Using the Grid to improve the effectiveness of Learning Classifier Systems

through clustering-based initialization

Fani Tzima, Fotis Psomopoulos, and Pericles Mitkas

Greece

www.eu-egee.org

e-infrastructure

Presentation Outline

Introduction

- Learning Classifier Systems (LCS)
- Different LCS flavors

ZCS-DM: Algorithmic description

- Rule representation
- Operation Cycle
- Clustering-based Initialization Component

Experiments and Results

- Experimental Setting for leveraging the Grid Infrastructure
- Qualitative Interpretation of Results
- Statistical Comparison of Results

Conclusions and Future Work

Learning Classifier Systems

- Learning Classifier Systems (LCS) [Holland, 1976] are a machine learning technique designed to work for both single-step and sequential decision problems

- LCS employ a population of classifiers (usually rules in the production system form) gradually evolving through the use of a reinforcement scheme and a GA-based search component

Michigan Vs. Pittsburgh

- Smith's approach, from the University of Pittsburgh → GA applied to a population of LCSs in order to choose the fittest
- "Michigan style" LCSs employ a population of gradually evolving, cooperative classifiers → each classifier encodes a fraction of the problem domain

Different LCS Flavors

- **Strength-based LCSs (ZCS)**
  - each classifier contains only one evaluation variable → both an estimation of the accumulated reward brought by its firing and its fitness for the population evolution
- **Accuracy-based LCSs (XCS)**
  - decoupling the RL process and the population evolution → fitness function not proportional to the expected reward, but to the accuracy of the latter's prediction
- **Anticipatory LCSs (ALCS)**
  - [Condition] [Action] → [Effect] classifiers (instead of [Condition]→[Action])
  - [Effect] represents the expected effect (next state)

Rule representation

- **Traditional production form of**

  IF *condition* **THEN** *action* [Strength] [Fitness]

- **Condition comprises predicates of the form**

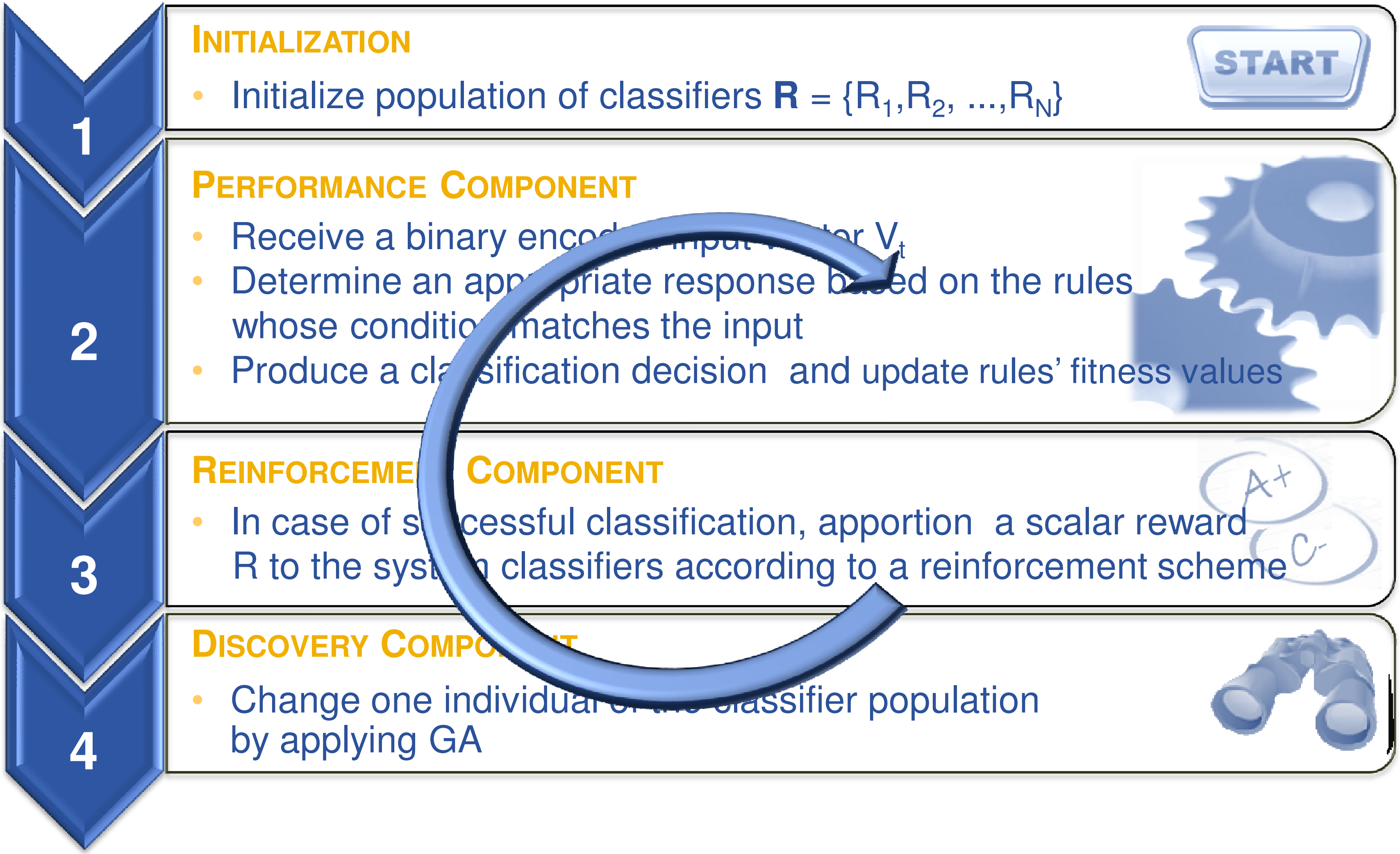
  <Attribute $\in$ SetOfNominalValues | NumericInterval>

