MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Evolutionary optimization of intrusion detection system in wireless sensor networks

DIPLOMA THESIS

Adam Saleh

Brno, spring 2008

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Adam Saleh

Advisor: Andryi Stetsko, Ph.D.

Acknowledgement

I would like to thank my supervisor ...

Abstract

Wireless Sensor networks are an interesting topic for security research because of their possible real world applications, their distributed nature and the constrains on their hardware. For example, setting up intrusion detection system on network of nodes so that it satisfies constrains on accuracy and hardware can be a timeconsuming. We believe that this process could be greatly improved by using genetic algorithms. Purpouse of this work is to provide a reasonably reusable framework for optimizations with evolutionary heuristics and calibrate it for optimizing of Intrusion Detection System on Wireless Sensor Network.

Keywords

Wsn,ids,spea2,nsgaii

Contents

1 Introduction

2 Evolutionary Optimization of Wireles Sensor Networks

Optimizing configuration of a wireless sensor network can be a time-consuming task. Nodes of the network are constrained devices, so tradeoffs between battery life, network, memory and processor usage have to be considered. Because it is difficult to to predict how will different configuration settings change the performance of the network, it is advisable to measure the impact of changing configuration with WSN simulation software. Unfortunately, simulation of such configuration can take anywhere between several minutes to several hours of processor time, rendering usage of simple exhaustive search on all configurations impractical for most WSN calibration purposes.

According to (b Talbi), optimization problems with problem space that is hard to characterize and expensive to search are good fit for heuristic approaches.

[copy figure 1.9 from talbi]

Definition 1 (WSN Optimization problem)**.** *To be more formal*[1]*, we define our optimization problem as a tuple* (S, f)*, where* $S \subset R^{n_i}$ *is the set of posible input parameters and* $f : S \rightarrow R$ *is the evaluation function. Solution* $s \in S$ *of problem* (S, f) *is then the* n_i*-tuple of input parameters with minimal value of* $f(s)$

Most of the heuristics used are variants on search through problem space, with different approaches to avoid undesirable local optima.

There are two basic principles that are used when designing a metaheuristic:

Diversification: an algorithm should explore the search space.

Random search can be considered an extreme application of difersification, where in every round we generate next solution randomly without using any information from previous (good) solutions.

---

1. and to maintain consistency in further chapters

Less extreme example is the family of population based heuristics, where in each iteration we are trying to improve a set of candidates, called a population.

Intensification: an algorithm should exploit information from the best solutions found.

Local search can be considered an extreme application of intensification, where next instance to evaluate is allways selected deterministicaly, based on the previous solution.

Local search and its variants belong to a family of single-solution based heuristics, where in each iteration we are basing our search of the problem-space of single solution. (b talbi)

Because of their nature, heuristic approaches are hard to evaluate and reason about, therefore they are often considered to be a method of last resort. (b Talbi) warns against using heuristics laizily on problems where more deterministic approaches (such as using a SAT-solver or linear programming) would be more efficient. Heuristic approaches are usually considered whe we need to solve our problem in order of magnitude faster than possible using conventional approaches, while sacrificing guarantees on optimality.

In our case, we are constrained by number of configuration evaluations we are able to perform in a given timeframe.

We are focusing on evolutionary heuristics, because they seem to have reasonable performance with regard of optimizing WSN (b Stetsko 2011). These are modeled natural proces of evolution of species. It is a family os stochastic, population-based heuristics (as opposed to deterministic, single-solution based).

2.1 Single objective evolution algorithms

Basic principles behind evolutionary algorithms can be shown on example of simple evolutionary algorithm.

Evolutionary algorithms are based of the fact, that natural proces of evolution can be considered an algorithm solving an optimization problem of adapting a species to its environment.

2.1.1 Template of an evolutionary algorithm. (b Talbi)

```
Generate(P(0)) ;
t=0;
While not Termination Criterion(P(t)) Do
Evaluate(P(t));
P'(t)= Selection(P(t)) ;
P'(t)= Reproduction(P'(t));
Evaluate(P'(t)) ;
P(t+1) = Replace(P(t),P'(t)) ;
t =t+1;
End While
Output Best individual or best population found.
```

If we continue with this metaphor, we can find these paralels:

- Population of individuals is a set of configurations. Their environment is the optimization problem.
- We decide surviving individuals based on their evaluation function score.
- From surviving individuals we generate next generation, either by mutation or crossover.
- Mutation of individual usualy consists of randomly selecting some configuration from its neghbourhood as its ofspring. This step provides intensification for evolution heuristic.
- Crosover usualy consists of creating an offspring by permutation of two individuals. This step provides diversification for evolution heuristic.

