GRASP-based Feature Selection for Intrusion Detection in CPS Perception Layer

Conference Paper · October 2020

DOI: 10.1109/CIoT50422.2020.9244207

5 authors, including:

Silvio Ereno Quincozes
Universidade Federal de Uberlândia (UFU)
37 PUBLICATIONS **121** CITATIONS
SEE PROFILE

Diego Passos
Universidade Federal Fluminense
66 PUBLICATIONS **943** CITATIONS
SEE PROFILE

Célio Vinicius N. Albuquerque
Universidade Federal Fluminense
179 PUBLICATIONS **2,065** CITATIONS
SEE PROFILE

Luiz Satoru Ochi
Universidade Federal Fluminense
217 PUBLICATIONS **3,233** CITATIONS
SEE PROFILE

GRASP-based Feature Selection for Intrusion Detection in CPS Perception Layer

Silvio E. Quincozes*, Diego Passos,
Célio Albuquerque, and Luiz Satoru Ochi
Computing Institute, Fluminense Federal University
Niterói, RJ, Brazil
**sequincozes@id.uff.br*

Daniel Mossé
Computer Science Department
University of Pittsburgh
Pittsburgh, PA, USA

***Abstract*—Cyber-Physical Systems (CPS) will form the basis for the world's critical infrastructure and, thus, have the potential to significantly impact human lives in the near future. In recent years, there has been an increasing demand for connectivity in CPS, which has brought to attention the issue of cyber security. Aside from traditional information systems threats, CPS faces new challenges due to the heterogeneity of devices and protocols. In this paper, we investigate how Feature Selection may improve intrusion detection accuracy. In particular, we propose an adapted Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic to improve the classification performance in CPS perception layer. Our numerical results reveal that GRASP metaheuristic overcomes traditional filter-based feature selection methods for detecting four attack classes in CPSs.**

I. Introduction

Cyber-Physical Systems (CPS) integrate physical processes, networking, and computation. The Internet of Things (IoT) also brings about a similar structure, with devices connecting to the physical world, networking for transmitting the data, and computations. These types of computational systems can remotely monitor and control physical processes through several tiny devices. These devices typically have sensing, actuation, computing, and communication capabilities. Thus, they capture physical-world data (*e.g.*, position, temperature, pressure) and transfer it to applications running in control centers through networking technologies, such as Wi-Fi, ZigBee, or LTE. Thereby, CPS can be defined in a three-layer architecture: perception, transmission, and application layers. These layers are responsible for, respectively, collecting physical data and send actuating commands, communicating with control centers, and monitoring and decision-making regarding the physical processes — often without human intervention [1].

The scope of CPS has reached several Industrial IoT systems, including chemical engineering, healthcare, manufacturing, transportation, automotive systems, power, petroleum, and the water industry [2], [3], [4], [5]. As an integral part of so many critical infrastructure systems, it is very important to consider security aspects of CPS, specially it resilience to attackers. In the last decade, harmful attacks were reported. In 2010, 900 MW were lost during 7 seconds of attack targeting a power plant [6]. Also, the Stuxnet worm attacks caused a serious deterioration in Iran's nuclear program. In the worst cases, event human life may be at risk. Malicious remote control of cars and airplanes has already been successfully demonstrated [6]. Currently, similar vulnerabilities are still being reported: TrapX Security reported a malware found on several automatically guided vehicles [7].

The increased connectivity between the physical and cyber domains is at the core of the unique security challenges in CPS [1]. Due to its importance, security must be a concern at design time. That includes analyzing possible attacks and building a robust security architecture that encompasses confidentiality, integrity, availability, and, especially, authenticity in all layers. However, as real-world experiences demonstrate, it is impossible to predict — and prevent — all security issues during design. In particular, devices at the CPS perception layer are often the most vulnerable targets due to their resource-constrained characteristics [1].

As a consequence, Intrusion Detection Systems (IDS) play a crucial role in CPS security. Their effectiveness depends upon both their detection time and accuracy. Detection time is being addressed by a different research lines, including the use of real-time Big Data processing technologies (*e.g.,* stream event processing [8]). Processing perception layer features at application layer may mitigate additional overhead. Accuracy, on the other hand, can be improved by selecting appropriate features to identify security threats. In particular, Feature Selection (FS) methods aim at reducing the number of features analyzed by an IDS, discarding irrelevant or redundant ones. As an added benefit, FS reduces the feature space, making the data processing for online intrusion detection faster.

Whereas intrusion detection is an online task — and, therefore, time-sensitive —, FS for IDS is typically offline in nature. Proper features to detect attacks for a specific scenario can be selected in design or deployment time. If necessary, the set of features may be updated over time during relatively long time scales. Moreover, the update process can be executed in parallel with intrusion detection itself. Thus, while certain FS methods are less time-consuming than others, for this particular application their accuracy should be the main concern.

This work investigates how FS techniques may improve intrusion detection accuracy in the perception layer. In particular, we propose an adapted GRASP Feature Selection metaheuristic that outperforms traditional filter approaches in

978-1-7281-9541-4/20/$31.00 ©2020 IEEE

an empirical analysis based on data collected from a Wireless Sensor Network (WSN) [9], representing the perception layer.

II. Cyber-Physical Systems

In this section, we provide a brief overview of the CPS architecture. Then, we discuss some existing applications categorized as CPS in different domains.