- **Encoded over the ternary alphabet 0,1,#.**
  - The symbol # (“wildcard” or “don’t care”) allows for generalization.

- **Actions are discrete**

Both inputs **11** and **10** are matched by the rule condition **1#**

ZCS-DM operation cycle



1

INITIALIZATION

- Initialize population of classifiers **R** = {R_1,R_2, ...,R_N}

2

PERFORMANCE COMPONENT

- Receive a binary encoded input vector V_t
- Determine an appropriate response based on the rules whose condition matches the input
- Produce a classification decision and update rules' fitness values

3

REINFORCEMENT COMPONENT

- In case of successful classification, apportion a scalar reward R to the system classifiers according to a reinforcement scheme

4

DISCOVERY COMPONENT

- Change one individual of the classifier population by applying GA

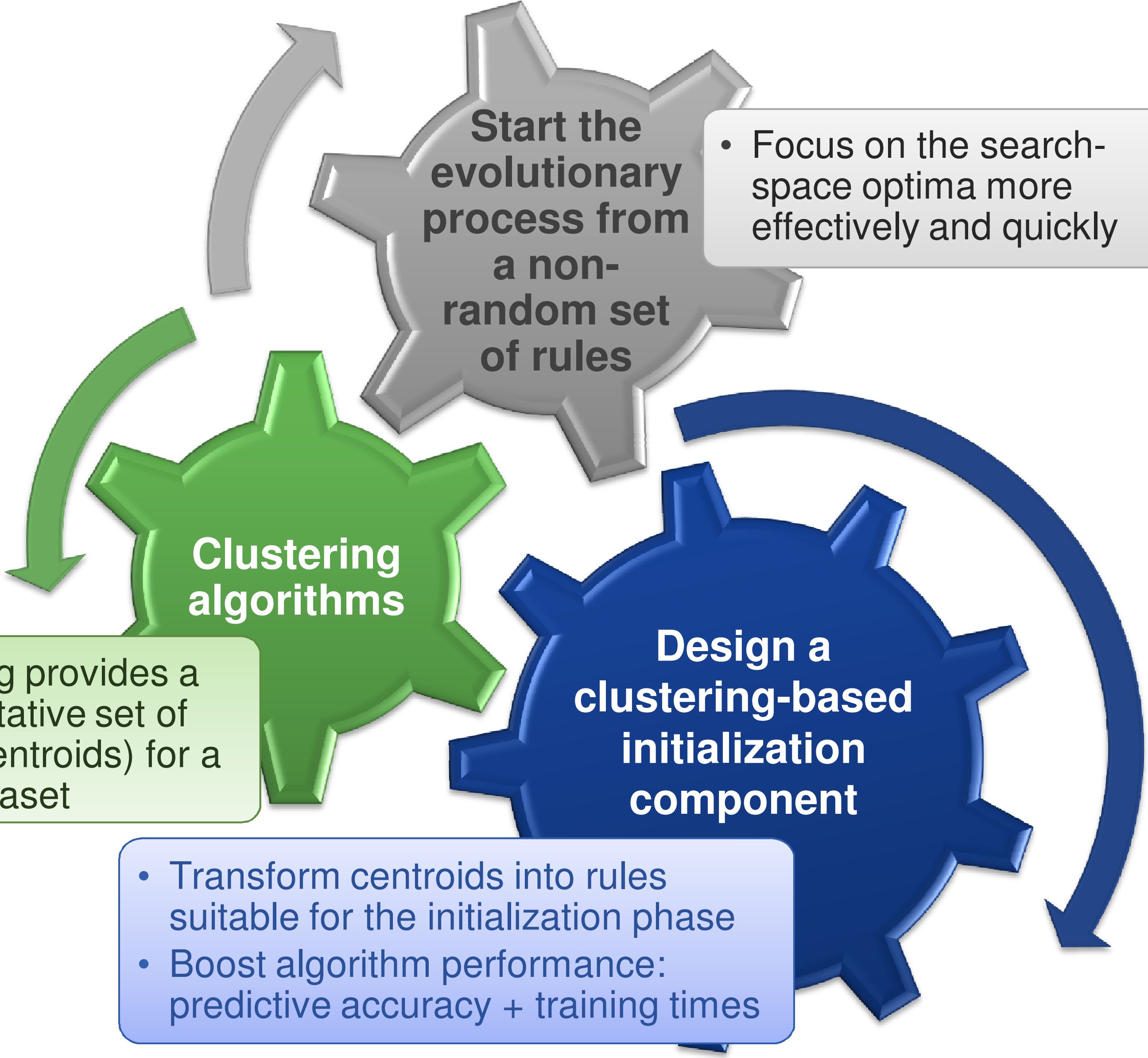
Method Pros and Cons

- ✓ Intuitive representation
- ✓ Applicable for datasets where there is no prior knowledge of the attributes’ probability distributions
- ✓ Production of models storable in a compact form
- ✓ Fast (post-training) classification of new observations
- ✓ Resulting ruleset is ordered

