding the global optimized respond. It has the flexibility of integration with other optimization techniques such as GA and PSO to form hybrid tools. Firefly algorithm suffers from diminishing returns once the swarm size grows or when the solution space grows immensely large and is impossible to predict the future moves of a firefly so there is no opportunity to allow pre-caching of values.

3.8.1 Firefly Algorithm and Finite State Automata

In firefly algorithm the agents or fireflies act as the regular expression and the values are initialized randomly. The regular expressions interact through bioluminescent glowing which allows estimating the cost function space efficiently. The degree of attractiveness is determined by the objective function and the attractiveness varies by distance between two regular expressions. The light intensity is estimated based on the cost function. The best solution is determined and the attractiveness is varied according to the best solution. This process is repeated until the terminating condition is satisfied.

When compared with GA and PSO, firefly algorithm deals with multi modal optimization problem and will use less time to group the regular expression. Firefly algorithm has high flexibility in hybridizing with GA and PSO algorithms. By hybridizing FA with GA and PSO optimization algorithms, it will provide a better memory utilization, high throughput, increased convergence speed and better regular expression grouping time.

3.9 Bat algorithm

Yang (2010) developed Bat Algorithm which was very efficient in terms of frequency tuning, automatic zooming and parameter control. Bat algorithm depends on the echolocation features of micro bats. There are around 1000 distinct types of bats (Tudge, 2000; Yilmaz and Kucuksille, 2013). The sizes are differed broadly, extending from the little bumblebee bat of around 1.5 to 2 grams to the mammoth bats with wingspan of around 2m and may weigh up to around 1 kg. Most bats uses echolocation to a specific degree; among every one of the species, micro bats use echolocation expansively, while mega bats don't. Micro bats typically use a type of sonar, called, echolocation, to sense prey, avoid obstacles, and position their roosting crevices in the dark. They can transmit a very loud sound pulse and listen for the reverberation that bounces back from the encompassing objects. Their pulses range in properties and can be corresponded with their searching procedures, relying on the species. Most bats utilize short, frequency-modulated signals to clear through around an octave, and every heartbeat keeps going a couple of thousandths in the frequency range of 25 kHz to 150 kHz (Yang, 2013).

BA has three salient features (Guang-Qiu et al., 2013) in which the first is it uses a frequency-tuning technique to improve the diversity of the solutions in the population, second feature is that it uses the automatic zooming to aim to balance exploration and exploitation throughout the search process by impersonating the variations of pulse emission rates and loudness of bats when searching for prey. Accordingly, it has quick convergence rate when compared with GA and PSO, and the last feature is that it uses parameter control which can vary the values of parameters such as frequency, velocity, solution, loudness and pulse emission rate as the iterations proceed. This provides a way to automatically toggle from exploration to exploitation as soon as optimal solution is impending.

Based on the above explanation and uniqueness of bat echolocation, Xin-She Yang (2010) built up the bat algorithm with the three idealized rules. The primary rule is that each one bats use echolocation to feel distance, and they also recognize the distinction between prey and heritage limitations in a few supernatural way. Second rule is that bats fly arbitrarily with speed v_i at position x_i with a frequency f_{min}, differing wavelength λ and loudness A_0 to hunt for prey. They can consequently modify the wavelength (or frequency) of their discharges heartbeats and change the rate of heartbeat emission $r \in [0, 1]$, depending on the proximity of their goal. The last rule is in spite of the fact that the loudness can change from multiple points of view; expect that the loudness fluctuates from a substantial (positive) A_0 to a minimal constant value A_{min}.

A. Initialize Bat Population

Initial population is randomly generated from real-valued vectors with the measurement d and number of bats n, by enchanting into account lower and upper limits.

$$x_{ij} = x_{\min\, j} + rand(0,1)(x_{\max\, j} - x_{\min\, j}) \tag{7}$$

where i=1, 2, n, j=1, 2….d, x_{minj} and x_{maxj} denotes lower and upper limits for dimension j respectively.

B. Update Frequency, Velocity and Solution

The frequency element controls step size of a solution in BA. This element is assigned to random value for each bat between upper and lower limits [f_{min}, f_{max}]. Velocity of a solution is relative to frequency and new solution relies upon its new velocity.

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{8}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) f_1 \quad (9)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (10)$$

in which β ϵ [0, 1] shows arbitrarily generated number, x^* represents current global best solutions.

For local search part of algorithm (exploitation) is chosen among the selected best solutions and random walk is applied.

$$x_{new} = x_{old} + \epsilon \overline{A}^t \quad (11)$$

Where $\overline{A}^t$ is average loudness of all bats, ϵ ε [0, 1] is random number and represents direction and intensity of arbitrary-walk.