In our framework selection and replacement steps are part of the evolutionary algorithm, while crossover and mutation functions are to be calibrated on case by case basis.

2.2 Multiobjective evolution algorithms

Problem that we are trying to optimize is unfortunately illsuited for single objective evolution. Objectives such as memory consumption

and IDS accuracy are ortoghonal, therefore it is hard to determine single criteria that would satisfactory cover both of them. A usual solution is to provide some sort of weighted average. In this case,because output of our algorithm would be only a single solution, we need to know emphasis on different criteria before we run our optimization.

Better approach is to recognize multi-objective nature of our problem.

Definition 2. *Multiobjective optimization problem A multiobjective optimization problem may be defined as:*

$$MOP =$$

Without the loss of generality, we assume minimization of all objectives

Multi objective algorithms on the other hand are based on idea of Pareto optimality and domination. We say that solution A dominates solution B, if A has no objective "worse" than B and at least one objective "better". We say that A is pareto-optimal, if there is no other solution that would dominate A. This means that no objective of A can be improved without deterioration of another objective. Set of all non-dominated solutions is then called Pareto front.

Subset of solutions where no solution from the superset dominate the one in the set is called Pareto optimal set.

We will call a subset that contains only solutions that don't dominate each other a Pareto aproximation.

Definition 3. *Pareto dominance An objective vector $u = (u_1, \cdots, u_n)$ is said to dominate $v = (v_1, \cdots, v_n)$ (denoted by $u \prec v$) if and only if no component of v is smaller[2] than the coresponding component of u and at least one component of u is strictly smaller:*

$$\forall i \in \langle 1, n \rangle : u_i \leq v_i \wedge \exists i \in \langle 1, n \rangle : u_i < v_i$$

Definition 4. *Pareto optimality A solution $x \in S$ is Pareto optimal if for every $x' \in S$, $F(x')$ does not dominate $F(x)$, that is*

$$\forall x' \in S, F(x) \not\prec F(x')$$

2. we assume minimization

Definition 5. *Pareto optimal set For given MOP(F,S), the pareto optimal set is defined as* $\mathcal{P}* = \S \in \mathcal{S} |\ \nexists \S' \in \mathcal{S}, \mathcal{F}(\S') \prec \mathcal{F}(\S)$

Definition 6. *Pareto front For given MOP(F,S), the pareto optimal set is defined as* $\mathcal{P}* = \S \in \mathcal{S} |\ \nexists \S' \in \mathcal{S}, \mathcal{F}(\S') \prec \mathcal{F}(\S)$

Definition 7. *Pareto aproximation For given MOP(F,S), we call set* $A \subset S$ *a pareto aproximation if* $\forall x \in A\ \nexists x' \in A, F(x') \prec F(x)$

Multiobjective heuristics then try to obtain best aproximation of Pareto front. To gauge how good an approximation of Pareto front is, we usualy use two criteria, first to measure convergence to Pareto optimal front and second to measure diversity in found pareto set.

[image about approximations]

Calibration of multi objective evolution algorithms is harder than their single criteria counter-parts, mainly because it is hard to reason about the convergence of output without having the real Pareto optimal front in the first place. Pareto optimal sets often aren't directly comparable. TODO We will discuss calibration in greater detail in later chapters.

2.2.1 NsgaII

In Non-dominated Sorting Genetic Algorithm, the Replace step is performed first by sorting union of old $P(t)$ and new $P'(t)$ in into ranks based on the ordering of pareto dominance. then sorting among members of each rank is performed based on similarity beween individuals. Best individuals are then chosen for the next iteration.

We can see that first sorting helps with general convergence to real Pareto front, while second improves diversity in the nondominate part of the population. To get the result we just need to extract non-dominated results from the final population.

2.2.2 Spea2

In Strength Pareto Evolutionary Algorithm, unlike previous NSGAii, stores non-dominated indivituals in archive separate from the rest of the population. Therefore after new population is generated, new contents of archive are determined in two passes, first using procedure to ensure convergence and then procedure to ensure diversity.

1. union of nondominated archive and current population is created. For each individual, algorithm computes
   (a) strength based on number of individuals evaluated individual dominates
   (b) raw fitness, that is determined by the sum of strengths of individuals that dominate the evaluated one [3]
   (c) density information
   (d) agregated fittness
2. new archive of nondominated solutions is created from the union
3. (a) if new archive has less individuals, than its desired capacity,
   (b) best remaining individuals based on agregated fittness will ba added
   (c) if archive has more individuals than its desired capacity,
   (d) remove some of them from the archive based on density information

The final result then contains just the nondominated individuals of the archive.

