FI MU

Faculty of Informatics
Masaryk University Brno

Towards Better Selective Forwarding And Delay Attacks Detection in Wireless Sensor Networks

by

Martin Stehlik
Vashek Matyas
Andriy Stetsko

FI MU Report Series **FIMU-RS-2016-01**

Copyright © 2016, FI MU **April 2016**

**Copyright © 2016, Faculty of Informatics, Masaryk University.
All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

Publications in the FI MU Report Series are in general accessible via WWW:

`http://www.fi.muni.cz/reports/`

Further information can be obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**

Towards Better Selective Forwarding And Delay Attacks Detection in Wireless Sensor Networks.

Martin Stehlik
Masaryk University
xstehl2@fi.muni.cz

Vashek Matyas
Masaryk University
matyas@fi.muni.cz

Andriy Stetsko
Masaryk University
xstetsko@fi.muni.cz

April 11, 2016

Abstract

A number of intrusion detection techniques have been proposed to detect different kinds of active attacks on wireless sensor networks (WSNs). Selective forwarding and delay attacks are two simple but effective attacks that can disrupt the communication in WSNs. In this work, we propose two parametrized collaborative intrusion detection techniques and optimize their parameters for a specific scenario using extensive simulations and multiobjective evolutionary algorithms. Moreover, we sample the whole search space to enable evaluation of evolution performance. The found optimized results are also compared to a simpler non-collaborative detection technique to demonstrate improvements of collaborative approach. We also evaluate the influence of changes of the number of malicious nodes on the intrusion detection performance. This technical report extends our paper [SMS16] by details of experiment settings and results.

1 Introduction

Wireless sensor networks (WSNs) are highly distributed *ad hoc* networks consisting of low-cost electronic devices (sensor nodes). The sensor nodes are powered by batteries and consist of a radio transmitter, sensors, limited micro-controller and memory. Their goal is to monitor (potentially large) environments for various physical phenomenons like temperature, humidity, movement or light intensity. WSNs find use in agriculture, ecology, military, building or industrial monitoring and automation, etc. The communication radius of the sensor nodes is often limited to tens of meters. Thus, the collected

measurements are routed towards a base station (BS) for further processing from node to node on *hop-by-hop* basis.

Since the sensor nodes are often deployed in open or even hostile environments and also given by the nature of wireless communication and other limitations, WSNs are highly vulnerable to various kinds of attacks. These attacks can range from passive eavesdropping to active interfering. An attacker can capture and reprogram or replace a sensor node with malicious one or just insert his own malicious device into the network. These devices can be believed as benign by other legal sensor nodes.

In our research, we aim at detecting active attacks by intrusion detection systems (IDSs). Several IDSs have been proposed to detect different attacks on WSNs. In this paper, we focus on a *distributed* IDS, where sensor nodes themselves monitor the overall network area by promiscuous listening on the transmissions among their neighbors [SMR+05].

However, most of proposed IDSs are designed for a specific purpose or topology and their proper reconfiguration for different purposes or topologies is not discussed or left to the user. Moreover, the limited sensor nodes' resources are also often left unresolved. In a recent work [SSMS14], we proposed and evaluated an *optimization framework* consisting of a *simulator* and an *optimization engine* that can optimize IDS for particular needs. The optimization engine designs IDS solutions that are evaluated by a network simulator and simulation results are provided back to the optimization engine for further optimization. This process continues until sufficiently good results are found. We experienced a very good performance of evolutionary algorithms to design new IDS configurations. For the simulations, we used the MiXiM framework [KSW+08] based on the widely used OMNeT++ network simulator. MiXiM enables precise and quite accurate simulations, as we investigated in [SSM11].

In [SSSM13], we demonstrated the benefits of multiobjective evolutionary algorithms (MOEAs) on a simple IDS where no collaboration among the sensor nodes was performed. The main benefit of MOEAs is elimination of limits given by single-objective evolutionary algorithms, where all objectives like memory consumption and IDS accuracy have to be blended into a single *fitness function* with given weights specified by a network operator before the optimization process. In case of changes in the requirements for the IDS's performance (giving more weight to one objective and less to another one), the evolution process has to be run again. Moreover, finding suitable weights for the fitness function can be a very hard task. Using MOEAs, the network operator

can choose any IDS setting from the *Pareto front*[1] [Tal09] approximation and change the selection to another optimized one any time according to current requirements. We evaluated 48 different MOEA's configurations for two widely used algorithms – NSGA-II [DPAM02] and SPEA2 [ZLT01]. For each of the evolution results, we compared convergence and diversification of the found solutions with optimal results found using exhaustive search. We also compared time consumption.

In this technical report, we extend our paper [SMS16] by details of experiment settings and results. In [SMS16] and this report, we propose and – based on the previous experience [SSSM13] – optimize two more complex collaborative detection techniques that utilize communication between IDS nodes before making a decision about a monitored node.

The *contributions* of this work are threefold:

1) We provide a novel technique detecting *delay attack* where an attacker intentionally delays ongoing packets. This detection technique is optimized using an enhanced IDS optimization framework.

2) We propose and optimize a collaborative IDS for selective forwarding attack based on [KDF07] and compare it with simple IDS evaluated in [SSSM13]. Various attacker strategies are discussed. We explore optimized IDS performance in case of changes in the number of malicious sensor nodes.

3) We enhance and evaluate a complex IDS optimization framework for WSNs with distributed simulations. MOEA is used to optimize the IDS and we show that MOEAs can be efficiently used even without a computation cluster.

Furthermore, we point out a non-trivial bug that we have found in ParadisEO [INR13, LJT11], a frequently used software framework for metaheuristics. The bug was reported to the authors.

The paper is organized as follows. In Section 2, we discuss related work. The optimized detection techniques and their parameters are presented in Section 3. Experiment settings are described in Section 4 and the approach that we used to evaluate the

[1] Pareto front is a set of non-dominated solutions with respect to all objectives. Thus, a network operator can easily choose between a solution A with a better IDS accuracy but higher resource consumption or solution B with a worse IDS accuracy but consuming lower resources. Solution C, that is dominated by A and B in all objectives is dominated and, thus, is not a member of the Pareto front.

solutions found by evolution are described in Section 5. Results of the experiments are presented and discussed in Section 6. The paper is concluded in Section 7.

2 Related Work

Selective forwarding attack has been among the most discussed attacks in WSNs in last years. Karlof et al. in [KW03] introduced selective forwarding attack in WSNs and discussed the possibilities of an attacker to place a malicious sensor node on a path between data source and base station. Silva et al. in [SMR+05] defined a "retransmission rule" as listening to a packet by an IDS whether it was forwarded by monitored sensor node or not. Krontiris et al. in [KDF07] set a threshold value for the percentage of packets dropped to 20%. Another proposals of detection techniques for selective forwarding attack where the parameter setting is left unresolved can be found, e.g., in [HH08, TACC09]. To the best of our knowledge, no work has been published on such a complex parameter optimization for collaborative detection of selective forwarding attack.