C. Update Pulse Emission Rate and Loudness

As the cycle continues the loudness and pulse emission rate must be upgraded. When the bats move in the direction of its food then there is an increase in the pulse emission rate and there is a drop in the loudness A. The following equation describes the how the Loudness A and pulse emission rate r are updated.

$$A_i^{t+1} = \alpha A_i^t \quad (12)$$

$$r_i^{t+1} = r_i^0 [1 - e^{(-\gamma t)}] \quad (13)$$

where α and γ are constants. r_i^0 and A_i represents the factors of random values and A_i^0 can generally be [1, 2], while r_i^0 can usually be [0, 1].

The pseudo code for Bat Algorithm is as follows:

1. Initialize bat population x_i and speed v_i , i=1, 2,..n
2. Describe frequency of pulse f_i at x_i
3. Compute loudness A_i and pulse rate r_i
4. while
5. Generate new solutions by adjusting frequency, velocities and solutions.
6. Choose a solution from the best solutions
7. Generate a local solution from the chosen best solution
8. Accept new solutions
9. Reduce loudness A_i and increase pulse rate r_i
10. Ranks the bats to obtain the current best x*
11. Until terminating condition is satisfied.

In this algorithm the behavior of bat is captured into fitness function of problem to be solved. The algorithm is very straightforward, flexible and simple to implement. It solves a wide range of non linear problems more efficiently and at initial stage it provides a very fast convergence by toggling from exploration to exploitation. Automatic control and auto zooming mechanism can be achieved by the loudness and pulse emission rate. If the algorithm enters the exploitation phase very rapidly, then stagnation may occur after some initial stage.

3.9.1 Bat Algorithm and Finite State Automata

In FSA designing, the bats are considered as the set of regular expressions. The number of regular expressions, the velocity, pulse rate, frequency and loudness are initialized. The frequency, velocity and locations are updated. The procedure is repeated till the terminating conditions are met and the best solution is attained. When compared to GA and PSO optimization algorithms bat algorithm is simple, easy to implement and solves non linear problems effectively. Thus it will provide low implementation cost, high processing throughput, reduced memory utilization, high detection rate and fast convergence.

3.10 Flower pollination algorithm

Xin She Yang in 2012 proposed the flower pollination algorithm (Yang, 2012) which is a bio-inspired optimization algorithm that evolved from the idea of processes of flower pollination. The flower pollination algorithm is formed and structured by four pollination policies based on the stimulation of flower plants. The four policies are defined as:

1.In global pollination process, biotic and cross-pollination are considered and pollinators that carry pollens move by obeying levy flights.
2.In local pollination process, biotic and self pollination are considered.
3.Insect pollinators expand flower constancy and are equal to the probability of reproduction which is relative to the match between two flowers that are involved.
4.The local and global pollinations interactions are controlled by a switch probability between 0 and 1.

The algorithm to perform flower pollination optimization is given below.

1.Define objective function
2.Initialize the population of n flowers
3.Find the best solution in the population

4.Define switch probability
5.Define stopping critera
6.While
7.For all n flowers in the population
    a. If rand<p
    b. Draw a d-dimensional step vector L that obeys Levy distribution and perform global optimization
    c. Else
    d. Draw from uniform distribution and perform local optimization
8. Evaluate new solutions
9. Update the better solutions in the population
10. Find current best solution
11. Repeat till the terminating condition is satisfied.

Flower Pollination Algorithm is very efficient than GA and PSO due to the two key components long distance pollinators and flower consistency. The pollinators can travel long distance and they have the ability to escape any local landscape and can explore larger search space. Flower consistency ensures that same pieces of flowers are chosen more frequently and guarantees convergence more quickly. These two components ensure that the algorithm is very efficient.

3.10.1 Flower Pollination Algorithm and Finite State Automata

Initialize the population by representing the set of regular expression as the flowers. Based on the regular expression define the switch probability and the stopping criteria. For all the set of regular expressions in the population perform global and local optimization using Levy distribution and uniform distribution. Evaluate new solutions and form groups. Update the group by better solutions and repeat this procedure until optimal solution is obtained. When compared to GA and PSO, the FPA is very efficient in terms of convergence and space because of long distance pollinations and flower consistency.

4. Discussions and proposed approach

The main reason that was analyzed from the finite state automata is that the number of states gets increased due to the state explosion problem. The problem with explosion in states can be competently alleviated by grouping the regular expression. Grouping the regular expression falls into two cases. The first one is if number of groups is given, the DFA states can be minimized and second is if the maximum number of DFA states is known, number of groups can be minimized (Back, 1996). The performance of the regular expression grouping method is assessed by considering these two cases based on the various practical demands.

