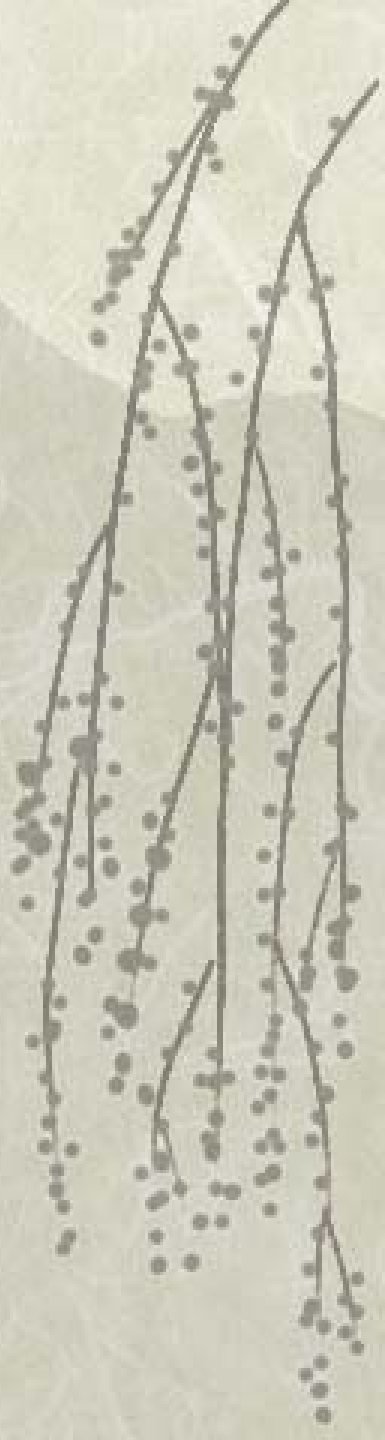
An Immunity-Based Technique to Characterize Intrusions in Computer Networks

By D. Dasgupta and F. Gonzalez,

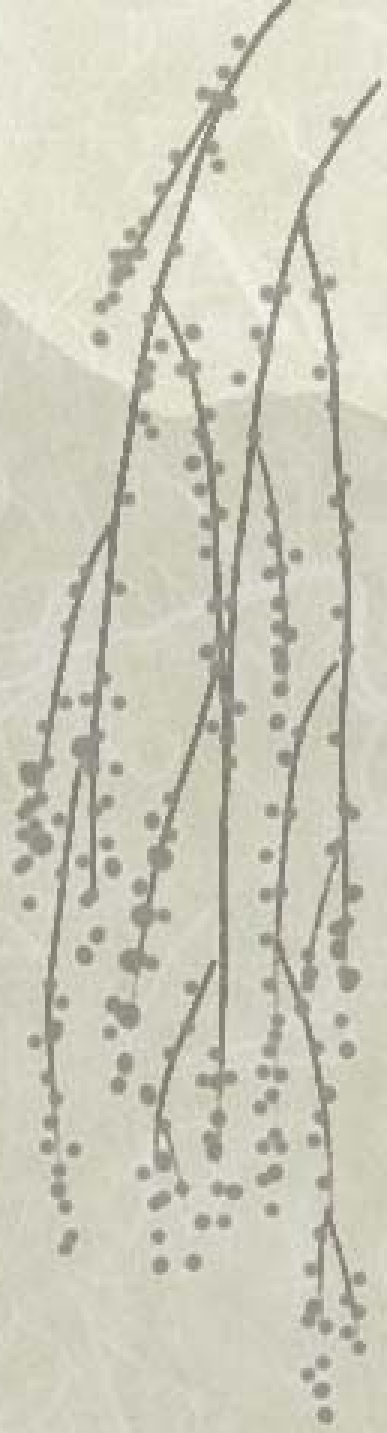
IEEE Transactions on Evolutionary Computation, 2002

Presenter: Markus Dale



Overview

- Artificial Immune Systems (AIS)
- Paper credentials
- New and Significant
- Results
- Other Methods
- Intro and Definitions
- Positive Characterization
- Negative Characterization
- Comparisons
- Criticism



Artificial Immune Systems

- Ideas first published in 1994
- Dedicated conference since 2001 - International Conference on Artificial Immune Systems (ICARIS)
- 805 entries found by Google Scholar
- DOD Office of Force Transformation described Network Immune Systems as hot trend. One of the companies listed, Okena, bought by Cisco.

Paper Credentials

- Published in IEEE Transactions on Evolutionary Computation - peer-reviewed quarterly journal of IEEE Computational Intelligence Society
- Google Scholar: Cited by 35
- Prof. Dasgupta:
  - 1998 editor of Springer book on AIS
  - AIS track organizer for Genetic and Evolutionary Computation Conference (GECCO) 2004/2005
  - Chair of IEEE Working Group on AIS

New and Significant

- Approach inspired by immune system
  - Extends previous two-class self/non-self approach to multi-class approach
  - No reliance on structured representation of data
  - Need only positive data to build normal profile of system
  - Applied to anomaly detection for network security but general approach can be applied to different anomaly detection problems

Results

- Negative Characterization (NC) approach compares very favorably to Positive Characterization (PC) approach
  - Only 10% of space needed
  - Results “closely” match PC results