A. Overview

In a CPS, physical objects are equipped with tiny devices that have sensing, actuation, computation and, networking capabilities. They collect data from the perception layer and transmit it to the application layer, where control centers handle physical processes. The application layer enables personalized applications such as monitoring, data analysis, and end-user applications [1], [5]. The traditional three-layer CPS architecture is presented in Figure 1.

The physical and cyber worlds may connect through both infrastructure and infrastructure-less network technologies, such as Wi-Fi, 3G/4G/5G, Bluetooth, and Zigbee. Therefore, the transmission layer also involves heterogeneous technologies and communication protocols — although some application protocols are frequently used, such as Message Queue Telemetry Protocol (MQTT) and its version adapted to sensor networks (MQTT-SN). These protocols adopt the publish-subscribe paradigm to transmit messages from multiple sensing devices to a centralized broker that delivers each message to the applications that subscribed for it. The reverse can also happen: applications can act as publishers sending messages to multiple actuators at the perception layer.

The perception layer is the main novelty of CPS compared to traditional information systems. It contains possibly heterogeneous devices that collect physical world information (*e.g.*, sensors and RFID tags) and respond to certain events (*e.g.*, actuators). These devices are generally resource-constrained, with limited power sources.

B. Applications

Many applications have become smarter and optimized due to CPS technology adoption. CPS applications span across several domains. Here, we discuss some of them.

1) Healthcare: Zhang *et al.* [5] propose Health-CPS, a system that deals with large amounts of complex healthcare data. To provide efficient patient-centric healthcare services, Health-CPS was built upon cloud and big data analytics technologies. It uses a unified data standard in the perception layer and extends the transmission layer to a data management layer for distributed storage and parallel computing. Finally, in the application layer, it employs a data-oriented approach, enabling various smart healthcare applications and services. Evaluation results show that adopting cloud and big data technologies in healthcare CPS can be promising in terms of performance. Regarding security, Health-CPS employs data encryption to protect user privacy. However, other security requirements are also important in this scenario, such as availability, authenticity and integrity. Additionally, detecting and preventing malicious activity can save lives (*e.g.*, preventing attackers from controlling in-body devices).

2) Smart Grids: Smart Grids add advanced monitoring, control, and communication to the electric power grid. By combining complex physical systems, network and cyber systems, they enable efficient use of energy for consumers, generators and distributors. Cyber security is an important issue in Smart Grids, as the proper working of computing devices, such as Intelligent Electronic Devices (IEDs), is essential for their correct behavior. Hence, new protection and intrusion detection techniques are required to ensure availability, confidentiality, and data integrity. In particular, the real-time communication between IEDs from different electrical substations is critical to protect assets and human lives (*e.g.*, switching-off a circuit breaker during a short circuit). One of the challenges in this scenario is coping with the stringent communication time requirements, while implementing security mechanisms. Therefore, this communication should be monitored constantly in order to detect anomalous and malicious behaviors [4].

3) Industrial Automation: Control and monitoring a have high impact on industrial automation. Collaboration between distributed devices and control software makes production systems more responsive, flexible and scalable. With real-time response capability, Industrial CPS play a key role in the concept of Industry 4.0 [3].

4) Vehicular Systems: Vehicular Ad Hoc Networks (VANET) is a promising approach for future intelligent transportation systems. Constraints such as driver behavior and high topology mobility result in particularities that are not experienced by other networks. Therefore, Vehicular CPS (VCPS) can be employed to improve energy efficiency and road capacity. The platoon-based approach, in which a vehicle follows another vehicle within the same common interest group and short distances are maintained between them, is a VCPS that cooperatively improves driving patterns. The need for cooperation in VANETs increases vulnerabilities and, therefore, requires mitigation techniques, such as authentication and misbehavior detection systems [2]. Insecure cooperative VANETs may expose vehicles to potential attacks regarding infrastructure (*e.g.*, altering traffic information such as speed limits) or other vehicles (*e.g.*, GPS poisoning or jamming), leading to crash risks and traffic chaos.

III. Threats and Detection Methods

The interaction between cyber and physical domains implies that the system is vulnerable to different attacks at each layer. Applications must be protected and monitored to avoid attackers gathering sensitive data, as well as controlling the physical-world behavior. Similarly, the data transferred in the transmission layer must be protected to avoid message manipulation and unauthorized access. However, the most critical point is the perception layer. The devices in this layer are often attached to the physical objects: once they are compromised, attackers may interact directly with the physical world. Thus, eventual security measures in the upper layers may not be

