Evolving Lightweight Intrusion Detection Systems for RPL-based Internet of Things

Ali Deveci[1][0000−0002−4990−0785], Selim Yilmaz[1,2][0000−0002−9516−6892], and Sevil Sen[1][0000−0001−5814−9973]

[1] WISE Lab., Dept. of Computer Engineering, Hacettepe University, Ankara, Turkey
alideveci1984@hotmail.com
ssen@cs.hacettepe.edu.tr
[2] Dept. of Software Engineering, Muğla Sıtkı Koçman University, Muğla, Turkey
selimyilmaz@mu.edu.tr

Abstract. With the integration of efficient computation and communication technologies into sensory devices, the Internet of Things (IoT) applications have increased tremendously in recent decades. While these applications provide numerous benefits to our daily lives, they also pose a great potential risk in terms of security. One of the reasons for this is that devices in IoT-based networks are highly resource constrained and interconnected over lossy links that can be exposed by attackers. The Routing Protocol for Low-Power and Lossy Network (RPL) is the standard routing protocol for such lossy networks. Despite the efficient routing built by RPL, this protocol is susceptible to insider attacks. Therefore, researchers have been working on developing effective intrusion detection systems for RPL-based IoT. However, most of these studies consume excessive resources (e.g., energy, memory, communication, etc.) and do not consider the constrained characteristics of the network. Hence, they might not be suitable for some devices/networks. Therefore, in this study, we aim to develop an intrusion detection system (IDS) that is both effective and efficient in terms of the cost consumed by intrusion detection (ID) nodes. For this multiple-objective problem, we investigate the use of evolutionary computation-based algorithms and show the performance of evolved intrusion detection algorithms against various RPL-specific attacks.

Keywords: IoT · RPL attacks· intrusion detection· multi-objective optimization· genetic programming.

1 Introduction

IoT, which enables a variety of devices to be connected to each other, is one of the most breakthrough advancements in our era. A great deal of IoT applications have found use in various domains including smart home, smart city, logistic monitoring, e-health, and the like. That's why the number of smart devices enabling such IoT applications has long been increasing. The total installed base of these devices is estimated to be 75 billion, a five-fold increase in ten years,

and machine-to-machine (M2M) connections are estimated to constitute half of global connections by 2030 [22,6].

Low Power and Lossy Networks (LLN) are a type of IoT that provide lossy communication among IPv6-enabled resource-constrained devices. They are characterized by their constrained communication with high packet loss, low throughput, and limited frame size [1]. In a typical implementation of LLN, each of the resource-constrained nodes can communicate with each other, but also connects to a special node, called the LLN Border Router (LBR) in order to connect to the Internet. In order to build routes among nodes in such a constrained network, RPL was developed by IETF-ROLL in 2012 [1], and is adopted as a standard routing protocol for LLNs today.

Although RPL is good at building efficient routes between nodes in an LLN, it is still very susceptible to attacks, especially insider attacks. The results of such attacks can be vital considering the applications of LLNs in critical systems such as healthcare, smart home. Therefore, researchers have been working on developing effective intrusion detection systems for RPL-based IoT. However, most studies in the literature mainly focus on detecting attacks against RPL and overlook the suitability of developed IDS to such low-power and lossy networks. Therefore, the main aim of this study is to explore on developing intrusion detection systems that show both high accuracy and low cost.

Here, a distributed intrusion detection architecture in which a global ID node is placed at the root node and some other nodes participate in intrusion detection by sending their local information to the root node is explored. Although involving the monitoring nodes brings about an additional burden to the network and devices, they enable the global ID node to capture intrusions on a global scale, hence more effectively. Here, the size of the information collected and sent by the monitoring nodes becomes important in terms of resource consumption. In addition to increasing communication cost, large packets sent by monitoring nodes might also lead to fragmentation. Therefore, in this study, while developing intrusion detection systems, beside their accuracy, the information used and sent by the monitoring nodes is taken into account. For simplicity, this information regarding the cost of intrusion detection and communication is taken as the number of features used for training in this study. Therefore, the detection accuracy and the number of features extracted from both the ID node and the monitoring nodes must be tuned simultaneously to generate an effective and efficient IDS, which is the main motivation of this study.

In order to solve this multi-objective optimization problem, we employ genetic programming (GP) due to its ability to explore search space efficiently for complex environments such as LLNs and to also handle multiple objectives (i.e., accuracy and the number of features in this study) simultaneously. The main objective of GP is to evolve a detection program (or model) that finds a good trade-off between accuracy and the minimum number of features used. Only the evolved features are extracted and sent by the monitoring ID nodes to the central ID node, which then periodically runs the evolved program. To handle multiple objectives by GP, we employ Non-dominated Sorting Genetic

Algorithm II (NSGA-II) [8], one of the most popular Pareto-based evolutionary multi-objective algorithms. The following four attacks are covered in this study: worst parent, hello flood, increased version, and decreased rank attacks. Various network scenarios with these four attacks in which attackers are placed in different locations are evaluated and discussed. The experimental results show that the increase in the number of nodes and the number of data packets used in intrusion detection also increases the number of features used as expected, resulting in an increase in power consumption and a decrease in network performance. For WP, HF, and IV type attacks, GP can produce a satisfactory ID program with an average detection accuracy of 94%. On the other hand, limiting the number of features has an adverse impact on the detection of DR.