Introduction

- Immune system: Distinguish self from non-self
- Computer system: Legitimate users and data from unauthorized users and viruses
- 1994 Stephanie Forrest at U of NM used negative-selection algorithm (detect only non-self)

Statistical Methods

- Simple model: individually different state variables.
  - Normalcy depends on time
  - Normalcy depends on correlations among parameters
- Temporal and multi-var correlations
  - Require too much training time
  - Infeasible due to large data set size

Data Mining

- Some success in capturing multi-variable correlations but
  - Data must have some degree of structure
  - Need information on which connections are normal versus which connections are attacks for some algorithms

Definitions

- System State Space
  - State S consists of vectors of features corresponding to all possible states of system.

$$x^i = (x_1^i, x_2^i, \ldots, x_n^i) \quad [0.0,1.0]^n$$

- Normal Subspace:

$$Self \quad S$$

$$Non\_Self = S - Self$$

More Definitions

- Characteristic function:

$$\mu_{self} : [0.0,1.0]^n \qquad [0.0,1.0]$$

$$m_{self,t}(\vec{x})$$

1 if > t
0 if <= t
Degrees of normalcy: 1 normal - 0 abnormal

Anomaly Detection Problem

- Given a set of normal samples build good estimate of μ_{self}
- This function decides whether or not observed state is anomalous

Positive Characterization (PC) Approach

- Use positive sample set for representation of *Self* space as *Self* set
- Characteristic function is distance from that element to nearest neighbor in *Self* set
  - Euclidean distance but also used:

$$D\ (\vec{x}, \vec{y}) = \max(|x_1 - y_1|, \ldots, |x_n - y_n|)$$

PC Implementation

- Use Bentley's KD-tree
  - Represents k-dimensional points
  - Generalization of one-dim binary search tree
  - Amortized cost of nearest-neighbor query is O(log N)

Data Sets for Experiments

- Data sets on intrusion detection from MIT Lincoln Labs
  - Normal and abnormal data from test lab with simulated attacks
  - Standard corpora for IDS
  - Network traffic (tcpdump), audit data (Solaris BSM) and file system data
  - Five weeks of data with ~300k per day compressed (3 training/labeled, 2 test)

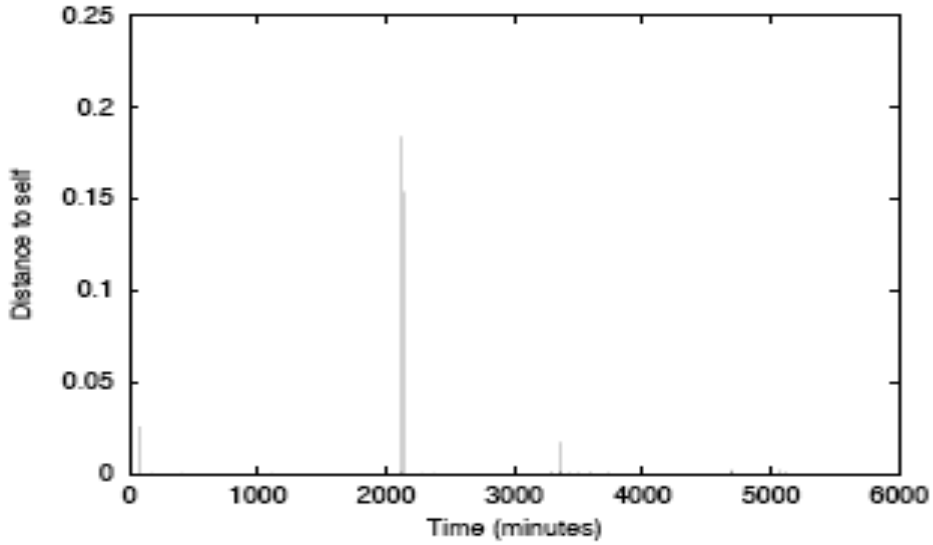
Experiment Setup

- Train using first week - attack free data
- Use second week for test - only network attacks
- Parameters used:
  - Number of bytes per second
  - Number of packets per second
  - Number of Internet Control Message Protocol (ICMP) packets per second
- Sample each minute. Built set S of normal descriptors from time series R in overlapping sliding windows with window size w:

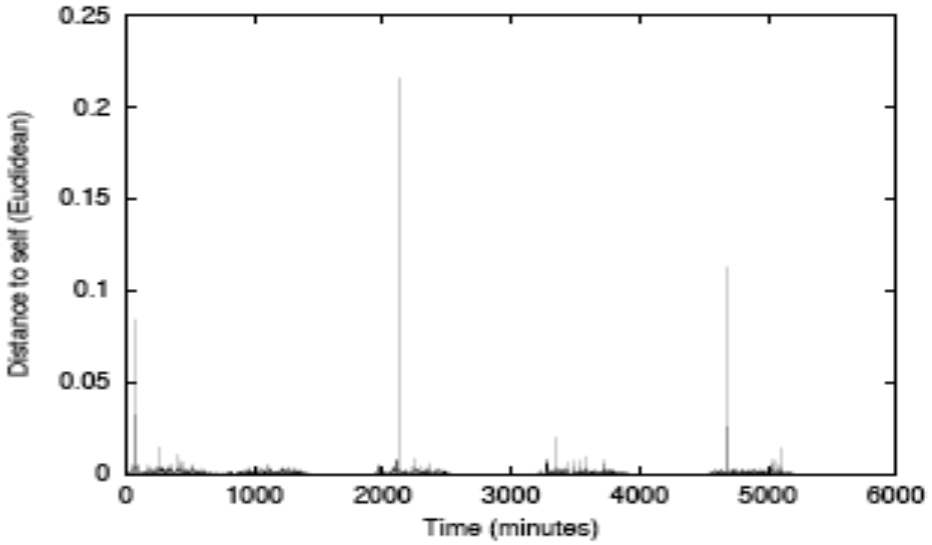
$$S = \{(r_1, \ldots, r_w), (r_2, \ldots, r_{w+1}), \ldots, (r_{n-w+1}, \ldots, r_n)\}$$