Grid Resources for Parameter Optimization
and Parallel Execution of Experiments

- X **Non-deterministic nature of the algorithm + Relatively long training times**
  - X multiple experiments to reach statistically sound conclusions
- X **Large number of tunable parameters**

Clustering-based Initialization


Start the evolutionary process from a non-random set of rules
Clustering algorithms
Design a clustering-based initialization component

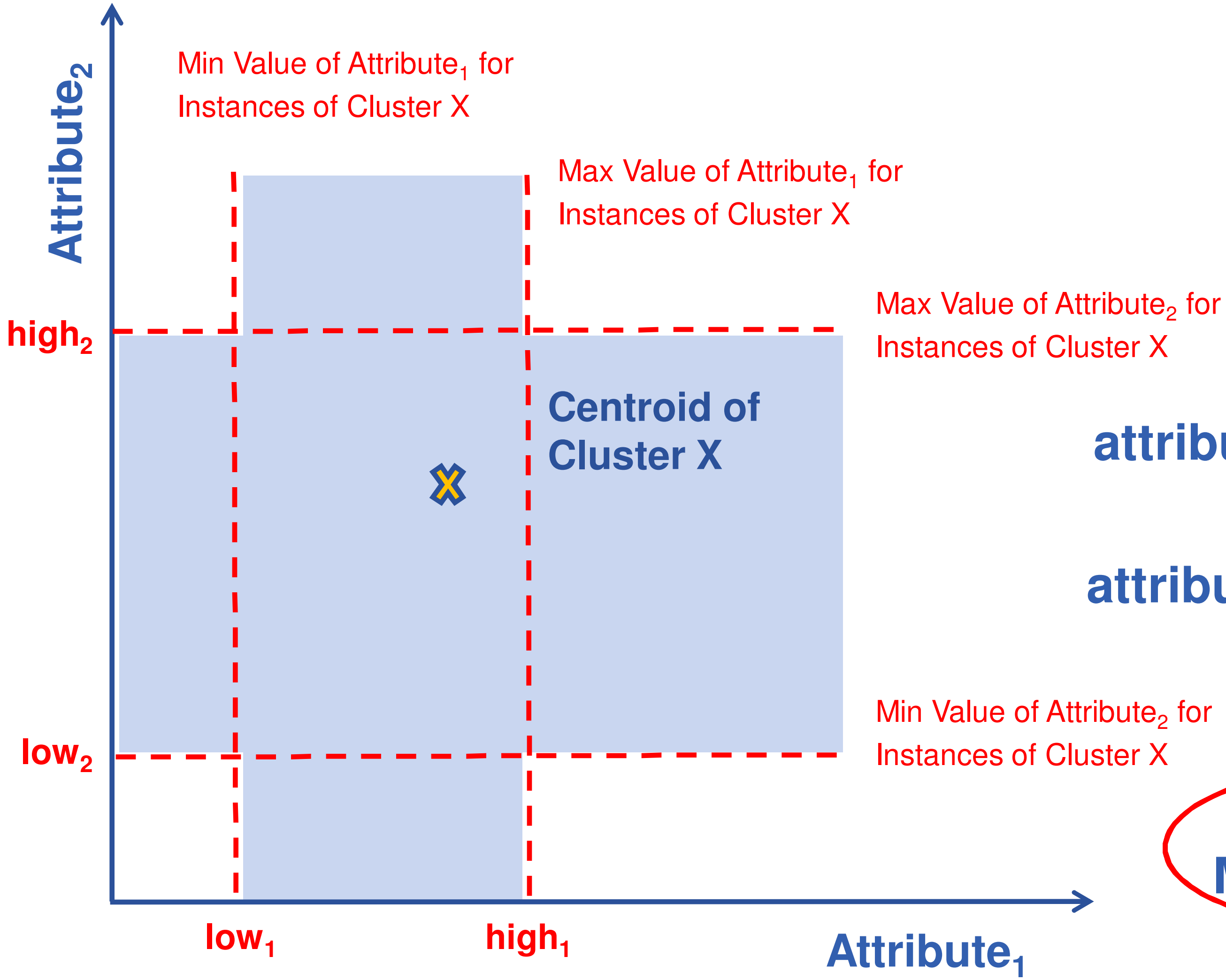
- Focus on the search-space optima more effectively and quickly

- Clustering provides a representative set of points (centroids) for a given dataset

- Transform centroids into rules suitable for the initialization phase
- Boost algorithm performance: predictive accuracy + training times

Centroids to Rules Transformation

- 3 possible condition parts for the case of 2 numeric attributes


$Attribute_2$
Min Value of $Attribute_1$ for Instances of Cluster X
Max Value of $Attribute_1$ for Instances of Cluster X
Max Value of $Attribute_2$ for Instances of Cluster X
$high_2$
Centroid of Cluster X
low_2
Min Value of $Attribute_2$ for Instances of Cluster X
low_1
$high_1$
$Attribute_1$