The paper is organized as follows. The background information on RPL and insider attacks against RPL as well as the methods used in this study, namely GP and NSGA-II are given in Section 2. The related studies in the field of intrusion detection in RPL are given in Section 3. The proposed approach is given in detail in Section 4. The experimental settings and results are provided in Section 5. The strengths, limitations, and possible future directions of this study are discussed in Section 6. Finally, Section 7 concludes this study.

2 Backround

This section gives background information on both RPL, specific attacks against RPL and the methods used for detecting these attacks, namely evolutionary computation techniques.

2.1 RPL

RPL is a distance vector and source routing protocol and becomes standard for low-power and lossy networks [13]. RPL aims to create Destination Oriented Directed Acyclic Graphs (DODAG). A gateway node, known as the root node, is responsible for the formation of DODAG by broadcasting control messages in RPL called the DODAG Information Object (DIO). DIO messages are initially transmitted only by the root node to construct an upward route from the sensor nodes to itself. Nodes receiving multiple DIO packets from their neighbors select the most suitable candidate parent nodes considering the rank values in the DIO packets, determine their rank, and transmit the modified DIO packet. The Destination Advertisement Object (DAO) packets, however, are used to construct downward routes. The DAO packet is unicast by all nodes to their selected parents. The downward routes in DODAG are operated in two modes: *storing mode* and *non-storing mode*. In the non-storing mode, only the root node keeps the routing table, and hence the nodes rely on only the root node for forwarding their packets to the destination nodes. In the storing mode, a routing table is kept by each node in the network; and instead of sending every incoming packets to the root node, the nodes forward them to the next hop that is on the route to the destination address. Another control packet, called DODAG Information

Solicitation (DIS), is broadcast by a new node to join DODAG. Upon receiving the DIS packet, a node returns with a DIO packet, thus sharing the DODAG configuration that is necessary for the requesting node. The objective function such as hop count specifies how a node computes its rank value that is used in parent selection. Although there are many OF types in the literature, Objective Function Zero (OF0) and Minimum Rank with Hysteresis Objective Function (MRHOF) are proposed as the default OF for RPL-based networks [9].

RPL Attacks: Although RPL has some security mechanisms specified in its RFC [1], it is susceptible to insider attacks. Attacks against RPL are classified according to what they primarily target in the literature [16]:

- **Attacks targeting resources:** Attackers aim to make legitimate nodes exhaust their resources by forcing them to perform unnecessary processing, causing also the available links to be down.
- **Attacks targeting topology:** Attackers dramatically affect the construction of the RPL topology in a non-optimal way or lead to the isolation of some nodes from the topology.
- **Attacks targeting traffic:** Attackers interfere with network traffic and try to change the traffic pattern.

This study focuses on the following attacks targeting resources and topology:

- Decreased Rank (DR): In this type of attack, the attacker node illegitimately advertises a lower rank value to other nodes in the network. As a result, benign nodes inevitably send their packet through the attacker node. Consequently, the entire network traffic may be controlled by this malicious node. This attack is considered a first step for the forthcoming attacks, increasing the severity of the attackers.
- Increased Version (IV): The version number is necessary for the global repair in RPL. It is propagated in DIO packets throughout the network and is increased only by the root node. In this type of attack, the attacker node illegitimately increases the value of the version field, resulting in unnecessarily rebuilding the networks.
- Hello Flood (HF): The main purpose here is to unnecessarily increase the size of network traffic by generating a large number of DIS packets, leading to a dramatic consumption of the network's resources.
- Worst Parent (WP): In this attack, the attacker node intentionally selects its worst parent to route incoming packets to degrade the routing performance of the network. The consequence of this attack is unoptimized routes between the nodes, which reduces the performance of the network.

2.2 Evolutionary Computation

Evolutionary computation is inspired by natural evolution and has been shown to be very effective in solving many problems in different domains, including intrusion detection [24][21]. Due to being very good at discovering complex characteristics of a system and being able to solve multi-objective optimization problems,

it is explored in this study. While there are many evolutionary computation techniques in the literature, GP which represents candidate individuals as trees is used here. In the following, GP and one of the multi-objective evolutionary algorithms, namely NSGA-II, are introduced in detail.

Genetic Programming: GP [14] is one of the most popular evolutionary-based computation techniques that is inspired by the 'survival of the fittest' theory [11]. Because GP is a very simple yet effective learning approach, it has been used to solve a wide range of real-world problems in many research domains.