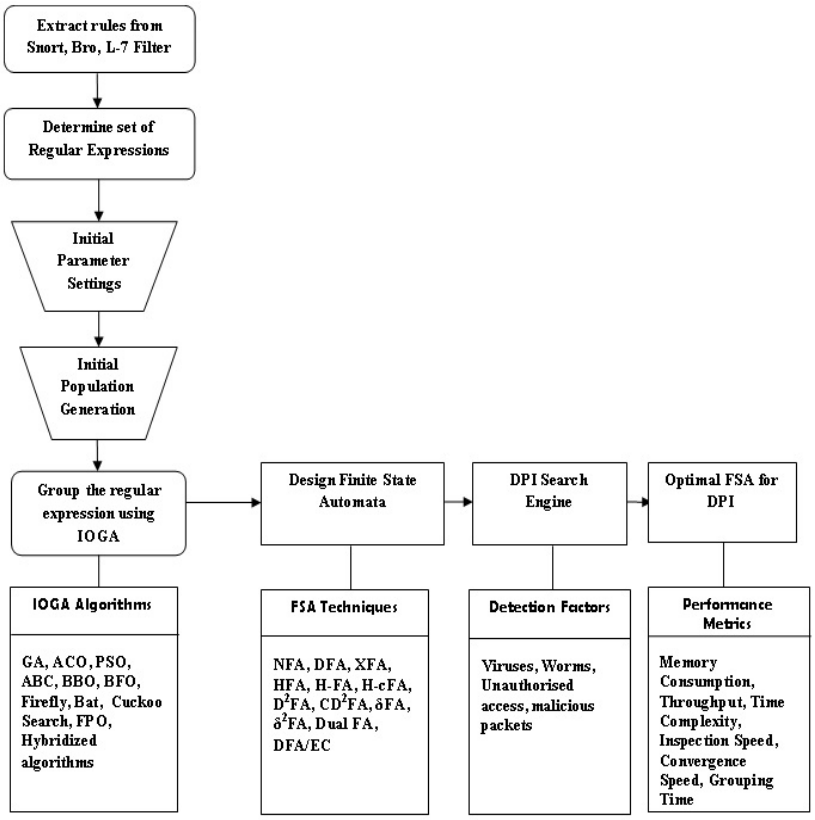

Fig.1. Flow graph of Proposed Approach

Grouping the regular expression can be done by Intelligent Optimization Grouping Algorithms. The various types of intelligent optimization grouping algorithms discussed in section 3 were studied and analyzed. These intelligent optimization grouping algorithms can be used to optimize the finite state automata and can efficiently alleviate the problem of state blowup thereby it will obtain the universal best distribution among the consumption of memory and number of groups. In Fig.1 the overall flow graph of the proposed approach for Finite State Automata in Deep Packet Inspection which is used to group the regular expression efficiently using the Intelligent Optimization Grouping Algorithms and memory efficient automata is shown. The packet payload files are extracted from the rule sets such as Snort (Roesch, 1999), Bro NIDS (Paxson, 1998), and Linux L-7 filter (Levandoski et al., n.d) are used as input. The set of regular expressions are determined using these packet payload files. Initially the parameters are assigned depending on the Intelligent Optimization Grouping Algorithms such as GA, ACO, PSO, BFO, BBO, CS, ABC, Firefly, Bat, FPA and the initial population is generated by randomly distributing the regular expression on the search space. As discussed in section 3 each IOGA algorithm has certain advantages and disadvantages and as discussed GA and ACO algorithms are already applied by authors in Fu et al. (2014) to group the regular expression. They found that by applying GA and ACO a global optimal solution can be obtained, the convergence to local optimum is avoided, accelerates convergence speed and reduces memory consumption but obtains in efficient time and the time cost increases exponentially with the number of regular expression groups. Hence by appropriately selecting IOGAs there will be a variation in the performance. The various IOGA algorithms can also be hybridized and applied with FSA so as to improve the performance metrics such as memory, throughput, time complexity, grouping time, inspection speed and convergence speed. Based on the performance of the IOGAs the parameters listed in Table.2 are adjusted and the process is continued until the optimal groups are formed or until the maximum iteration is obtained. Once the optimal solution is obtained, the regular expressions are grouped and the finite state automata is designed based on the design technique (Prithi et al., 2016) such as HFA, H-cFA, XFA, δFA etc and FSA is integrated with the DPI search engine to identify the packets that hold the viruses, unauthorized access and attacks. By using Intelligent Optimization Algorithms with Automaton based DPI, the state explosion problem can be reduced and the overall performance of the system can be enhanced. A comparative study about various IOGA's for FSA based DPI is analyzed and the anticipated advantages and disadvantages for each grouping algorithm are highlighted in Table 3.