2.2.3 Ibea

Both NsgaII and Spea2 differ mostly in their aproach to characterize the convergence to Pareto front and diversity. Indicator-Based evolutionary algorithm, takes more abstract approach, based on a concept of binary quality indicators. Indicator is a function, that takes two pareto sets of the same domain and outputs some quantification of a difference between the two.

Zitzler and Kunzli proposed two indicators,

1. additive ϵ-indicator that quantifies the minimal distance first pareto set has to be moved in each dimension in objective space such that the second pareto set is weakly dominated.

3. this criterion is to be minimized

2. hypervolume-distance indicator, that quantifies how much volume is dominated by the first pareto set but not dominated by the second, with respect to predefined reference point Z.

Both of these indicators are dominance preserving.

Because any single individual can be considered a pareto set of size one, IBEA uses given indicator to quantify fitness of an individual.

2.2.4 Choosing the algorithm

Design of both of these algorithms give some guarantees on convergence to Pareto front. Both of them are incremental improvements on their predecessor, and while Spea2 is newer, it doesn't mean it is better in every instance. It has to be noted, that even comparison between these two algorithms with regards to optimizing IDS for WSN (Stehlik 2013) came out inconclusive.

Of the more interesting concepts that IBEA uses is the hypervolume distance indicator, which might provide an interesting alternative to more popular NSGA and SPEA, if provided with good reference point. We will tallk more about hyper-volume as a metric of pareto set convergence in chapter on calibration.

We would advise to focus on calibration only one algorithm.

3 Our Framework

(b Stetsko 2011) showed, that using simple evolutionary algorithms is a viable and efficient way of optimizing Wireless Sensor networks. Unfortunately, prevoius experimental setup was too tightly coupled with optimization framework, rendering the code of previous experiments hard to reuse. Therefore we have created a micro-framework based on python and ParadisEO.

ParadisEO is a highly configurable framework for metaheuristics written in C++. Our micro-framework simplifies its usage by allowing to specify evaluation, mutation and crossover functions in python. Emphasis is on multiobjective algorithms, ability to evaluate population in parallel and having good facilities for experiment analysis.

We chose python, for these reasons:

1. it is an efficient glue language. Most often you will want to optimize already written application with minimal changes. Python is well suited for running external applications, specifying their inputs and parsing their outputs.

2. it has good facilities for statistical analysis. That allows us to include analysis of every optimization run into the executable itself. Having automatically generated experimental log has proven invaluable especialy in callibration of algorithms for our spepcific problem.

3. it is easily integrated with C++. We strive to make the integration with ParadisEO simple and extensible.

3.1 Evaluation function

To simplify configuration, inputs and outputs of evaluation function are allways a list of floating point numbers. We believe this is general enough, and that most WSN optimisation problems can be fitted to this constraint. Only problem might pose translation of combinatorial problems to continuous ones, fortunately so far we have been quite successfull with using just a simple rounding techniques. We

wanted to avoid usage of binary strings as input vectors. Many evolutionary algorithms support them and if we were using them we wouldn't need to concern ourselves with implementation details of mutation and crossover functions. On the other hand, with generalized mutation functions on binary strings, there are no guarantees that mutated individual will even be valid. CITATION In our opinion, vectors of reals are esier to reason about, and their convergence may provide valuable insight to structure of the problem.

3.1.1 Bounds

For inputs you specify their bounds, which are then used to initialize the population and to supply constraints for mutation and crossover parameters

[Example]

3.1.2 Objectives

Because we focus on multiobjective evolution, you need to specify number of objectives. To simplify configuration, all the parameters will be minimized.

[example]

It is advisable to have three objectives at maximum, because number of solutions on the Pareto optimal dramaticaly increase with new objectives. At minimum, $(n+1)$ objective problem will contain all the solutions of an n-objective problem. [CITATION? 309 talbi]

Then there is the problem of visualizing and evaluating more than 3-dimensional data.

3.1.3 Boinc Assisted Evolution

Because evolution is population-based heuristic, it is well suited for parallel evaluation of its individuals. ParadisEO already has two methods included, either by use of MPI protocol [CITATION], or by using SMP on multi-core machine. Unfortunately, SMP module of ParadisEO is not yet stable on all platforms, and MPI has specific demands on infrastracture and is usually hard to retro-fit on already written program.

Boinc on the other hand can simply utilize machines already preasent to form a simple computational grid. With its simple archhitecture consisting of Management server and several worker nodes it can be deployed in most settings. Its simple architecture prohibits co-operation between worker nodes, but that doesn't concern us while evaluating individuals.

We have decided to give the evolution algorithms ability to use a simple scheduler for creating Boinc work units, plugable from python.

