DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch082

Multi-Objective Optimization of Intrusion Detection Systems for Wireless Sensor Networks

Martin Stehlík[1], Adam Saleh[1], Andriy Stetsko[1] and Vashek Matyáš[1]

[1]Masaryk University, Brno, Czech Republic
xstehl2@fi.muni.cz

Abstract

Intrusion detection is an essential mechanism to protect wireless sensor networks against internal attacks that are relatively easy and not expensive to mount in these networks. Recently, we proposed, implemented and tested a framework that helps a network operator to find a trade-off between detection accuracy and usage of resources that are usually highly constrained in wireless sensor networks. We used a single-objective optimization evolutionary algorithm for this purpose. This approach, however, has its limitations. In order to eliminate them, we show benefits of multi-objective evolutionary algorithms for intrusion detection parametrization and examine two multi-objective evolutionary algorithms (NSGA-II and SPEA2). Our examination focuses on the impact of an evolutionary algorithm (and its parameters) on the optimality of found solutions, the speed of convergence and the number of evaluations.

Introduction

Recent advances in wireless communications and low-cost electronic devices enabled the development of *low-cost* and *high-performance wireless* networking technologies. Apart from widely used *cellular networks* known from mobile phones and *infrastructure local area networks*, there are also *ad hoc networks* operating without any given and fixed infrastructure, where the connections are established on demand in ad hoc manner (Dressler, 2007).

Wireless sensor networks (WSNs) can be considered as a type of ad hoc wireless networks with many specifics. The main difference against the "ordinary" ad hoc wireless networks is that the WSNs consist of a large number of usually *homogeneous*, *low-cost* and *resource restricted sensor nodes*. Their goal is to measure physical parameters like temperature, humidity, intensity of light, and send it to a *base station* (BS) for further processing. Since a node communication range is limited to tens of meters and it is not always feasible for the node to directly communicate with the BS, measurements are usually sent hop-by-hop from one node to another until they reach the BS.

Since WSNs are often deployed in physically open and sometimes even hostile environments, they can be subject to various security attacks ranging from *passive* eavesdropping to *active* interfering (Zhang and Lee, 2000). An active attacker may insert a node in a network, or capture and reprogram an existing one in order to, e.g., drop, delay, modify or reorder packets containing important sensor measurements or routing information (Karlof and Wagner, 2003).

An *intrusion detection system* (IDS) is an essential mechanism to protect a network against internal attacks. Sensor nodes can monitor only a small part of their surrounding. Hence, to enable intrusion detection even in the remote parts of the network, *intrusion detection agents* should be deployed at different nodes in all parts of the network to monitor malicious events *locally*, in a distributed fashion (da Silva et al., 2005; Roman et al., 2006).

An IDS for a WSN should be highly optimized for a given application scenario, i.e., it should not consume more energy (memory) than it is necessary to achieve a required level of detection accuracy. Otherwise, a higher detection accuracy will be at the expense of resources that are highly constrained in sensor nodes. For example, MICAz – a typical sensor node – is equipped with the 8 MHz Atmel Atmega128L microcontroller, 4 kB RAM, 512 KB flash memory, 802.15.4 compliant Texas Instruments CC2420 transceiver and two AA batteries (Crossbow, 2013).

One can expect different requirements in network security and resource consumption for different applications, e.g., one for emergency-response and another for agriculture. Since there is a variety of application scenarios, a single set of IDS parameters is not optimal for all of them. In (Stetsko, 2012), a framework that optimizes the parameters of an IDS for a given application scenario in terms of the detection accuracy and resource consumption was presented. The optimization was driven by multiple objectives that were represented by different evaluation metrics (e.g., number of true positives, number of true negatives, memory usage). A single-objective optimization algorithm was used with the need to provide weights for each objective before the optimization process takes place. If the network operator wants to change weights, the optimization process should run again.

In this work, we examine two multi-objective evolution-

ary algorithms (MOEAs) that eliminate the limitation mentioned above. Our contribution is threefold. First, we show that MOEAs can be utilized for optimization of an IDS in WSNs. Second, we compare effectiveness of NSGA-II and SPEA2 for our scenario that is described in the following sections. Third, we evaluate the impact of MOEA parameters on the optimization process.