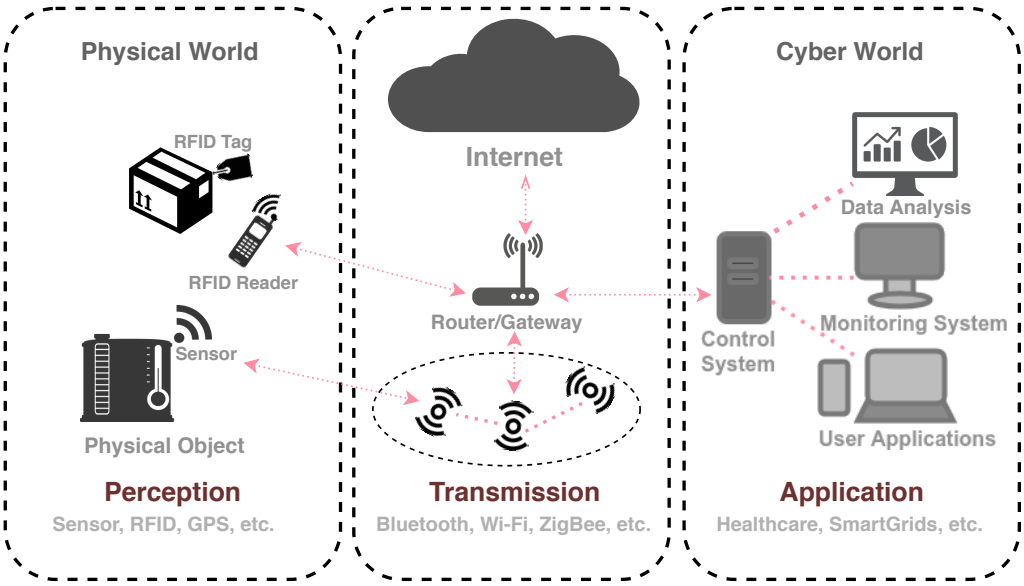


Fig. 1: Cyber-Physical Systems Architecture, including devices, Cloud, and applications.

effective if the attackers have direct access to the perception layer. In this section, we describe the main threats targeting CPS, especially the perception layer, as well as the existing detection methods.

A. Cyber Attacks and Vulnerabilities

While confidentiality is considered the most relevant security property for traditional information systems, in CPS, availability is often considered the most important requirement – due to the need for constant monitoring and acting –, followed by integrity, confidentiality, and authenticity [1].

These goals may be achieved by following different cyber attack approaches, which may target the different CPS layers [2][1]. Nevertheless, the perception layer may be considered the core of the security concerns due to multiple factors, including the hardware limitations and the heterogeneity in terms of devices and communication protocols.

Perception layer attacks target devices attached to physical objects, such as sensors and actuators. These devices usually lack security mechanisms due to their computational constraints, thus being vulnerable to unauthorized access or information manipulation. Other attacks at the perception layer include information disclosure, tracking, tampering, and availability compromising. In particular, even if lightweight security solutions are used, availability may still be compromised by Denial-of-Service (DoS) attacks [9].

Once the perception layer is unable to reliably collect information from the physical world (*e.g.*, if a sensor's energy is exhausted during a flooding attack), the entire CPS may be compromised. This means applications will not be aware of the systems' states. Consequently, they will not be able to take the proper response actions to match changes in the state of physical processes (*e.g.*, sounding a fire alarm, administering insulin to patients with low glucose levels, reducing the load on an overheated machine).

Common DoS attacks to the perception layer include grayhole, blackhole, scheduling and flooding. In [9], these attacks are explored through a simulated WSN with the Low Energy Aware Cluster Hierarchy (LEACH) routing protocol. This protocol assumes a fixed Base Station (BS) and clusters sensor nodes (see Figure 2). Each sensor cluster has a special node called Cluster Head (CH), responsible for aggregating the data received from the other sensor nodes within its cluster and for transmitting them to the BS.

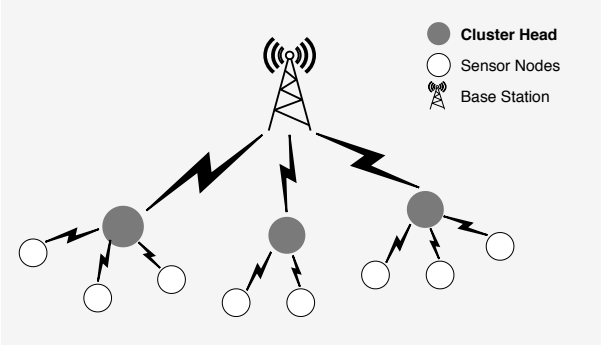


Fig. 2: A typical WSN topology [9].

The LEACH hierarchical structure aims at balancing the energetic burden among all nodes in the network by using different sensors as CH in each round. However, this behavior has an intrinsic vulnerability: a malicious node may become a CH and control all traffic inside the compromised cluster.

To run a blackhole attack, the malicious sensor exploits the LEACH protocol by advertising itself as a CH. When the attacker assumes the role of CH, it drops all messages received from each node in its cluster instead of forwarding them to the BS [9]. Similarly, in a grayhole attack, the malicious CH drops only part of the messages, chosen either randomly or

following some pattern created to disrupt the network while camouflaging the attack [9].

Flooding attacks targeting the LEACH protocol are performed by sending a large number of *Advertising Cluster Head* (ADV CH) messages with a high transmission rate. As practical effects, it causes the sensors to spend more time and energy to decide which CH to join. In addition, the attacker tries to confuse the victim to choose the malicious node as CH to perform the aforementioned attacks. Flooding is particularly harmful to those nodes which are located at multiple hops from the malicious CH, which will consume more energy from every intermediary sensor in the path [9].

B. Intrusion Detection

To identify malicious activities, IDSs collect and analyze data from different sources (*i.e.*, systems and networks) to identify possible threats, such as the above-mentioned attacks. IDSs can be classified according to their data source as network-IDS (NIDS) or host-IDS (HIDS). Furthermore, they can follow different approaches, such as statistical- or artificial intelligence (AI)-based. Finally, an IDS may employ anomaly-based or signature-based techniques. Signature-based methods can achieve better accuracy for known attacks (*i.e.*, those for which signatures are known), but anomaly-based methods aim at detecting any anomalous activity, even if no exact sample is available for training. In addition, the latter tends to be faster. On the other hand, anomaly-based techniques may result in a high number of false positives, because not every anomaly corresponds to an attack or malicious activity (*i.e.*, it could be a legitimate anomalous activity).