Experiment 1

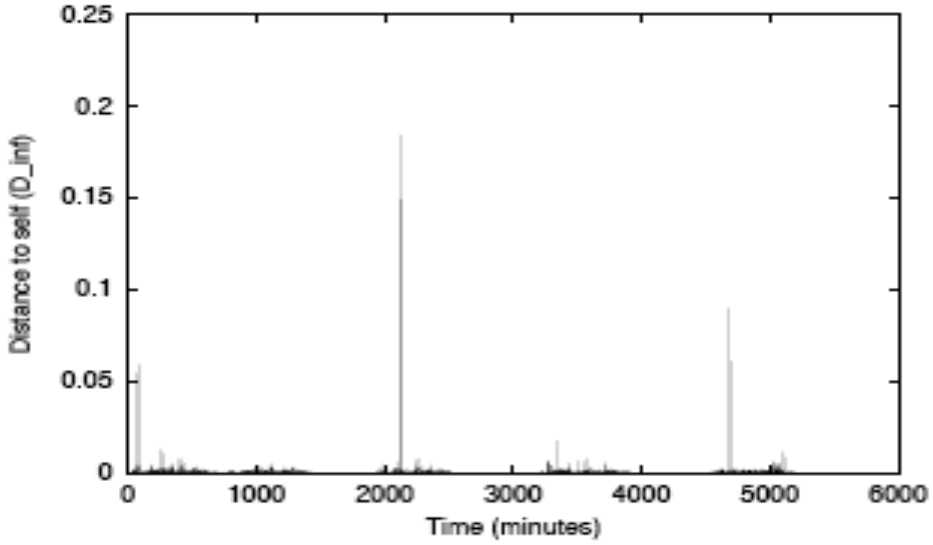
- Considers variables independently
  - Not enough to detect all five attacks
  - Higher window size increases sensitivity
  - Higher window size allows for detection of temporal patterns
  - Change of metric does not modify number/type of deviations detected