$attribute_1 \in [low_1, high_1]$

AND

$attribute_2 \in [low_2, high_2]$

⇩

Cluster X Majority Class

Experimental Setting

- Evaluation of 4 versions of the algorithm
  - **ClusterInit100**
    Clustering-based initialization – Full training time (100 iterations)
  - **RandomInit100**
    Random ruleset initialization – Full training time (100 iterations)
  - **ClusterInit75**
    Clustering-based initialization – Reduced training time (75 iterations)
  - **RandomInit75**
    Random ruleset initialization – Reduced training time (75 iterations)

“Vanilla-Setup” Parameters

Parameter	Description	Value
N	Number of rules	400
I	Number of iterations	100/75
detAS	[illegible]	True
S	[illegible]	100
R	[illegible]	1000
p	[illegible]els)	0.5
τ	Tax for classifiers in NOT**A**	0.1
ρ	GA invocation rate	0.5
c	Crossover probability	0.15
m	Mutation probability	0.005
g	Generalization probability	0.1
φ	Covering invocation threshold	0.1
NC	Number of clusters	10
gc	Clustering generalization rate	0.5

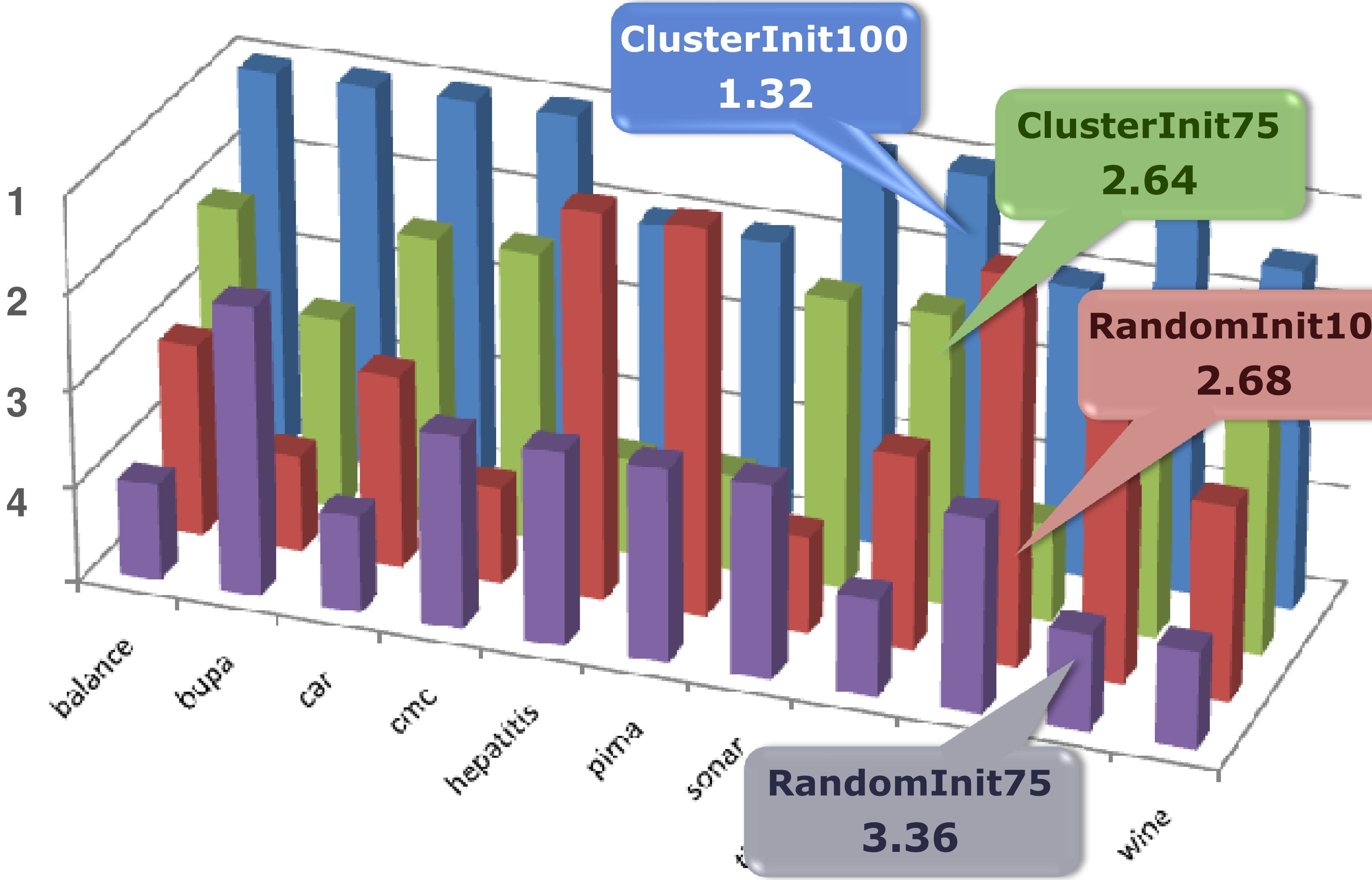
Number of iterations I expresses the number of complete passes through the training set during the algorithm training phase