Optimization Framework for WSN

In this section, we present our optimization framework that can be used for optimization of various detection techniques in WSNs. It consists of a *simulator* that simulates a specific configurable scenario of a WSN and provides statistics of the simulation. The statistics are input for an *optimization engine* that, based on them, designs new WSN configurations and provides them back to the simulator for further evaluation. The conceptual architecture of the framework is discussed in details in (Stetsko, 2012). In the following subsections, we present our use case that is consistent for all experiments discussed farther in this paper.

Simulator

We evaluate the performance of the IDS using the MiXiM simulator (Köpke et al., 2008) based on the OMNeT++ platform. In the past, we made a thorough comparison of available simulators for WSNs in (Stetsko et al., 2011). The MiXiM simulator provides complex and realistic models suitable for our research. The simulation models are following: *wireless channel* and *network topology* models regarding the global aspects of the WSN and models of *network*, *data link* and *physical* layers and *energy consumption* models of the sensor nodes.

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:

- *Wireless channel model* – An open changing environment is simulated using the *log-normal shadowing* model (Rappaport, 2001) that is the most widely used wireless channel model among the simulators (Stetsko et al., 2011). 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.

- *Network topology model* – Static topology with random uniform distribution of the sensor nodes in 2D square area is used in the experiments.

- *Network layer* – The network layer uses static routing tree generated using the following algorithm. A base station broadcasts a packet containing its identification together with the value h (number of hops to the base station) set to 0. A node waits until it receives a packet from a neighbour that is the closest one (has the highest signal strength). Then the node sets the neighbour as its parent, increases value h by 1 and broadcasts the value together with its identification.

- *Data link layer* – Protocol CSMA-CA according to the IEEE 802.15.4 standard is used.

- *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.

- *Energy consumption* – The energy consumption is not taken into account in this paper because we do not use any sleep mode of the nodes' transceivers that saves the energy in presented detection technique.

Optimization Engine

Various metaheuristics can be used to generate new candidate IDS configurations based on the previous ones evaluated by the simulator. We use *evolutionary algorithms* (EAs) as we found them advantageous in our previous work (Stetsko, 2012). The new generation of candidate configurations is evaluated by the simulator and the process continues until some stopping criterion is fulfilled (in our case predefined number of generations). In this paper, we show how the *multi-objective* approach can be utilized for IDS optimization in WSN. The MOEAs are discussed in section "Optimization Using MOEA". For experiments in this paper, we use a framework for metaheuristics ParadisEO (INRIA Lille, 2013; Liefooghe et al., 2011).

Distributed Computation

Apart from optimization using MOEAs, we decided to perform also *exhaustive search* for all our experiments. Since it is computationally demanding, we use the BOINC distributed computing platform (Anderson, 2001). We expect the network operators to optimize even more complex scenarios, where the exhaustive search would be unfeasible. However, to allow for a thorough comparison of the MOEAs, as one of the goals of this paper, we precomputed all configurations using BOINC on several tens of CPUs.

Intrusion Detection System

In our optimization scenario, we use a detection technique the goal of which is to reveal malicious sensor nodes that execute *selective forwarding attack* where the attacker forwards only a fraction of received packets (Karlof and Wagner, 2003). In this kind of attack, it is assumed that the traffic is routed also through these malicious nodes. These nodes

are supposed to forward the packets received from their children to their parents towards the BS. The intent of the malicious nodes is to filter the traffic and forward only some packets which are selected randomly or based on some criteria (e.g., based on the data content of the packets to suppress information on some specific event in the environment).

Detection Technique

We use a simple but *configurable* technique for detection of selective forwarding attack. Following notations are used in the text to explain the functionality of our IDS:

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 neighbours 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 neighbours 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 neighbours of this node.*

Notation 6 *The function $m : \mathbb{N} \rightarrow \mathbb{N}$ takes a sensor node index as an argument, and returns the amount of memory (in bytes) used by an IDS on this node.*

An IDS is running on a sensor node and continuously analyzing sent and overheard packets. A monitoring node $c_i \in C$ overhears to some extent both incoming and outgoing packets of all close enough monitored neighbours $b_j \in C \cup A$. Note that the set of monitored neighbours is a subset of all neighbours limited to p_1 (*max monitored nodes*). Neighbour of the node c_i is every node $b_k \in C \cup A$, such that c_i overheard at least one packet from b_k during the simulation. An IDS stores 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 a monitored node.