As a population-based learning algorithm, GP aims to find possibly the best solution across generations. A number of agents, called individuals, participate in the population, and each of them represents a candidate solution to the problem. Individuals are encoded with a tree structure, called GP tree, where terminal and non-terminal types of node take part. The terminals and non-terminals form the leaf and intermediate nodes of the GP tree, respectively.

Evolution in GP starts with an individual set that is initially generated randomly. The individuals are then evaluated and assigned a *fitness value* that indicates how well this candidate solution can solve the targeted problem. Afterward, individuals undergo three genetic operators in each generation that include *selection*, *crossover*, and *mutation* to breed their offspring. In selection, a pair of individuals is selected, and the fitness value of an individual plays a key role here to determine if it is reproduced in the next generation. The selection is made based on the selection operators such as tournament selection where a number of individuals is picked randomly first, then the fittest individual is selected from that subpopulation. They produce two new offspring individuals in crossover by replacing the subtrees rooted at the crossover point randomly determined. In mutation, however, subtrees of the offspring individuals are also replaced by, contrary to crossover, randomly generated new subtrees. Hence, better individuals are aimed to be evolved through generations. GP reaches the end of the generation once the termination condition is satisfied. There may be different conditions such as reaching the total number of generations, approximating well to the ideal or optimum solutions, and the like. The general steps of the GP algorithm are given in Algorithm 1.

Algorithm 1: Basic steps of GP.

```
1 Initialize population;
2 repeat
3 |   Calculate the individuals' fitness;
4 |   Sort and rank populations by fitness value;
5 |   Reproduce/Regenerate the new population using GP operators (mutation,
  |     crossover etc.);
6 until a termination criterion is satisfied;
7 return best-of-run individual
```

Multi-objective Optimization: Multi-objective optimization is a task that aims to solve problems that involve two or more conflicting objectives. Most of the problems in real life today fall into this category. Until now, a great deal of effort has been put into developing evolutionary-based heuristic approaches that effectively solve such problems. Among them, the Pareto-based approaches are the most popular, and the majority of studies adopt the Pareto strategy where a set of solutions is achieved (called non-dominated solution or Pareto optimal solution) rather than a single solution. In the Pareto-based approaches, a solution x is said to be better than another solution y provided that x 'dominates' y for every objective. So, the main goal of the Pareto-based approaches is to have a set of solutions, called a Pareto optimal set, that is not dominated at the end of the optimization process. The set of objective values corresponding to the Pareto optimal set is called the Pareto front.

As being one of the most popular evolutionary-based approaches targeting multi-objective problems, NSGA-II [8] relies on Pareto domination of solutions in the objective space. Here, the candidate solutions from the previous and current population are split into several fronts according to their Pareto dominance and crowding distances, and the solutions that belong to better fronts are allowed to survive to the next generations. By doing so, the non-dominated solutions survived at the best front are obtained for the problem. In this study, we adopt the selection and survival strategies of NSGA-II to handle multiple objectives.

3 Related Work

A considerable attention has been paid since the birth of the RPL protocol. These attempts are categorized mainly as studies that *i*) analyze the vulnerabilities of the RPL protocol under attack and *ii*) develop solutions to secure the protocol against different types of attacks. These studies are briefly discussed here.

In [2], the performance of RPL is investigated against version number attack with one to three attackers. It is shown that the number of attackers have a clear impact on packet delivery ratio. However, if the attacker is positioned closer to the root node, it increases end-to-end delay and power consumption. A comprehensive analysis is given in [9] to reveal how the performance of the RPL changes as a function of different objective functions when the network is subject to routing attacks with a varying number of attackers. It is shown that RPL is more vulnerable to these attacks when MRHOF is adopted.

A number of solutions have been proposed in the literature to secure RPL-based networks. For this purpose, the researchers have not only modified the protocol, but also developed IDSs that are integrated into the network. The first proposed IDS for RPL is SVELTE [20] which uses anomaly- and signature-based detection methods. Another anomaly-based study based on a game-theoretic model is proposed in [12]. It relies on two parts: *i*) a stochastic game for detection and *ii*) an evolutionary game for confirmation. The stochastic game model calculates the standard RPL rules as a zero-sum game, and the proposed scheme confirms the accuracy of the detection by applying evolutionary methods. In [18],

a trust-based model is proposed for the detection of rank and black hole attacks. Here, trust- and mobility-based metrics are evaluated. The proposed model has two parts: *i*)trust formation including trust metrics, trust index computation, trust rating, and *ii*)attack detection, including isolation of malicious nodes. Here, a fuzzy threshold-based system is used to calculate the trust formation and attack detection metrics. Rank values are checked with the sequence value of DIO messages for detection of rank attacks, the trust index value of the preferred parent is checked whether it is lower than threshold value for the detection of black hole attacks. The trustworthiness of the nodes is considered for the selection of parent nodes. In [17], a solution is proposed not only to detect the version number attack, but also to locate the attacker nodes. Here, monitoring nodes periodically send the collected feature data towards the root, which then detects attacks and locates the attacker by inspecting incoming data.