The delay attack detection has not been discussed very much either. Silva et al. in [SMR+05] defined a "delay" rule as a timeout before which a retransmission by monitored sensor node must occur. Liu et al. in [LCC07] used the forwarding delay time measurement for their complex insider detection technique, but its parametrization was left unresolved. To the best of our knowledge, we are the first presenting complex collaborative detection technique aimed at the delay attack detection.

The optimization problem in different aspects of WSNs has been considered in several papers. Anjum et al. in [ASSS04] presented strategy on optimal placement of IDS nodes. Two types of sensor nodes and specific clustered topology was assumed. Cheng et al. in [CCL04] proposed the energy-aware node placement of sensor nodes and the relay sensor node placement was considered in [CDWX07]. Khanna et al. used single-objective evolutionary algorithms for several optimization issues in WSNs in [KLC06, KLC07, KLC09]. In [KLC09], the placement problem of nodes monitoring intrusion was optimized. In [SSMS14], we presented a framework aimed at the optimization of intrusion detection system in terms of detection accuracy and resource consumption.

3 Detection Techniques

In this section, we describe details of detection techniques that we evaluate in this paper. All proposals are aimed at distributed detection where an IDS runs on each sensor node deployed in a WSN. Thus, all network area can be monitored to detect malicious behavior by sensor nodes themselves. On the other hand, the IDS consumes additional sensor nodes' resources like memory.

First, we present a non-collaborative technique detecting *selective forwarding* attack that we optimized in [SSMS14]. Executing the selective forwarding attack, malicious sensor nodes forward only a fraction of received packets. Second, we present a more complex collaborative detection technique evaluated in this work aimed at the same kind of attack. The idea behind the collaborative approach basically comes from [KDF07] regarding to voting scheme and time windows. However, we enriched the collaborative approach presented in [KDF07] by parameters "voting threshold" and "minimum received votes". Finally, we propose a novel collaborative technique detecting *delay* attack [SMR+05] where the forwarded packets are intentionally delayed by the malicious nodes.

We use the following notations to explain the functionality of the IDS [SSMS14]:

Notation 1. *The set* $A = \{a_1, ..., a_{n_m}\}$ *is a set of all malicious nodes in a network.*

Notation 2. *The set* $C = \{c_1, ..., c_{n_b}\}$ *is a set of all benign nodes in a network.*

Notation 3. *The function* $x : \mathbb{N} \rightarrow \mathbb{N}$ *takes a sensor node index as an argument, and returns a number of the neighbors that consider this node benign.*

Notation 4. *The function* $y : \mathbb{N} \rightarrow \mathbb{N}$ *takes a sensor node index as an argument, and returns a number of the neighbors that consider this node malicious.*

Notation 5. *The function* $n : \mathbb{N} \rightarrow \mathbb{N}$ *takes a sensor node index as an argument, and returns a number of the neighbors of this node.*

Neighbor $b_k \in C \cup A$ of a node $c_j \in C$ is each node such that c_j overheard at least one packet from b_k since the beginning of the WSN operation time.

Monitored neighbor $b_l \in C \cup A$ of a node $c_j \in C$ is such a *neighbor* of the node c_j that the IDS running on the node c_j collects the statistics of the packet forwarding of the node b_l. The selection process for the set of the monitored neighbors is described in Section 3.2.

Solution s is a specific configuration of the IDS in a form of a detection technique and specific values given to each of the parameters used by that technique.

Ranges of values of IDS parameters discussed in the following text are then discussed in more detail in Section 5.

3.1 Objectives

We use the three following. All the objectives are *minimized*.

Objective function 1. The number of *false negatives* (fn) of a solution s is calculated as follows:

$$fn(s) = \frac{1}{|A|} * \sum_{a_i \in A} \frac{x(a_i)}{n(a_i)}. \tag{1}$$

The values of fn range from 0 to 1. If every malicious node in the network is correctly detected by all of its neighbors, fn is equal to 0 and if none of malicious nodes is detected by any of its neighbors, fn equals to 1.

Objective function 2. The number of *false positives* (fp) of a solution s is calculated as follows:

$$fp(s) = \frac{1}{|C|} * \sum_{c_i \in C} \frac{y(c_i)}{n(c_i)}. \tag{2}$$

The values of fp range from 0 to 1. If every benign node in the network is considered benign by all of its neighbors, fp is equal to 0 and if all benign nodes are considered malicious by all of its neighbors, fp equals to 1.

Objective function 3. The consumed *memory* (mem) in a solution s is calculated as follows:

$$mem(s) = 8 * p_1 + 16 * p_2, \tag{3}$$

8 bytes are required for every monitored neighbor (4 bytes for node ID, 2 bytes for PR counter and 2 bytes for PF counter) and 16 bytes are required for one slot in the buffer (4 bytes for source address, 4 bytes for receiver address, 4 bytes for destination address in a case of multiple base stations in the WSN and 4 bytes for unique ID of a packet). The memory demands come from our real application (ProtectLayer for WSN [MSS+15]).

The values of mem range from 0, where the IDS is potentially switched off, to 288 bytes for our upper bounds of $p_1 = 30$ and $p_2 = 3$ used for selective forwarding attack, respective 400 bytes for our upper bounds of $p_1 = 30$ and $p_2 = 10$ used for the delay attack.

3.2 Non-collaborative Detection of Selective Forwarding Attack

In [SSSM13], we evaluated MOEAs on a simple IDS detecting selective forwarding attack. An IDS was running on each sensor node and continuously monitoring its own sent and also overheard packets addressed to all monitored sensor nodes whether they were forwarded or dropped by those monitored sensor nodes.

The basic principle is illustrated in Figure 1. The black dots represent sensor nodes that are placed within communication range of sensor node $b_i \in C \cup A$ and, thus, can monitor b_i for selective forwarding attack. However, the number of monitored neighbors is limited to p_1 (*max monitored nodes*), not only due to memory reasons – the IDS can have incomplete information about furthest neighbors (the IDS nodes can be interfered, far from the monitored node or hidden behind an obstacle) causing additional false positives. Thus, each IDS monitors at most p_1 nearest neighbors (according to received signal strengths). The arrows represent routing directions of the packets – b_i forwards all received packets to a parent node $b_j \in C \cup A$. The IDS maintains a table, where each of p_1 rows corresponds to a certain monitored node. The table contains the number of packets received (PR) and forwarded (PF) by each monitored node.

The IDS stores all overheard packets addressed to all monitored neighbors in a single buffer limited to p_2 packets (*buffer size*). Each time a packet P addressed to a monitored node b_i is overheard by the IDS, the PR counter of b_i is incremented and packet P stored in the buffer. Once the node b_i forwards the packet P, the IDS increments PF of the node b_i and packet P is removed from the buffer. In case the packet P is being the oldest one and the buffer is full, it is removed from the buffer without incrementing the PF counter.

Figure 1: Non-collaborative intrusion detection.