Several algorithms can be employed for each of these approaches. For example, clustering algorithms such as K-means and other distance-based learning algorithms are usually employed for anomaly detection. Bayesian networks and Decision tree algorithms, such as Naive Bayes and Random Tree, can perform data classification based on labeled training data (signature-based). Regardless of the technique, an IDS must be fed with relevant features. Of course, *relevance* is directly linked to what the system aims to detect. For example, while some types of DoS attacks can be detected based on traffic features (*e.g.*, number of connections or count of transmitted bytes), brute force attacks can be detected by observing features from application logs (*e.g.*, number of failed login attempts). Hence, a relevant feature for DoS may not be relevant for brute force attacks and vice-versa [10].

For anomaly-based techniques, in which attacks are not necessarily known, feature choice can be based on prior risk analysis and the identification of potential threats. However, choosing the right features manually requires specialized knowledge that may not be always available. Additionally, frequent behavioral changes and attack evolution make this a very difficult task, especially when considering zero-day attacks (*i.e.*, those not yet known to developers and application administrators). In this context, FS techniques can be adopted.

C. Feature Selection in the Perception Layer

The first step to deploy an IDS is to define which data may be analyzed to identify malicious patterns. However, this decision is not trivial when there is a large number of features available. Non-relevant or redundant features may increase the processing overhead and produce an inaccurate model and lead the classifiers to lower accuracy.

This problem can be handled by using FS methods, which apply some technique to select the best features for the problem. There are three main categories of FS methods: *filter, wrapping,* and *embedded*. Filter methods consider statistical characteristics, such as entropy or correlation, in order to rank each feature according to its relationship with each sample class label (*i.e.*, normal behavior or specific attack label). Wrapping methods select feature subsets based on machine learning outputs and have been shown to achieve more reliable results than purely statistical methods. However, wrapping evaluations are usually computationally expensive. The embedded methods are employed by classifier algorithms that do some feature discarding internally.

Filter methods are frequently chosen due to their simplicity compared to other methods. Some popular filter methods are Information Gain (IG), OneR and Information Gain Ratio (GR) [11]. Because IDSs using these methods perform worst than if wrapping methods were employed, a better approach is to combine both methods through a hybrid filter wrapping algorithm for FS [12], [11]. In this context, using metaheuristics may be promising to reduce the number of wrapping evaluations and find a suitable feature subset solution. More details are provided below.

IV. Related Work

The Greedy Randomized Adaptive Search Procedure (GRASP) [13] is an iterative multi-start metaheuristic proposed originally to achieve an approximate solution of graph planarization problems. It has since been generalized to solve other combinatorial optimization problems in different domains [14]. The general GRASP strategy can be carried in two iterative phases: *construction* and *local search*. These steps are repeated until a stop criteria is reached (*e.g.,* the maximum number of total iterations). For each iteration, a different random seed solution is generated. The construction phase relies on this seed to generate a Restricted Candidate List (RCL) and choose from it a randomized greedy solution. The RCL consists of a predefined number of candidates to compose a feasible solution. Typically, this initial greedy solution can be improved by neighbor solutions. Thus, in the local search phase, the greedy solution is the starting point for a neighborhood exploration.

Yusta [14] demonstrates that GRASP can outperform Sequential Forward Floating Selection (SFFS), Tabu Search, Genetic and Memetics algorithms in generating the best feature subset to classify samples from different databases (Spambase, Waveform, Ionosphera, Vehicle, Wincosin, and German). Esseghir and Amir [11] also apply GRASP for FS considering some of those datasets (Ionosphere, and SpamBase) and others

(Sonar, Audiology, and Arrhythmia). Whereas the former work used the K-Nearest Neighbours (KNN) classifier to wrapping evaluations, the latter explored Artificial Neural Networks.

Bermejo *et al.* [15] and Mohsen *et al.* [16] use GRASP to deal with large and high-dimensional datasets. Basically, Bermejo *et al.* focuses on reducing the number of wrapper evaluations by using the Incremental Wrapper Subset Selection (IWSS) algorithm in the construction phase. The main drawback of this approach is the possibility of excluding important features due to a premature stop caused by the current solution exceeding the solution cardinality threshold. Mohsen *et al.* combined GRASP with an extended version of Simulated Annealing as a local search procedure with the goal of introducing new parameters that allow the implicit weighting between accuracy and execution time.

Diez *et al.* [17] propose GRASP Forest (G-Forest) for constructing ensembles of decision trees. This method uses concepts of GRASP for both feature selection and choosing splitting points at each tree node. For selecting features for each level of the tree, G-Forest assembles an RCL composed of all features with an IG above a certain threshold. It then proceeds to choose features randomly from this RCL. Once a feature is chosen, G-Forest computes the IG of each possible split point for that feature and forms a new RCL composed of those with an IG above a certain threshold. The split point is then chosen randomly from this list. This procedure is repeated until the desired number of trees is created. In a later work [18], the same authors extend G-Forest to Annealed Randomness Forest (GAR-Forest). The key idea is to introduce a parameter that controls the randomness during the solution construction phase, which ranges from an entirely random procedure (*i.e.*, value 0) to a totally greedy one (*i.e.*, value 1). Diez *et al.* consider 62 datasets (including Waveform, Ionosphere, and Vehicle). However, none of those are related to the cyber security domain. Kanakarajan and Muniasamy [19] apply GAR-Forest to detect DoS, Probe, R2L and U2R attacks in traditional networks.