If the IDS on a node c_i overhears a packet P sent to a monitored node b_j and b_j should forward the packet (e.g., b_j is not a base station), then the IDS stores P in the buffer and increments the PR counter of the monitored node b_j. The number of buffered packets is limited by p_2 (*buffer size*). If a new packet arrives but the buffer is full, the oldest packet is removed from the buffer. When the IDS overhears the packet P being forwarded by the node b_j, it removes P from the buffer (if it is still there) and increments the PF counter of the node b_j. Since both the table and the buffer are limited by parameters p_1 and p_2, respectively, the IDS monitors only the closest nodes and the latest packets.

The detection is done at the end of the simulation, based on the collected statistics. The node c_i considers the node b_j as a selective forwarder if the dropping ratio of b_j, i.e., ratio of a number of packets dropped to a number of packets received, is higher than p_4 (*detection threshold*). If the node c_i overheard less than p_3 (*min received packets*) packets received by the node b_j during the simulation, the node b_j cannot be considered malicious by the node c_i because the number of overheard packets is small and there is a high level of uncertainty. In this case, the node b_j is considered benign. Note that the node c_i considers the neighbour node b_k benign if it is not a *monitored neighbour*. To summarize it, the detection decision is based on the following conditions:

- A node b_j is considered *malicious* by the neighbour node c_i if node b_j is the *monitored neighbour* and the observed dropping ratio is higher or equal to the *detection threshold* and the node c_i overheard at least *min received packets* addressed to the node b_j.
- A node b_j is considered *benign* by the neighbour node c_i if node b_j is not the *monitored neighbour* or if the dropping ratio is lower than the *detection threshold* or the IDS overheard less than *min received packets* addressed to the node b_j.

The IDS parameters that we optimize are shown in Table 1. The number of minimum received packets is in the range of $\langle 1, 100 \rangle$ and detection threshold can be set from all packets dropped to all packets forwarded. The other parameters are discussed in the following subsection.

Name	Description	Range	Step
p1	*Max monitored nodes*	$\langle 1, 50 \rangle$	1
p2	*Buffer size*	$\langle 1, 50 \rangle$	1
p3	*Min received packets*	$\langle 1, 100 \rangle$	1
p4	*Detection threshold*	$\langle 0.01, 1 \rangle$	0.01

Table 1: The list of IDS parameters.

Memory Consumption

The detection accuracy is influenced by the amount of memory allocated for the IDS. Each sensor node j requires the following amount of memory (in bytes) for the IDS:

$$m(j) = 8 * p_1 + 16 * p_2, \quad (1)$$

where 8 bytes are required for every monitored neighbour (4 B for node ID, 2 B for PR counter and 2 B for PF counter) and 16 bytes are required for one slot in the buffer (4 B for source address, 4 B for receiver address, 4 B for destination address in a case of multiple BSs in the WSN and 4 B for unique ID of a packet).

We determined the upper bound for the number of nodes being monitored by an IDS agent to 50. Table of neighbours can occupy 400 B ($50 * 8$ B $= 400$ B) at maximum. That would be already 10% of MicaZ RAM (4 kB). Additional memory is needed for the buffered packets. We set the upper bound of the buffer size to 50 because the proof-of-concept experiments showed that bigger sizes did not influence the IDS accuracy. Thus, $50 * 16$ B $= 800$ B can be allocated for the buffer at maximum that is 20% of the RAM.

Optimization Using MOEA

In this section, we show how MOEAs can be utilized for optimization of intrusion detection systems in complex environments of wireless sensor networks.

MOEAs can be useful for optimization of IDSs and other aspects in WSNs, providing the network operators with a set of *non-dominated* solutions. Then the network operator can choose between, e.g., an optimized solution A with better IDS accuracy at the cost of higher memory consumption and another optimized solution B with lower memory consumption at the cost of worse IDS accuracy. The set of non-dominated optimal solutions is called *Pareto front* (Talbi, 2009) and the goal of the MOEAs is to find a good approximation of the true Pareto front.