Benchmark Datasets

Dataset	Attributes	Classes	Missing Values	Instances
Balance Scale Weight & Distance	4 nominal	3	0	625
Bupa Liver Disorders	6 numeric	2	0	345
Car Evaluation	6 nominal	4	0	1728
Contraceptive Method Choice	7 nominal + 2 numeric	3	0	1473
Hepatitis	13 nominal + 6 numeric	2	167	155
Pima Indians Diabetes	8 numeric	2	0	768
Connectionist Bench (Sonar)	60 numeric	2	0	208
Tic Tac Toe Endgame	9 nominal	2	0	958
Congressional Voting Records	16 nominal	2	392	435
Breast Cancer Wiskonsin	9 numeric	2	16	699
Wine	13 numeric	3	0	178

- **20 x 10-fold stratified cross-validation runs**
- **Comparison of the results based on accuracy rate**

Comparative analysis of results



ClusterInit100
1.32

ClusterInit75
2.64

RandomInit100
2.68

RandomInit75
3.36

Statistical Evaluation of Results

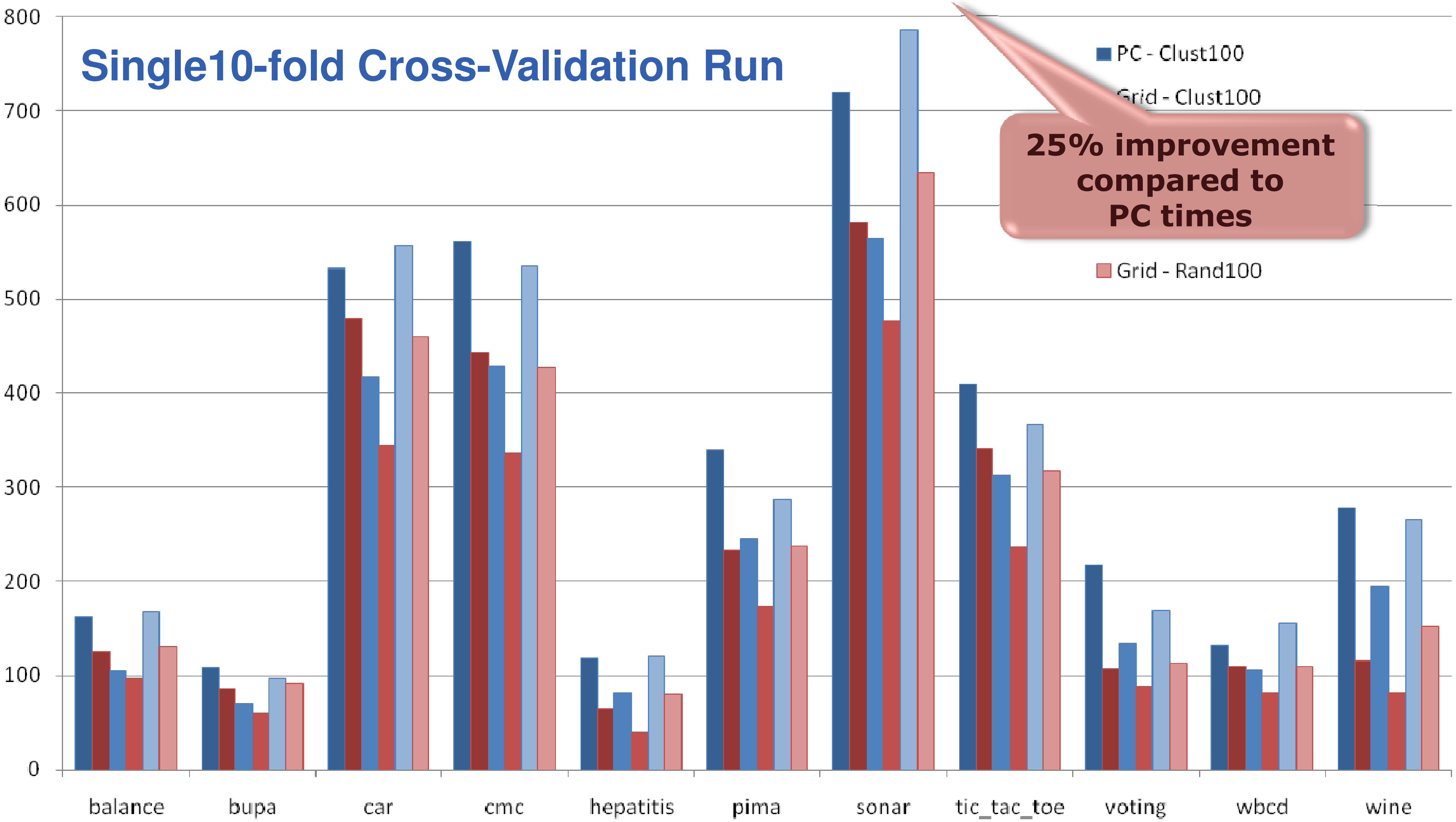
- **Statistical procedure [Demsar, 2006] for robustly comparing classifiers across multiple datasets**
  - use the **Friedman test** to establish the significance of the differences between classifier ranks
  - use a **post-hoc test** to compare classifiers to each other
- **In our case, the goal was to compare the performance of all algorithms to each other**
  - the **Nemenyi test** was selected as the appropriate post-hoc test

At $\alpha = 0.05$, the performance of the clustering-based initialization approach with full training times is *significantly better* than that of all its rivals.