Clearly, the use of the GRASP metaheuristics is already considered in the literature for FS. However, since its use in the domain of intrusion detection is little explored, it is not clear the real contribution of the use of this approach to the current machine learning-based IDSs. Furthermore, to the best of our knowledge, no previous work explores a GRASP-based metaheuristic in the context of CPS. In this work, we perform an assessment of a novel combination of RCL, classifiers used for wrapping evaluation, and local search in to the CPS domain. More details are given in Section V.

V. Proposed GRASP Feature Selection Algorithm

This section proposes a specialization of the GRASP algorithm to the feature selection problem. In particular, we adapt the generic GRASP approach to select features extracted from the CPS perception layer. These adaptations include the solution composition (features subset in our case) fitness function choice, neighborhood structure, and RCL generation. The implemented GRASP-based FS is shown in Algorithm 1.

Algorithm 1: GRASP Algorithm Adapted to FS

```
   input : fSet // full features set
           maxIt // number of iterations
           rclSize // RCL number of features
           numF // feature subset size
   output: bestFS // best feature subset
1  begin
2      bestFS ← ∅
3      bestFS.acc ← 0
4      while numIterations++ < maxIt do
5          greedyFS ← construct(numF, rclSize, fSet)
6          greedyFS.acc ← evaluate(greedyFS)
7          bestLocalFS ← localSearch(greedyFS, rcl)
8          if bestLocalFS.acc > bestFS.acc then
9              bestFS ← bestLocalFS
10         end
11     end
12 end
13 return bestFS
```

The input parameters are: the full feature list (`featureList`), the maximum number of GRASP iterations (`maxIt`), the number of features to compose the RCL (`rclSize`), and the desired feature subset size (`numF`). For each GRASP iteration, a greedy and randomized feature subset (`greedyFS`) is generated from the RCL features. This feature subset is further used as a seed to the local search procedure. At the end of the iteration, the best solution found (`bestFS`), that is, the best feature subset found among all construction and local searches iterations. The constructive (Line 5) and local search (Line 7) phases are detailed in Algorithms 2 and 3, respectively.

A. Construction Phase

In the construction phase, we employ the IG algorithm for feature ranking and RCL composition (Lines 2-8 of Algorithm 2). Note that once the feature list is ranked, this step is skipped in the next iterations. The `greedyFS` is constructed incrementally by choosing random features from the RCL until the `numFeatures` is achieved.

Algorithm 2: Constructive Phase

```
   input : fSet // full features set
           numFeatures // feature subset size
           rclSize // RCL number of features
   output: greedyFS // Greedy Random Feature Set
1  begin
2      if rcl = ∅ then
3          foreach feature ∈ fSet do
4              feature.IG ← calcIG(feature)
5          end
6          rcl ← selectTopRanked(fSet, rclSize)
7      end
8      greedyFS ← ∅
9      while (|greedyFS| < numFeatures) do
10         greedyFS ← ∪ selectRandom(feature ∈ rcl)
11     end
12 end
13 return greedyFS
```

Algorithm 3: LOCAL SEARCH

```
input : rcl // restricted candidate list
        greedyFS // Seed feature subset
        maxIt // Number of iterations
output: bestLocalFS // Best feature set found
1  begin
2      bestLocalFS ← greedyFS
3      numLSIterations++ ← 0
4      while numLSIterations++ <maxIt do
5          nbFS ← bitFlipt (greedyFS, rcl)
6          nbFS.acc ← evaluate(nbFS)
7          if nbFS.acc > bestLocal.acc then
8              bestLocalFS ← nbFS.acc
9          end
10     end
11 end
   return: bestLocalFS
```

B. Local Search Phase

In the local search phase (Algorithm 3), we employ the Bit-Flip [20] neighborhood structure to perform iterative simple movements from the seed feature subset. Each generated neighbor feature subset (`nbFS`) is evaluated through the selected classifier algorithm. The feature subsets quality is assessed by the classifiers' accuracy (`acc`). The local search procedure is also repeated at most `maxIt` iterations.

C. Discussion

As discussed before, filtering FS is faster than wrapping, however it is less accurate. On the other hand, wrapping provides high accuracy at a high computational cost. Thus, the main idea of using GRASP to perform feature selection is to reduce the number of wrapping evaluations necessary to increase an objective function (*e.g.*, intrusion detection accuracy) within a feasible computational cost. To achieve this goal, we use the a filter algorithm to compose a RCL.

The GRASP-based FS avoids irrelevant features in the constructive phase, providing a suitable seed feature subset to the local search phase. Thus, the expected number of iterations (and computation required) until convergence is reduced.

VI. EXPERIMENTAL ANALYSIS

To assess the efficiency of the proposed GRASP-based implementation in selecting features for intrusion detection at the perception layer, we conducted empirical evaluations considering blackhole, grayhole, and flooding attacks, from a public dataset [9] with 18 features shown in Table I. We experimented with five different classifiers: Naive Bayes, Random Tree, J48, Random Forest, and REPTree. Lastly, we implemented three filter-based FS methods to compare to our GRASP implementation: IG, GR, and OneR.

