Int. J. Nonlinear Anal. Appl. 13 (2022) 2, 2517–2540
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.27338.3563

Application of meta-heuristic algorithms in intrusion detection system

Amirreza Saba[a], Xiang Michelle Liu[a,*], Marjan Mansourvar[b]

[a]School of Technology and Innovation, Marymount University, Virginia, USA
[b]DTU Computer Bioscience Group, Technical University of Denmark (DTU), Lyngby, Denmark

(Communicated by Majid Eshaghi Gordji)

Abstract

With the Internet being the dominant tool for global communication in today's world, the issue of Internet information security has become quite a significant challenge. Wireless sensor networks, like other systems, can be penetrated and it appears that natural communities, due to their adequate capabilities in information processing, can be utilized as a model in these networks. Accordingly, in this study, we shall analyze a model that uses the energy and operational power of the three meta-heuristic algorithms GA, PSO & GSO that occur in natural communities. Furthermore, we introduced four Dos, D-dos, Wormhole, and Sinkhole attacks to these algorithms, and thereafter examined their latency, throughput, and energy. The findings revealed that among these algorithms, the GA algorithm has the highest energy and the PSO algorithm has the highest throughput.

Keywords: intrusion detection system, mobile ad-hoc network, glow worm optimization algorithms, genetic algorithms, particle swarm optimization algorithms
2020 MSC: 49N30, 65K10

1 Introduction

Over the last few decades, the utilization of Internet networks has skyrocketed and at present, almost all organizations use computer networks to store and transmit their data. The use of Internet applications and services on various platforms such as e-learning and e-commerce is increasing every day, which raises security and privacy concerns [1]. Evidence suggests that along with these uses, we have seen an increase in cyber security breaches with the use of phishing tools and additional and newer hacks [2], [3]. Unfortunately, the advent of new cyber-attacks has made defending against them an important security problem in wide area networks [4]. With the spread of new threats and attacks against computer networks, in case of non-compliance with the security infrastructure by an organization, all data/information of that organization are at risk and therefore it is necessary to detect and neutralize these attacks in a timely manner.
One way to protect a network from attacks is through intrusion detection. Building an efficient intrusion detection model is a challenging task because an intrusion detection system must simultaneously have a high attack detection

*Corresponding author
Email addresses: amirsaba1980@gmail.com (Amirreza Saba), xliu@marymount.edu (Xiang Michelle Liu), marjma@dtu.dk (Marjan Mansourvar)

Received: November 2021 *Accepted:* February 2022

rate as well as a low error warning rate. In an ideal intrusion detection system, the TP value is 100 and the FP and FN values are zero. But in practice, in a penetration detection system the TP value is less than 100 and the FP and FN values are more than zero. In an intrusion detection system, a balance must be struck between FP and FN, and in order for it to be effective, the FN and FP rates must be kept to a minimum. Consequently, the intrusion detection system requires a high detection rate and a low false alarm rate. The significant factor about evaluating the various proposed algorithms for reducing false positives is that just reducing the false positive warning rate is not sufficient (since some false positive reduction techniques cause a decrease in accuracy). As a result, the technique used should reduce false positive warnings while not limiting accuracy.

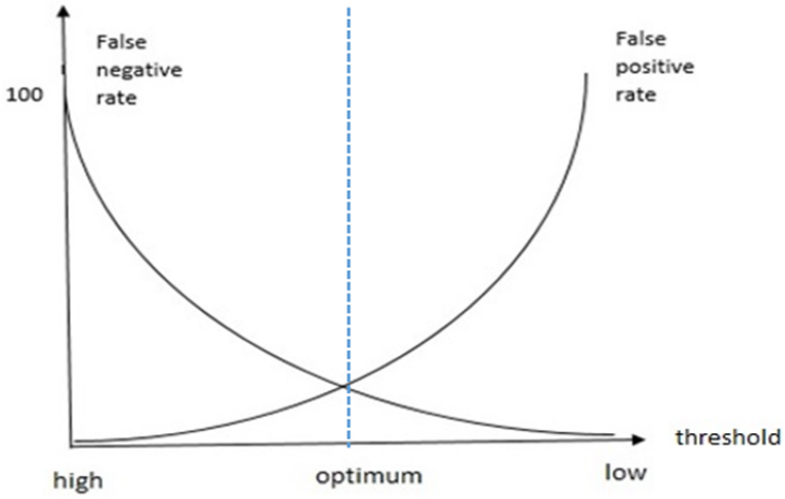

Figure 1: Balance between FP and FN

Intrusion detection is the process of detecting malicious activity. An intrusion detection system is a defense system to help identify, evaluate, and report unauthorized network activity. The intrusion detection system does not actually detect intrusion but detects traffic activities that may create an intrusion with constant network monitoring for unusual activities [5].

Over the past few years, wireless and mobile communication networks have become much more developed, and due to the lack of infrastructure and node allocation, the management of this network, like traditional routers in the fixed network, has encountered many new challenges. The first challenge is how to maintain a network connection. The routing problem in networks without wireless mobile infrastructure has been resolved via some protocols such as AODV, DSR and DSDV. The second challenge is security, due to features such as changing dynamic topology, heterogeneous and decentralized control, limited resources and unfriendly environment, mobile networks with various attacks such as fraud, package modification and distributed denial of service, etc. Therefore, to maintain the security of networks, a prevention-based approach is not enough, and hence, intrusion detection mechanisms need to developed [6]. Consequently, to detect attacks and prevent them, a technique with four characteristics should be utilized. These four characteristics are: accurate attack detection, efficiency, intelligence and high security.

In mobile networks, malicious attacks are quite dangerous because they not only block the victim node but also disrupt the entire network. Up to now, various algorithms and protocols have been proposed in the wireless sensor network, the most important of which are error routing and tolerance, security issues such as authentication and intrusion detection, etc. Algorithms and protocols provided for these issues should include and take into consideration items such as consumption, low power and low latency, bandwidth limitations, long service life, and service quality issues. A wireless sensor network without infrastructure also has certain traffic. Such networks are usually subject to attacks that reduce the efficiency of the network.

With the comparison help of several important meta-heuristic algorithms, it is possible to check the accuracy of intrusion detection in these methods. In fact, meta-heuristic algorithms such as: ant colony optimization (ACO), genetic algorithm (GA), firefly algorithm (GSO), particle mass optimization algorithm (PSO), etc., are among the types of approximate optimization algorithms that have local optimal ext strategies, and are applicable vis a vis a wide range of issues (9,8,7,10).

Therefore, in this research, to enhance the performance of intrusion detection system in networks without wireless mobile infrastructure, several meta-innovative algorithms were analyzed and examined to identify general network attacks and improve detection rates for various types of network attacks. Thereafter, their latency, throughput and

energy were compared and the most efficient and functional was introduced.

2 Literature review

Several intrusion detection systems have been introduced both in research as well as commercially. Even though these systems differ in terms of data collection and processing technology, most of them are grounded on a specific architecture. Their sensors collect data from the system and sends them to the attack detection engine with the aim of detecting attacks. If the attack is detected, the alert is sent by the detection engine to the response component to react appropriately to the attack. Multiple studies have been conducted in this regard and the following are the findings of some of them.

In a 1995 study, Sepford et al. implemented a genetic algorithm-based approach, wherein intrusion detection is performed utilizing software and a genetic algorithm. In this approach, utilizing a trust framework, a set of classification rules is proposed to judge the proportionality function [10]. In 2006, Weise developed a genetic programming framework that automatically detects algorithms for distributed data problems in sensor networks. [11].

In 2009, Ghose and Krishnanand, optimized a nature-inspired multi model function called the Firefly Algorithm with mass robotics. This study's findings points out that GSO is similar to ACO and PSO, but there are important differences. The simulation utilized in this approach reveals that the GSO algorithm is effective in several optimal modes of the multi-model function [10]. In 2008, Owais et al. with the assistance of genetic algorithms discovered malicious intrusion in the network. The results of this study showed that genetic algorithms are used in different ways in the intrusion detection system and numerous researchers employ evolutionary algorithms, especially genetic algorithms [12]. In a 2008 study, Kulkarni et al, localized a central network in a wireless sensor network and utilized a particle swarm algorithm for localization. In this approach, they utilized a scenario focusing on rapid target localization and the convergence of wireless sensor network nodes around the intended target. This study's findings demonstrated that the proposed algorithm is quite effective in the navigation-oriented network [13].

In 2008, Ghose et al. developed a firewall congestion optimization algorithm that simultaneously analyzed multiple states in several function models. In this approach, a comparison of GSO and PSO was presented and with the experimental Niche-PSO comparison, another type of PSO was presented which simultaneously designed several optimal modes [14]. Chang and ping, in 2009 introduced a new particle swarm algorithm that detected an attack by reducing the false alarm rate. False alarms are a serious problem that occur in intrusion detection systems. The experiment was performed via the 1999 KDD cup database. The study's findings indicated that by utilizing this efficient algorithm, the error warning rate can be curtailed [15].

In 2013, Mostaque and Morshedur introduced an intrusion detection system using genetic algorithms and fuzzy logic to effectively detect different types of intrusion activity in a network. This system, based on fuzzy logic, was able to detect an attacker in computer network activities and establish a superior collection of rules. The proposed intrusion detection systems was tested and evaluated with the KDD Cup 99 database. The experimental findings clearly proved that the proposed system has a higher accuracy rate in identifying normal and abnormal nodes and that its detection rate is acceptable [12]. In 2014, Feng and Pan, enhanced the performance of the particle swarm algorithm with the aid of hybrid techniques. This approach focused on solving the problem of slow convergence and optimizing the performance of the particle swarm algorithm. In this approach, alternative search memory was integrated with particle swarm algorithm, introduced as CRS-PSO [16].

Haghighat et al., in their 2007 study assessed the combined genetic and fuzzy algorithms (using three algorithms: combined genetic, PSO & SFL). They introduced an efficient solution to complex optimization problems [17]. In 2008, Sabaahi & Mowghar reviewed the intrusion detection system. This approach provides a new classification of intrusion detection systems able to be utilized for surveys and classifications [18]. Einy et al. [16] introducing an IDS (NIDS) network called MOPSO-FLN and using a combination of multi-objective particle swarm optimization (MOPSO-) algorithm based on feature subset selection (FSS) and learning network Acted quickly (FLN).In this study, KDD Cup99 and data set were used for feature selection, network training and model testing. The results of the simulation showed that this method can improve the IDS performance in terms of evaluation criteria compared to other previous methods by balancing the goals of the number of identifying features and educational errors based on evolutionary power [19].