Several MOEAs producing approximations of the true Pareto front have been proposed. However, a good MOEA should produce solutions close to the true Pareto front with high *diversity* and should *converge* in a relatively short time. The *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) proposed in (Deb et al., 2002) and the *Strength Pareto Evolutionary Algorithm 2* (SPEA2) proposed in (Zitzler et al., 2001) belong to the most widely used MOEAs (Talbi, 2009) and we compare them in this paper. Both algorithms are implemented in the evolutionary framework ParadisEO and also in MOGAlib (MOGALib, 2013).

NSGA-II features two main criteria to provide good *convergence* and *diversification*, respectively: 1) *ranking using non-dominance concept* to sort the solutions according to the number of other solutions they are dominated by; and 2) *crowding distance* to keep the solutions spread as far from each other as possible.

The fitness values calculated for the solutions found by SPEA2 are based on the number of dominating solutions and their strength of dominance (to achieve *convergence*) and on the density estimation function (to achieve *diversification*).

Objective Space

In the optimization of our IDS, we consider three following objective functions: number of false positives fp, number of false negatives fn, amount of consumed memory mem.

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

$$fn(x) = \frac{1}{|A|} * \sum_{a_i \in A} \frac{x(a_i)}{n(a_i)}. \quad (2)$$

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

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

$$fp(x) = \frac{1}{|C|} * \sum_{c_i \in C} \frac{y(c_i)}{n(c_i)}. \quad (3)$$

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

Objective function 3. The consumed *memory* (mem) in a solution x is averaged over all benign nodes in the WSN as follows:

$$mem(x) = \frac{1}{|C|} * \sum_{c_i \in C} m(c_i), \quad (4)$$

where $m(c_i)$ is calculated using formula 1.

The values of mem range from 0, where the IDS is potentially switched off, to $8 * p_1 + 16 * p_2$ that is 1200 bytes for our upper bounds of $p_1 = 50$ and $p_2 = 50$.

All three objectives are *minimized*.

Pareto Front Discussion

The true Pareto front of a *sparse topology* (discussed in the next section) found by exhaustive search is shown in Fig. 1. The goal of MOEA is to find IDS configurations that are close to the points of the true Pareto front as much as possible. In Fig. 1 (a), the view of the whole objective space is shown from the perspective of mem, fp and fn. Fig. 1 (b) depicts the trade-off between fp and fn, where the resulting values depend on the *detection threshold* of the IDS (parameter p_4). Fig. 1 (c) shows a slight increase of fp (see scale) with higher number of *monitored nodes* (parameter p_1 influencing mem). The fp increase is caused by the fact that an IDS monitors a higher number of legitimate neighbours, each of which may be falsely considered malicious. Finally, Fig. 1 (d) shows a rapid decrease of fn with a higher number of *monitored nodes* (parameter p_1) that is caused by the fact that a higher number of neighbours (and hence malicious nodes) is monitored (detected).

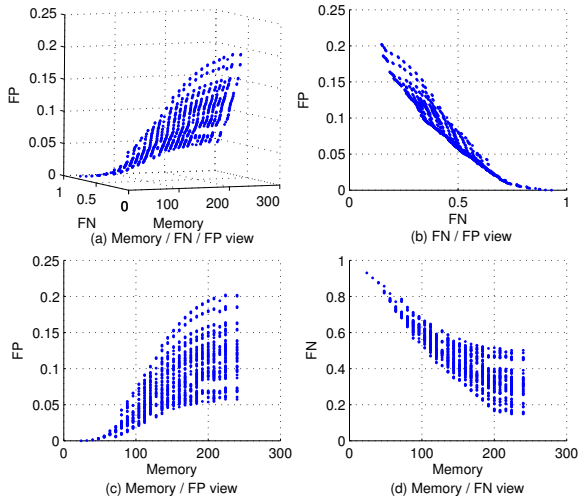

Figure 1: Optimal solutions on the Pareto front in our *objective space*.

Comparison Methodology

In this section, we present a methodology that we used to compare NSGA-II and SPEA2 using multiple settings. We tried different configurations (see Table 2) of the algorithms on our problem and compare the outcome according to four metrics that are defined further in the text.