At $\alpha = 0.05$, the performance of the clustering-based initialization approach with reduced training times is *NOT significantly different* than that of the baseline approach with full training times.

Comparison of Execution Times (1/2)

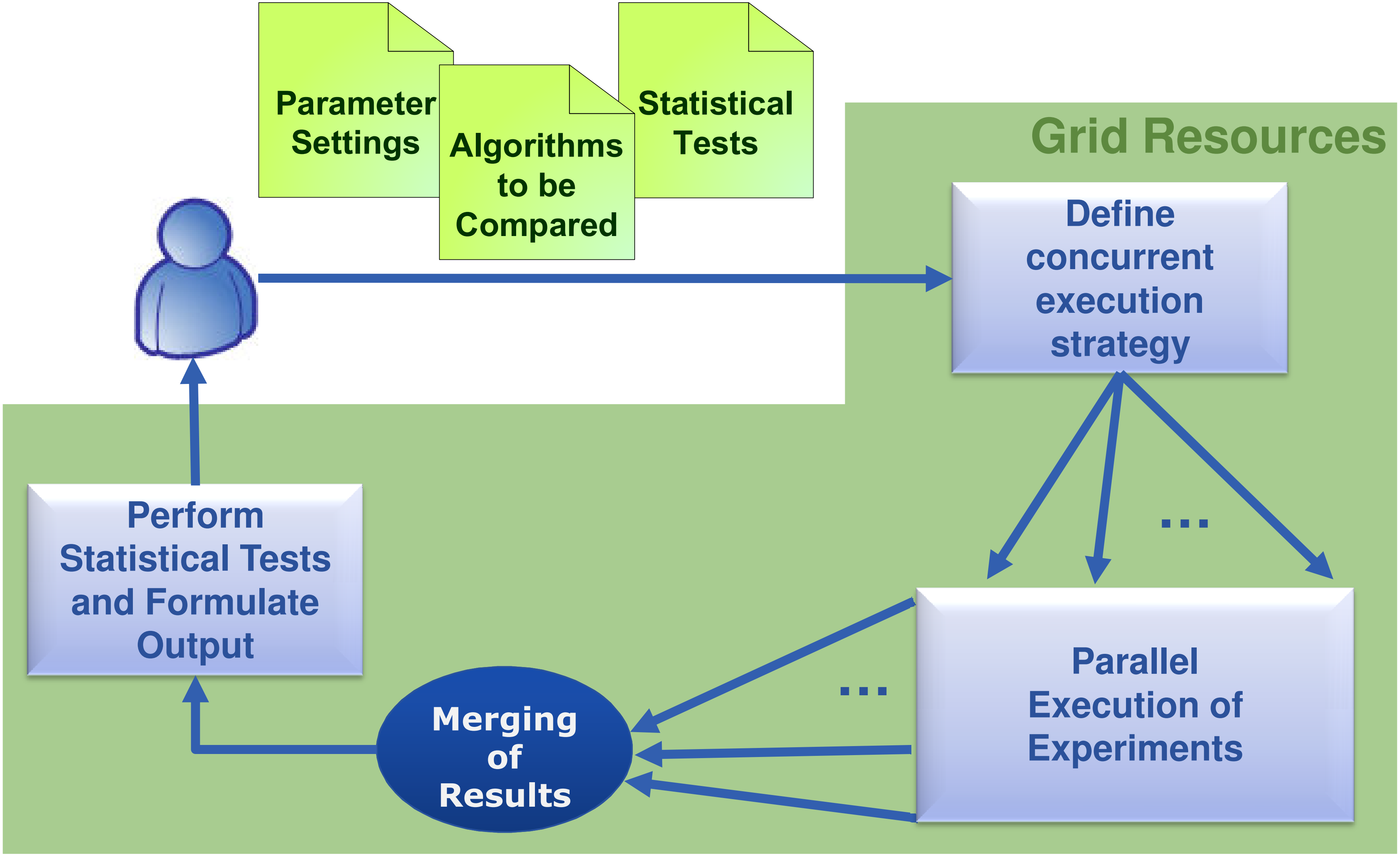
- Execution time (sec) on **personal computer** (Intel Core 2 Duo, CPU @2.00GHz – 4,00 GB RAM) Vs. **the Grid Infrastructure**