Alzaqebah et al [3] in a study using the Gray Wolf Optimization (GWO) method and the UNSWNB-15 data set tried to detect normal and abnormal network traffic. The results showed that the proposed model has performed better than other methods in minimizing the cross-error rate and false positive rate to less than 30% [20]. Rethinavalli and Copinta (2020) proposed three intrusion prevention selection strategies and an artificial neural network (ANN) classification model to improve the classification of Sybil nodes in MANET [21]. Chahal and Harit [9] used a multi-valued discrete particle swarm optimization (DPSO) study, including encryption and decoding phases, to optimize the process of identifying an optimal path for efficient data dissemination in VANET. The obtained results show that the

proposed algorithm is better than other related designs in the existing literature [22]. Acharya and Singh [1] used the Smart Water Drop Algorithm (IWD) to improve IDS performance. The authors evaluated the IWD model using the KDDCUP99 dataset in terms of false alarms, detection rate, and accuracy, and provided recommendations [23].

3 Research methodology

Metaheuristic algorithms for pptimization, siimulated annealing

Simulated annealing is based on the metal annealing processing [33]. Unlike the gradient-based methods and other deterministic search methods, the main advantage of simulated annealing is its ability to avoid being trapped in local optima. Metaphorically speaking, this is equivalent to dropping some bouncing balls over a landscape, and as the balls bounce and lose energy, they settle down in some local minima. If the balls are allowed to bounce long enough and to lose energy slowly enough, some of the balls will eventually fall into the globally lowest locations, hence the global minimum will be reached. Essentially, simulated annealing is a search along a Markov chain, which converges under appropriate conditions.

In simulated annealing, the actual search moves trace a piecewise path. With each move, an acceptance probability is evaluated, which not only accepts changes that improve the objective function (for a minimization problem, a lower objective value), but also keeps some changes that do not improve the objective (a larger objective value). The acceptance probability p is given by

$$p = \exp[-\Delta E/kBT]$$

where kB is the Boltzmann's constant, T is the temperature for controlling the annealing process and ΔE is the change in energy. This transition probability is based on the Boltzmann distribution in statistical mechanics. The change in the objective function, Δf, can be related to ΔE in the following way

$$\Delta E = \gamma \Delta f,$$

where γ is a real constant (typically, $\gamma = 1$ for simplicity). It is clear that $p \to 0$ as $T \to 0$. That is, the system virtually becomes a hill-climbing method at very low temperature. The way to control the temperature variations essentially controls how the algorithm behaves and its efficiency.

There are many ways to control the cooling rate or the temperature decrease. Two commonly used annealing schedules (or cooling schedules) are linear and geometric. For a linear cooling schedule, we have

$$T = T_0 - \beta_t,$$

where T_0 is the initial temperature and t is a pseudo time that replaces the iterations. β is the cooling rate and should be chosen in such a way that $T \to T_f$ when $t \to T_f$ (or the maximum number N of iterations). This usually gives

$$\beta = (T_0 - T_f)/t_f.$$

On the other hand, a geometric cooling schedule essentially decreases the temperature by a cooling factor $0 < \alpha < 1$ so that T is replaced by α_T or

$$T(t) = T_0 \alpha_t, \qquad t = 1, 2, \cdots, t_f.$$

The advantage of the second method is that $T \to 0$ when $t \to \infty$, and thus there is no need to specify the maximum number of iterations. For this reason, the geometric cooling schedule is more widely used. The cooling process should be slow enough to allow the system to stabilize. In practice, $\alpha \in [0.7, 0.99]$ is commonly used.

In addition, for a given temperature, multiple evaluations of the objective function are needed. If there are too few evaluations, there is a danger that the system will not stabilize and subsequently will not converge to its global optimality. Conversely, too many evaluations are time-consuming and the system will usually converge too slowly, as the number of iterations to achieve stability might be exponential with the problem size. Therefore, there is a fine balance between the number of evaluations and solution quality. We can either perform many evaluations at a few temperature levels or do only a few evaluations at many temperature levels. There are two major ways to set the number of iterations: fixed or variable. The first uses a fixed number of iterations at each temperature, while the second intends to increase the number of iterations at lower temperatures to fully explore local minima.

Genetic algorithms

Genetic algorithms (GAs) are probably the most popular evolutionary algorithms with a diverse range of applications. A vast majority of well-known optimization problems have been solved by genetic algorithms. In addition, genetic algorithms are population-based and many modern evolutionary algorithms are directly based on, or have strong similarities to, genetic algorithms.

Genetic algorithms, developed by John Holland and his collaborators in the 1960s and 1970s, are a model or abstraction of biological evolution based on Charles Darwin's theory of natural selection. Holland was the first to use crossover, recombination, mutation and selection in the study of adaptive and artificial systems. These genetic operators are the essential components of genetic algorithms as a problem-solving strategy. Since then, many variants of genetic algorithms have been developed and applied to a wide range of optimization problems, from graph colouring to pattern recognition, from discrete systems (such as the travelling salesman problem) to continuous systems (e.g., the efficient design of airfoil in aerospace engineering), and from financial markets to multi objective engineering optimization.

The essence of genetic algorithms involves the encoding of solutions as arrays of bits or character strings (chromosomes), the manipulation of these strings by genetic operators and a selection based on their fitness to find a solution to a given problem. This is often done through the following procedure:

1) definition of an encoding scheme;
2) definition of a fitness function or selection criterion;
3) creation of a population of chromosomes;
4) evaluation of the fitness of every chromosome in the population;
5) creation of a new population by performing fitness-proportionate selection, crossover and mutation;
6) replacement of the old population by the new one.

Steps 4), 5) and 6) are then repeated for a number of generations. At the end, the best chromosome is decoded to obtain a solution to the problem.

Each iteration, which leads to a new population, is called a generation. The fixed-length character strings are used in most genetic algorithms at each generation although there is substantial research on variable-length strings and coding structures. The coding of the objective function is usually in the form of binary arrays or real-valued arrays in adaptive genetic algorithms. An important issue is the formulation or choice of an appropriate fitness function that determines the selection criterion in a particular problem. For the minimization of a function using genetic algorithms, one simple way of constructing a fitness function is to use the simplest form $F = A - y$ with A being a large constant (though $A = 0$ will do if the fitness does not need to be non negative) and $y = f(\boldsymbol{x})$. Thus the objective is to maximize the fitness function and subsequently to minimize the objective function $f(\boldsymbol{x})$. However, there are many different ways of defining a fitness function. For example, we can assign an individual fitness relative to the whole population

$$F(x_i) = f(\xi_i)) \sum N_i = 1 f(\xi_i),$$

Where ξ_i is the phenotypic value of individual i, and N is the population size. The form of the fitness function should make sure that chromosomes with higher fitness are selected more often than those with lower fitness. Poor fitness functions may result in incorrect or meaningless solutions.

Another important issue is the choice of various parameter values. The crossover probability pc is usually very high, typically in the interval $[0.7, 1.0]$. On the other hand, the mutation probability pm is usually small (typically, in the interval $[0.001, 0.05]$). If pc is too small, then crossover is applied sparsely, which is not desirable. If the mutation probability is too high, the algorithm could still 'jump around' even if the optimal solution is close.

Selection of the fittest is carried out according to the fitness of the chromosomes. Sometimes, in order to make sure that the best chromosomes remain in the population, they are transferred to the next generation without much change, which is called elitism. A proper criterion for selecting the best chromosomes is also important, because it determines how chromosomes with higher fitness are preserved and transferred to the next generation. This is often carried out in association with a certain form of elitism. The basic form is to select the best chromosome (in each generation) which will be carried over to the new generation without being modified by the genetic operators. This ensures that a good solution is attained more quickly.

Other issues include multiple sites for mutation and the use of various population sizes. The mutation at a single site is not very efficient, so mutation at multiple sites typically increases the evolution of the search. On the other hand, too many mutants will make it difficult for the system to converge or even lead the system toward wrong solutions. In real ecological systems, if the mutation rate is too high under high selection pressure, then the whole population might

become extinct. In addition, the choice of the right population size is also very important. If the population size is too small, there will not be enough evolution, and there is a risk for the whole population to converge prematurely. In the real world, ecological theory suggests that a species with a small population is in real danger of extinction. In a small population, if a chromosome with a fitness substantially larger than the fitness of the other chromosomes in the population appears too early, it may produce enough offspring to overwhelm the whole (small) population. This will eventually drive the system to a local optimum (not the global optimum). On the other hand, if the population is too large, more evaluations of the objective function are needed, which will require extensive computing time.

As a simple example, an initial population is generated (Fig. 2) and its final locations aggregate towards optimal solutions (Fig. 3).

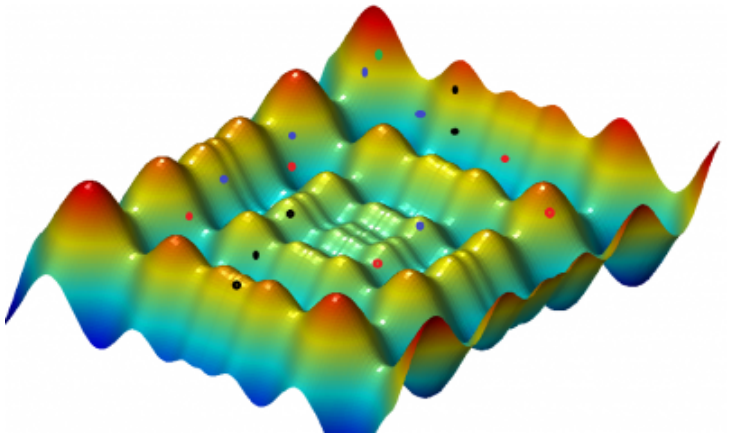

Figure 2: Genetic algorithm (initial population and locations)

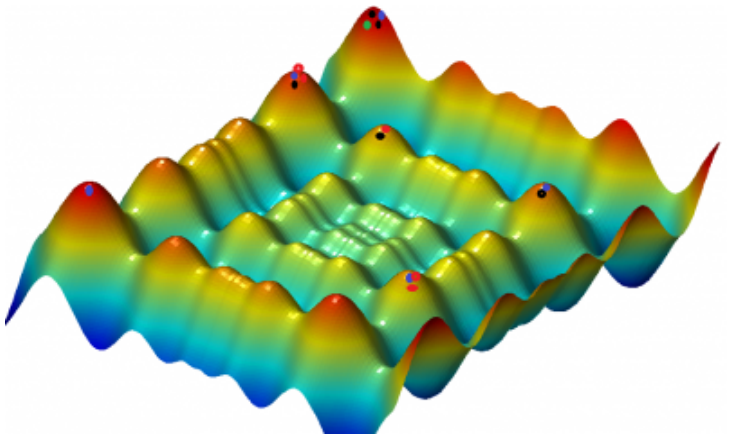

Figure 3: Genetic algorithm (final locations)

Differential evolution

Differential evolution (DE) was developed by R. Storn and K. Price in their nominal papers in 1996 and 1997. It is a vector-based evolutionary algorithm, and can be considered as a further development of genetic algorithms. Unlike genetic algorithms, differential evolution carries out operations over each component (or each dimension of the solution). Almost everything is done in terms of vectors, and DE can be viewed as a self-organizing search, directed towards the optimum.