Recently, machine learning-based IDS has also been proposed for intrusion detection in RPL. A multi-class classification based detection model is proposed in [27] against rank and wormhole attacks. Here, the light gradient boosting machine algorithm with one-sided sampling method is used in attack detection. In another ML-based IDS, a deep neural network approach is used in [25] to detect decreased rank, hello flooding, and version number attacks. Moreover, they introduce a dataset called IRAD. Another deep learning-based model is developed for the detection of hello flooding attack in [4]. Here, Gated Recurrent Unit (GRU)-based deep learning method with a Recurrent Neural Network (RNN) approach is used for classifying the nodes. Another neural network-based system is proposed in [15] to identify the normal behavior of the nodes. A recent neural network-based IDS is proposed in [5]. Contrary to other studies, they consider not only the routing layer, but also the link layer to extract features that are fed to the network to learn a model. The involvement of features related to the link layer has been reported to decrease the false positive rate. The use of evolutionary-based algorithms for the generation of the IDS model is investigated in [3].

Transfer learning-based approaches have also been proposed for intrusion detection in IoT. A deep transfer learning (DTL) approach called MultiMaximum Mean Discrepancy AE (MMD-AE), based on AutoEncoder (AE) and allows the transfer knowledge is proposed in [23]. Here, although no IoT protocol is targeted, general attacks such as TCP/UDP flooding attacks are targeted, and a labeled dataset is transferred to an unlabeled dataset in accordance with the proposed model. The results show that the proposed transfer learning approach has better experimental results (Area Under Curve (AUC) score) than the traditional approach. In [26], the knowledge is transferred to detect new types of attack and to evolve intrusion detection algorithms for new types of devices with different constraints. Here, while the energy usage of the devices is minimized, the detection accuracy is maximized.

While there are a few studies based on evolutionary computation in the literature [3][26], the current study differs from those by exploring different trade-offs between the intrusion detection accuracy and the cost of the evolved algorithm

in terms of energy consumption and communication cost. Therefore, a different intrusion detection architecture is explored here, and communication cost is taken into account for the first time.

4 Evolving Intrusion Detection Algorithms

The main aim of this study is developing a suitable IDS for RPL-based IoT networks. Therefore, a central ID node is placed at the root node, which runs the evolved intrusion detection algorithm. In order to analyze the network traffic far from the central ID node, some monitoring nodes in which periodically collect the local data in their neighbourhood and sent it to the central ID node are participated in intrusion detection. Although these monitoring nodes enable the central ID node to detect attacks with a more satisfactory performance, it can have an adverse impact on the average lifetime of the network due to collecting their local information and on the communication cost due to their sending such information regularly to the central node. Therefore, the trade-offs between detection accuracy and cost need to be investigated. Hence, this study aims to evolve a lightweight ID model in terms of communication cost and energy consumption while effectively detecting malicious network traffic.

An ID program essentially contains a conditional statement represented by a GP tree. As stated earlier, there are terminal and non-terminal nodes in a GP tree. The terminal nodes are leaf nodes in the GP tree that represent the features collected by the nodes that are extracted from the RPL control and data packets in a flow. In this study, we used 35 different features that were proposed in [26]. Traffic flows are used for the construction of these features. To do that, the flows are first windowed within the specific time intervals by both the root and the monitoring nodes. The optimum interval time of the window is found to be 60 s in this study. The windowed feature data is then collected by the monitoring nodes and aggregated at the root node and provided to the GP tree as input data. In addition to the RPL-related features, randomly generated numbers are also assigned to leaf nodes to enable a more effective search by GP individuals. The non-terminal nodes represent arithmetic, comparison, and logical operators in the evolved model. It is worth stating that the root node in the GP tree is constrained to be either a comparison or a logical operator, so that it returns a Boolean value. An example of a GP tree that represents a candidate ID program is given in Fig. 1, and the program corresponding to this tree is given thereafter. The number of features employed in the ID model is of very high importance in determining to what extent the model leads to additional cost in terms of the memory and energy consumption of the monitoring nodes and communication load in the network. Moreover, the high number of features might result in fragmentation and the increase in the number of packets. As shown in the results of the preliminary experiment given in Figure 2, the overall power consumption of the nodes increases linearly with increasing data packets. Therefore, in addition to the detection accuracy, the number of distinct features employed in the intrusion detection algorithm is taken as the second objective