Table 2. Parameters used in various IOGA Algorithms

Genetic Algorithm	Ant Colony Optimization	Particle Swarm Optimization	Artificial Bee Colony	Firefly Algorithm	Bat Algorithm	Cuckoo Search	Bacterial Foraging Optimization	Biogeography Based Optimization	Flower Pollination Algorithm
Number of Chromosome	Number of Ants	Population Size	Colony Size	Population of Fireflies	Population of Bat	Search Dimension & Number of Host Nests	Number of Bacterium	Population Size	Leaf area index of each plant & Specific leaf area
Chromosome size	Weight of Pheromone Trail	Initial Position	Number of Food Sources	Light Intensity	Initial Velocity	Tolerance	Maximum Number of Steps	Number of Habitats	Light Extinction coefficient
Number of generations	Weight of Heuristic Information	Initial Velocity	Food Source Limit	Absorption Coefficient	Pulse Frequency	Discovery Rate	Number of Chemo tactic, elimination dispersal & reproduction steps	Maximum Immigration & Emigration Rate	Maintenance Respiration
Selection Method	Pheromone Evaporation Parameter	Cognitive Factor	Number of Employed bees	Attractiveness	Pulse Emission Rate	Upper & lower bounds of search domain	Elimination Dispersion Probability	HIS and SIV Value	Leaf Carbon content
Crossover & Mutation Type	Initial Pheromone Level	Social Factor	Number of Onlooker bees	Maximum Number of Generations	Loudness	Levy Exponent & Step Size	Size of the Step	Number of elites	Light use efficiency
Crossover & Mutation Rate	Constant for Pheromone updating	Maximum Number of Iterations	Maximum Number of Iterations	Maximum Number of Iterations	Maximum Number of Iterations	Maximum Number of Iterations	Swimming Length	Maximum Number of iterations	Fraction of Mass in Leaves

Table 3 Comparative study of various IOGAs for FSA Based DPI

Intelligent optimization grouping algorithms	Proposed year	Proposed by	Algorithm basis	Objective function defined by	Advantages of IOGA for FSA based DPI	Disadvantages of IOGA for FSA based DPI
Genetic algorithm	1975[9, 38]	Holland	Principles of evolution	Collection of genes or chromosome	Reduces memory consumption improves compression rate, reduces number of groups	Converges towards local optima, does not operate on dynamic data
Antcolony o**ptimization	1992[10]	Marco Dorigo, VittoriManiezzo, Alberto Colorni	Interaction of ant species	Pheromone value	Increases convergence speed, improves computational time, reduces number of group and memory space	Sequence of random decisions is dependent, time to convergence is uncertain, theoretical analysis is difficult.
Particle swarm optimization	1995[1]	Eberhart and Kennedy	Swarming behavior of fish/bee/bird	Particles position and velocity	High throughput, improved time complexity and reduces number of groups	Slow convergence in advanced search stage, does not support scattering and non co-ordinate system problem
Bacterial foraging optimization	2002[12]	Kevin Passino	Foraging behavior of *E. coli* bacteria	Movement of bacteria (swimming/tumbling)	Reduces memory space, better computational and grouping time, high throughout	Slows down the convergence speed
Artificial bee colony algorithm	2005[58]	Karaboga D	Intelligent foraging behavior of honey bee swarm	Food source memory	Improves grouping time, faster convergence speed, limited memory utilization	Premature convergence, not accurate
Firefly algorithm	2008[16]	Xin-She Yang	Flashing behavior of swarming fireflies	Light intensity/brightness and attractiveness	Less grouping time, improves convergence speed, better memory utilization and high throughput	Pre-caching of values not allowed, poor exploiting capability
Biogeography based optimization	2008[14]	Simon D	Geography concept of biological organisms	Immigration rate and emigration rate	Very less computational and grouping time	Exploiting capability is very poor and in some case infeasible solutions obtained.
Cuckoo search	2009[15]	Xin-She Yang, Suash Deb	Brooding parasitism of cuckoo bird	Color of eggs	Increase convergence speed, improves memory consumption and throughput	Number of iterations is high, difficult to find best solution, accuracy is poor
Bat algorithm	2010[17]	Xin-She Yang	Echolocation behavior of mciro bats	Pulse emission rate, loudness frequency and velocity	Reduced memory utilization, high detection rate, high throughput and improves convergence rate	Unable to memorize any history of better solution, unable to get rid of local search complexity
Flower pollination algorithm	2012[18]	Xin-She Yang	Stimulation of pollination process of flower plants	Local and global pollination	Improves memory utilization, high throughput and better inspection speed	Global search ability gets weaker, convergence speed gets worse and the quality of the solution gets weaker for solving multidimensional problems.