Finally, during the evaluation phase, a sensor node b_i is considered as attacker by the IDS node if two following conditions hold:

(i) The IDS node has overheard (or sent) at least p_3 packets (*min received packets*) addressed to b_i.

(ii) The ratio of forwarded and received packets (PF/PR) is lower than p_4 (*detection threshold*).

3.3 Collaborative Detection of Selective Forwarding Attack

In this work, we demonstrate the usability of the enhanced optimization framework incorporating MOEAs on two more complex detection techniques, where the IDS nodes cooperate on decision, whether a monitored node $b_i \in C \cup A$ is malicious or not.

We extended the non-collaborative IDS in the following way. The monitored nodes are not evaluated by the IDS nodes at the end of the simulation. Instead, the simulation time is divided into windows of size p_5 (*time window*). The time windows are of the same fixed size among all the IDS nodes, but they are asynchronous – the first window of each IDS node is started randomly within the time interval of p_5. At the end of each time window, all monitored neighbors are evaluated by the IDS node and if an attack was detected, a voting process can be executed.

An example situation is depicted in Figure 2 where an IDS node c_k locally considers a monitored node b_i malicious since the end of window marked as "Attack!" (where c_k observed too many dropped packets) in Figure 3 until the end of the whole WSN operation time. This decision is based on the same principle as for non-collaborative IDS discussed in Section 3.2. However, to label the node as malicious, a "global" decision has to be made. Thus, c_k broadcasts a voting request to its neighbors (arrows from c_k point the neighbors of c_k that can also monitor node b_i). Each of the asked nodes that also monitors node b_i answers at the end of its own time window. If an asked

Figure 2: Collaborative intrusion detection.

Figure 3: Time windows. Since the end of the window in which the attack was detected, node c_k always vote positively about node b_i. Node b_i can be globally considered malicious by c_k since the end of window following the one in which the attack was detected if voting result is positive.

IDS node consider b_i as an attacker (either locally or globally), it answers positively, otherwise negatively. Node c_k waits the following time window to collect the responses. Finally, the monitored node b_i is considered an attacker globally by c_k if two following conditions hold:

(i) At least p_6 votes (*min votes received*) were received.

(ii) The ratio of positive and all responses is at least p_7 (*voting threshold*).

The collaborative approach should eliminate false positives. IDS nodes that, e.g., cannot overhear all forwarded packets can falsely label a monitored node as malicious. In the collaborative approach, a voting among other neighboring IDS nodes has to be performed. Other IDS nodes participating in the voting can deny the false decision of IDS node that has invoked the voting. At least some of the IDS nodes participating in the voting are placed at better positions overhearing the outgoing packets of monitored node more precisely. On the other hand, no or only negligible increase of false negatives is assumed since each benign node participating in the voting can recognize dropping easily.

3.4 Collaborative Detection of Delay Attack

Time related attacks result in long delays and traffic imbalance [RLG08]. We believe that a WSN should guarantee the delivery time for some applications (e.g., movement monitoring or fire detection). In [SMR+05], the *delay detection rule* is defined as follows: "The transmission of a message by a monitor's neighbor must occur before a *defined timeout*". We adapt this rule to our IDS detecting delay attack.

A technique that is similar to that one used for selective forwarding attack can be extended to detect intentional delays. Using the buffering technique discussed above, a packet can finally be considered forwarded even though some malicious node delayed its transmission. If the size of the buffer is not exhausted, the monitored packet can be stacked in the buffer for a long time and finally forwarded by the monitored neighbor with a big delay before being removed from the buffer by the IDS. Thus, such a packet is undetected for selective forwarding attack, yet useless for a base station if real-time sensing is required.

An important issue to consider is how long the IDS should wait until the packet is considered delayed and how many packets have to be delayed to consider a monitored neighbor as *delay* attacker. We suggest to assign a *time attribute* to each of the buffered packets. If a predefined timeout passes, the packet is considered delayed. As for the selective forwarding attack, an alert is produced when p_5 time units pass and the ratio of delayed packets is higher than p_4. In such a case, *majority voting scheme* is applied for decision about *delay attack*.

Our proposed and evaluated detection technique for delay attacks is an extension of the selective forwarding attack detection technique specified in Sections 3.2 and 3.3. We incorporate another parameter p_8 that is a timeout when a packet in the IDS buffer is marked as delayed.

4 Experiment Settings

In this section, we describe experiment settings and optimization scenarios that we use for evaluation of our IDSs. We also provide simulation settings of the MiXiM simulator [KSW+08], wireless channel model used and its parametrization. Basically, the simulation models are set up according to [SSSM13] to enable a comparison.

Note that the simulated WSN evaluated in this work is near-realistic application inspired by the mobile police unit scenario in [MSS+15]. Each sensor node sends peri-

odically every 1 second "still alive" packets that can be either dropped or delayed by malicious sensor nodes. The main goal of our optimization framework is to optimize the IDS for given specific scenario (application, topology, environment, etc.), not to provide a general IDS setting for any WSN. We assume a role of a network administrator that is responsible for the specification of the target WSN before optimizing according to the reality. However, attacker strategies that also have to be simulated, can hardly be predicted by the network administrator. Our approach to the attacker strategies is discussed in Subsection 4.3.

In our case, we simulate a WSN consisting of sensor nodes equipped with the CC2420 transceiver (widely used by MICAz and TelosB platforms) in an open environment. The description of settings of different simulation models follows:

4.1 Wireless Channel and Low Layers Models

All the following settings correspond to the experiments presented in [SSSM13].

An open changing environment is simulated using the *log-normal shadowing* model [Rap01] that is the most widely used wireless channel model among the simulators [SSM11]. The pass loss exponent representing the signal propagation was set up to 2 (outdoor environment). The variations in received signal are reflected by a Gaussian random variable with zero mean and standard deviation set up to 2. The time interval of the changes was set up to 0.001 s.

Protocol CSMA-CA according to the IEEE 802.15.4 standard is used on data link layer.

On the physical layer, the radio model represents the CC2420 transceiver that is compliant to the IEEE 802.15.4 standard and is used by MICAz and TelosB sensor nodes. The transmitting power is set up to -25 dBm (0.00316227766017 mW) for all sensor nodes.

4.2 Topology and Routing

We build on the topology and routing same as in [SSSM13], so that we are able to compare the collaborative and non-collaborative IDS results. The network consists of 250 uniformly distributed sensor nodes deployed in an area of 200 x 200 meters. The average area for one node is 160 m^2 and the distance between two nearest neighbors is 12.65 m on average. During the simulation, a node $b_j \in C \cup A$ has 41 neighbors (nodes from which b_j heard at least one packet during the simulation) on average.

Figure 4: Topology of the evaluated WSN. The sensor nodes are represented by circles while the base station is represented by red diamond. The black circles represent malicious sensor nodes for the scenario with 2% malicious sensor nodes and together with the gray circles for the scenario with 10% malicious sensor nodes.