For a d-dimensional optimization problem with d parameters, a population of n solution vectors $\boldsymbol{x}_i, (i = 1, 2, \cdots, n)$. is initially generated. For each solution $\boldsymbol{x}_i$ at any generation t , we use the conventional notation

$$\boldsymbol{x}_{ti} = (x_{t1}, i, x_{t2}, i, \cdots, x_{td}, i).$$

This vector can be considered as a chromosome. Differential evolution consists of three main steps: mutation, crossover and selection. Mutation is done as follows. For each vector $\boldsymbol{x}_i$ at any time or generation t, we first randomly choose three distinct vectors $\boldsymbol{x}_p, \boldsymbol{x}_q$ and $\boldsymbol{x}_r$, and then generate a so-called donor vector by the following mutation scheme

$$\boldsymbol{v}t + 1i = \boldsymbol{x}_{tp} + F(\boldsymbol{x}_{tq} - \boldsymbol{x}_{tr})$$

where $F \in [0, 2]$ is a parameter, often referred to as the differential weight. The minimum population size is $n \geq 4$. In principle, $F \in [0, 2]$, but in practice, a scheme with $F \in [0, 1]$ is more efficient and stable. The perturbation $\mathbf{w} = F(\mathbf{x}_q - \mathbf{x}_r)$ to the vector $\mathbf{x}_p$ is used to generate a donor vector $\mathbf{v}_i$, and this perturbation is directed and self-organized.

The crossover is controlled by a probability $C_r \in [0, 1]$. The actual crossover can be carried out in two ways: binomial and exponential. The binomial scheme performs crossovers on each of the d components or variables/parameters. By generating a uniformly distributed random number $r_i \in [0, 1]$, the jth component of $\boldsymbol{x}_i$ is set as follows

$$\boldsymbol{x}_t + 1_{j,i} = \boldsymbol{v}_t + 1_{j,i} if r_i \leq C_r, \quad (j = 1, 2, \cdots, d),$$

otherwise, it remains unchanged. This way, it is decided randomly whether to replace a component using the donor vector or not. The basic DE/Rand/1/Bin scheme is carried out in above equation, but at least 10 different schemes have been formulated. The interested reader is referred to Price et al. [46] for details.

Ant colony optimization

Ant colony optimization was pioneered by Marco Dorigo in 1992 and is based on the foraging behaviour of social ants. Many insects such as ants use pheromone as a chemical messenger. Ants are social insects and live together in organized colonies consisting of approximately 2 to 25 million individuals. When foraging, a swarm of ants or mobile agents interact or communicate in their local environment. Each ant lays scent chemicals or pheromone to communicate with others. Each ant is also able to follow the route marked with pheromone laid by other ants. When an ant finds a food source, it will mark it with pheromone and also mark the trail to and from it.
However, the pheromone concentration ϕ decays or evaporates at a constant rate γ. That is,

$$\phi(t) = \phi_0 \exp[-\gamma_t],$$

where ϕ_0 is the initial concentration at $t = 0$. Here the evaporation is important, as it ensures the possibility of convergence and self-organization.
From the initial random foraging route, the pheromone concentration varies and the ants follow the route with higher pheromone concentration. In turn, the pheromone is enhanced by the increasing number of ants. As more and more ants follow the same route, it becomes the favored path. Thus, some favorite routes emerge, often the shortest or more efficient ones. This is actually a positive feedback mechanism. As the system evolves, it converges to a self-organized state, which is the essence of any ant algorithm.

Bee algorithms

Bee algorithms are another class of metaheuristic algorithms, inspired by the foraging behaviour of bees. A few variants exist in the literature, including honeybee algorithm, artificial bee colony, bee algorithm, virtual bee algorithm, and honeybee mating algorithms.
Honey bees live in a colony and they forage and store honey in their constructed colony. Honey bees can communicate by pheromone and 'waggle dance'. For example, an alarming bee may release a chemical message (pheromone) to stimulate attack response in other bees. Furthermore, when bees find a good food source and bring some nectar back to the hive, they will communicate the location of the food source by performing the so-called waggle dance as a signaling system. Such signaling dances vary from species to species, however, they are aimed at recruiting more bees by using directional dancing with varying strength so as to communicate the direction and distance of the food source. For multiple food sources such as flower patches, studies show that a bee colony seems to be able to allocate forager bees among different flower patches so as to maximize their total nectar intake[47].

It seems that the Honey Bee Algorithm (HBA) was first formulated around 2004 by Craig A Tovey at Georgia Tech in collaboration with Sunil Nakrani then at Oxford University as a method to allocate computers among different clients and web-hosting servers. Later in 2004 and in early 2005, Xin-She Yang at Cambridge University developed

a Virtual Bee Algorithm (VBA) to solve continuous optimization problems. At about the same time, Pham et al. [45] developed the bee algorithms. Slightly later in 2005, Haddad and Afshar and their colleagues presented a Honey-bee mating optimization (HBMO) algorithm which was subsequently applied to reservoir modelling and clustering. Around the same time, D Karabogo in Turkey developed an Artificial Bee Colony (ABC) algorithm for numerical function optimization.

Ant and bee algorithms are more suitable for discrete and combinatorial optimization and have been applied in a wide range of applications.

Particle swarm optimization

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995, based on swarm behaviour observed in nature such as fish and bird schooling. Since then, PSO has generated a lot of attention, and now forms an exciting, ever-expanding research subject in the field of swarm intelligence. PSO has been applied to almost every area in optimization, computational intelligence, and design/scheduling applications. There are at least two dozens of PSO variants, as well as hybrid algorithms obtained by combining PSO with other existing algorithms, which are also increasingly popular.

PSO searches the space of an objective function by adjusting the trajectories of individual agents, called particles. Each particle traces a piecewise path which can be modelled as a time-dependent positional vector. The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted toward the position of the current global best $\boldsymbol{g}^*$ and its own best known location $\boldsymbol{x}^*i$, while exhibiting at the same time a tendency to move randomly.
When a particle finds a location that is better than any previously found locations, then it updates this location as the new current best for particle i. There is a current best for all particles at any time t at each iteration. The aim is to find the global best among all the current best solutions until the objective no longer improves or after a certain number of iterations.

Let $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ be the position vector and velocity, respectively, of particle i. The new velocity vector is determined by the following formula

$$\boldsymbol{v}_t + 1_i = \boldsymbol{v}_{ti} + \alpha_\epsilon 1[\boldsymbol{g}^* - \boldsymbol{x}_{ti}] + \beta_\epsilon 2[\boldsymbol{x}^* i - \boldsymbol{x}_{ti}]$$

where ϵ_1 and ϵ_2 are two random vectors, and each entry takes a value between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, which are typically equal to, say, $\alpha \approx \beta \approx 2$.

The initial locations of all particles should be distributed relatively uniformly so that they can sample over most regions, which is especially important for multimodal problems. The initial velocity of a particle can be set to zero, that is, $\boldsymbol{v}_t = 0_i = 0$. The new position can then be updated by the formula

$$\boldsymbol{x}_t + 1_i = \boldsymbol{x}_{ti} + \boldsymbol{v}_t + 1_i.$$

Although $\boldsymbol{v}_i$ can take any value, it is usually bounded in some range $[0, \boldsymbol{v}\max]$. There are many variants which extend the standard PSO algorithm, and the most noticeable improvement is probably to use an inertia function $\theta(t)$ so that $\boldsymbol{v}_{ti}$ is replaced by $\theta(t)\boldsymbol{v}_{ti}$ where θ takes a value between 0 and 1. In the simplest case, the inertia function can be taken as a constant, typically $\theta \in [0.5, 0.9]$. This is equivalent to introducing a virtual mass to stabilize the motion of the particles, and thus the algorithm is expected to converge more quickly.

As the iterations proceed, the particle system swarms and may converge towards a global optimum. For details, please refer to the Scholarpedia article on particle swarm optimization. As a simple example, an animation of particles is shown in Fig. 4 where all particles move towards the global optimum.

Tabu search

Tabu search was developed by Fred Glover in the 1970s, bur his seminal book was published much later in 1997. Tabu search explicitly uses memory and the search history is a major component of the method. As most algorithms are memoryless or only use results of the last or two last steps, it is initially difficult to see the advantage of using the search history. The subtleties of memory and history could introduce too many degrees of freedom, and a mathematical analysis of the algorithm behaviour becomes intractable. However, Tabu search remains one of the most successful and widely used metaheuristics in optimization.

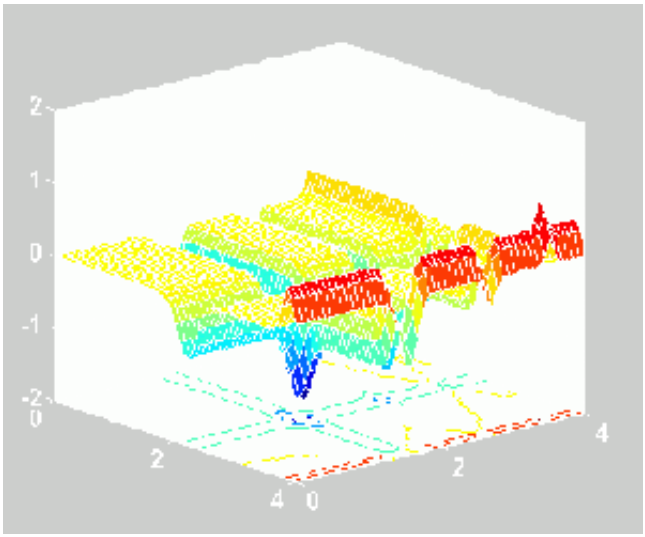

Figure 4: Particle swarm optimization (demo)

In essence, Tabu search can be considered as an intensive local search, and the appropriate use of search history avoids revisiting local solutions by recording recently tried solutions in tabu lists. Over a large number of iterations, these tabu lists could save a significant amount of computing time, leading to improvements in search efficiency. For example, studies show that the use of tabu lists with integer programming can save computing effort by at least two orders of magnitude for a given problem, as compared with standard integer programming [22, 23]. Many hybrid algorithms have been developed by combining Tabu search with other metaheuristics.