A. Parameters and Metrics

To demonstrate that few features can provide good results, the target feature subset length is set to 5 features. For the GRASP implementation, we define two stop criteria: the maximum number of iterations (*i.e.,* 50 iterations) and the maximum of consecutive GRASP iterations without improvement (*i.e.,* 20 iterations). These values were chosen because preliminary experiments have shown that these represent convergence points and computational efficiency.

TABLE I: Perception Layer Features (based on WSN-DS [9]).

Index	Description
1	Unique sensor node identification.
2	Current simulation time of the node .
3	Flag to distinguish CH and normal sensor.
4	The identification of the current CH node.
5	Distance between the node and its CH.
6	Number of advertise broadcast sent by CH.
7	Number of advertise CH messages received from CHs.
8	Number of join request sent to CH.
9	Number of join request received by the CH.
10	Number of advertise TDMA messages sent.
11	Number of TDMA messages received from CH.
12	Order of the node within the TDMA schedule.
13	Number of data packets sent to CH.
14	Number of data packets received from CH.
15	Number of data packets sent to the BS.
16	Distance between the CH and the BS.
17	Cluster sending code.
18	Energy consumption in the previous round.

We choose **accuracy** to compare the classifiers performance under different inputs (*i.e.*, feature subsets), each provided by a different FS method. Equation 1 shows that accuracy is computed using True Positives (TP), True Negatives (TN), False Positive (FP), and False negatives (FP).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

B. Results and Discussion

It is important to observe that filter-based methods evaluate each feature by considering their intrinsic characteristics to define how representative is it. Therefore, once three different attack classes are considered, each filter-based method selects three different feature subsets. We note that features highly correlated to the class attribute are commonly chosen by multiple methods. For example, the number of advertising broadcast sent by CH (*i.e.,* index 6) is closely related to the attack models in Section III. Accordingly, this feature is among the top-ranked ones by IG, GR, and OneR methods. Table II shows the five most relevant features (among all features found in Table I) selected by each filter-based methods.

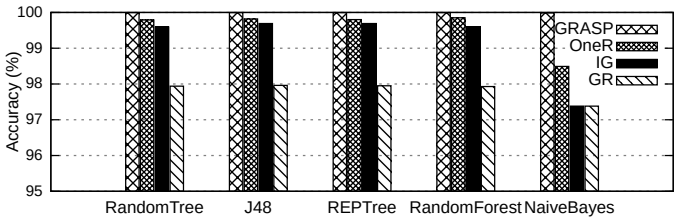
TABLE II: The five best ranked features by each filter method.

Attack Type	Best Ranked Features (Column Index)		
	Information Gain	Gain Ratio	OneR
Grayhole	3, 6, 12, 13, 18	3, 6, 8, 12, 13	3, 6, 7, 15, 18
Flooding	3, 6, 7, 13, 16	3, 6, 12, 13, 17	4, 6, 7, 8, 17
Blackhole	3, 6, 12, 13, 18	3, 6, 9, 10, 13	3, 4, 6, 7, 18

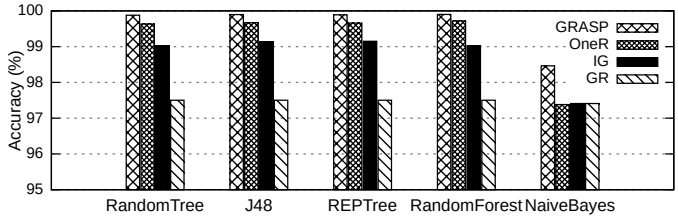
In contrast to purely filter-based methods, GRASP selects feature subsets that fit to the multiple classifiers' built model. Therefore, for each combination of attack model and classifier algorithm, different subsets is found (see Table III).

TABLE III: The feature subsets selected by GRASP.

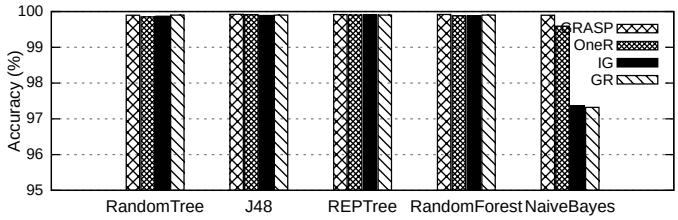
Classifier	Blackhole	Grayhole	Flooding
RandomTree	3, 10, 15, 1, 14	15, 12, 4, 9, 3	17, 12, 6, 1, 2
J48	15, 18, 12, 14, 3	15, 5, 4, 3, 9	18, 1, 2, 17, 6
REPTree	8, 9, 14, 15, 7	6, 18, 15, 3, 9	1, 6, 11, 4, 2
NaiveBayes	16, 18, 8, 1, 6	7, 11, 8, 6, 2	17, 8, 6, 1, 2
RandomForest	1, 14, 16, 6, 9	2, 13, 6, 14, 15	12, 4, 2, 6, 14



(a) Accuracy for detectign blackhole attacks.



(b) Accuracy for detectign grayhole attacks.



(c) Accuracy for detectign flooding attacks.

Fig. 3: Accuracy for perception layer attacks detection.