The routing tree is static with longest branches of 8 hops. The topology and the routing tree are depicted in Figure 4.

4.3 Attacker Strategies

We distinguish the attacker strategy into the three following categories – 1) behavior of a malicious node; 2) patterns in deployment of malicious nodes; and 3) number of deployed malicious nodes.

4.3.1 Attacker behavior

Ongoing packets can be dropped (delayed) purely randomly. Random dropping can be parametrized by the percentage of packets dropped. A network administrator specifies a percentage of packets that have to be dropped[2] (delayed) to consider a node as malicious and this behavior is then simulated during the optimization process. If the

[2]Some of the packets can be lost (delayed) by interference, other aspects of unreliable wireless communication or by malfunction, not by maliciousness.

percentage of dropped (delayed) packets is lower, the behavior is acceptable. Otherwise, the malicious nodes are detected even more reliably.

Other strategy can be dropping (delaying) the packets in a bulk – a malicious node drops (delays) some number of consecutive packets but from the longer perspective, the percentage is below the IDS threshold. (Not only) to detect this kind of attack, we introduce the time windows in our IDS. Dropping (delaying) several consecutive packets leads to a dropping (delaying) above a given IDS threshold within a specific time window in which the maliciousness can be detected. A network administrator has to decide on the maximum window size to detect such a consecutive dropping (delaying).

Finally, an attacker can drop the packets based on their contents (importance). Regarding to the example of mobile police unit scenario discussed above, an attacker can, e.g., drop a packet that informs the base station about presence of an intruder. To detect this kind of attack, we encourage a network administrator to use an IDS with a separate buffer and table for such important packets. This would mean additional (approx. doubled) requirements on the amount of memory used by the IDS but the optimization process would be analogous.

4.3.2 Deployment of malicious nodes

An attacker can deploy malicious sensor nodes into a WSN in specific "patterns" [JSM14, YX06]. Apart from random deployment considered in this work and most of other publications, the malicious nodes can be deployed to all neighboring places in a specific area (e.g., around the base station), around a line segment on which an attacker passed through the environment of WSN, etc. These deployment strategies are out of the scope of this paper and we plan to elaborate on them in future. However, to provide robust IDS solutions for various deployment strategies, all such strategies should be simulated for each IDS setting considered during the evolution. Performance of each IDS setting in each of the deployment strategy could be then, e.g., averaged during the evolution.

4.3.3 Number of deployed malicious nodes

The number of present malicious nodes can vary for each deployment strategy. For this reason, we elaborate on the changes of IDS behavior in cases where the percentage of

malicious nodes is different during the simulation. The analysis can be found in Section 6.1.2.

4.3.4 Our test case

In this work, we assume presence of 5 (2%) and 25 (10%) randomly placed malicious sensor nodes that drop randomly 50% of packets that should be forwarded for selective forwarding attack. Evaluating delay attack, the same sensor nodes delay all packets randomly within interval $< 0,5 >$ seconds. Since the delay detection is similar to the selective forwarding attack detection, we consider only the case with 5 malicious sensor node for the delay attack. Detailed topology illustration including malicious nodes placement can be found in Figure 4.

Note that we do not assume any "edge" sensor node to be an attacker since it can neither be detected by the IDS (no packets are addressed to these nodes), nor efficiently perform selective forwarding or delay attack (these nodes only produce their own packets).

5 IDS Optimization

In this section, we describe two methods that we use to compute Pareto front approximations – MOEA and sampling. MOEA is used as an efficient optimization tool. Using sampling, we show that MOEA can find comparable or even better results than sampling through all the objective space in much shorter time. Results of both methods are compared in Section 6.

5.1 Optimization Using MOEA

We optimize the IDS parameters using MOEA to efficiently speed up the process of finding good Pareto front approximation as dicussed in Section 6. For both detection techniques, we evolved the IDS parameters with the widely used NSGA-II algorithm [DPAM02]. The evolution parameters were set up in two ways, their specific values can be found in Appendix A.

During the evolution running on a server, we distribute the evaluation of all the individuals in the population in a form of simulation configurations to multiple computers

Name	Description	Range	Step	Sampling
p1	*Maximum monitored nodes*	$\langle 1, 30 \rangle$	1/3	$3, 9, 27$
p2	*Buffer size*	$\langle 1, 3 \rangle / \langle 1, 10 \rangle$	1	$1, 2, 3/3, 6, 9$
p3	*Minimum received packets*	$\langle 1, 30 \rangle$	1/5	$1, 15, 30$
p4	*Detection threshold*	$\langle 0.05, 0.95 \rangle$ / $\langle 0.1, 0.9 \rangle$	0.05/0.1	$0.25, 0.5, 0.75$
p5	*Time window*	$\langle 10, 300 \rangle$	10/30	$10, 150, 300$
p6	*Minimum received votes*	$\langle 1, 10 \rangle$	1/2	$1, 5, 10$
p7	*Voting threshold*	$\langle 0.1, 1 \rangle$	0.1	$0.25, 0.5, 0.75$
p8	*Delay timeout*	$\langle 1, 5 \rangle$	1	$1, 3, 5$

Table 1: The list of IDS. If multiple values are presented and divided by "/", the first values were used for the detection of the selective forwarding attack and the second values for the delay attack detection.

using BOINC distributed computing platform [And01]. Once all simulation results are uploaded to the server, new generation is produced.

5.1.1 Parameter Ranges

In Table 1, we summarize all eight parameters, their maximum and minimum values, step and sampling values used for sampling discussed later.

5.1.2 ParadisEO and Bug Discovered

For the evolution optimization, we use a widely used framework for metaheuristics – ParadisEO [INR13, LJT11]. The framework provides various techniques for evolution algorithms including MOEA. Using ParadisEO, one can easily switch among different algorithms like NSGA-II, SPEA2 or IBEA.

During our early experiments, we encountered an unexpected low diversity of the resulting population. Based on this observation, we evaluated the population selection during the optimization process step-by-step after each generation. A behavior inconsistent with the definition of NSGA-II was encountered – disappearance of non-dominated solutions that evinced extreme values of any of the objectives. Note that these solutions should definitely preserve in the population based on NSGA-II diversity criterion. This observation led us to a thorough investigation and discovery of a wrong comparison of solutions that are in the same "dominance" front in ParadisEO.

Figure 5: Sampling Pareto front approximation for the collaborative selective forwarding attack detection compared to the true Pareto front for the non-collaborative selective forwarding attack detection. All the solutions found by sampling for the collaborative detection dominate the solutions found by exhaustive search for the non-collaborative detection.

This bug significantly influences selection of all population members – not only the solutions at the edges of current Pareto front approximation. More details can be found in Appendix B.

5.2 Sampling

In order to show that evolution found good enough results, we compared the results found by MOEAs with a true Pareto front found using exhaustive search on multiple computers in [SSSM13]. However, we are not able to compute all possible settings for this more complex IDS with additional parameters even if we can run about 200 simulations in parallel in our computational cluster. The exhaustive search would require $148,770,000$ simulation runs for the scenario with the selective forwarding attack and $2,479,500,000$ for the scenario with the delay attack if all possible settings searched by evolution would be evaluated. One simulation takes approx. $5-8$ minutes.

