IDS using Hybrid Ant Colony-Genetic Algorithm (GAAPI) for Attack Classification and Detection

M. S. Kale, Dr. P. K Deshmukh, S. B. Javheri
1,2,3Dept. of Computer Engineering, Rajarshi Shahu College of Engineering University of Pune
Tathawade, Pune, India

Abstract
In recent fast growing internet world network security is become an essential need. Latterly one of the most popular research topics in network security is Intrusion Detection Systems (IDSs) which try to keep safe our network from intruders. This paper describes an IDS using Hybrid Ant Colony Genetic algorithms (GAAPI). It is hybridization of a special class of ACO called API and a Real Coded Genetic Algorithm (RCGA). To use characteristics of both algorithms to solve unconstrained problems. The proposed system is evaluated by KDD cup dataset. It is therefore the goal of this paper to provide a comprehensive overview of the application of GAAPI to solve complex global continuous problems. Proposed GAAPI is to detect and classify attacks of KDD dataset. Aim is to minimize the time and classify attacks with global optimal solution.

Keywords
Internet, Ant Colony Optimization (ACO), Genetic Algorithm (GA), Knowledge Discovery and Data Mining (KDD) Cup 99 Dataset

I. Introduction
In recent computer world internet expansion, information security and privacy have become more important Security methods like cryptography and firewalls do not satisfy user’s needs. This causes the use of more complex security systems, such as Intrusion Detection Systems (IDS’s), to be crucial.

IDS play an important role in network security. IDS gather information from a computer or computers network and attempts to detect intruders or system abuse. Generally, IDS will notify a human analyst of a possible intrusion and take no further action, but some newer systems take active steps to stop an intruder at the time of detection.

Intrusion Detection Systems are like a burglar alarm for your computer network they detect unauthorized access attempts. They are the first line of defence for your computer systems. IDSs are categorized into two major groups: network based and host-based.

Network-based IDSs detect attacks based on network traffic analysis but host-based IDSs use system information like CPU load, system calls and etc., for detection purpose.

IDS techniques are arranged into three general groups: Anomaly Detection, Signature Detection (Misuse) and hybrid. Anomaly detection IDSs model normal behavior of system and consider each event that differs from this model more than a threshold value as an intrusion. Signature based IDSs have a database of previous attack signatures and compare each behavior with database entries. If a match found, report it as intrusion.

Modeling normal behavior of a system, in anomaly IDSs, is so cumbersome due to its complexity. But in other hand, they are best suited for detection of new attacks.

Hybrid IDSs take advantages of Ant Colony and Genetic Algorithm fort attack classification and detection. This paper adopts the hybridization of a special class of ACO called API and a Real-Coded Genetic Algorithm (RCGA). Compared to its ACO surrogates, who were mainly applied to discrete optimization problems, API was particularly designed for continuous optimization problems.

Several IDSs employ intelligent method. This hybrid Ant Colony and Genetic algorithm use sets of features, one is limited to basic KDD features and the other consists of all 21 attacks and 41 packet features. This proposed GAAPI algorithm finds optimal solution to solve complex problems may arise in computing world. This Hybrid algorithm adopts some favorable features of Ant Colony Optimization (ACO) and Genetic Algorithm (GA).

The paper is organized as follows: Section II, introduces Ant Colony Optimization (ACO) and Real Coded Genetic algorithm. Section III, describes proposed Intrusion Detection system. Section IV, introduces KDD cup 99 data set. Section V, presents the conclusion of this work.

II. Ant Colony Optimization
In computer science and operations research, the ant Colony Optimization Algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. This algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the first algorithm was aiming to search for an optimal path in a graph, based on the behaviour of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behaviour of ants.

Metaheuristic algorithms are algorithms which, in order to escape from local optima, drive some basic heuristic: either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one. The metaheuristic part permits the low level heuristic to obtain solutions better than those it could have achieved alone, even if iterated. Usually, the controlling mechanism is achieved either by constraining or by randomizing the set of local neighbor solutions to consider in local search (as is the case of simulated annealing or tabu search), or by combining elements taken by different solutions (as is the case of evolution strategies and genetic or bionomic algorithms).