Evolution Parameters

The evolutionary algorithms have several parameters influencing the evolution process. The setting of the parameters is presented in this subsection.

Initial population. The initial population consists of randomly generated individuals (more specifically, the values of the IDS parameters are generated randomly within the predefined range of values).

Population size. The size of the population $PopSize$ is set to following values: $PopSize \in \{50, 100, 200\}$.

Crossover. The multi-point crossover operation is applied with the probability $PCross \in \{0.01, 0.1, 0.25, 0.5\}$.

Mutation. The mutation operation is applied with probability $PMut \in \{0.01, 0.1, 0.25, 0.5\}$ to every parameter $p_1, ..., p_4$ separately. When applying mutation, the parameter is changed randomly within an interval around the previous value of that parameter covering 10% of the overall parameter range (5% in both directions).

Number of generations. The number of generations $NGen$ is set to $NGen = 200$.

Having the setting of the evolutionary parameters specified above, the IDS parameters were optimized with $|PopSize| * |PCross| * |PMut| = 3 * 4 * 4 = 48$ different settings of both algorithms NSGA-II and SPEA2.

Metrics

Following notations are used to define metrics for the comparison of the algorithms:

Notation 7 *The set P is a set of all vectors (solutions) on the true Pareto front, $\mathbf{p} \in P$ is a Pareto optimal vector and $p = |P|$ is a size of the set P.*

Notation 8 *The set N is a set of all vectors (solutions) found by the evolution, $\mathbf{n} \in N$ is a non-dominated vector found by the evolution and $n = |N|$ is a size of the set N.*

Since we calculate the Euclidean distance between the solutions in the objective space in Metric 2 and Metric 3 defined below, we "normalize" the amount of the consumed memory in the following way:

Notation 9 $memn(x) = mem(x)/1200$, *where* 1200 *is the maximal amount of consumed memory in bytes and the range of the function* $memn$ *is* $< 0, 1 >$.

The MOEAs maintain an *archive* of the found non-dominated solutions. Two following aspects are expected of the solutions kept in the archive:

- *Convergence* – The approximation of the Pareto front should converge to the true Pareto front with new generations.
- *Diversification* – The found solutions should be uniformly distributed in the objective space in ideal case.

To be able to measure the effectiveness of the differently set MOEAs with respect to the aforementioned performance aspects, we use the following metrics to measure *convergence*:

Metric 1. The value of M_1 is the number nd of non-dominated solutions $\mathbf{nd} \in P$ found by the MOEA. The complexity of the calculation of this metric is $O(nd * p)$. This metric is used to measure *convergence*.

Metric 2. The value of M_2 (*generational distance metric*) is the average of Euclidean distances from all found solutions $\mathbf{n} \in N$ to the nearest solution $\mathbf{p} \in P$ on the true Pareto front (Deb et al., 2002). This metric is also used to measure *convergence*. Having n solutions and p Pareto dominant solutions, the complexity of the calculation of this metric is $O(n * p)$.

Several metrics can be used to measure *diversification*, yet with an assumption that it is straightforward to find a neighbouring solution in the objective space. However, the definition of the neighbouring solution is easy for a two-dimensional objective space, but much more complicated for three- or more-dimensional objective spaces. In our case,

we have three objective functions. Thus, we use the following metric for diversification specified by Schott in (Schott, 1995) as *Spacing metric*. Note that the calculation of this metric does not require the set of the true Pareto front P found in the exhaustive search:

Metric 3. *Diversification* is measured using M_3 as follows: $M_3 = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d}-d_i)^2}$, where $d_i = min_j\{|fn(i)-fn(j)|+|fp(i)-fp(j)|+|memn(i)-memn(j)|\}$ and $\overline{d}$ is an average of all distances $d_i, i \in \{1,...,n\}$. If $M_3 = 0$, the solutions are spaced equally in the objective space. Having n solutions, the complexity of the calculation of this metric is $O(n^2)$.

We also calculate the time requirements of the MOEA, where the runtime of one simulation needed to evaluate a single individual in a population is the most time consuming element (around 8 minutes for our *Sparse topology* discussed farther). Note that the overall time is also dependent on the number of available CPUs:

Metric 4. M_4 is the number of simulations needed in the whole evolution process.