Single10-fold Cross-Validation Run

25% improvement compared to PC times

Grid DAG workflow



Parameter Settings

Algorithms to be Compared

Statistical Tests

Grid Resources

Define concurrent execution strategy

…

Parallel Execution of Experiments

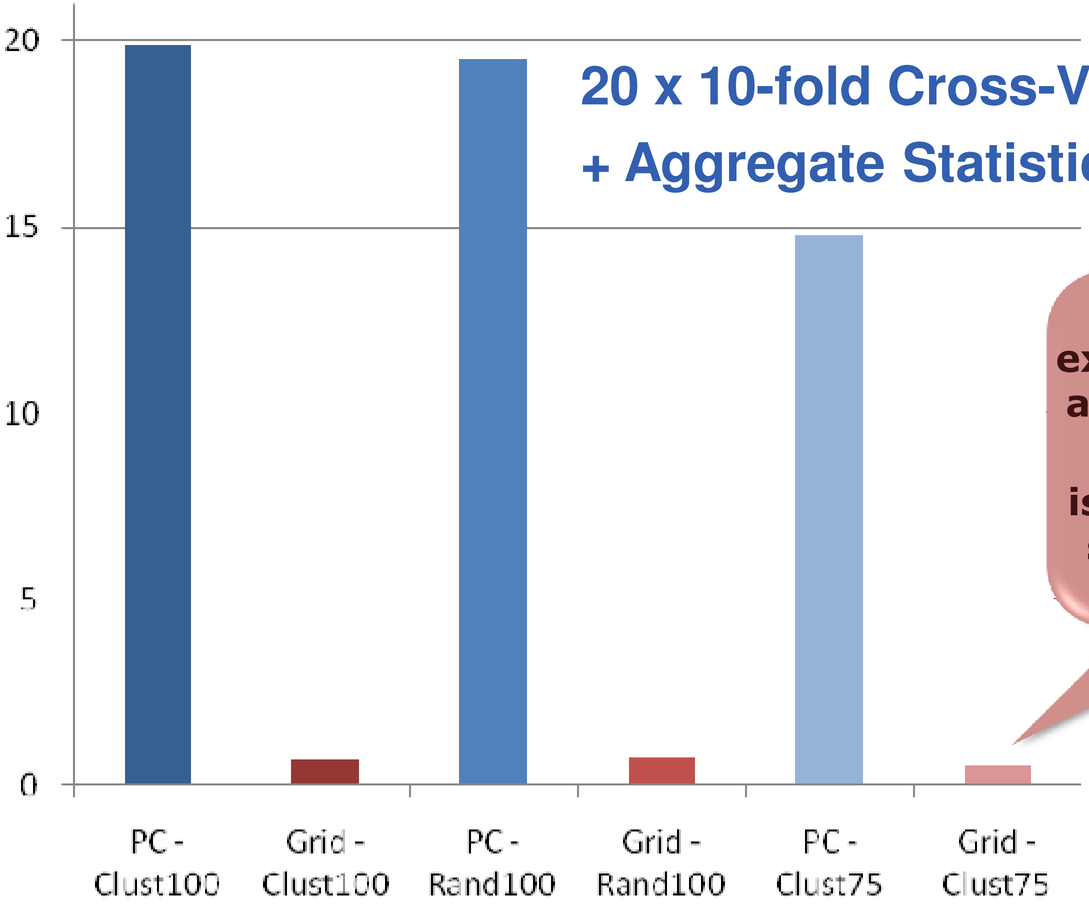
…

Merging of Results

Perform Statistical Tests and Formulate Output

Comparison of Execution Times (2/2)

- Execution time (hrs) on **personal computer** (Intel Core 2 Duo, CPU @2.00GHz – 4,00 GB RAM) Vs. **the Grid Infrastructure**



20 x 10-fold Cross-Validation Runs
+ Aggregate Statistical Tests

Scripted parallel execution on the Grid + automatic aggregation of results
is 27 times faster than sequential execution on PC

Outlook - Conclusions

- **Clustering-based initialization proved to be a useful component**
  - achieving **the best prediction accuracy** (on average) when full training times were employed
  - performing **equally well with the baseline approach**, even when **reduced training times** were employed
- **The concurrent utilization of Grid resources allowed for an effective and time-efficient way to perform parameter optimization and/or algorithm comparison experiments**
- **The Grid is the ideal execution environment due to the embarrassingly parallel nature of the problem**
  - jobs submitted simultaneously (organized in a DAG workflow)
  - different parameter set → independence of jobs

Future Work

- **Design and implementation of a more in-depth parameter exploration strategy to be evaluated on the Grid infrastructure**
  - effect on system performance
- **Post-training processing steps**
  - consistency and compactness of evolved rulesets
- **Evaluation of the algorithm as an on-line data-mining tool for real-world domains (such as urban Air Quality)**
  - the nature of the algorithm and the capability of LCS to tackle multi-step decision problems are encouraging

Thank you for your attention!