In our micro-framework it consists of three functions, schedule_work_unit, wait_for_completion and gather_result.

Even though the evaluations themselves will be done in parallel, we wanted to avoid any threaded code in our framework itself. Because we have written it with research in mind, we understand that accessing auxiliary data is important to evaluate experiments. In threaded environment, this could lead to deadlocks.

3.2 Calibration

We can think of calibration on an evolutionary algorithm as a meta-optimization problem with two objectives,

1. maximizing the fittness of result

2. minimizing the number of evaluations

With single-objective algorithms, optimizing for these two objectives is easy, because both of them are usualy represented by numbers. With multi-objective problems, resuls are Pareto sets, which are often not directly comparable.

Quality of a multiobjective result can be judged based on several criteria:

1. number of results in Pareto set.

2. distance from true Pareto front

3. regularity in distance beween results

Because the main reason we use multi objective algorithms is to avoid the need to do the expensive computation of true pareto front, we could either calibrate on a smaller sample of the problem

Underlying assumption behind sampling is, that the Pareto front of the solutions of the problems subset will be the subset of true pareto-front.

We could use different convergence criterion. One of the more popular is maximizing the hypervolume of the objective space dominated by the resulting approximation. Hypervolume indicator is based on idea of measuring the volume that a given individual dominates based on some reference point in the solution-space. This works with assumption that the solution-space is homogenous, compact, and that we can specify a reasonable reference point. A good reference point is usualy the worst individual [CITATION NEEDED] and because because we usualy don't know how will this individual look like, by guessing this reference point we are expresing certain biases (or expectations) on the shape of the solution space as well.

3.2.1 Population size and Number of generations

These two parameters set the boundaries on minimal and maximal number of evaluations. Lets mark the size of population N and number of generations G Because the whole initial population needs to be evaluated, number of evaluations will at least the size of populations. Because at each new generation, at most N new individuals are generated, therefore in one optimization run, at most $N \times G$ evaluations.

3.2.2 Mutation and Crossover

Probability of mutation, or crossover directly influence number of generated individuals in each generations.

4 Experiments

To test our framework we have decided to optimize a simple intrusion detection system on a model wireless sensor network.

4.1 Wireless sensor network

Implementation of WSN was reused from previous research (Stetsko 2012), with just a slight modifications to decouple it from previous optimisation framework. Nodes are simulated with MiXiM framework, with test application on each of them, that sends a packet containing arbitrary information through the network to a base station. Network has a static routing tree based on TODO algorithm.

We have marked some of the nodes as "droppers", so that instead of forwarding all of the packets through the network, they drop certain percentage. In our case, dropping ratio was set to 0.5.

4.2 Intrusion Detetection System

Our nodes implement a simple IDS capable of detecting nodes that seem to be intentionaly dropping packets. If we look at the wireless network stack of our node, IDS is part of the medium access controll sublayer. There it can eavesdrop on neighbouring nodes and check whether they behave accordingly.

For each neighbour it updates

- number of packets recieved
- number of packets forwarded.

We wanted to use IDS that is simple, but highly configurable. We have these four parameters to optimize:

1. number of monitored nodes
2. size of buffer for eavesdropped packets
3. treshold for minimal number of recieved packets for a node to be considered malicious

4. treshold for ratio between forwarded and recieved packets for a node to be considered to be malicious

At the end of simulation, based on a preset tresholds and ratio between forwarded and reieved packed it decides whether node is malicious or benign. More specificialy, the set of malicious nodes is:

$$M = \{n \in N | r_n \geq r_{min} \wedge \frac{f_n}{r_n} \geq d_{max}\}$$

And the set of benign nodes is:

$$B = \{n \in N | r_n < r_{min} \vee \frac{f_n}{r_n} < d_{max}\}$$

You can see, that both sets are disjoint, and their union is the set of neighbours N

4.3 Setup

We want to use our framework to optimize IDS in this boundaries:

Because this experiment is aimed primarily on showcasing the functionality of our framework, we have attempted meta optimization with the help of sample pareto-front, as well as TODO. We have used exhaustive search to sample the Pareto front and determine the TODO, therefore we could check, whether our assumptions hold.

Our mutation function had three parameters,probability that individual will be mutated $pMut$ and relative δ neighbourhood around the old value, that the new value will come from:

TODO CODE,IMAGE

Our cross-over function, with parameters $pCross$ (probability that two individuals will be crossed over) and $crossProb$ (probability that params of the two individuials will be swapped):

Calibration was first done on fixed number of generations and population size, with optimizing for values $pMut$, δ, and the type of algorithm, without using cross-over at all ($pCross = 0$).

4.4 Results