We compare the number of individual evaluations during MOEA with number of evaluations needed for the exhaustive search for each of the experiments. In the exhaustive search, we evaluated $25,000,000$ IDS configurations using our BOINC distributed computation platform.

Our Test Case

We evaluated the performance of the NSGA-II and SPEA2 in two following different simulation scenarios consisting of 250 sensor nodes and 1 BS in 1) *sparse* and 2) *dense* topology. In both topologies, the goal of the IDSs was to detect five malicious nodes. Their placement can be found in (Stehlik et al., 2013).

- **Sparse topology** (Topology #1) – The sensor nodes are placed in the area of 200 x 200 m. The average area for one node is $160\ m^2$, i.e., the distance between two nearest neighbours is 12.65 m in average.
- **Dense topology** (Topology #2) – The sensor nodes are placed in the area of 100 x 100 m. The average area for one node is $40\ m^2$, i.e., the distance between two nearest neighbours is 6.33 m in average.

Results and Discussion

In this section, we discuss the results of NSGA-II and SPEA2 configured in different ways. Evolutionary algorithm is a *stochastic* process – it means that multiple runs should be done for each configuration to get average behaviour of the algorithm. We ran the evolution 10 times for all the settings and we provide the *average* value Avg_x and *standard deviation* σ_x for every metric M_x computed from results obtained for all ten evolution runs. Since we are limited in space, detailed results can be found in (Stehlik et al., 2013), as well as IDS settings of Pareto optimal solutions.

Table 2 shows how the results of the metrics using different set of evolution parameters are numbered in the charts presented in this section. MOEA settings used for sets $1,...,16$ ($PopSize = 50$) are presented in the Table 2. Sets No. 17-32 and 33-48 have analogous MOEA settings for $PopSize = 100$ and $PopSize = 200$, respectively.

No.	1	2	3	4	5	6	7	8
$PCross$	0.01	0.01	0.01	0.01	0.1	0.1	0.1	0.1
$PMut$	0.01	0.1	0.25	0.5	0.01	0.1	0.25	0.5

No.	9	10	11	12	13	14	15	16
$PCross$	0.25	0.25	0.25	0.25	0.5	0.5	0.5	0.5
$PMut$	0.01	0.1	0.25	0.5	0.01	0.1	0.25	0.5

Table 2: The MOEA parameters settings for $PopSize = 50$.

Sparse Topology

The lowest number of neighbours is characteristic for the *sparse topology*. A node $b_j \in C \cup A$ has 41 neighbours in average. Hence, a lower amount of memory is needed to achieve IDS accuracy comparable to that in *dense topology*.

Exhaustive Search The true Pareto front obtained for the *sparse topology* is shown in Fig. 1. The results of the exhaustive search showed that $2,340$ IDS configurations (out of $25,000,000$) are Pareto dominant resulting in 996 unique solutions in the objective space (some IDS configurations have same results).

Convergence Fig. 2 shows solutions found by a single run of the MOEAs for set No. 45 that provided good results in a relatively short time. We consider a case where NSGA-II found 121 mutually non-dominated solutions, where 29 were Pareto optimal. SPEA2 found 80 mutually non-dominated solutions, where 20 were Pareto optimal. Objective space for evolution set No. 48, where solutions found by SPEA2 are spread better, can be found in (Stehlik et al., 2013).

In Fig. 3 (a) and (b), the impact of the evolution parameters on the convergence is shown. NSGA-II found more solutions on the Pareto front (metric M_1) in most cases than SPEA2. Nevertheless, SPEA2 has better results measured by metric M_2. This is caused by NSGA-II solutions that have redundant amount of consumed memory remained in the population archive (e.g., solutions on the right-top corner in Fig. 2). These solutions were not dominated by other solutions with lower fn, fp and mem during the evolution.

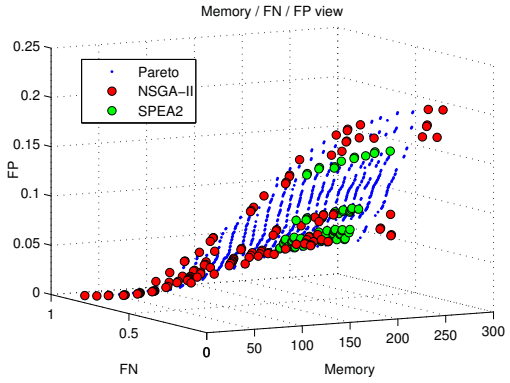

Figure 2: Solutions found by MOEAs for sparse topology using evolution parameters No. 45.