Fotis Psomopoulos

fpsom@issel.ee.auth.gr

Intelligent Systems and Software Engineering Labgroup

Informatics and Telematics Institute
Centre for Research and Technology-Hellas

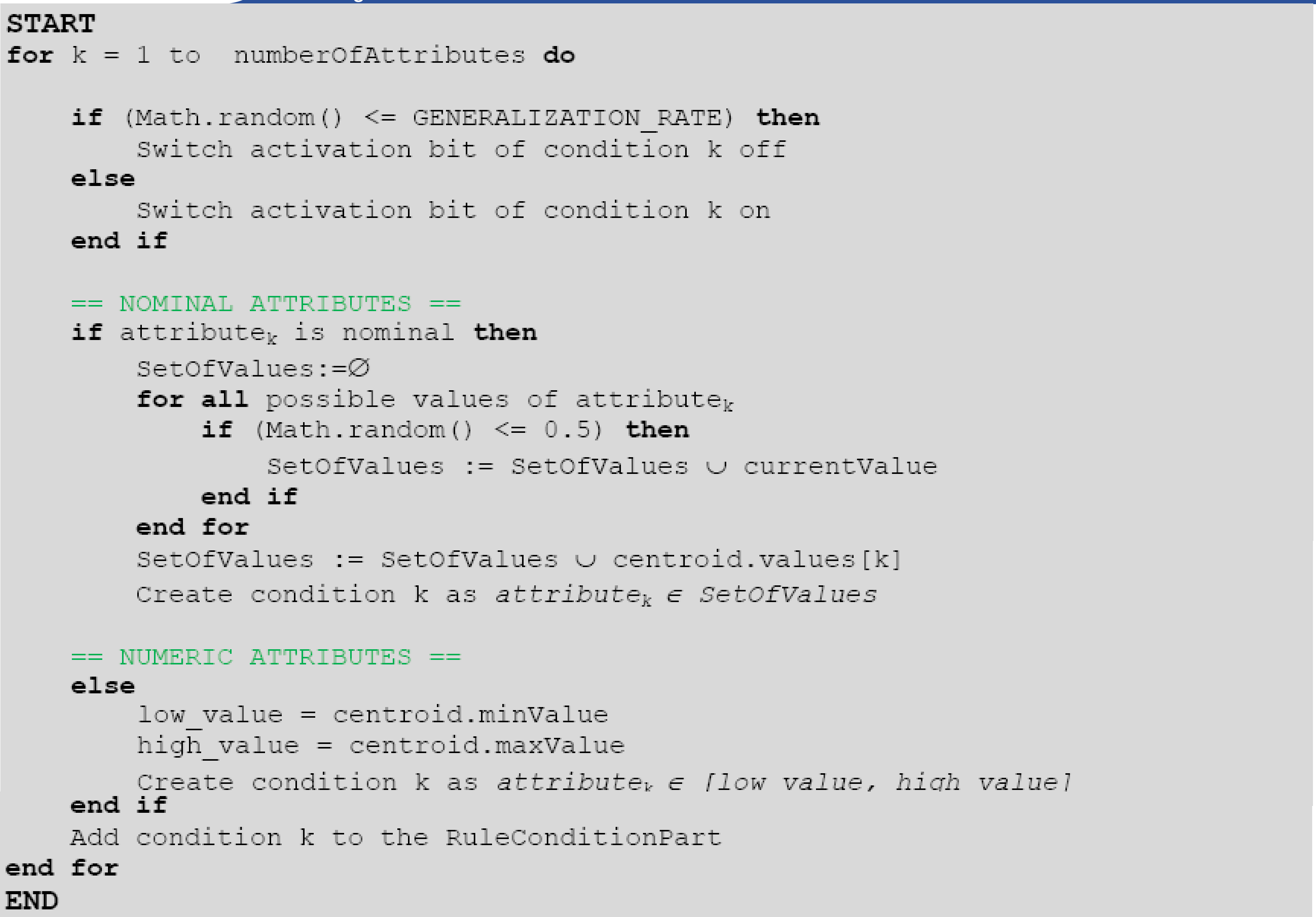
Electrical and Computer Eng. Dept.
Aristotle University of Thessaloniki

Thessaloniki, Greece

www.eu-egee.org

e-infrastructure

Centroids to Rules Transformation



```
START
for k = 1 to  numberOfAttributes do

    if (Math.random() <= GENERALIZATION_RATE) then
        Switch activation bit of condition k off
    else
        Switch activation bit of condition k on
    end if

    == NOMINAL ATTRIBUTES ==
    if attribute$_k$ is nominal then
        SetOfValues:=$\emptyset$
        for all possible values of attribute$_k$
            if (Math.random() <= 0.5) then
                SetOfValues := SetOfValues $\cup$ currentValue
            end if
        end for
        SetOfValues := SetOfValues $\cup$ centroid.values[k]
        Create condition k as $attribute_k \in SetOfValues$

    == NUMERIC ATTRIBUTES ==
    else
        low_value = centroid.minValue
        high_value = centroid.maxValue
        Create condition k as $attribute_k \in [low\ value,\ high\ value]$
    end if
    Add condition k to the RuleConditionPart
end for
END
```

Friedman Test

- Non-parametric **statistical test** for evaluating the differences between more than two related sample means
  - Performances of **k classifiers** across **N target datasets** (average ranks)

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$$

R_j: average rank of j-th algorithm on i-th dataset

  - **Null hypothesis** (all classifiers perform the same and any observed differences are merely random) rejected if $F_F > F_{critical}$ **(k-1,(k-1)*(N-1))**

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

statistic distributed according to the F-distribution with k–1 and (k–1)*(N–1) degrees of freedom

Nemenyi Post-Hoc Test

- Evaluates the relative performance of all classifiers to each other

- The performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference CD

$$CD = q_\alpha \sqrt{k(k+1)/6N}$$

critical values q_a are those of the Studentized range statistic divided by $\sqrt{2}$ with a significance level of α and k degrees of freedom