The experiments are evaluated in future on the various performance metrics such as memory consumption, throughput, time complexity, inspection speed, grouping time and convergence speed in the optimal FSA based DPI search engine and the expected outcome is analyzed and plotted in fig.2, 3 and 4. Fig. 2 depicts the memory consumption and throughput for the various IOGA algorithms and it shows that the hybridized approach reduces the number of states to a great extent and thereby it will provide a reduced memory consumption and high throughput.

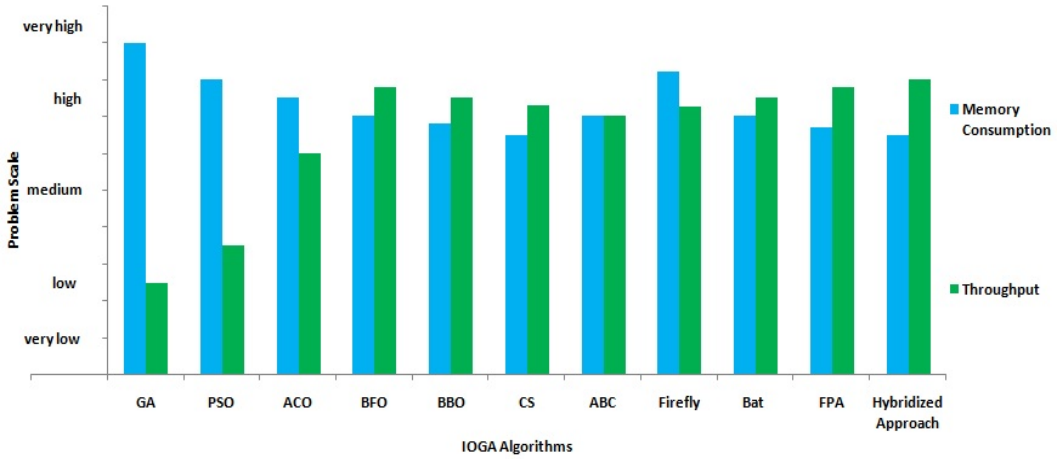

Fig.2. Memory Consumption and Throughput for the IOGA algorithms

Fig. 3 shows the inspection speed and convergence speed for the various IOGA algorithms. As discussed in 3.2.1 and 3.3.1 by grouping the regular expression through ACO/PSO the convergence speed can be increased and it can also improve the inspection speed and thus it depicts that the hybridized approach has high inspection speed and fast convergence speed.

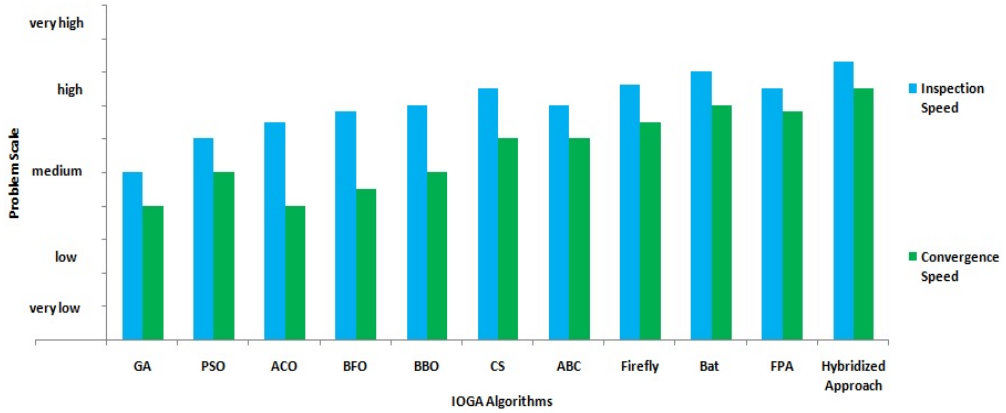

Fig.3. Inspection Speed and Convergence Speed for the IOGA algorithms

Fig. 4 illustrates the time consumption and grouping time for the various IOGA algorithms. It shows that by using the IOGA algorithms such as firefly/ABC the grouping time of the regular expression can be reduced and thus by implementing the hybridized approach it will provide a better time complexity and improved grouping time.