We decided to sample the search space in the following way. For each parameter p_i, where $i \in \{1, ..., 7\}$ for selective forwarding attack and $i \in \{1, ..., 8\}$ for delay attack, we chose three carefully considered "sampling" values presented in Table 1. The selection is based on experience and results obtained during early experiments. Then, we iterate over all parameters $p_1, ..., p_7$ for selective forwarding attack, respective $p_1, ..., p_8$ for delay attack. For each of the parameters, we evaluate all settings within their ranges using steps provided in Table 1. For each value of each parameter, we evaluate all "sampling" settings of all other parameters. Using this approach, we reduce the number of simulations to $84,564$ for selective forwarding attack and to $122,472$ for the delay attack.

Having the set of solutions obtained from aforementioned sampling, we extract only those solutions that are not strictly dominated by any other solutions within this set. We call this extracted set *Sampling Pareto front approximation*. This set is compared to Pareto front approximations found by evolutions. Note that finding *Sampling Pareto front approximation* is much more computationally demanding than finding Pareto front approximations using evolution as discussed in Section 6.

5.2.1 Parameter Ranges

In Table 1, we summarize all eight parameters, their maximum and minimum values, step and sampling values.

5.2.2 Coverage Metric

As mentioned above, we are not able to compare the Pareto front approximations found by different optimization techniques (sampling or evolution) with the true Pareto front. Thus, we use a *coverage metric* that was introduced by Zitzler et al. in [ZT99] to compare solutions found by two different optimization processes. This metric is used to compute the percentage of solutions found in Pareto front approximation A that are not dominated by any solution found in Pareto front approximation B, and vice versa. See a sample comparison in 6.1.1.

6 Experiment Results

In this section, we provide experimental results of the optimized IDSs that we obtained both using the sampling and evolution. We also compare the performance of IDS de-

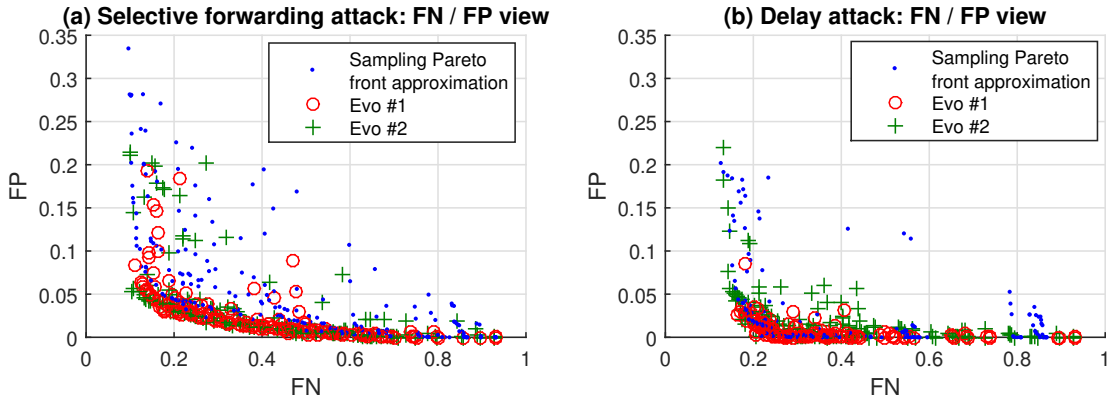

Figure 6: Results for detection of selective forwarding attack (a) and delay attack (b) found by evolution compared to results of Sampling Pareto front approximations.

tecting selective forwarding attack presented in [SSSM13] with collaborative version presented in this paper (same metrics and simulation setting were used).

6.1 Selective Forwarding Attack

First, we present results for detection of selective forwarding attack found by sampling (the *Sampling Pareto front approximation*[3]). We compare the results with the true Pareto front of non-collaborative IDS to show the improvements of IDS performance. In Figure 5, we show different views of the optimized solutions in the three-dimensional objective space.

In Figure 5 (a), we show that all sampled non-dominated solutions found for collaborative IDS dominate the Pareto optimal solutions found for non-collaborative IDS. In Figure 5 (b), we confirm the assumption that collaboration among the IDS nodes can significantly decrease the number of false positives. This reduction is caused by the fact that a node can be labeled as attacker only if a consensus among several IDS nodes is made. Also, a decrease of false negatives can be observed from the same view. It is caused by dividing the monitoring time into smaller windows, where, in each of them, a potential dropping can be recognized and agreed among the neighbors. The view depicted in Figure 5 (c) shows that even better solutions regarding number of false positives are reachable even if the same amount of memory is allocated for the IDS. However, the solutions with a higher number of false positives come with lower false negatives. Finally, the view in Figure 5 (d) confirm the assumption that collaborative

[3]We extracted 201 non-dominated solutions out of all 84, 564 samples.

decision making does not increase number of false negatives. In Figures 5 (c) and (d), we can see that higher memory consumption caused particularly by a higher number of monitored neighbors decrease false negatives on one hand (more neighbors being monitored means also higher number of truly recognized droppers), but increase false positives on the other hand (if a neighbor is not monitored, it can neither be labeled as malicious one truly, nor falsely).

Evolution can speedup the process of finding good enough solutions or solutions that even dominate the solutions in the *Sampling Pareto front approximation*. We present results of two multiobjective evolution runs (marked as "Evo #1" and "Evo #2") set according to the experience from [SSSM13]. The evolution settings can be found in Appendix A.

The solutions found by evolution for the selective forwarding attack are depicted in Figure 6 (a). 13,502 simulation runs were needed for the "Evo #1" and 23,558 for "Evo #2". However, the evolutions converged to the found results set already after approx. 100 generations, that would reduce the number of simulations needed to about a half. We found out that most solutions found by evolution dominate the solutions obtained in the *Sampling Pareto front approximation* set, especially regarding to the number of false positives. For detailed results, see Appendix A.

6.1.1 Coverage Metric

Using coverage metric, we computed that "Evo #1" found 94 out of all 144 solutions (65%) that are not dominated by any solution of "Evo #2". On the other hand, "Evo #2" found 126 out of all 153 solutions (82%) that are not dominated by any solution of "Evo #1". Comparing the two evolution runs, we can conclude that "Evo #2" found "better" Pareto front approximation at the cost of higher number simulation runs.