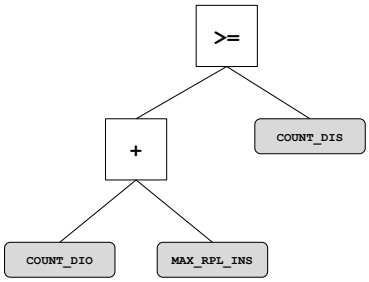

Fig. 1. An example GP tree.

```
1 if ((COUNT_DIO + MAX_RPL_INS) >= COUNT_DIS) then
2 |   alert(intrusion)
3 end
```

which needs to be optimized simultaneously in order to ensure an efficient and high-performance ID model. The GP algorithm is used to learn an ID model (program) that is optimal with respect to these objectives. Hence, the GP tree corresponds to the detection algorithm. While detection accuracy is measured by running the evolved program in the network with and without attackers, the number of features is measured just after the candidate GP tree is constructed by counting the distinct features in the leaf nodes. In order to optimize these objectives simultaneously, the selection and survival strategies of NSGA-II [8] are integrated into the GP algorithm. Therefore, the set of Pareto dominant

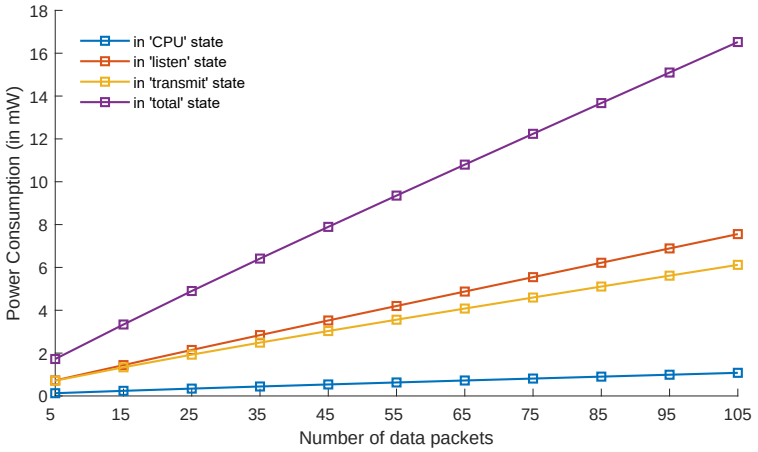

Fig. 2. The change in average power consumption under different states with varying number of packets.

individuals found after each evolution step is taken into account to determine both the survival of the parent individuals and the breeding of new offspring for the next steps. In GP, each individual represents a candidate program that is evaluated according to Pareto dominance. The GP individuals that dominate others (i.e., give higher accuracy with fewer distinct features) are called *Pareto dominant ID programs*, which we aim to learn in this study.

In this study, ECJ [10], a Java-based evolutionary computation toolkit, is used for the implementation of the GP and NSGA-II algorithms. The GP program is terminated at the 1000^{th} generation. In order to point an ideal value for the number of generations, we have performed pre-experiments and found that the change in the performance of GP is not significant after the 1000^{th} generation. The parameter settings of GP are listed in Table 1, and the other settings not listed in the table are the default parameters of the ECJ.

5 Experimental Results

An overview of the simulation environment is given in this section in detail. In addition, the experimental results obtained from the standalone and collaborative IDS architectures are also discussed comparatively thereafter.

5.1 Simulation Environment

In the experiment, a grid topology, shown in Figure 3, is used with 30 nodes, including the root node. Among them, three nodes (10% of the nodes) are set as attacker nodes, where their positions are randomly chosen. The nodes in the topology are positioned in the network such that each node is 20 m away from another node, and the transmission range between the nodes is limited to 25 m so that the nodes can communicate with their neighbor nodes. The arrows in Figure 3 represent the preferred parent of the child nodes in DODAG in a benign environment. The Cooja simulator [19] that emulates the LLN nodes is used in the experiments. Supported by Cooja for the emulated nodes, the Contiki operating system [7] (version 2.7) that also involves the implementation of RPL is used. Zolertia Z1 platform is chosen as the mote type for the nodes.

Table 1. GP parameters and their values.

Parameters	**Value**
Non-terminals	+, -, *, /, sin, cos, log, ln, sqrt, abs, exp, ceil, floor, max, min, pow, mod, $<$, $\leq$, $>$, $\geq$, ==, ! =, and, or
Terminals	features in [26] and rnd(0,1)
Generation Size	1000
Population Size	100
Crossover and mutation probability	0.9 and 0.1
Max. depth of GP tree	20