A. ACO for Continuous Search Domains
This paper adopts the hybridization of a special class of ACO called API and a Real-Coded Genetic Algorithm (RCGA) similar to the approach of the evolutionary strategy with float representation. Compared to its ACO surrogates, which were mainly applied to discrete optimization problems, API was particularly designed for continuous optimization problems.

The proposed ant-based algorithm differs in terms of search strategy from the basic ACO in that the ants have to navigate by...
using their memory of the visual landmarks encountered along with the familiar routes, instead of using pheromones. Further, API aims at maximizing the prey instead of minimizing the path. API has also a good “downhill” (gradient descending) search behavior. One of its drawbacks though, is that it may end up quickly in a local minimum due to a constant movement of the nest only in the best position found by the ants (the search agents).

ACO started to be analyzed from the continuous optimization perspective in the analytical way only recently, in spite of the fact that the first proposal to adapt ACO for continuous optimization dates back to 1995. Bilech and Parmee’s proposal, entitled continuous ant colony optimization, initializes a nest at a given point of the search space. Then, it generates random vectors corresponding to the directions that will be followed by each ant in its attempt to improve the solution. If an ant is successful in such a pursuit, the direction vector chosen is updated. The continuous pheromone model proposed later on is a more complex approach which uses a Gaussian probabilistic approach for the pheromone update.

This model consists of a Gaussian pdf centered on the best solution found so far in the search process (up to the current iteration). The variance vector of this pdf starts with a value three times greater than the range of each variable of the problem (e.g., each ant takes a proportion from the solution space to explore). Then, this variance value is modified according to a weighted average of the distance between each individual (ant) in the population and the best solution found so far. The variance vector depends only on the number of ants. The main drawback of this model is the fact that “it only investigates one promising region of the problem at a time”, and it may therefore suffer from premature convergence.

An ACO algorithm for the continuous domain is proposed; this algorithm can avoid premature convergence, and therefore local optima trapping. The proposed algorithm uses an archive that holds a predefined number of the best solutions found so far. Each solution corresponds to the center of a different Gaussian pdf.

B. Genetic Algorithm

The term genetic algorithm, almost universally abbreviated nowadays to GA, was first used by John Holland. Genetic Algorithms are a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures as to preserve critical information. Genetic algorithms are often viewed as function optimizer, although the ranges of problems to which genetic algorithms have been applied are quite broad. An implementation of genetic algorithm begins with a population of (typically random) chromosomes.

One then evaluates these structures and allocated reproductive opportunities in such a way that these chromosomes which represent a better solution to the target problem are given more chances to ‘reproduce’ than those chromosomes which are poorer solutions. The ‘goodness’ of a solution is typically defined with respect to the current population. Genetic algorithms are metaheuristics used for solving problems with both discrete and continuous variables. The population is the main element of genetic algorithms, and the genetic operations like crossover and mutation are just instruments for manipulating the population so that it evolves towards the final population including a “close to optimal” solution. The requirements set on the population also change during the execution of the algorithm.

C. Hybrid ACO-GA

In recent years, there were proposals to hybridize ACO and GA in a number of optimization applications. Both of them refer to the combinatorial optimization model of ACO. Migration of solutions from one algorithm to the other occurs whenever any of them finds an improved potential solution after iteration. Thus, a percentage of the best solutions of ACO (say K %) are added to the GA population pool which will follow the breeding process proportional to their fitness. Then, another percentage of the best individuals in the GA (L% of the GA population) are used to add “fitness-proportional pheromone” in the ACO search process and a new percentage of the worst fitted individuals of GA (M %) are used to “evaporate a constant amount of pheromones” in the ACO search. When both algorithms find an improvement, or both algorithms do not improve after iteration, then no migration takes place. The application of the hybrid ACO-GA.