To strengthen the observation that evolution outperformed sampling not only regarding the time demands and observation in Figures 6 and 8, we use the coverage metric also to compare both evolution runs with the sampling. The results are the following. Any of 144 solutions found by "Evo #1" is not dominated by any solution found by sampling (100%) and only 3 out of 153 solutions (98%) are dominated for "Evo #2". However, 45 out of 201 solutions (22%) and 39 out of 201 solutions (19%) found by the sampling are not dominated by any solution found by "Evo #1" and "Evo #2", respectively. We found some of these solutions behind the "edges" of Pareto front approximations of "Evo #1" and "Evo #2" that is an advantage of the sampling comparing to

	Evo #1	Evo #2	Sampling
Evo #1	–	65% (94/144)	100% (144/144)
Evo #2	82% (126/153)	–	98% (150/153)
Sampling	22% (45/201)	19% (39/201)	–

Table 2: Coverage metric results for the selective forwarding attack. For each Pareto front approximation in a row, the values specify the number of found solutions that are not dominated by any solution within the result set of Pareto front approximation specified in the column.

evolution. More specifically, sampling found slightly lower minimal values of the false negatives (0.096) in comparison with the evolution (0.112 and 0.101 for "Evo #1" and "Evo #2", respectively). Nevertheless, we consider this difference negligible. All results obtained using coverage metric are summarized in Table 2.

6.1.2 Comparison of Results For 2% And 10% Malicious Sensor Nodes

As we outlined in Subsection 4.3, we analyze the influence of increased or decreased percentage of malicious sensor nodes on IDS performance.

In Figures 7 (a-c), we show how the performance of IDS settings optimized for WSN with 2% malicious sensor nodes[4] changes in a WSN with 10% malicious sensor nodes. Figures 7 (d-f) show how the IDS performance of IDS optimized by evolution for 10% malicious sensor nodes[5] changes in the WSN with 2% malicious sensor nodes.

The memory consumed by the IDS is same for both testing scenarios (the IDS is configured in the same way and, thus, the memory demands are constant). In Figures 7 (b) and (e), we can see that changes in percentage of malicious sensor nodes (both up and down) have negligible impact on number of false positives. Some improvement of false positives if the percentage of malicious nodes is higher (better observable for solutions with higher numbers of false positives) is caused by higher ratio of "edge" sensor nodes (that can never be marked as malicious by the IDS – see Subsection 4.3.4) and decreased number of other benign sensor nodes.

[4]Pareto front approximation optimized by "Evo #2" (see Figure 6 (a)).

[5]157 non-dominated solutions were found using "Evo #2".

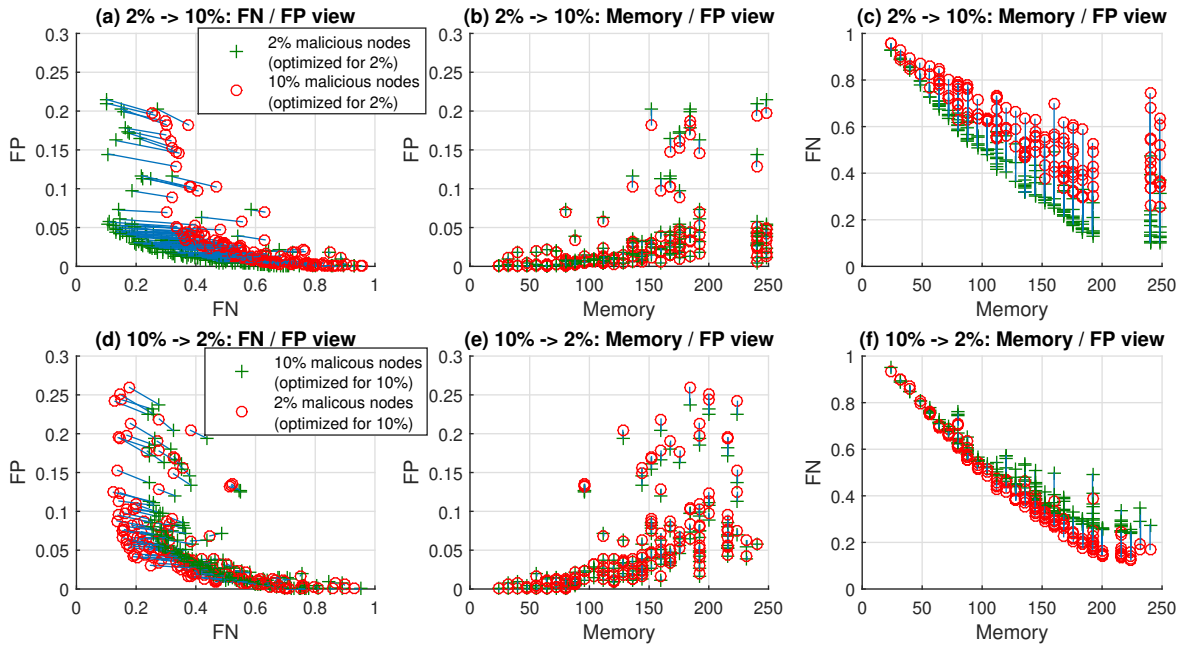

Figure 7: Influence of changed percentages of malicious sensor nodes on performance of each optimized IDS setting. Green crosses represent IDS performance in an environment for which the IDS was optimized (2% for Figures (a-c) and 10% for Figures (d-f)). Red circles represent IDS performance in an environment with increased (Figures (a-c)) and decreased (Figures (d-f)) percentage of malicious sensor nodes. Lines connect equal IDS settings.

The most significant change in IDSs performance in case of increased percentage of malicious sensor nodes is higher number of false negatives (see Figures 7 (a) and (c); Figures 7 (d) and (f) depict opposite trend for decreasing percentage of malicious sensor nodes). Investigating the number of false negatives on "per-node" basis, we found out that malicious sensor nodes close to the edges of the WSN in the situation with increased percentage of malicious sensor nodes evince higher number of false negatives. Such "close-to-edge" malicous sensor nodes cannot be detected as malicious by sufficient number of neighbors[6]. In some IDS settings, we encountered individual false negatives differing in range of approx. $\langle 0.2, 1 \rangle$ based on the position in the WSN. To enable more reliable detection of "close-to-edge" sensor nodes, we suggest (if possible in real WSN) to decrease the number of the voting threshold and possibly the number of the minimum received votes individually for neighboring IDSs.

6.2 Delay Attack

In this section, we present non-dominated results for detection of delay attack found both by the *Sampling Pareto front approximation* and evolution.

In Figure 6 (b) we can see that the number of solutions found by *Sampling Pareto front approximation*[7] is reduced comparing to selective forwarding attack. It is caused by the fact that we were not able to compute such "dense" sampling since the search space is much larger for delay attack and hence we lost some non-dominated solutions. Since the basic principle of delay attack detection is similar to the selective forwarding attack detection, we can observe a similar pattern in Figure 6 and in all views in Figure 9 in Appendix A.

We present results of evolution runs "Evo #1" and "Evo #2" that were set in the same way as for the selective forwarding attack detection. Also in this case, the evolution found results that dominate the results found by the *Sampling Pareto front approximation*. Only $13,539$ simulation runs had to be executed for "Evo #1" that is about nine times less than for the "sampling" and $23,953$ simulation runs were required for "Evo #2" (about five time less than for sampling). For detailed results, see Appendix A.