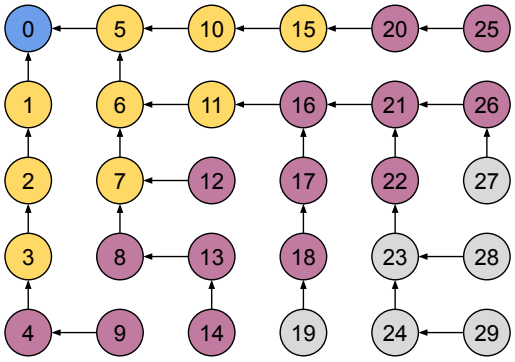

Fig. 3. Simulation network

By adopting this experimental environment and the settings, we have thoroughly evaluated the performance of the proposed evolutionary-based IDS against four targeted routing attacks; WP, HF, IV, and DR. To do that, eight different scenarios are generated, and the learning is repeated 10 times for each scenario that is individually simulated for three hours. The main motivation of the scenario-based evaluation is to thoroughly discuss how well the evolved ID program can detect when monitoring or attacker nodes are re-positioned after the learning step. Therefore, these scenarios differ from each other in terms of the position of the monitoring and attacker nodes in the training and testing network environment. These scenarios are outlined in Table 2. It is seen from the table that the monitoring nodes in the network are grouped according to their proximity to the root node that is represented with blue in Figure 3. The monitoring nodes are either placed closely to the root node (1-3 hops away from the root node, represented with yellow in Figure 3) or far (4-6 hops away, represented with purple in Figure 3) from the root node. It is worth pointing out here that each of the monitoring nodes is randomly chosen from a different hop level (that is, a single monitoring node is chosen per hop and the average of these nodes' data is used in the experiments). Hence, in each scenario 10% the nodes are responsible for monitoring. In addition, in order to see the effect of attackers' positions, in some scenarios, the attackers are placed randomly, but differently from its corresponding training setting.

5.2 Results

In order to evaluate the performance of the proposed IDS, we have considered the Pareto front set that represents the objectives of the Pareto dominant individuals found for each simulated scenario. Because we have run the GP algorithm 10 times for each scenario, we have aggregated 10 Pareto front sets and extracted the extreme points from these sets to reveal how well the GP algorithm could achieve the best performance in terms of accuracy (ACCR) and number of features

Table 2. Position settings of the monitoring ID nodes and the attacker nodes used in the experiments.

Scenario	Location of monitoring ID nodes		Location of attackers
	In training	In testing	in training and testing
S1	close	close	same
S2	close	close	different
S3	far	far	same
S4	far	far	different
S5	close	far	same
S6	close	far	different
S7	far	close	same
S8	far	close	different

(NoF). These extreme points are shown in Table 3 separately for each of the two objectives. For example, '0.945 (8)' stated for the ACCR implies that the GP could reach 0.945 accuracy with a model that has eight distinct features. Similarly, '1 (0.850)' stated for NoF implies that the GP model has only one distinct feature and could reach 0.850 accuracy.

The results enable us to evaluate the performance of the proposed IDS from different points of view. The first is that DR is a harder-to-detect routing attack as compared to other attacks when considering the average of the accuracies from eight scenarios (it is 0.88 overall). The difference in the average accuracy performances obtained from the other attacks is not significant, and GP succeeds to evolve a satisfactory ID program for these attacks (it is 0.94 overall). As for

Table 3. The best performances with respect to accuracy (ACCR) and number of features (NoF) obtained in the Pareto front set.

Scenario	Objective	WP	HF	IV	DR
S1	**ACCR**	0.945 (8)	0.950 (13)	0.955 (12)	0.880 (11)
	NoF	1 (0.850)	1 (0.865)	1 (0.870)	1 (0.810)
S2	**ACCR**	0.940 (7)	0.945 (12)	0.950 (12)	0.870 (10)
	NoF	1 (0.840)	1 (0.855)	1 (0.870)	1 (0.800)
S3	**ACCR**	0.950 (11)	0.945 (10)	0.935 (8)	0.870 (10)
	NoF	1 (0.845)	1 (0.860)	1 (0.860)	1 (0.820)
S4	**ACCR**	0.940 (8)	0.940 (9)	0.925 (6)	0.865 (8)
	NoF	1 (0.840)	1 (0.855)	1 (0.850)	1 (0.820)
S5	**ACCR**	0.935 (8)	0.930 (9)	0.940 (10)	0.880 (11)
	NoF	1 (0.835)	1 (0.845)	1 (0.830)	1 (0.835)
S6	**ACCR**	0.935 (12)	0.925 (11)	0.945 (12)	0.875 (10)
	NoF	1 (0.830)	1 (0.840)	1 (0.825)	1 (0.810)
S7	**ACCR**	0.930 (9)	0.925 (12)	0.935 (10)	0.900 (12)
	NoF	1 (0.835)	1 (0.830)	1 (0.835)	1 (0.800)
S8	**ACCR**	0.925 (8)	0.920 (7)	0.910 (9)	0.890 (12)
	NoF	1 (0.825)	1 (0.830)	1 (0.810)	1 (0.805)

the evaluation according to the average of NoF, it is seen that the ID program requires the more features for the DR attack (it is 10.5 in overall). This clearly suggests that GP is unable to evolve an ID program that gives higher detection accuracy even after a rigorous evaluation of massive feature data. Furthermore, the difference in the accuracy performances obtained from the extreme points with respect to ACCR and NoF ranges from 4.5% to 12.0%. They are the WP and DR attacks that give the highest and least difference in performance, respectively. This shows that limiting the number of features in ID program has an adverse impact on the detection capability to some degree as expected, and this varies according to the targeted attack. Note that these give the largest difference in the performances of GP, and even a few increases in the number of features in ID program yield much better accuracy.