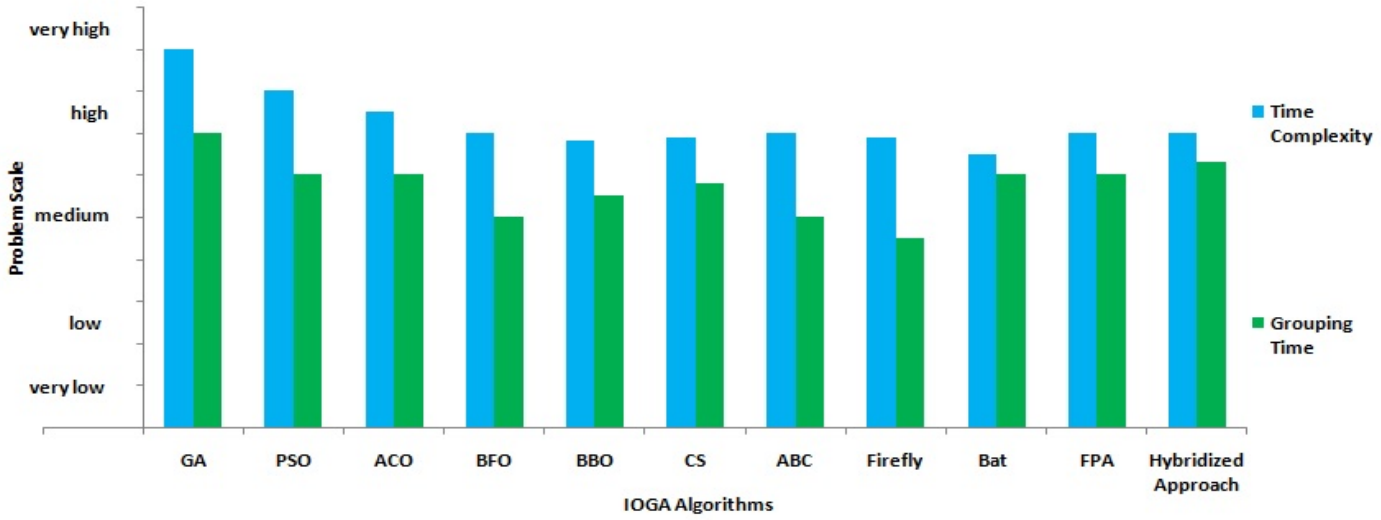

Fig.4. Time Complexity and Grouping Time for the IOGA algorithms

5. Conclusions and future scope

In this paper, the various Intelligent Optimization Grouping Algorithms for FSA in DPI has been surveyed and analyzed. The first discussion that was done is on the basic features of regular expression and how the regular expression works in Deep Packet Inspection. A brief discussion is given on the finite state automata; the problems that occur in designing the FSA and how efficiently automation based DPI approach can be designed. The various Intelligent Optimization Grouping Algorithms such as GA, ACO, PSO, ABC, BBO, BFO, Cuckoo Search, Firefly, Bat and FPA algorithms are presented and how it can be used with FSA are illustrated. The strengths and weakness of each algorithm is studied. GA can perform efficiently for complex space search and GA cannot support constraint optimization problems. Since GA can find global optimal solution in distributing the regular expression, the system can reduce memory consumption and reduce number of groups. PSO occupies bigger optimization ability and all the particles can quickly converge to the best solution and it can provide increased convergence speed, improved time complexity and reduced number of groups. It is also easy to implement but suffers from partial optimism and cannot work out on scattering problems. ACO can be used in dynamic applications and avoids premature convergence. Positive feedback is made easily by introducing the pheromones which results in increase in the inspection speed, improves computational time and reduces memory but convergence time is tentative and coding is not straight forward. BFO algorithm will provide a reduced memory size and high throughput. When used with FSA, BFO will provide a reduced computational and grouping time but does not improve the convergence speed. BBO has very less computational time when compared with other optimization algorithms and can group the regular expression in very less time. It can be easily designed and even complex regular expressions can be easily grouped. CS increases the speed of convergence and can be easily hybridized with other optimization algorithms in order to provide improved memory space, throughput and better inspection speed. ABC is fast and uses less time to group the regular expression. Because of its fast convergence and fewer setting parameters it can provide limited memory utilization and faster convergence speed. Firefly algorithm deals with multi modal optimization problem and is highly flexible in hybridizing with GA and PSO and also provides better memory utilization, high throughput, increased convergence and better regular expression grouping time. When compared to GA and PSO bat algorithm is simple, easy to implement and will provide low implementation cost, high processing throughput, reduced memory utilization, high detection rate and fast convergence. FPA is very efficient when compared to GA and PSO because of its long distance pollinators and flower consistency. These algorithms can be used along with the implementation of finite state automata in DPI so as to reduce the state explosion problem, to provide optimal and memory efficient finite state automata and to provide high throughput along with accuracy at very high inspection speed.