[6]These sensor nodes receives, at extreme case, packets only from one descendant. Such traffic can be overheard by less (if any) number of neighbors comparing to a sensor node placed closer to the BS receiving packets from several directions.

[7]We extracted 139 non-dominated solutions out of all $122,472$ samples.

	Evo #1	Evo #2	Sampling
Evo #1	–	61% (72/118)	97% (115/118)
Evo #2	79% (112/141)	–	99% (139/141)
Sampling	22% (30/139)	14% (19/139)	–

Table 3: Coverage metric results for the delay attack. For each Pareto front approximation in a row, the values specify the number of found solutions that are not dominated by any solution within the result set of Pareto front approximation specified in the column.

6.2.1 Coverage Metric

Such as for the selective forwarding attack, we found out that evolution outperformed sampling and "Evo #2" provided better results than "Evo #1". Since the observations are similar to observations discussed in 6.1.1, we only summarize all obtained results in Table 3.

7 Conclusion

Selective forwarding and delay attacks are quite simple but effective attacks on WSNs. Especially the selective forwarding attack and its detection have been often discussed by the community during the last decade. Several detection techniques have been proposed, but to the best of our knowledge, none of the works discussed parametrization for a specific application, topology, environment or attacker strategy.

In our work, we propose two highly parametrized detection techniques for selective forwarding and delay attacks and showed the influence of careful parameter settings on the IDS performance. We evaluated the IDSs using extensive simulations and optimized using two approaches: 1) computationally demanding sampling, where we utilized our computational cluster where we computed up to 250 simulations in parallel; and 2) multiobjective evolutionary algorithms that speedup the optimization process in such a way that the results can be computed also on a single computer in reasonable

time. We also discussed various attacker strategies and evaluated the impact of changes in deployment of malicious sensor nodes in the WSN on the IDS performance.

We believe that the approach where we can choose from a set of non-dominated solutions based on current WSN application, security and other requirements anytime after the optimization process can be easily adapted to practical applications. However, the optimization should be performed on a carefully configured simulator with an accurate model of target WSN.

Both proposed detection techniques can be easily combined into a single IDS distinguishing selective forwarding and delay attack.

Acknowledgment

We would like to thank Ludek Smolik, Lukas Sekanina and colleagues from CRoCS for the discussions and suggestions. This work was supported by the Czech research project VG20102014031, programme BV II/2 - VS. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

[And01] D.P. Anderson. BOINC: a system for public-resource computing and storage. In *Proceedings of IEEE/ACM Workshop on Grid Computing*, pages 4–10, 2001.

[ASSS04] F. Anjum, D. Subhadrabandhu, S. Sarkar, and R. Shetty. On optimal placement of intrusion detection modules in sensor networks. In *Proceedings of First International Conference on Broadband Networks*, pages 690–699, 2004.

[CCL04] P. Cheng, C. N. Chuah, and X. Liu. Energy-aware node placement in wireless sensor networks. In *Global Telecommunications Conference, 2004. IEEE GLOBECOM'04*, volume 5, pages 3210–3214, 2004.

[CDWX07] X. Cheng, D. Z. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. In *Wireless Networks*, volume 14, pages 347–355. IEEE Computer Society, January 2007.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[HH08] T. Hoang Hai and E. Huh. Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge. In *Seventh IEEE International Symposium on Network Computing and Applications, 2008. NCA '08.*, pages 325–331, July 2008.

[INR13] INRIA Lille. DOLPHIN project-team. ParadisEO - A Software for Metaheuristics. [Online; accessed 9/8/2015], 2013. `http://paradiseo.gforge.inria.fr/`.

[JSM14] F. Jurnecka, M. Stehlik, and V. Matyas. On node capturing attacker strategies. In *Security Protocols XXII - 22nd International Workshop Cambridge, UK, March 19-21, 2014 Revised Selected Papers*, pages 300–315, 2014.

[KDF07] I. Krontiris, T. Dimitriou, and F. C. Freiling. Towards intrusion detection in wireless sensor networks. In *Proc. of the 13th European wireless conference*, 2007.

[KLC06] R. Khanna, H. Liu, and H. H. Chen. Self-organization of sensor networks using genetic algorithms. In *IEEE International Conference on Communications, 2006. ICC'06*, volume 8, pages 3377–3382, 2006.

[KLC07] R. Khanna, H. Liu, and H. H. Chen. Dynamic optimization of secure mobile sensor networks: A genetic algorithm. In *IEEE International Conference on Communications, 2007. ICC'07*, pages 3413–3418, 2007.

[KLC09] R. Khanna, H. Liu, and H. H. Chen. Reduced complexity intrusion detection in sensor networks using genetic algorithm. In *IEEE International Conference on Communications, 2009. ICC'09*, pages 1–5, 2009.

[KSW+08] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. Klein Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating wireless and mobile networks in OMNeT++ – the MiXiM vision. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems*, pages 1–8, 2008.

[KW03] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols, Elsevier*, 1(2):293–315, 2003.

[LCC07] F. Liu, X. Cheng, and D. Chen. Insider attacker detection in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1937–1945, 2007.

[LJT11] Arnaud Liefooghe, Laetitia Jourdan, and El-Ghazali Talbi. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. *European Journal of Operational Research*, 209(2):104–112, 2011.

[MSS+15] V. Matyas, P. Svenda, A. Stetsko, D. Klinec, F. Jurnecka, and M. Stehlik. *Cyber Physical Systems, Chapter 5: WSNProtectLayer -âĂŞ security middleware for wireless sensor networks*, pages 119–162. CRC Press, 2015. ISBN 9781498700986.

[Rap01] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2nd edition, 2001.

[RLG08] R. Roman, J. Lopez, and S. Gritzalis. Situation awareness mechanisms for wireless sensor networks. *Communications Magazine, IEEE*, 46(4):102–107, April 2008.

[SMR+05] A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong. Decentralized intrusion detection in wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, pages 16–23, 2005.

[SMS16] M. Stehlik, V. Matyas, and A. Stetsko. Towards better selective forwarding and delay attacks in wireless sensor networks. In *Proceedings of the 13th IEEE International Conference on Networking, Sensing, and Control*, 2016.

[SSM11] A. Stetsko, M. Stehlik, and V. Matyas. Calibrating and comparing simulators for wireless sensor networks. In *Proceedings of the 8th IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 733–738, Los Alamitos, USA, 2011.

[SSMS14] A. Stetsko, T. Smolka, V. Matyas, and M. Stehlik. Improving intrusion detection systems for wireless sensor networks. In I. Boureanu et al., editor, *Applied Cryptography and Network Security*, volume 8479 of *Lecture Notes in Computer Science*, pages 343–360. Springer International Publishing, 2014.

[SSSM13] M. Stehlik, A. Saleh, A. Stetsko, and V. Matyas. Multi-objective optimization of intrusion detection systems for wireless sensor networks. In P. LiÃš et al., editor, *Advances in Artificial Life, ECAL 2013, Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems*, pages 569–576, Cambridge, MA 02142-1493 USA, 2013. MIT Press.