(a) ws = 1



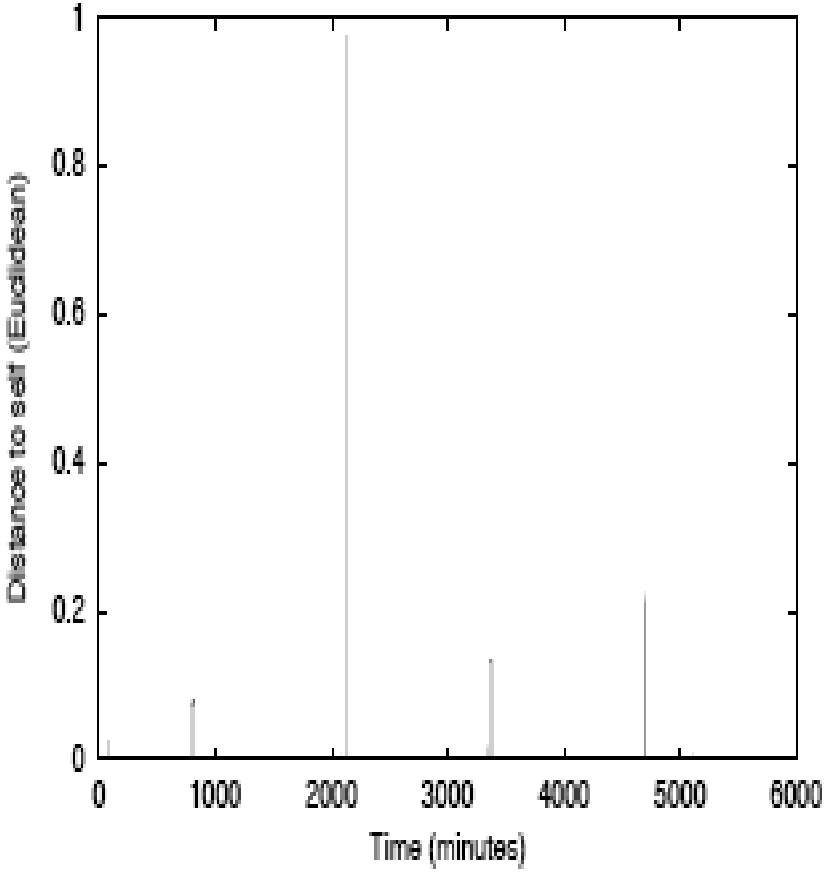
(b) ws = 3, Euclidean



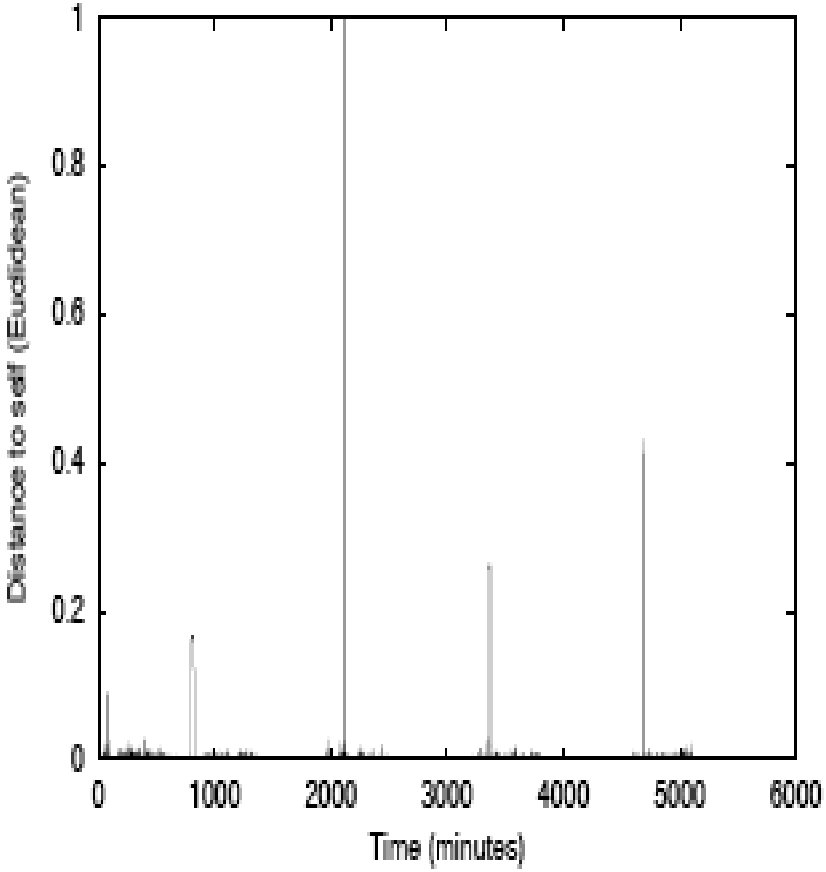
(c) ws = 3, D_α

Experiment 2

- Uses all three network params as features
- Use window size 1 and 3
- Detects all five attacks
- Increasing window size increases sensitivity


Distance to self (Euclidean)
Time (minutes)
0
0.2
0.4
0.6
0.8
1
0
1000
2000
3000
4000
5000
6000

(a) ws = 1


Distance to self (Euclidean)
Time (minutes)
0
0.2
0.4
0.6
0.8
1
0
1000
2000
3000
4000
5000
6000

(b) ws = 3

Positive Characterization

- Worked well on performed experiments
- Problem:
  - Must store all samples that constitute normal profile
  - Infeasible for large data generated by network traffic!

Human Immune System

- Self/Non-Self distinction with T-Cells
  - T-Cell receptors made with pseudo-random genetic rearrangement process.
  - Negative selection: Censoring process in thymus. Destroy T-cells that react against self-proteins
  - Circulate throughout body to detect non-self proteins (antigens).

Original Negative Selection Algorithm

- Forrest et al. 1994
- Self: S - collection of strings of length l over finite alphabet (e.g. normal activity segmented into equal-sized substrings)
- Set of detectors R - fail to match any string in S
- Monitor S for matches of R - change!
- Used binary representation (either Self or Non-Self), O(N) runtime (N = number of strings in S)

Extended Negative Selection

- Use real-valued representation for detector rules

R_1: If Cond_1 then non-self (or level of variability)