As result, GRASP achieved better accuracy than filter-based methods. The accuracy achieved by using the above-mentioned feature subsets for each classifier to detect three different attacks is shown in Figure 3. These results represent an average of a 5-fold cross-validation.

For blackhole attacks, GRASP enabled almost classifiers to reached an accuracy of 99.99%, that is, better performance than every classifier combined to the filter-based methods. From the 2010 blackhole attacks in the dataset, the filter selected by IG, GR, and OneR, respectively, enables the classifiers to detect, on average, 1816, 1510, and 1933 blackhole attacks, while GRASP supports the detection of 2009 attacks.

Regarding grayhole attacks, the best accuracy was also achieved by GRASP features, with an accuracy of about 99.9%. Not shown in the graph are the FN (attacks that were missed): Both J48 and Random Forest presented 26 and 22 false negatives, lower than the average supported by IG and OneR features, but higher than GR features (which had 0 FN). However, the GR features lead to J48 and Random Forest to experience 1,770 false positives each, lowering their accuracy to 97.50% for GR features. By using the GRASP features, there are only 45 and 47 FNs, respectively.

To detect flooding attacks, all classifiers using the GRASP features outperformed their own results when using IG, GR, and OneR features, except for Random Tree. This classifier achieved the same 99.90% accuracy by using both GRASP and GR. The difference is that GRASP presented 36 false positives and 35 false negatives, while GR presented 70 false positives and 1 false negative (not shown).

We observed that the node ID and the number of packets received from CH are common features that yield the highest accuracy to detect blackhole attacks. Similarly, the number of data packets sent to the BS belong to both best feature subsets that provide the best accuracy for grayhole attacks. For flooding attack detection, the current simulation time and number of advertising broadcasts sent by CH were relevant.

Finally, it is important to mention that we set accuracy as GRASP objective function. To optimize any other metric, simply replace the objective function by the desired one.

C. Other Metrics

Besides accuracy, we examine whether other metrics are improved by using GRASP as an FS method. In particular, when considering datasets with very few attacks, **accuracy** may not be the best measure to represent the classifiers' ability to detect samples that belong to classes with few samples in unbalanced data sets. In contrast, F1-Score in Equation 2 is a commonly used metric in machine learning, which, in our case, focuses on the classifiers' ability to detect the specific attack classes (*e.g.,* blackhole, grayhole, and flooding attacks). This metric is derived from Precision (*i.e.,* $\frac{TP}{TP+FP}$) and Recall (*i.e.,* $\frac{TP}{TP+FN}$). Note that TN is not a factor in F1-Score, that is, it doesn't take into account the benign traffic.

$$F1 - Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} 2 \tag{2}$$

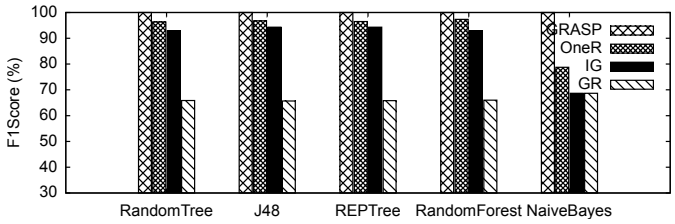
As can be seen in Figure 4, the feature subsets provided by GRASP (Table III) improve the F1-Score for all classifier algorithms, when compared to features provided by filter-based methods (Table II). All classifiers using the features selected by GRASP have their F1-Score increased significantly in detecting blackhole and grayhole. Detection of flooding has a large improvement for Naive Bayes, but all other classifiers produce the same high F1-Score.

The most relevant features found by GRASP to improve the F1-Score in detecting blackhole attacks are 1, 6, 8, 9, 14, and 16. These features are selected for at least two subsets that lead classifiers (*i.e.,* REPTree, Naive Bayes, and Random Forest) to reach an F1-Score of 100%, as shown in Figure 4(a). The average F1-Score by classifiers using features provided by OneR, IG, and GR, to detect blackhole attack samples was, respectively: 93.14%, 88.61%, and 66.40%.

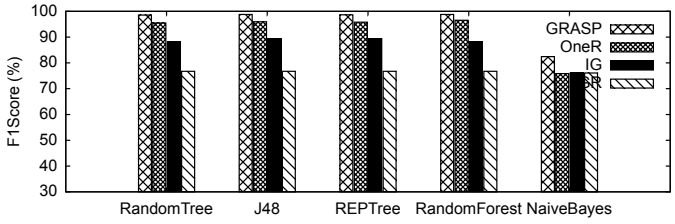
According to Figure 4(b), all algorithms have the best F1-Score for grayhole attacks detection when using features provided by GRASP; they reach about 98.80%, not counting

Naive Bayes. The average for classifiers using features provided by OneR, IG, and GR, to detect this attack was (not counting Naive Bayes) 95.94%, 88.72%, and 76.73% (91.93%, 86.19%, and 76.61% including Naive Bayes), respectively.

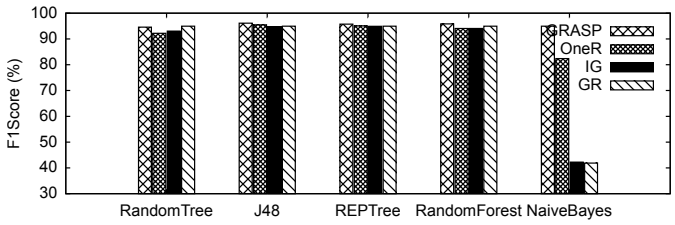
Finally, flooding attacks are best detected by J48 algorithm, as shown in Figure 4(c), with an F1-Score of 96.12% (not counting Naive Bayes). This result is reached by using the feature subset found by GRASP. Features selected by OneR, IG, and GR lead classifiers to reach, on average, an F1-Score of 94.19%, 94.19%, and 94.94% (91.83%, 83.80%, and 84.32% including Naive Bayes), respectively.