When it comes to the change in IDS performance when only attackers are repositioned with the same configurations of the monitoring nodes (that is, the comparison of S1 with S2, S3 with S4, and so forth), a slight performance degradation is often observed, and the change here is no more than 2.5%. This is not surprising because the locations of attackers are positioned randomly from the entire network, and there are cases studied in these scenarios where the ID program evolved and tested when the attacker nodes are, respectively, in the vicinity and away from the monitoring or root node. As for the change in IDS performance as a function of different configurations of monitoring nodes by keeping the attacker's positions the same and different (that is, comparison of S1 with S3 and S2 with S4), it is seen that the performance of the evolved ID program slightly improves (up to 2.5%) when monitoring nodes are positioned within the first three hops.

In order to reveal how monitoring nodes are helpful in improving the attack detection capability of the ID program, we replicated the simulations by adopting standalone architecture where the root node is in charge of intrusion detection alone. The simulation here is run with two attacker configurations that are denoted 'same' and 'different' which again represent the cases where attackers are positioned at the same and different locations when ID program is evaluated in the testing environment, respectively. The results are shown in Table 4. Note that S1 through S4 in Table 3 should only be considered to ensure a fair comparison between the performances of collaborative and standalone architectures. The results suggest that the performance of the ID program increases with the collaborative architecture, and the difference reaches 4%. However, no significant difference is observed in terms of NoF.

6 Strengths, Limitations, and Future Directions

The primary target of our attempt is to reduce the operational cost of IDS considering the constrained resource of LLN nodes. Therefore, without sacrificing detection accuracy, we aim to minimize the number of distinct features that cause exhaustive resource consumption of nodes, as well as adversely affect the communication cost of monitoring nodes. The obtained results show the appli-

Table 4. The extreme points with respect to ACCR and NoF in the Pareto front sets obtained by the standalone architecture.

Position	Objective	WP	HF	IV	DR
Same	**ACCR**	0.910 (9)	0.920 (12)	0.935 (11)	0.865 (10)
	NoF	1 (0.835)	1 (0.840)	1 (0.830)	1 (0.790)
Different	**ACCR**	0.905 (8)	0.915 (11)	0.925 (12)	0.860 (9)
	NoF	1 (0.825)	1 (0.830)	1 (0.825)	1 (0.780)

cability of the proposed multi-objective approach to effectively and efficiently detect intrusions in LLNs. Because the depth of GP tree is highly correlated to the number of terminal nodes (i.e., feature nodes) in most cases, we implicitly control the size of the tree, and hence the length of the program. Therefore, the code bloating problem in GP is handled in our approach. However, even few, the LLN nodes also suffer from the frequency of the operations in the ID program that are to be executed. Although this execution cost is not included as an individual objective, it can be easily minimized by penalizing solutions that require intensive computation.

The existence of monitoring nodes may reduce the communication performance of LLNs, but they are undeniably important in detecting intruders effectively. We here perform the simulations by grouping the monitoring nodes as close (1-3 hops away from the root node) and far nodes (4-6 hops away from the node); however, it is of high importance to thoroughly investigate the number and the position of these nodes, which could also be studied in the future. In addition, the proposed approach is tested on a centralized architecture in which the root node is responsible for raising the alarm. That's why, the single point of failure, which occurs when the root node is down by the intruders, is the drawback of our detection system. To overcome this, one can rely on a fully decentralized architecture where multiple nodes are in charge of the detection task simultaneously. For this architecture, developing different local detection programs for each individual node is a must. To do that, it is worth studying the federated learning approach to have local programs that are informed globally and constructed collaboratively.

As the main objective of this study is to show applicability of multi-objective GP in developing IDS for resource-constrained LLNs, we only targeted four attack types, and it can be extended by involving other types of attacks. In addition, the positions of the LLN nodes are stationary in the experiment; however, if not all, in most real-world IoT applications, a portion of these nodes is mobile. Therefore, it is worth testing our proposed approach in a mobile environment, which could be another future direction of this study.

7 Conclusion

In this study, we explore the use of the Pareto-based multi-objective approach to efficiently and effectively detect four different attack types specific to RPL.

To the best of the author's knowledge, this is the first study that aims to simultaneously optimize the detection accuracy of ID programs and their costs including communication cost of ID nodes. To do that, a massive number of simulations are generated, and the different ID programs are evolved by using these simulations. The evaluations are made on the basis of Pareto sets obtained from the evolved programs. Among all experiments, the average accuracy was 92.2%, while the variance in these accuracies was 0.08%, demonstrating that the proposed approach provides satisfactory results in detecting targeted attacks. It is also worth stating here that our IDS model converges to an accuracy level above 90% after reaching around 30% of the generations. In the future, we plan to explore the applicability of our approach when a portion of LLN nodes are mobile.