…

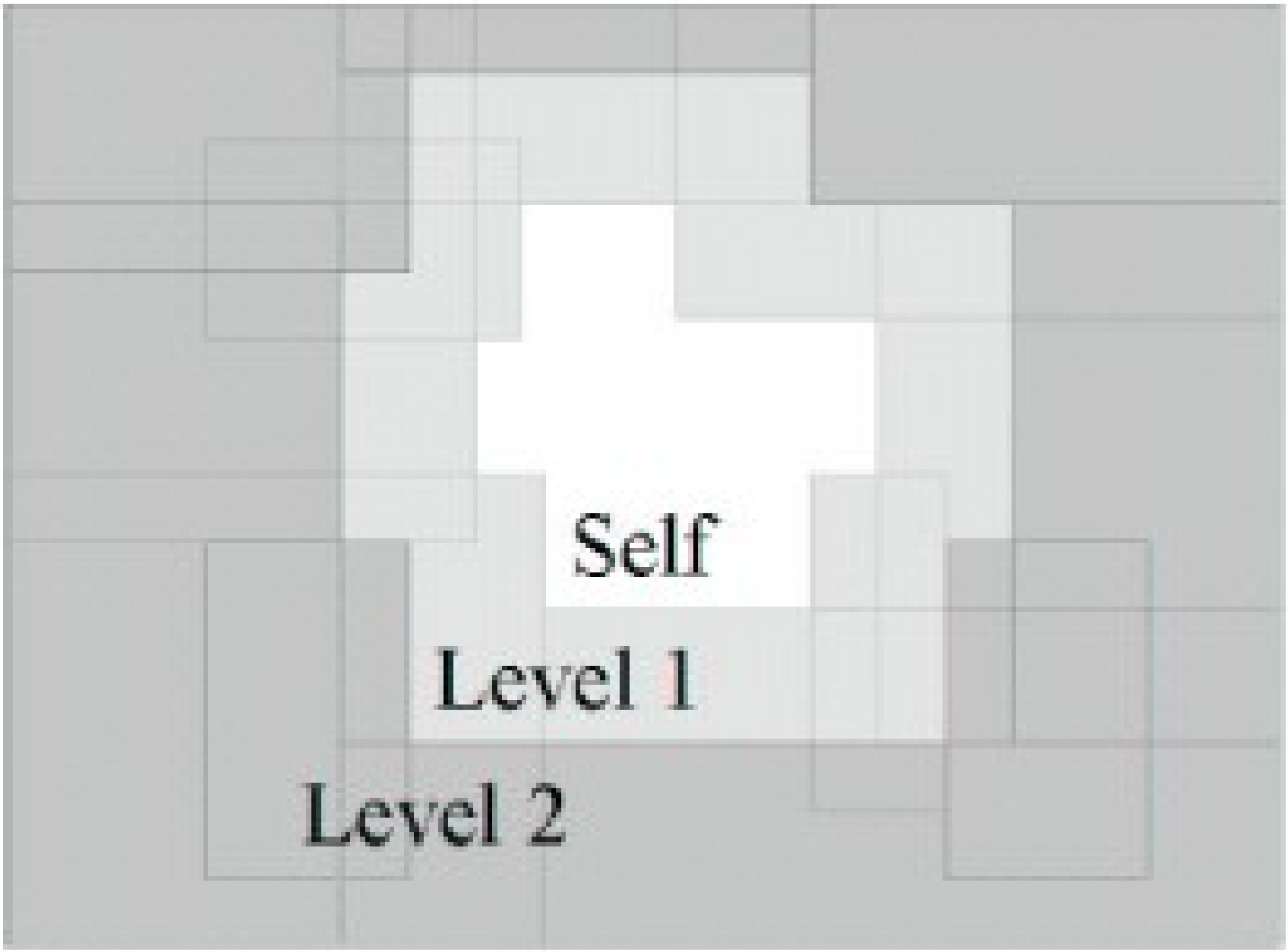
R_m: if Cond_m then non-self

***Cond_i**: consists of ranges between a lower and upper value for each feature of a feature vector. The condition is met if all features fall within their respective ranges.*

***Levels**: Different levels of variability v within normal space. Match highest level rule.*

Detector Rules

- Conditions define hypercube in descriptor space
- Set of rules try to cover non-self space with hypercubes
- If a feature vector matches a condition it must be part of non-self space or that variable level (pick highest matching rule)
- How to create rules? Genetic algorithm


Self
Level 1
Level 2

Criteria for GA evolution

- Multi-objective optimization problem
  - Don't cover positive examples
  - Large area (try to minimize number of rules)
  - Minimize overlap with other rules (use sequential niching algorithm)

GA Input

- Feature vectors set S of normal behavior vectors (training set)
- v: Level of variability
- maxRules: max rules in solution set
- minFitness: min value for rule to be included in solution set
- maxAttempts: max number of attempts to evolve a rule with fitness > *minFitness*

GA Representation

- Each chromosome = condition part of a rule
- Used fixed eight bits for each element [low,high]
- String of bits mapped to interval [0.0,1.0]

Fitness Evaluation

- Rule R matched for feature vector x^j if hypersphere centered at x^j and radius v intercepts hyperrectangle defined by points (low_1,high_1)…(low_n,high_n) of rule.
- Volume of R:

$$volume(R) = \prod_{i=1}^{n} (high_i - low_i)$$

Raw Fitness and Sequential Niching

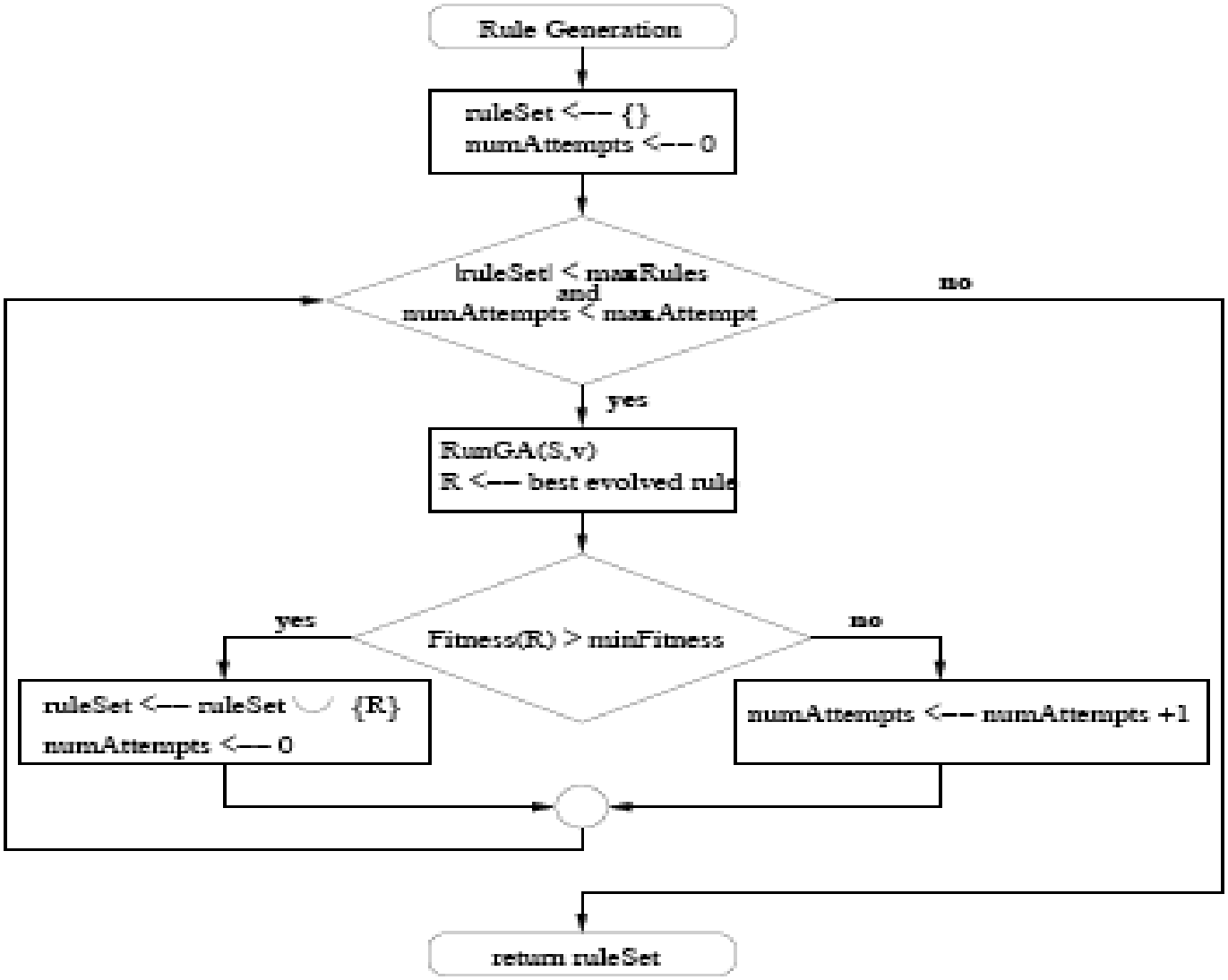
$$raw\_fitness_R = volume(R) - C \cdot num\_elements(R)$$

$fitness_R \leftarrow raw\_fitness_R$

for each $R^j \in ruleSet$ do

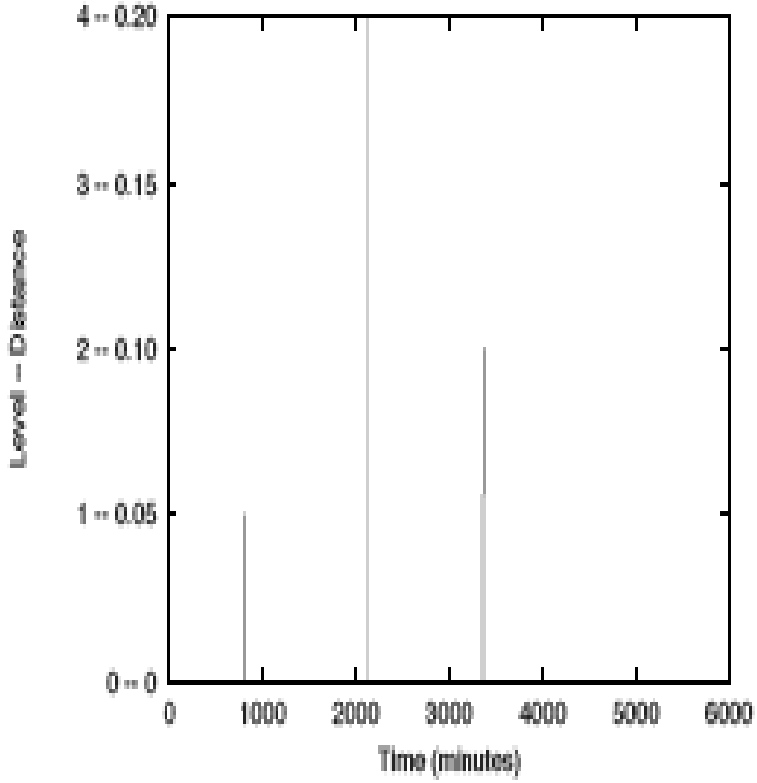
$fitness_R \leftarrow raw\_fitness_R - volume(R \cap R^j)$