(a) F1Score for detecting blackhole attacks.



(b) F1Score for detecting grayhole attacks.



(c) F1Score for detecting flooding attacks.

Fig. 4: F1Score for perception layer attacks detection.

VII. Conclusion

At the perception layer of CPS and IoT systems, the tiny sensor devices are particularly susceptible to threats, due to their stringent constraints. Proper planning and design of secure architectures for CPS are important, but using IDSs is essential. Accordingly, FS becomes an important approach to enable accurate attack detection.

Our study quantified how a GRASP-based FS implementation can improve the accuracy of five classifiers to detect blackhole, grayhole, and flooding attacks targeting the CPS perception layer. Experimental results reveal that the GRASP metaheuristic provides improved feature subsets, overcoming traditional filter-based FS. GRASP enabled the classifiers to increase the detection up to 31.8% (blackhole), 12.26% (grayhole), and 7.70% (flooding), that is, detected on average about 17.25% more than with other FS techniques. These gains impact directly on the CPS availability. Also, with GRASP, the trade-off of computational cost and accuracy may be adjusted through the number of GRASP iterations.

As future works, we plan to investigate an unsupervised (e.g., parametric-less) GRASP method without user inputs. Also, other traditional algorithms such as Relief, Sequential Feature Selection (SFS), and Recursive Feature Elimination will be compared. Finally, we plan to extend our analysis to cover other CPS layers through other datasets and metrics.

References

[1] Y. Ashibani and Q. H. Mahmoud, "Cyber physical systems security: Analysis, challenges and solutions," *Computers & Security*, vol. 68, pp. 81–97, 2017.

[2] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, "A survey on platoon-based vehicular cyber-physical systems," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 263–284, 2016.

[3] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Computers in Industry*, vol. 81, pp. 11–25, 2016.

[4] X. Yu and Y. Xue, "Smart Grids: A Cyber–Physical Systems Perspective," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1058–1070, 2016.

[5] Zhang, Yin and Qiu, Meikang and Tsai, Chun-Wei and Hassan, Mohammad Mehedi and Alamri, Atif, "Health-CPS: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. 11, no. 1, pp. 88–95, 2017.

[6] A. Nourian and S. Madnick, "A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet," *Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 2–13, 2015.

[7] S. Week, "IoT Devices Infected via Supply Chain Attack," https://www.securityweek.com/, Feb. 2020.

[8] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A survey," *International Journal of Information Management*, 2018.

[9] Almomani, Iman and Al-Kasasbeh, Bassam and Al-Akhras, Mousa, "WSN-DS: a dataset for intrusion detection systems in wireless sensor networks," *Journal of Sensors*, vol. 2016, 2016.

[10] S. E. Quincozes, C. Raniery, R. Ceretta Nunes, C. Albuquerque, D. Passos, and D. Mossé, "Counselors Network for Intrusion Detection," *International Journal of Network Management*, p. e2111, 2020.

[11] M. A. Esseghir, "Effective wrapper-filter hybridization through GRASP schemata," in *Feature Selection in Data Mining*, 2010, pp. 45–54.

[12] E. Elhariri, N. El-Bendary, and S. A. Taie, "Using Hybrid Filter-Wrapper Feature Selection With Multi-Objective Improved-Salp Optimization for Crack Severity Recognition," *IEEE Access*, vol. 8, 2020.

[13] C. C. Ribeiro and M. G. Resende, "Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP," *ACM Transactions on Mathematical Software (TOMS)*, vol. 25, no. 3, pp. 341–352, 1999.

[14] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," *Pattern Recognition Letters*, vol. 30, no. 5, 2009.

[15] P. Bermejo, J. A. Gámez, and J. M. Puerta, "A GRASP algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets," *Pattern Recognition Letters*, vol. 32, no. 5, pp. 701–711, 2011.

[16] M. Moshki, P. Kabiri, and A. Mohebalhojeh, "Scalable feature selection in high-dimensional data based on GRASP," *Applied Artificial Intelligence*, vol. 29, no. 3, pp. 283–296, 2015.

[17] J. F. Diez-Pastor, C. García-Osorio, J. J. Rodríguez, and A. Bustillo, "GRASP forest: a new ensemble method for trees," in *International Workshop on Multiple Classifier Systems*. Springer, 2011, pp. 66–75.

[18] J.-F. Díez-Pastor, C. García-Osorio, and J. J. Rodríguez, "Tree ensemble construction using a grasp-based heuristic and annealed randomness," *Information Fusion*, vol. 20, pp. 189–202, 2014.

[19] N. K. Kanakarajan and K. Muniasamy, "Improving the accuracy of intrusion detection using gar-forest with feature selection," in *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA)*. Springer, 2016, pp. 539–547.

[20] Z. A. Dahi and E. Alba, "The grid-to-neighbourhood relationship in cellular GAs: from design to solving complex problems," *Soft Computing*, vol. 24, no. 5, pp. 3569–3589, 2020.