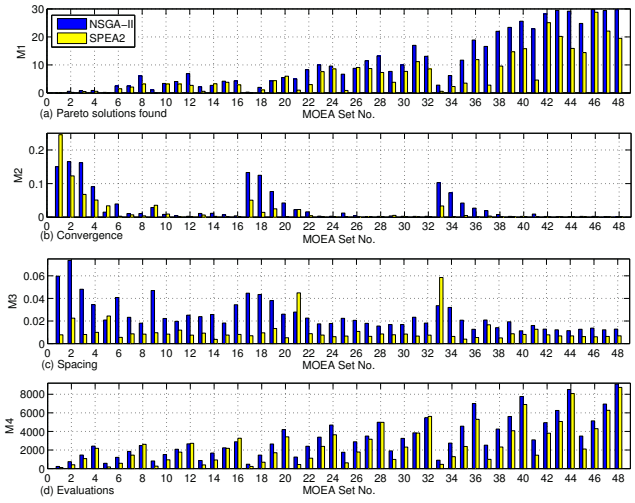

Figure 3: Results of metrics for sparse topology.

We found out that higher crossover probability has a higher impact on the speed of convergence than mutation probability from the perspective of metric M_2. See quartets in Fig. 3 (b). It is possible to obtain good results with high crossover probability and low mutation probability, but much more difficult if the crossover probability is very low. However, both parameters, as well as the population size, have a positive impact on the convergence.

Diversification There is a difference between *diversification* of NSGA-II and SPEA2 for our problem. Fig. 3 (c) suggests that the solutions are spread better using SPEA2. However, checking the objective space (Fig. 2) provides additional information on spreading of the solutions within the whole objective space that is better for NSGA-II. Note that SPEA2 found better spread solutions if mutation probability is higher.

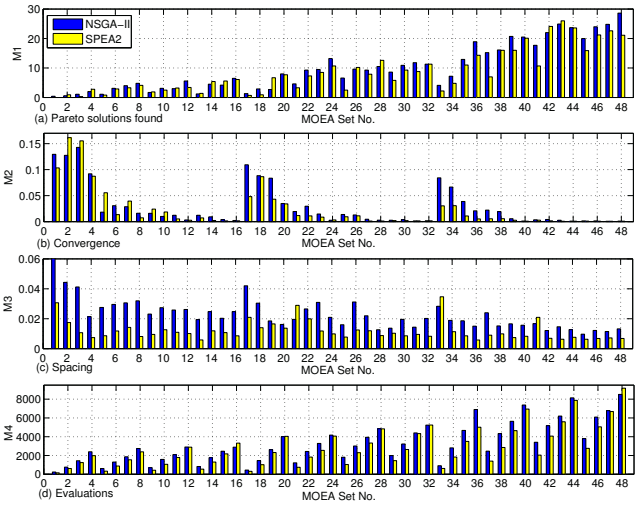

Figure 4: Results of metrics for dense topology.

Evaluations The number of simulations needed to evaluate the individuals in the population using a simulator are depicted in Fig. 3 (d). All MOEA parameters influence the number of evaluations, but $PMut$ and $PopSize$ have a higher impact than $PCross$ (see, e.g., sets No. 36, 40, 44 and 48 with different values of $PCross$ vs. sets $33 - 36$ with different values of $PMut$).

Note that, e.g., set No. 45 required only $3,497$ simulations for NSGA-II and $2,101$ simulations for SPEA2 in average. 24.8 and 14.4 Pareto dominant solutions were found for NSGA-II and SPEA2, respectively. Since the solutions cover different parts of the Pareto front (especially for NSGA-II), the evolution is much more efficient than evaluation of $25,000,000$ configurations in case of exhaustive search.

Dense Topology

In the *dense topology*, a node $b_j \in C \cup A$ has 127 neighbours in average.