D. Background Information And Concepts

The algorithm proposed in this paper addresses optimization problems in the continuous domain of the form

$$\min f(x) \quad x \in H$$

where $$x \in \mathbb{R}^n$$, and $$H = \{ x \in \mathbb{R}^n | li \leq xi \leq ui \}$$, $$i = 1, \ldots, n$$, with $$li$$ and $$ui$$ being the lower and upper bounds of $$xi$$, respectively. We are particularly interested in unconstrained optimization problems, and thus it is assumed that the set $$H$$ is wide enough such that $$H \supseteq \mathbb{H}_0 = \{ x \in \mathbb{R}^n | f(x) \leq c \}$$, for a sufficiently large real number $$c$$.

Further, it is assumed that the function $$f(x)$$ is continuous on $$H$$, and that $$H \cap \mathbb{H}_0$$ is a nonempty and compact set for a real number $$c$$. An interpretation of the $$\mathbb{H}_0$$ and $$c$$ for the algorithm proposed in this seminar is given in following section of the paper.

Suppose that $$x^* = \{x_1^*, x_2^*, \ldots, x_n^* \}$$ is a globally optimal solution and $$\varepsilon > 0$$ is a sufficiently small number. If a candidate solution $$\tilde{x} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n \}$$ satisfies

$$|x_i - \tilde{x}_i| \leq \varepsilon, \quad i = 1, \ldots, n$$

or

$$|f(x) - f(\tilde{x})| \leq \varepsilon$$

then, $$\tilde{x}$$ is called an $$\varepsilon$$-optimal solution of the problem defined in (1).

Fig. 1: Search Space Division According to API Strategy. S = R^2 Denotes a Bi-dimensional Solution Space; s1, s2, s3 are Sites Randomly Generated Around the nest, their Maximum Distance from the Nest Being Given by Aant. The Small Squares Denote Local Exploration of the Site (Points Situated at a Maximum Distance of Asite from the Site Center).
In the case of the algorithm proposed in this seminar, to find ε-optimal solutions, the feasible solution space \([l, u]\) is divided into smaller solution spaces with different amplitudes (defined as a percentage of the search space) from the initial domain, where overlapping is allowed. Fig. 1 shows how the initial solution space is divided into smaller search spaces. The example in fig. 1 is given for a 2-D search space. This approach is the one adopted by Monmarche in his thesis when proposing the API algorithm.

The nest \(N\) initially takes a random position in the feasible search space \([l, u]\), \(l = (l_1, l_2, ..., l_n)\) and \(u = (u_1, u_2, ..., u_n)\) are the lower and upper bound vectors for each dimension, respectively, delimitating the feasible solution space in \(\mathbb{R}^n\) (\(n\) is the dimension of the problem). Therefore, \(N = (N_1, N_2, ..., N_n)\) is the initial position of the nest in the feasible solution space. The amplitudes for search space division change dynamically.

The formula used to determine the search amplitude of each agent (ant) is given by

\[
A_{\text{ant}_i} = \left(1 - \frac{k}{N_{\text{ants}}}\right) G_{\text{ant}_i}
\]

where \(A_{\text{ant}_i}\) is the radius from the nest, delimitating the solution space ant \(i\) can cover; \(k\) is the current index (iteration of the search loop) of ant \(i\); \(N_{\text{ants}}\) is the total number of search agents, and \(G_{\text{ant}_i}\) is the age of the ant and it is a parameter that increases as ant \(i\) performs its tasks with time, and is computed by

\[
G_{\text{ant}_i} = \frac{T_{i}}{T_{\text{ant}_i}}
\]

This parameter was inspired from the real behavior of pachycondyla apicalis ants. \(T_{i}\) is the current number of iterations after the movement of the ant \(i\), and \(T_{\text{ant}_i}\) is the maximum number of iterations between two movements of the ant \(\text{ant}_i\).

### III. Proposed intrusion detection system

This section of the paper describes the proposed Intrusion Detection system with KDD Cup 99 dataset. Proposed IDS uses GA-API algorithm that is described in section IV. In this paper IDS will work as following phases.