Harmony search

Harmony Search (HS) is a relatively new heuristic optimization algorithm first developed by Z. W. Geem et al. in 2001. Harmony search is related to the improvisation process of a musician. When a musician is improvising, he or she has three possible choices:

(1) play any famous piece of music (a series of pitches in harmony) exactly from his or her memory;
(2) play something similar to a known piece (thus adjusting the pitch slightly); or
(3) compose new or random notes. If we formalize these three options for optimization, we have three corresponding components: usage of harmony memory, pitch adjustment, and randomization.

From a Markov chain point of view, pitch adjustment is a random walk which generates a new solution from the current solution xold by

$$\boldsymbol{x}_t + 1_{\text{new}} = \boldsymbol{x}_t\text{old} + bp\boldsymbol{e}_{ti}$$

where $\boldsymbol{e}_{ti}$ is a random number drawn from a uniform distribution $[-1, 1]$ and bp is the bandwidth, which controls the local range of pitch adjustments.

Firefly algorithm

The Firefly Algorithm (FA) was developed by Xin-She Yang [61] and is based on the flashing patterns and behaviour of fireflies. In essence, FA uses the following three idealized rules:

1. Fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex;
2. The attractiveness is proportional to the brightness and both decrease as the distance between two fireflies increases. Thus for any two flashing fireflies, the brighter firefly will attract the other one. If neither one is brighter, then a random move is performed;
3. The brightness of a firefly is determined by the landscape of the objective function.

As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, we can now define the variation of attractiveness β with the distance r by

$$\beta = \beta_{0e} - \gamma_r 2$$

where β_0 is the attractiveness at $r = 0$. The movement of a firefly i, attracted to another more attractive (brighter) firefly j, is determined by

$$\boldsymbol{x}_t + 1_i = \boldsymbol{x}_{ti} + \beta_{0e} - \gamma_r 2_{ij}(\boldsymbol{x}_{tj} - \boldsymbol{x}_{ti}) + \alpha \boldsymbol{e}_{ti}$$

where the second term is due to the attraction. The third term is a randomization with α being the randomization parameter, and $\boldsymbol{e}_{ti}$ is a vector of random numbers drawn from a Gaussian distribution or uniform distribution at time t. If $\beta_0 = 0$, it becomes a simple random walk. Furthermore, the randomization $\boldsymbol{e}_{ti}$ can easily be extended to other distributions such as Lévy flights. A demonstrative example is shown in Fig. 5

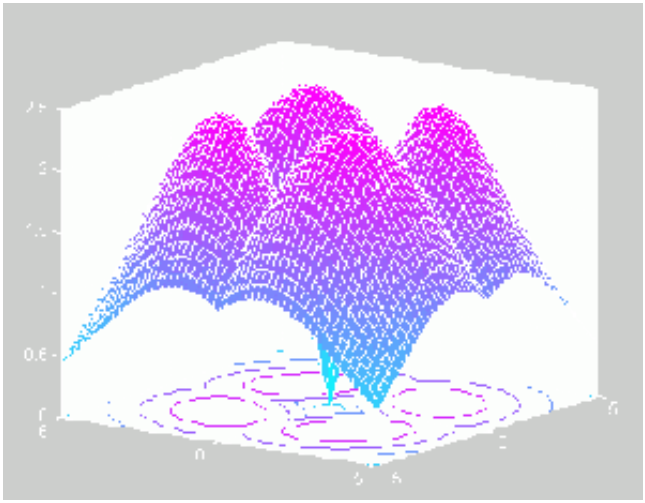

Figure 5: Firefly algorithm and fireflies move towards 4 global optima

Cuckoo search

Cuckoo search (CS) is one of the latest nature-inspired metaheuristic algorithms, developed by Xin-She Yang and Suash Deb in 2009. CS is based on the brood parasitism of some cuckoo species [64]. In addition, this algorithm is enhanced by the so-called Lévy flights, rather than by simple isotropic random walks [44]. Recent studies show that CS is potentially far more efficient than PSO and genetic algorithms [62].
Cuckoos are fascinating birds, not only because of the beautiful sounds they make, but also because of their aggressive reproduction strategy. Some species named *ani* and *Guira* lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage in the mandatory brood parasitism by laying their eggs in the nests of other host birds (often other species).

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
- The best nests with high-quality eggs are carried over to the next generations;
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $pa \in [0, 1]$, depending on the similarity of a cuckoo egg to its host eggs. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

When generating new solutions $\boldsymbol{x}(t + 1)$ for, say, a cuckoo i, a Lévy flight is performed

$$\boldsymbol{x}(t + 1)i = \boldsymbol{x}(t)_i + \alpha L(s, \lambda),$$

where $\alpha > 0$ is the step size which should be scaled to the problem of interest. The above equation is essentially the stochastic equation for a random walk. In general, a random walk is a Markov chain whose next status/location

only depends on the current location (the first term in the above equation) and the transition probability (the second term). Here the random walk via Lévy flight is more efficient in exploring the search space, as its step length is much longer in the long run.
The Lévy flight essentially provides a random walk whose random step length (s) is drawn from a Lévy distribution

$$L(s, \lambda) \sim s - \lambda, (1 < \lambda \leq 3),$$

which has an infinite variance and infinite mean. Here the steps essentially form a random walk process with a power-law step-length distribution with a heavy tail. Some of the new solutions should be generated by the Lévy walk around the best solution obtained so far to speed up the local search. However, a substantial fraction of the new solutions should be generated by far field randomization, that is, the locations should be far enough from the current best solution to make sure that the system will not get trapped in a local optimum.
For simplicity, a demonstrative animation is shown here.

3.1 Routing protocol in manet

The Manet network is an effective way to find attackers in intrusion detection systems. Intrusion detection systems are the most crucial part of the security infrastructure. This network is a set of mobile nodes equipped with a receiver and a transmitter for wireless communication.
A network intrusion detection system monitors suspicious activity and alerts the system (including blocking the user or the source IP address to access the network). To improve network security, the routing protocol should be used. Protocols that can be utilized for routing in the Manet network include DSDV (real time), AODV (Reactive), PDR (packet delivery rate) and PDRR (packet removal rate). In this research, DSDV routing protocol was utilized [24]. AODV routing protocol is based on DSDV algorithm, the difference being that due to routing, only in the required time the amount of data distribution is reduced. The path detection algorithm only starts when there is no path between two nodes. The DSDV protocol is a table-based routing protocol, wherin each node of the network has a routing table.
In this study, incremental packages are utilized to reduce the volume of traffic caused by updating the routes in the network. The only advantage of this method is to avoid routing loops in networks including mobile routers. Nodes are now available to use the route or not.

3.2 Attacks within wireless sensor network

Attacks can be divided into active and non-active depending on the type of activity. In active attacks, which is a way to exploit information on the network, the hacker modifies the data in the target node as well as the data routed to the target. Passive/non-active attacks enter the network and monitor and scan some open and vulnerable ports. Dos, D-dos, Wormhole, and Sinkhole are the most common non-active attacks.
In a DOS attack, the infiltrator changes how a communication and information system operates or is managed, preventing users from having access to network resources. This attack can affect all layers of the OSI connection model. After turning on the system, the attacker steals, alters, or infiltrates an information source. All risks in local area networks, especially wireless types, are due to one of the above risks [25].
The DDoS attack is one of the powerful attacks and the biggest threat to the Internet infrastructure and the IT environment. In this attack, the attacker program starts from several systems that are in the same area of the network. In this attack, all systems send data packets to the victim simultaneously and at maximum power [26]. Even if a DDoS attack is detected, there is no way to correctly identify the attack packets, because the attacker deletes all signs that indicate the intended operation, as well as any evidence of their culpability. They hide the attack script by giving them vague and suspicious names. As a result, users are not suspicious [27].
In a Sinkhole attack, an attempt is made to collect network traffic. Network traffic is then slowly removed or changed, which in turn disrupts the network. A Sinkhole attack increases the network overhead, plus reduces the life of the network (through high energy consumption), where eventually the network is destroyed [28].
The wormhole is another of the most dangerous attacks; Because this type of attack does not require compromise with the network sensor. Moreover, wormhole attacks cannot be forestalled via cryptographic methods [29]. Wormhole attack occur by two malicious nodes through connection outside the band. The adversary uses this attack to convince the nodes in the hop that their distance from the base station is only one or two hop. But in fact, the attacker is located near the base station, By creating a wormhole, it creates a pause in routing [30].

3.3 False alarms in intrusion detection system

The main goal of intrusion detection systems is to detect all intrusions. The primary issue with these systems is the high number of alerts/alarms, which sometimes include false/inaccurate or unimportant alerts. It is impossible to tackle and handle thousands of alerts a day, especially if 99% of them are false alarms.
In false positive alerts or FPs, no attack occurs on the system but it is classified as an attack. Multiple methods have been proposed to reduce false positives. All of these methods can be divided into two general ones, specifically: The Configuration Method & Detection Technique. The first method includes steps that operate in the detection phase, and the second method is the alarm generated after detection [31].

$$\text{False Alarm Rate} = \frac{FP}{\text{Number of Attacks}} \tag{3.1}$$

$$\text{False positive rate} = \frac{FP}{FP + TN} \tag{3.2}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \times 10 \tag{3.3}$$

Equation (3.1) calculates the false alarm rate. Equation (3.2) shall be utilized to calculate the false positive warning rate. Equation (3.3) is used to obtain accuracy in classifying events. False Negative (FN) occurs when no warning is provided when an attack or intrusion takes place (when it actually has). Equation (3.4) is for obtaining the false negative rate [32].

$$\text{Falsenegativerate} = \frac{FN}{FN + TP} \tag{3.4}$$

3.4 Research algorithms

In this study, genetic metaheuristic, particle swarm and firefly algorithms were assessed, and compliant to a series of specific and standard criteria, the algorithms were compared and the efficiency of each of them was analyzed and examined.
Genetic algorithm is among the first meta-heuristic methods mimicking the processes observed in natural evolution and utilizing information in a population to create new and enhanced solutions. The genetic algorithm applies optimal survival rules to a series of problem answers in the hope of obtaining better ones. The process commences with the initial population being randomly constructed. The population then evolves for generations. The quality of the members gradually improves to increase the appropriateness value as a measure of quality value. During each generation, three general genetic operators, including selection, integration, and mutation, are used for each individual with a specific probability. They are usually the most widely used in genetic algorithms. After the first generation of the population, if the number of individuals is in line with the required number, then the genetic operations are performed on each individual, otherwise, more individuals are created. Each time a similar operation is repeated and then stopped [33].
The PSO particle swarm algorithm utilizes a phenomenon called swarm intelligence. In this algorithm, each solution is called a particle. Each particle is calculated by the merit function, indicating the merit of that particle. The more suitable a particle is, the closer it is to its target. In fact, each particle is drawn to a particle that has a higher suitability function. Each particle also has a velocity that indicates the motion of the particle. Each particle continues to move in the problem space by following the optimal particles. Henceforth, a group of PSO particles are randomly formed at the initiation of the work and find their optimal solution by updating the generations [15].
The GSO firefly algorithm is also one of the newest modeling algorithms that has proven its ability to solve optimization problems. In this algorithm, fireflies are randomly placed in the target function space and move to a better location in order to target function. In light of the probability function, Firefly is more likely to move to more luciferin neighbors. The size of the local decision domain is variable, affected by the number of neighbors. When the number of neighbors is small, the local decision domain to find more neighbors increases. If the number of neighbors is large, then the local decision domain decreases. The movement of fireflies creates the desired state [34]. Firefly algorithm consists of 4 stages, including the placement phase of fireflies, the luciferin updating phase, the motion/movement phase and finally the local decision domain update phase.