Exhaustive Search The true Pareto front obtained for the *sparse topology* can be found in (Stehlik et al., 2013). The results of the exhaustive search showed that $20,072$ IDS configurations (out of $25,000,000$) are Pareto dominant resulting in $2,219$ unique solutions in the objective space.

Convergence, diversification and evaluations Results for evolution sets No. 45 and 48 can be found in (Stehlik et al., 2013). Similarly to the *sparse topology*, optimized solutions are spread better for SPEA2 in case of evolution set No. 48. The experiment showed similar characteristics of performance based on evolution settings as for *sparse topology*. The results of all performance metrics are shown in Fig. 4.

Conclusion

In this work, we extended our *optimization framework* to utilize MOEAs in the process of IDS parametrization. The multi-objective approach is beneficial since it provides the network operator with a set of optimized solutions. He/she can select a solution according to the purpose of the WSN and change the IDS settings according to current needs. We showed that the knowledge of the Pareto optimal solutions is advantageous. However, the computation using exhaustive search is extremely time-demanding. MOEAs proved to be a good compromise between the quality of Pareto front approximation and the optimization time.

We compared two widely used MOEAs: NSGA-II and SPEA2. The results suggest that NSGA-II might be better for our needs. However, one should be careful with a definite conclusion. Various metrics, as well as visualization of the objective space, provide different views of the algorithm performance.

We also focused on the impact of MOEA parameters on the speed of convergence, number of evaluations and quality of Pareto front approximations. Higher population size provides better results at the cost of higher number of evaluations. We found out that a higher crossover probability does not increase the number of evaluations as much as a higher mutation probability and has a better impact on the quality of Pareto front approximations.

In the future, we plan to optimize techniques to detect other attacks than selective forwarding. We would also like to extend the optimization framework to design robust solutions in complex environments. Finally, we plan to adapt and use our framework for the optimization of a whole network stack.

Acknowledgments

We would like to thank Lukas Sekanina for his invaluable advice. This work was supported by the GAP202/11/0422 project of the Czech Science Foundation.

References

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

Crossbow (2013). MicaZ Datasheet. [Online; accessed 4/12/2013]. http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=164.

da Silva, A. P. R., Martins, M. H. T., Rocha, B. P. S., Loureiro, A. A. F., Ruiz, L. B., and Wong, H. C. (2005). 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.

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

Dressler, F. (2007). *Self-Organization in Sensor and Actor Networks*. John Wiley & Sons.

INRIA Lille (2013). DOLPHIN project-team. ParadisEO - A Software for Metaheuristics. [Online; accessed 4/12/2013]. http://paradiseo.gforge.inria.fr/.

Karlof, C. and Wagner, D. (2003). 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.

Köpke, A., Swigulski, M., Wessel, K., Willkomm, D., Haneveld, P. T. K., Parker, T. E. V., Visser, O. W., Lichte, H. S., and Valentin, S. (2008). 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.

Liefooghe, A., Jourdan, L., and Talbi, E.-G. (2011). A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. *European Journal of Operational Research*, 209(2):104–112.

MOGALib (2013). MOGALib – MuleGA. [Online; accessed 4/12/2013]. http://mulega.uni-pannon.hu/index.php/MOGALib.

Rappaport, T. (2001). *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2nd edition.

Roman, R., Zhou, J., and Lopez, J. (2006). Applying intrusion detection systems to wireless sensor networks. In *IEEE Consumer Communications & Networking Conference (CCNC 2006)*, pages 640–644, Las Vegas (USA).

Schott, J. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics.

Stehlik, M., Saleh, A., Stetsko, A., and Matyas, V. (2013). Multi-Objective Optimization of Intrusion Detection Systems for Wireless Sensor Networks – webpage. [Online; accessed 4/14/2013]. http://www.fi.muni.cz/~xstehl2/ECAL/.

Stetsko, A. (2012). *On intrusion detection in wireless sensor networks*. PhD thesis, Masaryk University.

Stetsko, A., Stehlik, M., and Matyas, V. (2011). 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.

Talbi, E. G. (2009). *Metaheuristics - From Design to Implementation*. Wiley.

Zhang, Y. and Lee, W. (2000). Intrusion detection in wireless adhoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 275–283, New York, NY, USA. ACM.

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