References

Aho A.V., Corasick M.J., 1975. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, Vol. 18, No. 6, pp. 333–340. https://dl.acm.org/doi/10.1145/360825.360855

Bai Q. 2010. Analysis of particle swarm optimization algorithm. *Computer and Information Science*, Vol. 3, No.1. https://doi.org/10.5539/cis.v3n1p180

Basturk B, Karaboga D. 2006, An artificial bee colony (ABC) algorithm for numeric function optimization. *IEEE Swarm Intelligence Symposium*

Back T. 1996: *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming and Genetic Algorithms*. Oxford University Press US, January

Bahriye A., Dervis K. 2012. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, Vol. 23, No. 4, pp.1001-1014

Civicioglu P., Besdok E. 2011: A Conceptual Comparison of Cuckoo-Search, Particle Search Optimisation, Differential Evolution and Artificial Bee Colony Algorithms. Springer Science and Business Media (2011) https://doi.org/10.1007/s10462-011-9276-0

Commentz-Walter B. 1979. A string matching algorithm fast on the average. *Proceedings of ICALP*, pp. 118–132 (1979)

Chan F.T.S., Tiwari M.K. 2007. Swarm Intelligence: Focus on Ant and Particle Swarm Optimization. Itech Education and Publishing, Vienna, Austria. ISBN 978-3-902613-09-7, pp. 532 (2007)

Darwin C. 2007, *On the Origin of Species*. John Murray, sixth edition. http://www.gutenberg.org/etext/1228 (1859). Accessed 05 August 2007

Dorigo M., Maniezzo V. and Colorni A. 1991. *Positive Feedback as a Search Strategy*. Politecnico di Milano, Italy, Technical Report, Report No. 91-016

Farahani S.M., Abshouri A.A., Nasiri B., Meybodi M.R. 2011. A gaussian firefly algorithm, *International Journal of Machine Learning and Computing*, Vol.1, No. 5, pp.448-453 (2011)

Fu Z., Wang K., Cai L., and Li J. 2014, Intelligent grouping algorithms for regular expressions in deep inspection, *IEEE/ACM Transaction on Networking*, Vol. 22, No. 2, pp. 644-651. https://doi.org/10.1109/ICCCN.2014.6911804

Guang-Qiu H., Wei-Juan Z., Qiu-Qin L.: Bat algorithm with Global Convergence for Solving Large-Scale Optimization Problem. Application Research of Computers, Vol. 30, No. 3, 1-10 (2013)

Ghalia,M.B.: Particle Swarm Optimization with an Improved Exploration-Exploitation Balance. 51st Midwest Symposium on Circuits and Systems (2008) https://doi.org/10.1109/MWSCAS.2008.4616910

Goldberg D.E., Deb K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. Urbana, 51 61801-2996 (1991)

Harrison,M.A. 1978, *Introduction to Formal Language Theory*. Addison-Wesley Longman Publishing Co., Inc.

Hopcroft J.E., Motwani R., Ullman J.D. 2001. Introduction to Automata Theory, Languages, and Computation. Second Edition, Addison-Wesley Series in Computer Science, Addison-Wesley-Longman, ISBN 978-0-201-44124-6, pp. I-XIV, 1-521

Hopcroft J. 1971. An nlogn Algorithm for Minimizing States in a Finite Automaton. Stanford University Stanford, CA, USA, STANCS-71-190

Janakiriman, S., Vasudevan,V. 2009, ACO based distributed intrusion detection system. *International Journal of Digital Content Technology and its Applications*, Vol. 3, No.1, pp.66-72

Karaboga D. 2005. *An Idea Based on Honey Bee Swarm for Numerical Optimization*. Technical Report - TR06, Erciyes University Press

Karaboga D., Akay B. 2009: A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation,* Vol. 214, pp.108–132. https://doi.org/10.1016/j.amc.2009.03.090

Kleene, S. C.: Representation of Events in Nerve Nets and Finite Automata. RAND Research Memorandum RM-704, RAND Corporation (1951)

Konar,A. 2005: Computational Intelligence: Principles, Techniques and Applications. Springer, Berlin Heidelberg New York

Holland,J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

Koza J.R. 1992. *Genetic Programming, on the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA

Koza J.R., 1994. Genetic Programming II, Automatic Discovery of Reusable Programs. MIT Press, Cambridge, MA

Levandoski J., Sommer E., Strait M.: Application Layer Packet Classifier for Linux. http://l7-filter.sourceforge.net/