3.5 Simulation parameters

Simulation models are generated from a set of data taken from a random system that requires the validity of statistical data to be checked and verified with statistical distributions and thereafter tested/analyzed together upon

simulation completion. The simulated output data produces only one possible estimate of real-world events utilizing appropriate methods to increase the accuracy of the output data. Pursuant to final controls, the primary parameters of this study are stipulated in Table 1

Table 1: Simulation parameters

Row	Parameters	QTY
1	Simulation Area	500 x 500 m^2
2	Transfer Domain	100 m^2
3	Initial Energy	0.5J
4	MAC Protocol	802.11
5	Transfer Force	20*0.000000001W
6	Receiving Force	20*0.000000001W
7	Number Of Nodes	100
8	Traffic Demand	1Mb/s – 5Mb/s
9	Gateway Candidate	5

3.6 Simulation performance criteria

The performance metrics utilized in this simulation are as follows:

3.6.1 Throughput

Throughput is the level of data that can be exchanged in a network over a period of time. This information/data may be delivered over a physical or logical link or from a specific network node. Equation (3.5) shall be utilized for this purpose.

$$\text{Throughput}=\frac{\mu}{t} \tag{3.5}$$

μ is the number of bytes received per t unit time. For the entire network, it will be the average throughput of all nodes.

3.6.2 End-to-end latency

The average time it takes the packet to move from one point of the network to another determines the packet delay. End-to-end latency is determined by parameters such as: leading conduction delay, queuing delay, transmission delay, and overclocking delay. Equation (3.6) shall be utilized to calculate the end-to-end latency.

$$D=T_d-T_s \tag{3.6}$$

D: Latency of a package
T_d: Time of package reception at the destination
T_s: Time of sending package at origin

3.6.3 Rate of data delivery

Success rate for receiving messages is more than messages sent on a communication channel. To obtain a successful data delivery rate, Equation (3.7) shall be utilized.

$$PDR=\frac{\text{Receive}_{\text{Pckt}}}{\text{Sent}_{\text{Pckt}}} \tag{3.7}$$

3.6.4 Energy consumption of nodes

The primary task of nodes is to collect information from its surrounding environment. In fact, nodes have the ability to collect data from these environments (not possible via other means to obtain). Due to the advances made in network types, users and applicants of these networks demand an acceptable level of reliability of these systems. Energy consumption in the sensor network is actually the most significant challenge regarding increasing reliability. Often, due to the utilization of this type of network in harsh and inaccessible environments, it is not possible to

recharge or replace the sensor nodes. Therefore, the most important problem in designing wireless sensor networks is to reduce energy consumption toward prolonging the network's lifespan.

$$\text{Node}_{\text{RemEng}}=\text{Node}_{\text{RemEng}}-(E_{tx}\times N_t+E_{rx}\times N_r+E_{ix}\times E_{sx}) \tag{3.8}$$

Equation (3.8) is the energy calculation formula.

$$E_{tx}$$

represents the energy per packet sent,

$$N_t$$

the number of sending nodes,

$$E_{rx}$$

the energy received per packet,

$$N_r$$

the number of receiving nodes &

$$E_{ix}$$

is the initial network energy, which is a constant parameter in the NS2 simulator, and finally the

$$E_{sx}$$

available energy for network density, a constant parameter in the NS2 emulator.

4 Simulation and findings:

The defined scenario for the simulation environment is 100 nodes in an environment with dimensions of 500×500. NS2 emulator is used to define the attack in this report. To accomplish this, the relevant libraries were utilized for the attack. As evidenced in Table 1, five gateways are defined as candidates. DSDV routing protocol, a type of CBRP pattern, is utilized for routing between nodes in the simulation environment.

4.1 Latency comparison of analyzed algorithms

To compare the latency of three GSO-PSO-GA algorithms, simulations were plotted for packet length changes and number of repetitions in the network. Figure 6 indicates the end-to-end latency for the three genetic, particle swarm and firefly algorithms. In this figure, the PSO algorithm has the highest latency by increasing the packet length. The next as far as latency belongs to the GSO algorithm (less latent than the PSO algorithm). Compared to the two algorithms PSO and GSO, the GA algorithm has the lowest latency by increasing the packet transmission length in the network. At first, the distance within the graph is small, but when there is a high number of repetitions, the distance between the three algorithms increases in the graph.

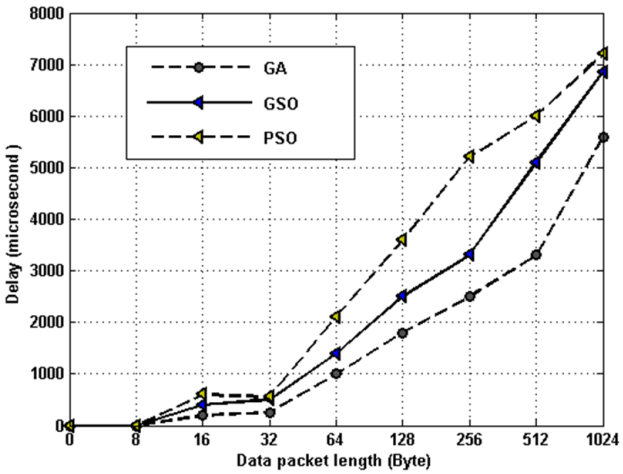

Figure 6: End-to-end delay

4.2 Comparison of algorithm throughputs

To calculate the throughput, in the data entry section of the NS2 parameter related to the node, a counter is fixed that sends 3000 bytes (packets of specified length bytes at different data rates per byte per second). This information moves between the specified node of origin and destination. With each execution of the proposed algorithm in this mode, the sections related to CRC and the confirmation of the submitted frame become inactive. With consideration to the obtained findings, the changes in the data rate set in the transmitter node, induces differing throughput amounts. This scenario is based on the number of repetitions, and Dos, D-dos, Wormhole, Sinkhole attacks are applied on them. In this study, to increase reliability, each step was repeated 7 times to ensure the performance and accuracy of this simulation. Furthermore, we utilized the 802.11 protocol to send packages. These packages are a set of Media Access Controls (MAC) and Physical Layer Properties (PHY) for implementing wireless local area networks (WLANs). Moreover, for data transmission, the ARQ (Automatic Repeat Request) Protocol was utilized to control errors.

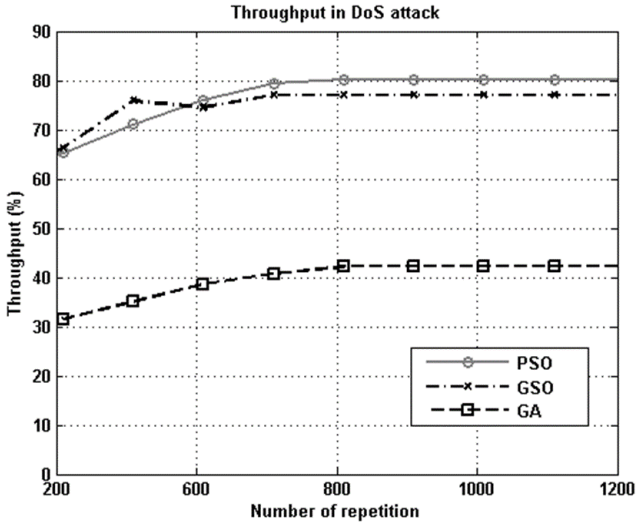

Figure 7: Throughput in dos attack mode

As seen in Figure 7, the PSO algorithm had a higher throughput against the applied Dos attack. The GA algorithm, with increased repetitions compared to the PSO algorithm had less throughput during the Dos attack. In light of the PSO algorithm's structure compared to the GA algorithm plus its upper space complexity, it had a higher throughput. Moreover, the GSO algorithm had the lowest throughput compared to the PSO & GA algorithms during the Dos attack.

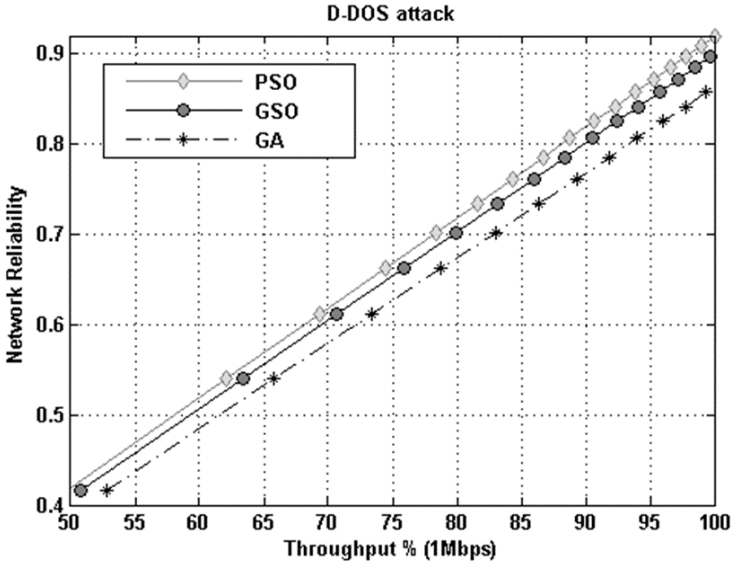

Figure 8: Throughput in *D*-dos attack mode

Figure 8 shows throughput to reliability. In this scenario, the data rate is 1Mbps. As is clear in this figure, the PSO algorithm had a higher reliability in the D-dos attack. As the data rate increases, GSO algorithm's reliability in D-dos attack was less than the PSO algorithm. Therefore, the GA algorithm is less reliable than PSO & GSO algorithms.