[TACC09] M. Tiwari, K.V. Arya, R. Choudhari, and K.S. Choudhary. Designing Intrusion Detection to Detect Black Hole and Selective Forwarding Attack in WSN Based on Local Information. In *Fourth International Conference on Computer Sciences and Convergence Information Technology, 2009. ICCIT '09.*, pages 824–828, Nov 2009.

[Tal09] E. G. Talbi. *Metaheuristics – From Design to Implementation*. Wiley, 2009.

[YX06] B. Yu and B. Xiao. Detecting selective forwarding attacks in wireless sensor networks. In *20th International Parallel and Distributed Processing Symposium, 2006. IPDPS 2006.*, pages 8 pp.–, April 2006.

[ZLT01] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: improving the strength pareto evolutionary algorithm. Technical report, Eidgenössische Technische Hochschule Zürich (ETH), 2001.

[ZT99] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.

A Detailed Evolution Settings And Results

We used the NSGA-II algorithm [DPAM02] that was set up in the following way: The population consisted of 200 individuals, evolution ran for 200 generations, probability of multi-point crossover was 0.5 and mutation probability for each parameter was 0.01 for the "Evo #1" and 0.25 for the "Evo #2".

Figure 8: Results for detection of selective forwarding attack found by evolution compared to results of Sampling Pareto front approximation.

In Figure 8 (a), we show that solutions found by evolution are similar or even dominate the solutions found by the *Sampling Pareto front approximation*. Figure 8 (b) clearly shows that evolution can find solutions that strictly dominate solutions found by the *Sampling Pareto front approximation* with regards to the number of false positives and false negatives. In Figure 8 (c), we can see that some of the solutions found by evolution also strictly dominate the solutions found by *Sampling Pareto front approximation* with regards to the number of false positives and consumed memory. Finally, in the view depicted in Figure 8 (d), we can see that the solutions are equally distributed comparing to the *Sampling Pareto front approximation* with regards to the number of false negatives and consumed memory.

Figure 9 shows the three-dimensional objective space of the found non-dominated solutions for the delay attack. Comparing the selective forwarding attack, the found results come with lower number of false positives both for the *Sampling Pareto front approximation* and evolution. To reach comparable number of false negatives, higher memory consumption is required. This demand is caused by higher number of packets stored in the IDS buffer. With regards to the consumed memory, the distribution of found non-dominated solutions come with similar patterns. The views in Figures 9 (b-

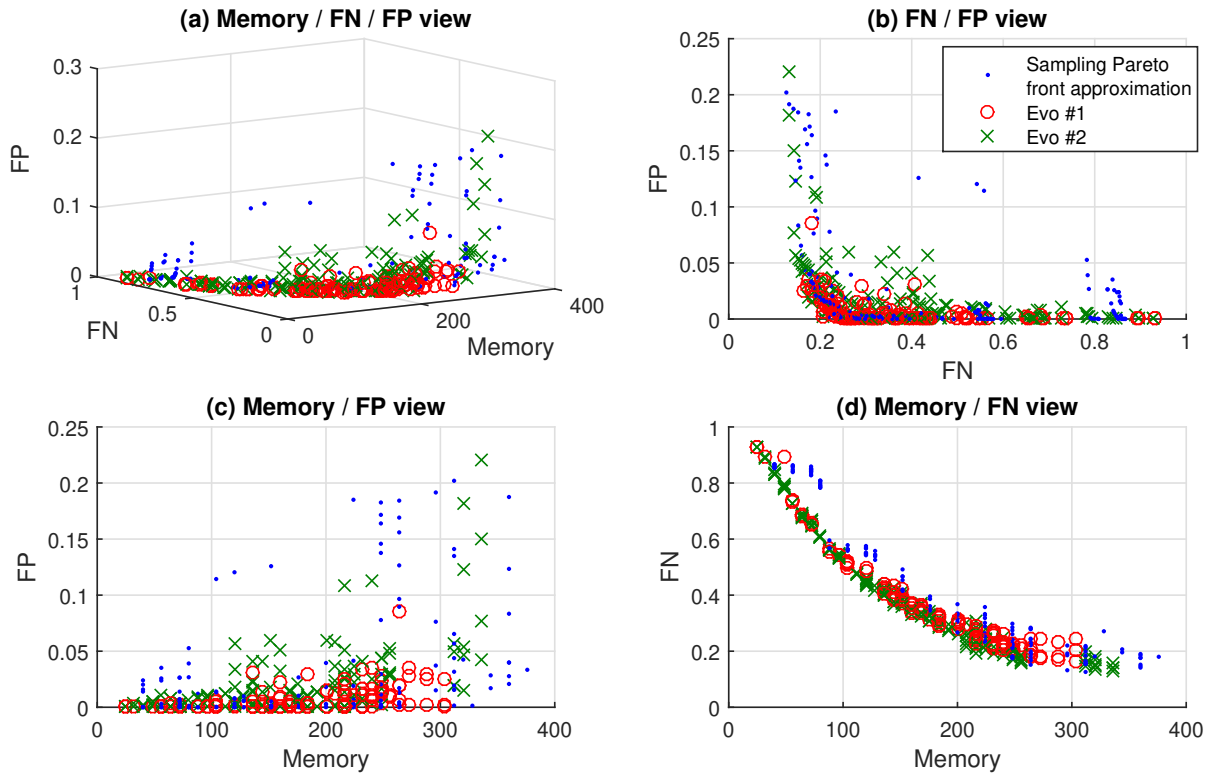

Figure 9: Results for detection of delay attack found by evolution compared to results of Sampling Pareto front approximation.

d) show that the solutions found by evolution are equally distributed comparing to the *Sampling Pareto front approximation*.

B Bug in ParadisEO

As discussed in Subsection 5.1.2, we revealed a bug in ParadisEO – framework for meta-heuristics [INR13, LJT11], the latest version 2.0.1.

The bug was related to wrong evaluation of the individuals for diversification in the population before selection in each generation. The erroneous part was the calculation of the distances to closest neighbors for each individual, because of incorrect sorting. The bug was found in the header file `moeoFrontByFrontCrowdingDiversityAssignment.h` in the function `setDistances()`, line 127: `std::sort(sortedptrpop.begin(), sortedptrpop.end(), cmp2);`. This sorting is performed consecutively for each of the objectives, but should be done only within the same "dominance" front (bounded by variables `a` and `b`). However, all the population is stored in `sortedptrpop` and since all the population is being sorted at line 127, this leads to a

mixing of the individuals across different "dominance" fronts. Each time this sorting is called, it should be performed only on the corresponding part of the population containing individuals with the same dominance rank that is specified by interval ⟨a;b⟩.

We correct the discussed bug by the following:

```
std::sort(sortedptrpop.begin()+a, sortedptrpop.begin()+b+1, cmp2);
```