end-For

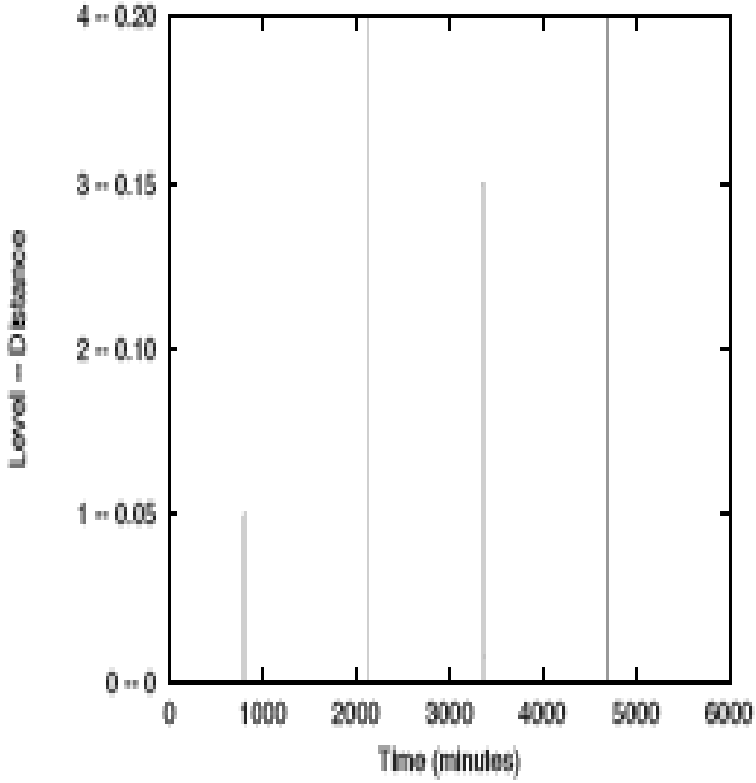

Rule Generation
ruleSet <— {}
numAttempts <— 0
|ruleSet| < maxRules
and
numAttempts < maxAttempt
no
yes
RunGA(S,v)
R <— best evolved rule
yes
Fitness(R) > minFitness
no
ruleSet <— ruleSet ◡ {R}
numAttempts <— 0
numAttempts <— numAttempts +1
return ruleSet

Experiments

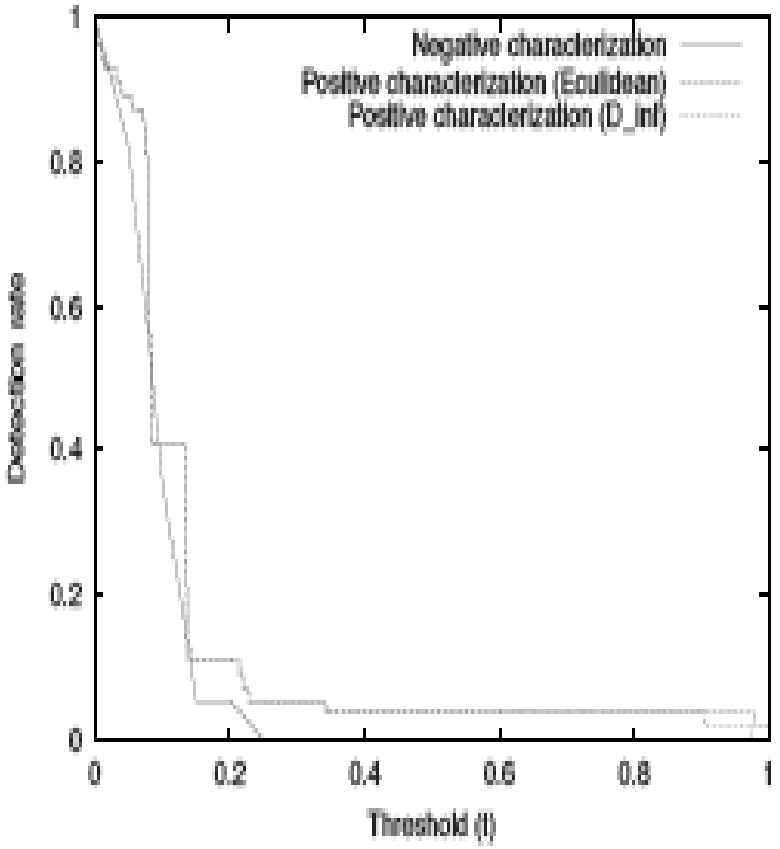
- GA params:
  - Population size 100
  - Number of generations 1500
  - Mutation rate 0.2
  - Crossover rate 1.0
  - Sensitivity 1.0
- Variability: 0.05, 0.1, 0.15 and 0.2
- Run each process 10 times against test, report average results


4 -- 0.20
3 -- 0.15
2 -- 0.10
1 -- 0.05
0 -- 0
Level – Distance
0
1000
2000
3000
4000
5000
6000
Time (minutes)