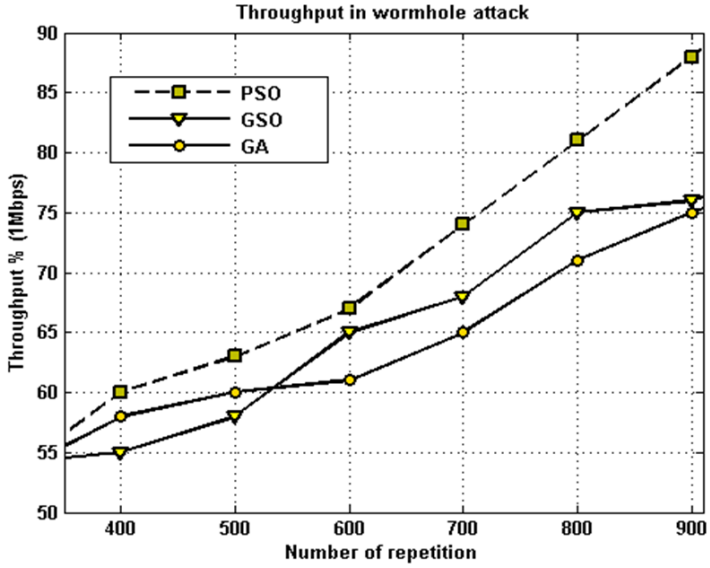

Figure 9: Throughput in wormhole attack mode

Figure 9 shows the throughput or in other words the amount of data that can be exchanged on the network for the three algorithms (GA, PSO & GSO) in a Wormhole attack. The PSO algorithm had the highest throughput compared to the other two algorithms GA & GSO, which is among this algorithm's advantages. As evidenced, the GSO algorithm had a lower throughput in the initial repetitions compared to the GA algorithm (because the simulation complexity was low in initial repetitions). However, as the number of repetitions increased, the complexity increased, and eventually the GSO algorithm registered more throughput than the GA algorithm during the Wormhole attack.

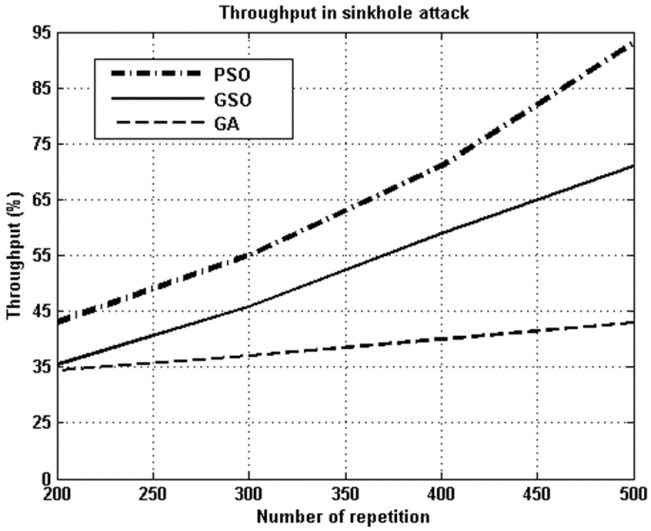

Figure 10: Throughput in sinkhole attack mode

Figure 6 demonstrates the throughput of the three studied algorithms in a Sinkhole attack. The PSO algorithm attacked by Sinkhole had the highest throughput by increasing the number of repetition. The throughput is a useful and effective feature in the network, an added advantage for the PSO algorithm, which this feature. As shown in the figure, the GA algorithm had a lower throughput at higher repetition times than the other two algorithms during the Sinkhole attack.

Figure 11 displays the throughput of the three studied algorithms during a Dos attack. As can be seen in the figure, the PSO algorithm had the highest reliability in the Dos attack (compared to the other two algorithms GA & GSO). The GA algorithm had the lowest reliability and the lowest throughput during a Dos attack (compared to the other two algorithms PSO & GSO).

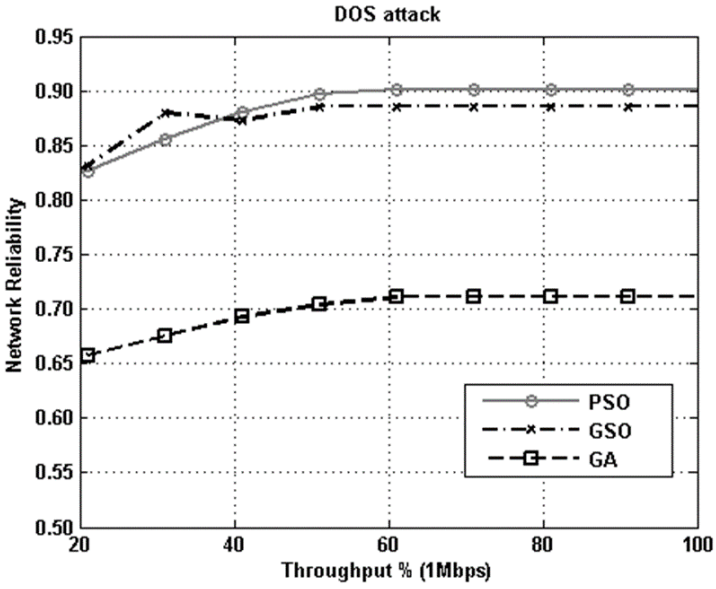

Figure 11: Throughput and reliability in dos attack mode

4.3 Energy consumption comparison of research algorithms

As observable in this section's figures, energy consumption is shown vis a vis time. The findings revealed that over time, due to spatial complexity, the PSO and GSO algorithms consume more energy and the GA model consumes less energy compared to PSO & GSO.

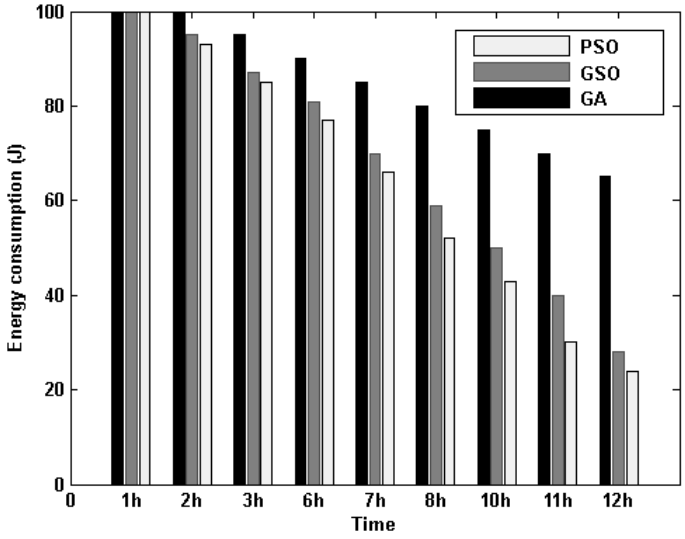

Figure 12: Energy consumption vs time

Figure 12 shows that the PSO algorithm had increased power consumption over time. One of the reasons for the high power consumption in this algorithm is its complexity. The studies revealed that the energy consumption of GA algorithm over time was less than that of PSO & GSO algorithms. The simplicity of the GA algorithm and its less complex nature compared to the others resulted in lower energy consumption and therefore better performance.
Figure 13 indicates the energy consumption of the three algorithms studied vis a vis the number of repetitions (sending & receiving). PSO and GSO algorithms have high energy consumption with increasing number of repetitions, but the GA algorithm displayed less energy consumption and had more energy remaining in it (in comparison with the two other algorithms PSO & GSO and their number of repetitions). Low energy consumption is a great feature for GA algorithm. The oscillations observed in the figure are due to the presence of possible noise in the simulation environment.

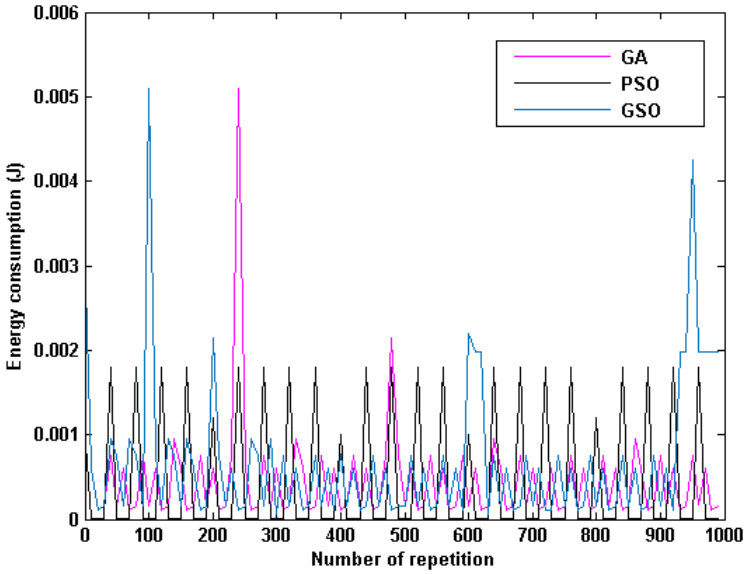

Figure 13: Energy consumption vis a vis the number of repetitions

In Figure 14, the Sinkhole attack is performed to examine the dead nodes of the three PSO, GA & GSO algorithms vis a vis the number of iterations. During the Sinkhole attack against the GA algorithm, more dead nodes were detected

than in the PSO & GSO algorithms. Throughput is among the reasons for high dead nodes. The GA algorithm has more dead nodes with lower throughput. After the Sinkhole attack, PSO algorithm had the fewest dead nodes considering the number of repetitions compared to GA & GSO. The PSO algorithm had a high throughput, hence a lower number of dead nodes in it.