#### A. Attack Conversion

In this phase first KDD 99 attacks are convert from ASCII values to number values.

#### B. Attack Normalization

In this phase attacks are normalized and vertical mining is performed on converted number values of attacks.

#### C. Discrete Function

We construct this function to divide the attacks in several parts to further perform processing.

#### D. Proposed GA-API Algorithm

This algorithm use for the attack classification and detection process. A GA-API algorithm that incorporates some favorable features of GA and API algorithms is proposed in this paper. The idea in GA-API is to keep the algorithm focused toward continuously improving in the solution, while avoiding trapping in local optima. Therefore, the API algorithm was intended to be the core of the GA-API, keeping the search tracked toward improvement in the solution, while RCGA was intended to provide the escape mechanism from local optima when API is trapped. Thus, when API is at the search level of sites (the lowest search level) and continuously improves the solution, RCGA is in a passive mode. In this passive mode, the population of RCGA is formed by all the best solutions generated by API at the ant level only (there are no sites to be forgotten).

When API is slow in improving the solution (there are sites to be forgotten due to failure in improving the solution), RCGA switches to an active role. This time, its population uses the information of forgotten sites as well (the population is more heterogeneous than in the former case), and thus the solution generated by the RCGA has more chances to be far from the local optimum in which the API was trapped.
Fig. 3: Proposed GAAPI Flowchart

GAAPI has a well-established balance between exploration (with API and RCGA) and exploitation (API). API keeps the algorithm focused toward the global optimum, moving the nest position (the point where exploitation starts) only in the best solution found so far, while RCGA helps the ants to use useful information of less explored regions (forgotten sites). The strong influence of API with its “down-hill” (gradient descending) behaviour may increase the speed of convergence toward the global when compared to other powerful global search techniques such as PSO, EAs, or GAs, where the exploration behaviour may play a strong role.

IV. KDD Cup 99 dataset
This is the data set used for the Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 the Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between “bad” connections, called intrusions or attacks, and “good” normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. Since 1999, KDD’99 has been the most wildly used data set for the evaluation of anomaly detection methods. This data set is prepared by Stolfo et al. and is built based on the data captured in DARPA’98 IDS evaluation program. KDD training dataset consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labelled as either normal or an attack, with exactly one specific attack type. The simulated attacks fall in one of the following four categories:

A. Denial of Service Attack (DoS)
Is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.

B. User to Root Attack (U2R)
Is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

C. Remote to Local Attack (R2L)
Occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.

D Probing Attack
Is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls. The datasets contain a total number of 24 training attack types, with an additional 14 types in the test data only. KDD ’99 features can be classified into three groups:

1. Basic Features
This category encapsulates all the attributes that can be extracted from a TCP/IP connection. Most of these features leading to an implicit delay in detection.

2. Traffic Features
This category includes features that are computed with respect to a window interval and is divided into two groups:

3. Content Features
Unlike most of the DoS and Probing attacks, the R2L and U2R attacks don’t have any intrusion frequent sequential patterns.

V. Conclusion
In this paper we have presented a GAAPI algorithm was proposed to solve global unconstrained continuous optimization problems. This algorithm is appropriate for optimization problems whose decision variables take values from the real number domain. The hybrid meta-heuristic algorithm proposed in this paper was created by combining some unique characteristics of two other robust meta-heuristic algorithms: RCGA and API. There are at least two main reasons why GAAPI performs better than other powerful heuristic techniques. First, the balance in exploration and exploitation given by the two chosen algorithms API and GA is one of the reasons. API has a strong influence targeting the search toward a continuously improved solution (the nest is moved only in the best solution found at each iteration by its ants), while GA has an active role in the solution search, only when API reduces its speed of convergence (the solution does not improve much from one iteration to another, or there are many failures in exploiting different sites). Proposed GAAPI algorithm finds optimal solution for Intrusion detection system with KDD Cup 99 dataset attack classification and detection. We are working on experiments and results of this proposed IDS using GAAPI algorithm.
References