References

1. Alexander, R., Brandt, A., Vasseur, J., Hui, J., Pister, K., Thubert, P., Levis, P., Struik, R., Kelsey, R., Winter, T.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Mar 2012). https://doi.org/10.17487/RFC6550, https://www.rfc-editor.org/info/rfc6550
2. Arış, A., Oktuğ, S.F.: Analysis of the rpl version number attack with multiple attackers. In: 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA). pp. 1–8. IEEE (2020)
3. Aydogan, E., Yilmaz, S., Sen, S., Butun, I., Forsström, S., Gidlund, M.: A central intrusion detection system for rpl-based industrial internet of things. In: 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS). pp. 1–5. IEEE (2019)
4. Cakir, S., Toklu, S., Yalcin, N.: Rpl attack detection and prevention in the internet of things networks using a gru based deep learning. IEEE Access **8**, 183678–183689 (2020)
5. Canbalaban, E., Sen, S.: A cross-layer intrusion detection system for rpl-based internet of things. In: Ad-Hoc, Mobile, and Wireless Networks: 19th International Conference on Ad-Hoc Networks and Wireless, ADHOC-NOW 2020, Bari, Italy, October 19–21, 2020, Proceedings 19. pp. 214–227. Springer (2020)
6. Cisco: Visual networking index: Forecast and trends, 2017–2022 white paper, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html, last accessed 04 Apr 2020
7. Contiki-Ng: contiki-ng/contiki-ng. https://github.com/contiki-ng/contiki-ng/wiki (2004), last accessed 13 Jul 2021
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
9. Dogan, C., Yilmaz, S., Sen, S.: Analysis of rpl objective functions with security perspective. In: SENSORNETS. pp. 71–80 (2022)
10. ECJ: A java-based evolutionary computation research system. https://cs.gmu.edu/~eclab/projects/ecj (2017), last accessed 04 Apr 2022
11. Eiben, A.E., Smith, J.E., et al.: Introduction to evolutionary computing, vol. 53. Springer (2003)

12. Gothawal, D.B., Nagaraj, S.: Anomaly-based intrusion detection system in rpl by applying stochastic and evolutionary game models over iot environment. Wireless Personal Communications **110**(3), 1323–1344 (2020)
13. Herberg, U., Clausen, T.: A comparative performance study of the routing protocols load and rpl with bi-directional traffic in low-power and lossy networks (lln). In: Proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks. p. 73–80. PE-WASUN '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2069063.2069076, https://doi.org/10.1145/2069063.2069076
14. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and computing **4**(2), 87–112 (1994)
15. Li, F., Shinde, A., Shi, Y., Ye, J., Li, X.Y., Song, W.: System statistics learning-based iot security: Feasibility and suitability. IEEE Internet of Things Journal **6**(4), 6396–6403 (2019)
16. Mayzaud, A., Badonnel, R., Chrisment, I.: A taxonomy of attacks in rpl-based internet of things. Int. J. Netw. Secur. **18**, 459–473 (2016)
17. Mayzaud, A., Badonnel, R., Chrisment, I.: A distributed monitoring strategy for detecting version number attacks in rpl-based networks. IEEE transactions on network and service management **14**(2), 472–486 (2017)
18. Muzammal, S.M., Murugesan, R.K., Jhanjhi, N.Z., Humayun, M., Ibrahim, A.O., Abdelmaboud, A.: A trust-based model for secure routing against rpl attacks in internet of things. Sensors **22**(18), 7052 (2022)
19. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with cooja. In: Proceedings. 2006 31st IEEE conference on local computer networks. pp. 641–648. IEEE (2006)
20. Raza, S., Wallgren, L., Voigt, T.: Svelte: Real-time intrusion detection in the internet of things. Ad hoc networks **11**(8), 2661–2674 (2013)
21. Sen, S.: A survey of intrusion detection systems using evolutionary computation. In: Bio-inspired computation in telecommunications, pp. 73–94. Elsevier (2015)
22. Statista: Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, last accessed 10 Apr 2022
23. Vu, L., Nguyen, Q.U., Nguyen, D.N., Hoang, D.T., Dutkiewicz, E.: Deep transfer learning for iot attack detection. IEEE Access **8**, 107335–107344 (2020)
24. Wu, S.X., Banzhaf, W.: The use of computational intelligence in intrusion detection systems: A review. Applied soft computing **10**(1), 1–35 (2010)
25. Yavuz, F.Y., Devrim, Ü., Ensar, G.: Deep learning for detection of routing attacks in the internet of things. International Journal of Computational Intelligence Systems **12**(1), 39 (2018)
26. Yılmaz, S., Aydogan, E., Sen, S.: A transfer learning approach for securing resource-constrained iot devices. IEEE Transactions on Information Forensics and Security **16**, 4405–4418 (2021)
27. Zahra, F., Jhanjhi, N., Brohi, S.N., Khan, N.A., Masud, M., AlZain, M.A.: Rank and wormhole attack detection model for rpl-based internet of things using machine learning. Sensors **22**(18), 6765 (2022)