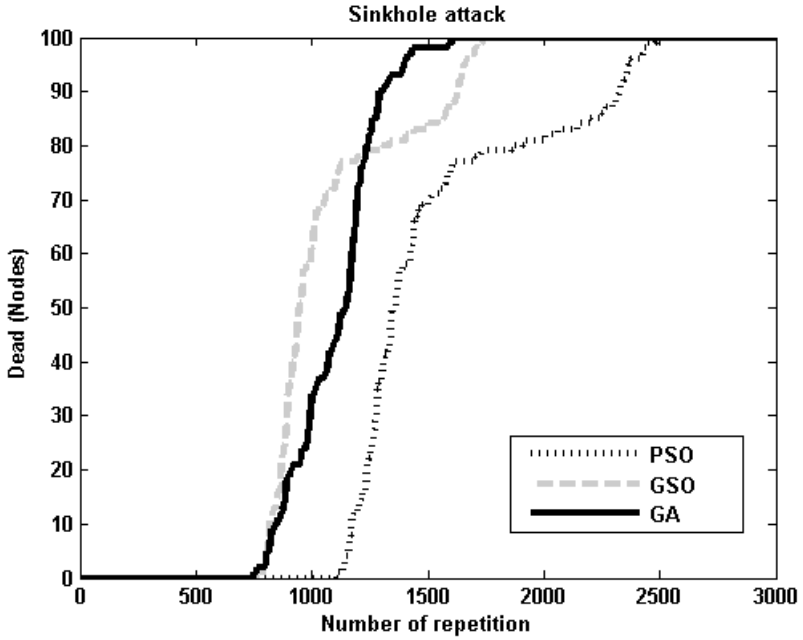

Figure 14: Investigation of dead nodes against sinkhole attack

In Figure 15, the status of dead nodes against the number of repetitions with Wormhole attacks can be assessed. As can be seen in the figure, when the PSO algorithm was attacked by a Wormhole, the number of dead nodes was less than the number of iterations and disappeared later in the nodes. Conversely, when the GA algorithm was attacked by a Wormhole, its nodes were lost in less time than the other two algorithms, and the loss of nodes is among GA algorithm's disadvantages.

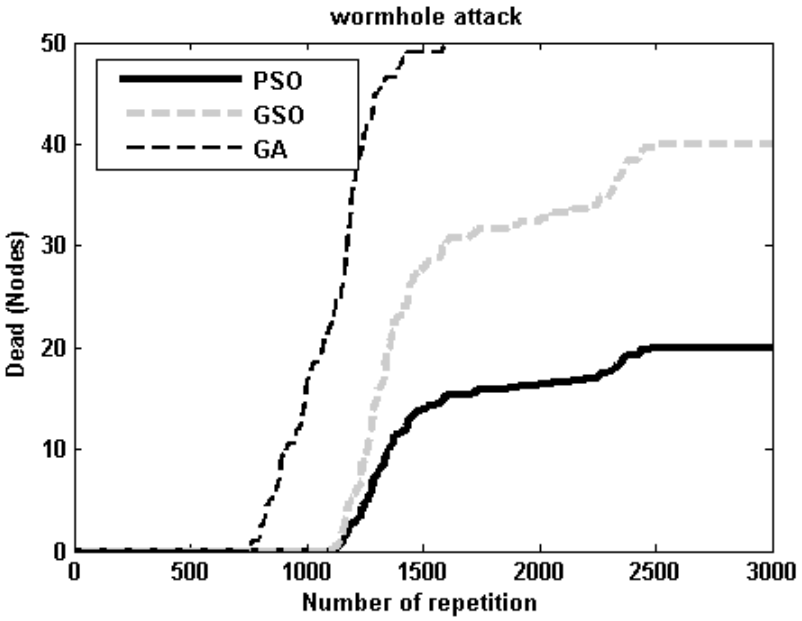

Figure 15: Analysis of dead nodes against wormhole attack

Figure 16 displays the status of living nodes in our algorithms against the repetitions. The findings made clear that in the PSO algorithm, the number of live nodes was equal to the high number of repetitions (about 15%). The GSO algorithm had fewer live nodes than the PSO algorithm and the GA algorithm had fewer live nodes (about 10%) than the other two (PSO & GSO) algorithms due to its low throughput.

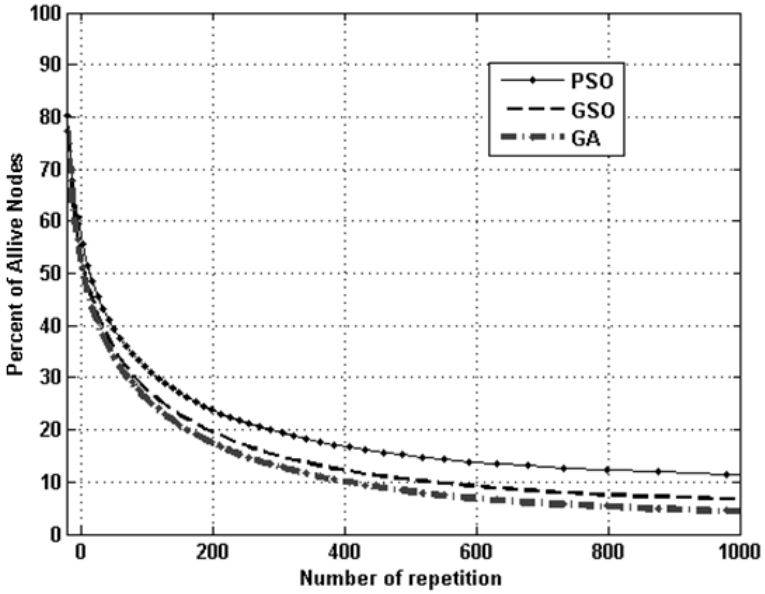

Figure 16: Analysis of live nodes against the number of repetitions

Figure 17 shows the status of live nodes in the three algorithms after the Dos attack. With the Dos attack on the GA algorithm, the percentage of nodes that remained alive in repetitions were low, but in the PSO algorithm, and due to the increasing number of repetitions, the percentage of live nodes was higher than the other two algorithms. Analysis pointed out that in 1300 iterations, in the GA algorithm only $1-2\%$ of nodes, $25-30\%$ in the GSO algorithm, and in the PSO algorithm, more than 90% of the nodes were alive and sending information.

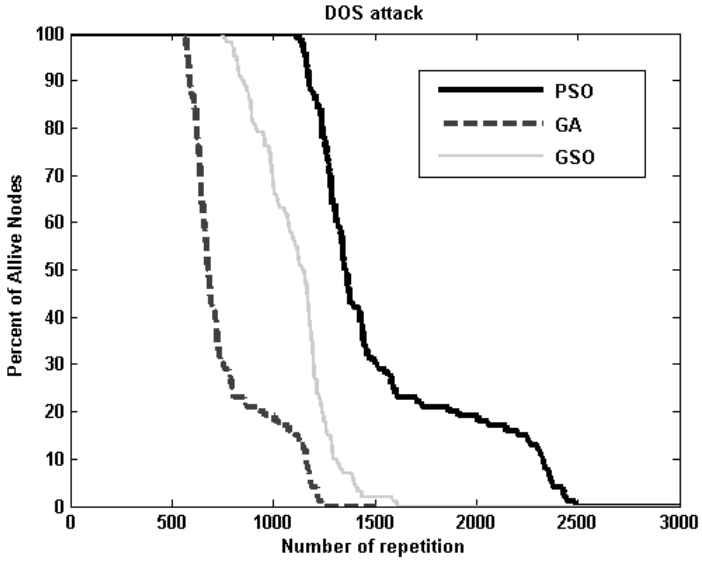

Figure 17: Assessment of live nodes against dos attack

4.4 Comparison of inaccurate warnings

As can be observed in Figure 18, we analyzed false negative warnings against repetition times. As evidenced, the GA algorithm generated a high false negative warning and proceeded upwards. In the GSO algorithm, it first proceeded with a gentle slope. However, due to the increasing complexity of the simulation, the slope became steeper and produced more false negative warnings. In contrast, the PSO algorithm followed a gentle slope against the frequency of repetitions, and its false negative signal production trend was less than the other two algorithms.

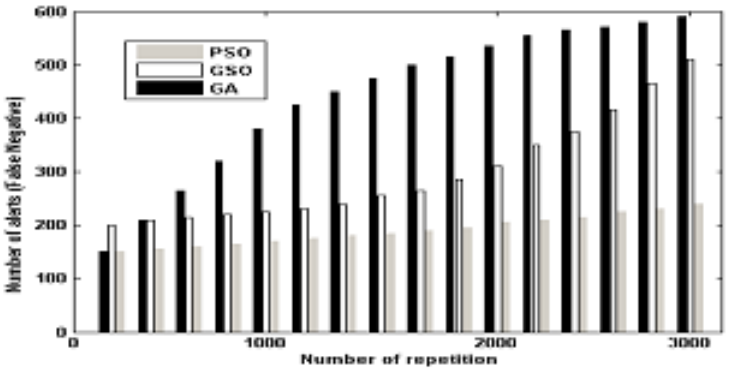

Figure 18: Investigation of false negative warning

The status of false positive alerts is depicted Figure 19. In the GA algorithm, the number of false positive alerts it generates is increased from 200-550, and therefore produces multiple false positive alerts. In the same number of iterations, the GSO algorithm's false negative warning increased from 150 to 500 and therefore this algorithm also produces a lot of false positive alarms. It is only the PSO algorithm that generates a lesser number of alerts than the other two algorithms (increased from 200 to 450).

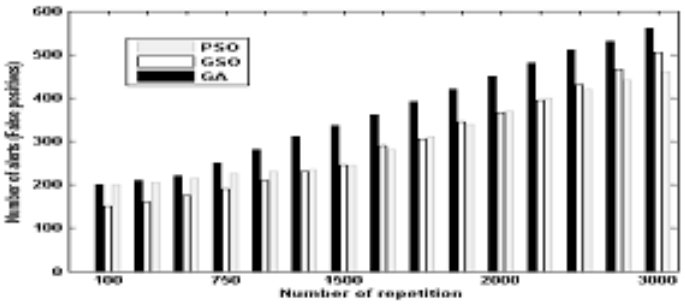

Figure 19: Assessment of false positive warnings

5 Conclusion

There are attacks on Internet networks that are generally destructive. Researchers deploy intrusion detection systems to detect attacks in various environments. Intrusion detection systems utilize meta-innovative methods to improve their performance. For this purpose, in this study, we assessed three meta-heuristic algorithms GA, PSO & GSO, and moreover introduced 4 attacks of Dos, D-dos, Wormhole and Sinkhole. Thereafter, we evaluated the efficiency of these algorithms in line with the four criteria of delay, throughput, energy consumption and the level of warnings.

The findings of our study indicated that the GA algorithm has the highest energy and the PSO algorithm has the highest throughput (among the other two algorithms). Since the issue of throughput and energy as well as the number of alerts is a question of network efficient, hence by merging the two algorithms PSO & GA, a new algorithm can be obtained possessing both high energy and high throughput with fewer false alarms, and consequently, very high network efficiency.

References

[1] N. Acharya and S. Singh, *An IWD-based feature selection method for intrusion detection system*, Soft Comput. **22** (2018), 4407–4416.

[2] S. Alqahtani and R. Gamble, *DDoS attacks in service clouds*, 48th Hawaii Int. Conf. Syst. Sci., 2015, pp. 5331–5340.

[3] A. Alzaqebah, I. Aljarah, O. Al-Kadi and R. Damaevicius, *A modified grey wolf optimization algorithm for an intrusion detection system*, Math. **10** (2022), no. 6, 999.