Liu T., Liu A.X., Shi J., Sun Y., and Guo L. 2014. Towards fast and optimal grouping of regular expressions via DFA size estimation. *IEEE Journal on Selected Areas in Communications*, Vol.32, No.10, pp. 1797 - 1809. https://doi.org/10.1109/JSAC.2014.2358839

Mabu S., Hirasawa K., Hu J. 2007, A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning, *Evolutionary Computation*, Vol. 15, No. 3, pp. 369-398. https://doi.org/10.1162/evco.2007.15.3.369

Nitin T., Singh S.R., Singh P.G. 2012. Intrusion detection and prevention system (IDPS) technology - network behavior analysis system (NBAS). *ISCA Journal of Engineering Sciences*, Vol. 1, No. 1, pp.51-56, July (2012)

Paxson V. 1998, Bro: A system for detecting network intruders in real time. *Proceedings of the 7th UNISEX Security Symposium*

Passino K.M. 2002, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Systems*, Vol.22, No. 3, pp. 52-67. https://doi.org/10.1109/MCS.2002.1004010

Prithi,S, Sumathi.S. 2016, A review on deterministic finite automata compression strategies for deep packet inspection. *International Journal of Innovations & Advancement in Computer Science*, Vol. 5, No. 6, pp. 7323-7325

Robinson J., Rahmat-Samii Y., 2004. Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, Vol. 52, No. 2, pp. 397 – 407. https://doi.org/10.1109/TAP.2004.823969

Roesch M. 1999. Snort: Light weight intrusion detection for networks. *Proceedings of the 13th USENIX Conference on System Administration*, pp. 229-238

Rohrer J., Atasu K., Van Lunteren J., Hagleitner C. 2009. Memory-efficient distribution of regular expressions for fast deep packet inspection. *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, pp. 147–154. https://doi.org/10.1145/1629435.1629456

Sidhu R., Prasanna V.K. 2001. Fast regular expression matching using FPGAs. *Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 227-238

Simon D. 2008. Biogeography-based optimization, *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 6, pp. 702-713, December 2008

Tereshko V. and Loengarov A. 2005. Collective decision making in honey-bee foraging dynamics. *Computing and Information Systems*, Vol. 9, No. 3, p. 1

Tudge C.: The Variety of Life. Oxford University Press, Oxford (2000) ISBN: 0-19-850311-3

Wang J., Hong X., Ren R.-R., Li T.-H. 2009. A real-time intrusion detection system based on PSO-SVM. *Proceedings of International Workshop on Information Security and Application*

Wu S., Manber U. 1994. A Fast Algorithm for Multi Pattern Searching. Technical Report TR-94-17, University of Arizona

Yang X.-S. 2009. Firefly Algorithm for Multimodal Optimization. Proceedings of the Stochastic Algorithms, Foundations and Applications (SAGA 109), Lecture notes in Computer Sciences Springer, Vol.5792. https://doi.org/10.1007/978-3-642-04944-6_14

Yang X.-S. 2013. Bat algorithm: Literature review and applications. *International Journal on Bio-Inspired Computation*, Vol. 5, No. 3, pp. 141–149. https://doi.org/10.1504/IJBIC.2013.055093

Yang X.-S., 2010, Nature-Inspired Metaheuristic Algorithms. Second Edition, University of Cambridge, Luniver Press

Yilmaz S., Kucuksille E.U. 2013, Improved Bat Algorithm (IBA) on Continuous Optimization Problems. Lecture Notes on Software Engineering, Vol. 1, No. 3. https://doi.org/10.7763/LNSE.2013.V1.61

Yang X.-S. 2010: Nature - Inspired Metaheuristic Algorithms. Second Edition, Universtiy of Cambridge, Luniver Press, UK

Yang X.-S., Deb S. 2009, Cuckoo search via levy flights. *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, IEEE Publications, USA. https://doi.org/10.1109/NABIC.2009.5393690

Yang X.S. 2012, Flower pollination algorithm for global optimization. *In: Unconventional Computation and Natural Computation 2012, Lecture Notes in Computer Science*, Vol. 7445, 240-249. https://link.springer.com/chapter/10.1007/978-3-642-32894-7_27

Yu F., Chen Z.F., Diao Y., Lakshman T.V. and Katz R.H. 2006.: Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection. University of California at Berkeley, Technical Report No.UCB/EECS-2006-76. http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/ EECS-2006-76.html (2006)

Biographical notes

S. Prithi is of the Department of Computer Science and Engineering, Sri Rajalakshmi Engineering College, Chennai. Tamil Nadu, India and **S. Sumath**i is in the Department of Electrical and Electronics Engineering, PSG College of Technology, Coimbatore , India