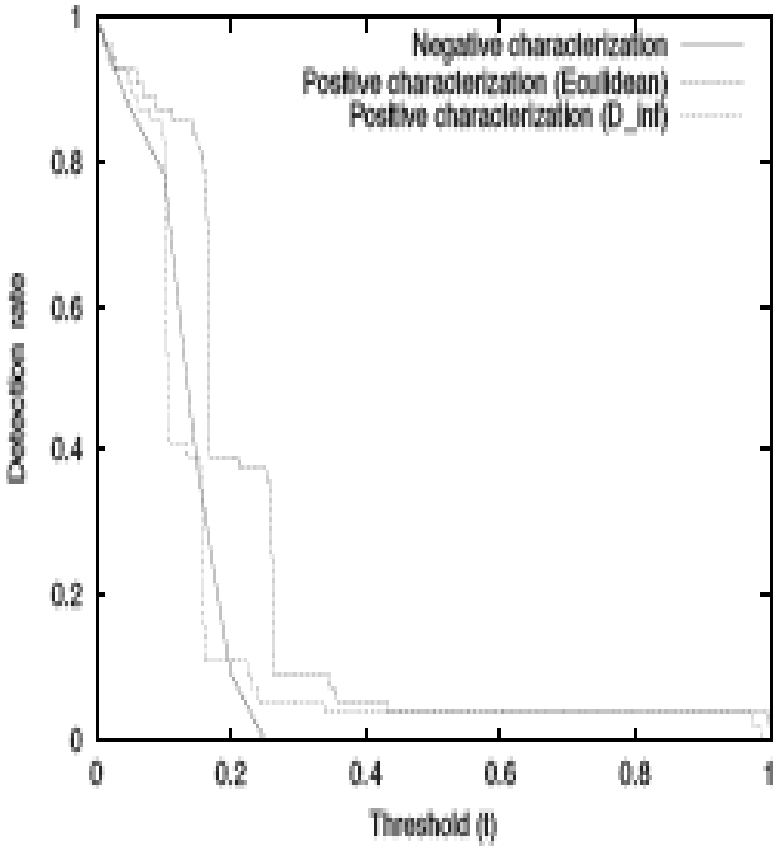
(a) ws = 1


4 -- 0.20
3 -- 0.15
2 -- 0.10
1 -- 0.05
0 -- 0
Level – Distance
0
1000
2000
3000
4000
5000
6000
Time (minutes)

(b) ws = 3


Negative characterization
Positive characterization (Euclidean)
Positive characterization (D_inf)
Detection ratio
Threshold (t)
0
0.2
0.4
0.6
0.8
1

(a) ws = 1


Negative characterization
Positive characterization (Euclidean)
Positive characterization (D_inf)
Detection ratio
Threshold (t)
0
0.2
0.4
0.6
0.8
1

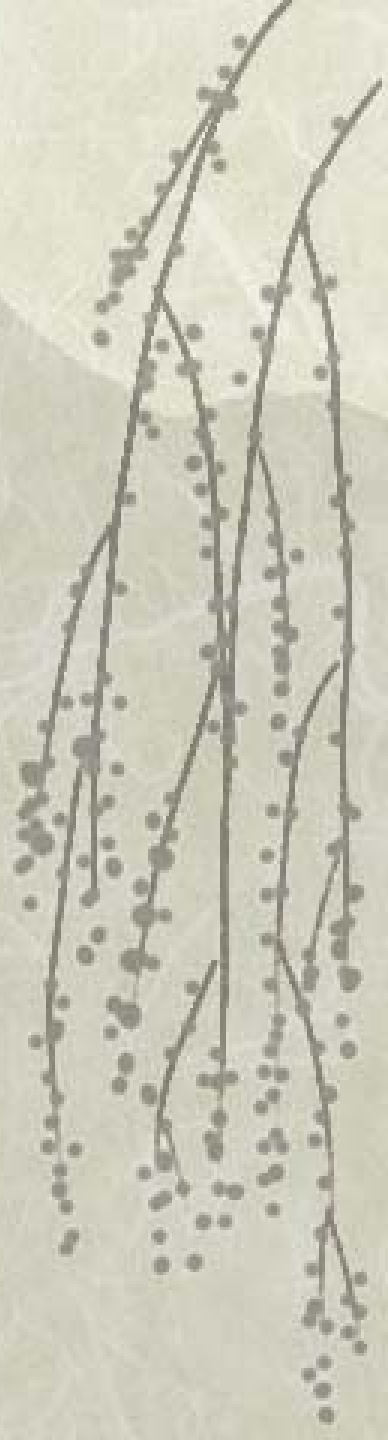
(b) ws = 3

Table 4: Best true positive rates for the different techniques with a maximum false alarm rate of 1%.

Detection Technique	Window size 1	Window size 3
Positive Characterization (Euclidean)	92.8%	96.4%
Positive Characterization (D_∞)	92.8%	92.8%
Negative Characterization	82.1%	87.5%

Tradeoffs

- NC more efficient in space than PC
- Tradeoff between compactness and accuracy (but similar)
  - NC uses only 10% of space
  - Detects 4 vs. PC 5 attacks



Future Work

- Different strategies for covering non-self space (e.g. hyperspheres)
- Apply to different IDS data
- Development of new algorithms to generate non-self covering rules

Criticism

- How does difference between PC/NC scale for number of attacks?
- How does approach scale to larger data sets?
- How does approach compare to best alternative approach?

References

- D. Dasgupta and F. Gonzalez, “An Immunity-Based Technique to Characterize Intrusions in Computer Networks”, IEEE Transactions on Evolutionary Computing, 6(3), pages 1081-1088, June 2002
- MIT Lincoln Lab Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/