[4] J. Arora, *Introduction to Optimum Design*, McGraw-Hill, 1989.

[5] N.A. Azeez, B.B. Salaudeen, S. Misra, R. Damaevicius and R. Maskeliunas, *Identifying phishing attacks in communication networks using URL consistency features*, Int. J. Electron. Secur. Digit. Forensics **12** (2020), 200–213.

[6] M. Basha, N. Vivekananda and H. Bindu, *Evaluating the effect of attack on MANET routing protocols using intrusion detection system*, IJECT **5** (2014), 64–71.

[7] H. Bersini and F. J. Varela, *Hints for adaptive problem solving gleaned from immune networks*, Parellel Problem Solving from Nature, PPSW1, Dortmund, FRG, 1990.

[8] C. Blum and A. Roli, *Metaheuristics in combinatorial optimization: overview and conceptual comparison*, ACM Comput. Surv. **35** (2003), 268–308.

[9] M. Chahal and S. Harit, *Optimal path for data dissemination in Vehicular Ad HocNetworks using meta-heuristic*, Comput. Electric. Engin. **76** (2019), 40–45.

[10] Z.H. Chang and W. Wei-ping, *An improved PSO-based rule extraction algorithm for intrusion detection*, Int. Conf. Comput. Intell. Natural Comput., 2009, pp. 56–58.

[11] C.H. Chongeikloo, M. Munng, C.H. Leckie and M. Palaniswami, *Intrusion detection for routing attacks in sensor networks*, Int. J. Distributed Sensor Networks **2** (2006), 313–332.

[12] J. Chunlin, Zh. Yangyang, G. Shing, Y. Ping and L. Zhe, *Particle swarm optimization for mobile ad hoc networks clustering*, Int. Conf. Network. Sens. Control, Taipei. Taiwan, March 21-23, IEEE, **1** (2004), 372–375.

[13] G.B. Dantzig, *Linear programming and extensions*, Princeton University Press, 1963.

[14] P. Dixit, R. Kohli, A. Acevedo-Duque, R.R. Gonzalez-Diaz and R.H. Jhaveri, *Comparing and analyzing applications of intelligent techniques in cyberattack detection*, Secur. Commun. Netw. **2021** (2021), 5561816.

[15] M. Dorigo and T. Stutzle, *Ant colony optimization*, MIT Press, 2004.

[16] S. Einy, Oz. Cemil and Y. Dorostkar Navaei, *Network intrusion detection system based on the combination of multiobjective particle swarm algorithm-based feature selection and fast-learning network*, Wireless Commun. Mobile Comput. **2021** (2021), 6648351, 12 pages.

[17] J. D. Farmer, N. Packard and A. Perelson, *The immune system, adaptation and machine learning,* Phys. D. **2** (1986), 187–204.

[18] M. Feng and H. Pan, *A modified PSO algorithm based on cache replacement algorithm*, 10th Int. Conf. Comput. Intell. Secur., 2014, pp. 558–562.

[19] Z.W. Geem, J. H. Kim and G.V. Loganathan, *A new heuristic optimization: Harmony search*, Simul. **76** (2001), 60–68.

[20] D. Ghose and K. Krishnanand, *Glowworm Swarm Optimization for simultaneous capture of multiple local optima of multimodal functions*, **3** (2009), no. 2, 87–124.

[21] F. Glover and G.A. Kochenberger, *Handbook of Metaheuristics*, Springer, 2003.

[22] F. Glover and M. Laguna, *Tabu Searc*, Kluwer, Boston, 1997.

[23] F. Glover, *Future paths for integer programming and links to artficial intelligence*, Comput. Oper. Res. **13** (1986), 533–549.

[24] D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Reading, Mass, Addison Wesley, 1989.

[25] A. Gupta and S.P. Ranga, *Wormhole Detection Methods in Manet*, Int. J. Enterprise Comput. Bus. Syst. 2 (2012), no. 2, 1–8.

[26] O.B. Haddad, A. Afshar and M.A. Marino, *Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization*, Water Resources Manag. **20** (2006), 661–680.

[27] A. Haghighat, M. Esmaeili, A. Saremi and V.R. Mousavi, *Intrusion detection via fuzzy-genetic algorithm combination with evolutionary algorithmsll*, 6th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS 2007). IEEE, 2007, pp. 587–591.

[28] J. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[29] M. Hosseinzadeh-Aghdam and P. Kabiri, *Feature selection For intrusion detection system using ant colony optimization*, Int. J. Network Secur. **18** (2016), no. 3, 420–432.

[30] P. Kabiri and M. Aghaei, *Feature Analysis for Intrusion Detection in Mobile Ad-hoc Networks*, Int. J. Network Secur. **12** (2011), no. 1, 42–49.

[31] D. Karaboga, *An idea based on honey bee swarm for numerical optimization*, Technical Report TR06, Erciyes University, Turkey, 2005.

[32] J. Kennedy and R.C. Eberhart, *Particle swarm optimization*, Proc IEEE Int. Conf. Neural Networks Piscataway, 1995, pp. 1942–1948.

[33] S. Kirkpatrick, C.D. Gelatt and M. P. Vecchi, *Optimization by simulated annealing*, Sci. **220** (1983), no. 4598, 671–680.

[34] K. Krishnanand and D. Ghose, *Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications*, Multiagent Grid Syst. **2** (2006), no. 3, 209–222.

[35] R. Kulkarni, A.Venayagamoorthy Miller and C. Dagli, *Network-centric localization in MANETs based on particle swarm optimization*, IEEE Swarm Intell. Symp., IEEE, 2008.

[36] M. Kumar, M. Hanumanthappa and S. Kumar, *Intrusion detection system-false positive alert reduction technique*, **2** (2011), no. 3, 37–40.

[37] L. Lifang He, X. Xiong and S. Huang, *A glowworm swarm optimization algorithm with improved movement rule*, Fifth Int. Conf. Intell. Networks Intell. Syst., 2012, pp. 109–112.

[38] A. Mokarian, A. Faraahi and A. Ghorbannia-Delavar, *False positives reduction techniques in intrusion detection systems-A review* , IJCSNS Int. J. Comput. Sci. Network Secur. **13** (2013), no. 10, 128–134.

[39] P. Moscato, *On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms*, Caltech Concurrent Computation Program (report 826), 1989.

[40] Md. Mostaque and H. Morshedur, *Network intrusion detection system using genetic algorithm and fuzzy logic*, Int. J. Innov. Res. Comput. Commun. Engin. **1** (2013), no. 7, 1435–1445.

[41] N. Neha Rai and KH. Rai, *Genetic Algorithm Based Intrusion Detection system,* Int. J. Comput. Sci. Inf. Technol. **5** (2014), 4952–4957.

[42] S. Owais, P. Pavel Krmer and A. Abraham, *Survey: Using genetic algorithm approach in intrusion detection systems techniques*, 7th Comput. Inf. Syst. Ind. Manag. Appl., IEEE, 2008, pp. 300–307.

[43] K. M. Passino, *Biomimicry of bacterial foraging for distributed optimization and control*, IEEE Control Syst. Mag. **22** (2002), no. 3, 52–67.

[44] I. Pavlyukevich, *Lévy flights, non-local search and simulated annealing,* Comput. Phys. **226** (2007), 1830–1844.

[45] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. Zaidi, *The Bees Algorithm*, Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.

[46] K. Price, R. Storn and J. Lampinen, *Differential evolution: A practical approach to global optimization*, Springer, 2005.

[47] R.F. Moritz and E.E. Southwick, *Bees as superorganisms*, Springer, 1992.

[48] S. Rethinavalli and R. Gopinath, *Classification approach-based sybilnode detection in mobile ad HOC networks*, Int. J. Adv. Res. Engin. Technol. **11** (2020), no. 12, 3348–3356.

[49] K. Rhee, *Detecting inner attackers and colluded nodes in wireless sensor networks using hop-depth algorithm*, J. Instit. Electron. Engin. Korea CI **44** (2007), no. 1, 113–121.

[50] O.J. Rotimi, S. Misra, A. Agrawal, E. Azubuike, R. Maskeliunas and R. Damasevicius, *Curbing Criminal Acts on Mobile Phone Network*, Cyber Security and Digital Forensics; Springer:Berlin/Heidelberg, Germany, 2022, 99-111.

[51] R.Y. Rubinstein, *Optimization of computer simulation models with rare events*, Eur. J. Oper. Res. **99** (1997), 89–112.

[52] S. Nakrani and C. Tovey, *On honey bees and dynamic server allocation in Internet hosting centers*, Adaptive Behav. **12** (2004), 223–240.

[53] F. Sabahi and A. Movaghar, *Intrusion detection: A survey* , Third Int. Conf. Syst. Networks Commun., 2008, pp. 23–26.

[54] S. Sevil and J. Clark, *Evolutionary computation techniques for intrusion detection in mobile ad hoc networks*, Department Comput. Engin. **55** (2011), no. 15, 3441-3457.

[55] J. Soryal and T. Saadawi, *IEEE 802.11 Denial of Service Attack Detection in MANET,* Wireless Telecommun. Symp. (WTS), (2012), 1-8.

[56] R. Storn and K. Price, *Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces*, J. Glob. Optim. **11** (1997), 341–359.

[57] E. G. Talbi, *Metaheuristics: From Design to Implementation*, John Wiley & Sons, 2009.

[58] C. Trang, H. Kong and H. Lee, *A distributed intrusion detection system for AODV*, Asia-Pacific Conf. Commun., IEEE, 2006, pp. 1–4.

[59] S. Voss, *Meta-heuristics: the state of the art*, Local Search for Planning and Scheduling (Ed. A. Nareyek), LNAI **2148** (2001), 1–23.

[60] T. Weise, *Genetic programming for sensor networks*, Technical Report (2006), 1–16.

[61] X.S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, UK, 2008.

[62] X.S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley & Sons, 2010.

[63] X.S. Yang and S. Deb, *Engineering optimization by cuckoo search*, Int. J. Math. Modell. Num. Optim. **1** (2010), no. 4, 330–343.

[64] X.S. Yang, *Firefly algorithms for multimodal optimization*, 5th Symp. Stochastic Algorithms Found. Appl. LNCS **5792** (2009), 169–178.

[65] X.S. Yang, *Engineering optimization via nature-inspired virtual bee algorithms*, IWINAC Lecture Notes Comput. Sci. **3562** (2005), 317–323.

[66] Y. Zhou, G. Cheng, S. Jiang and M. Dai, *Building an efficient intrusion detection system based on feature selection and ensemble classifier*, Comput. Netw **174** (2020), 107–247.