COMPUTATIONAL INTELLIGENCE

LABORATORY CLASSES

Implementation a simplistic version of
the AGDS network for some inference reasons

AGH

Adrian Horzyk

IMPLEMENTATION OF THE SIMPLISTIC AGDS OR AANG

Implement the simplistic version of an AGDS structure for Iris data.

You can but there is no need to use AVB-trees or AVB+trees to create it.
You can use sorted lists or tables of Value Vertices instead of AVB+trees.

You can use hash-tables, classic sorting algorithms, sorted lists, dictionaries,
but the result should be similar and correct.

You don't need to present your results or a network structure graphically,

Only text form of results is required.

Independently from your choice of implementation AGDS use the selected one to:

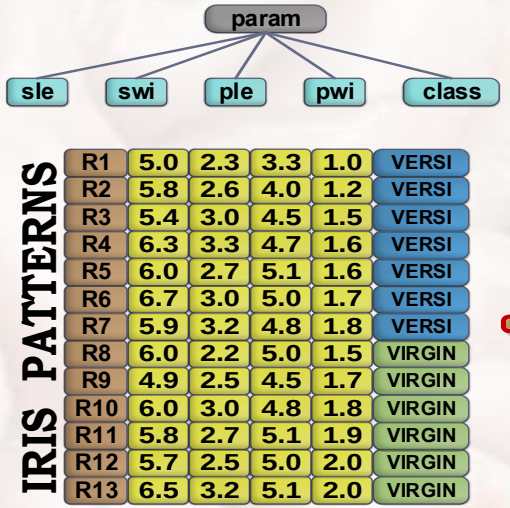
1. Find the most similar object(s) to the given object or a group of objects.
2. Find the most similar object(s) to any combination of input data given on the input.
3. Using associations generate the list of all objects sorted by the similarity to the give object (Task 1) or any combination of input data (Task 2).
4. Try to implement a classifier using AGDS structure.
5. Compute min, max, average, medians using classic algorithms and computed the time to similar operations on AGDS structures.
6. Implement knowledge-based inferences on AGDS structures.

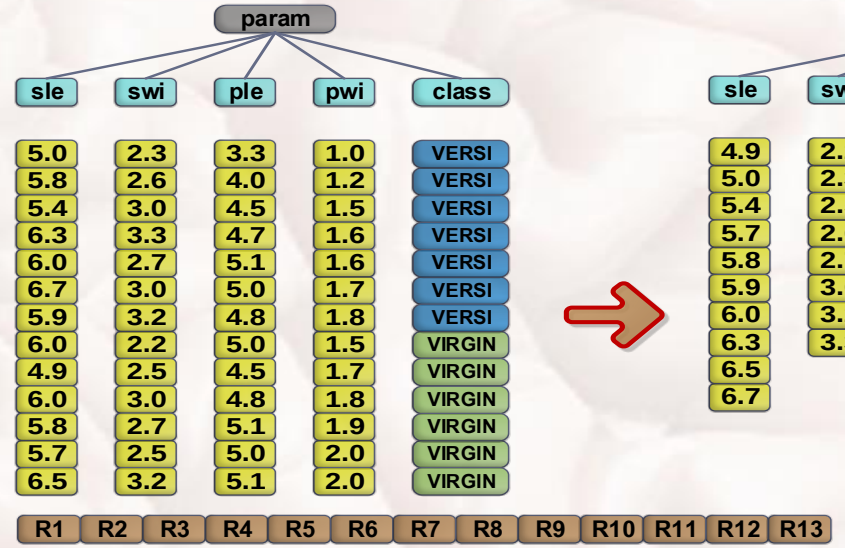
You can create this graph in the following steps for a given data set:

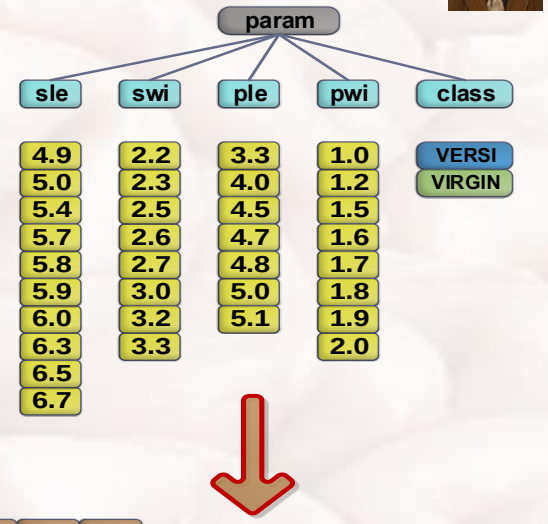
ASSOCIATIVE TRANSFORMATION

IRIS PATTERNS

param	sle	swi	ple	pwi	class
R1	5.0	2.3	3.3	1.0	VERSI
R2	5.8	2.6	4.0	1.2	VERSI
R3	5.4	3.0	4.5	1.5	VERSI
R4	6.3	3.3	4.7	1.6	VERSI
R5	6.0	2.7	5.1	1.6	VERSI
R6	6.7	3.0	5.0	1.7	VERSI
R7	5.9	3.2	4.8	1.8	VERSI
R8	6.0	2.2	5.0	1.5	VIRGIN
R9	4.9	2.5	4.5	1.7	VIRGIN
R10	6.0	3.0	4.8	1.8	VIRGIN
R11	5.8	2.7	5.1	1.9	VIRGIN
R12	5.7	2.5	5.0	2.0	VIRGIN
R13	6.5	3.2	5.1	2.0	VIRGIN

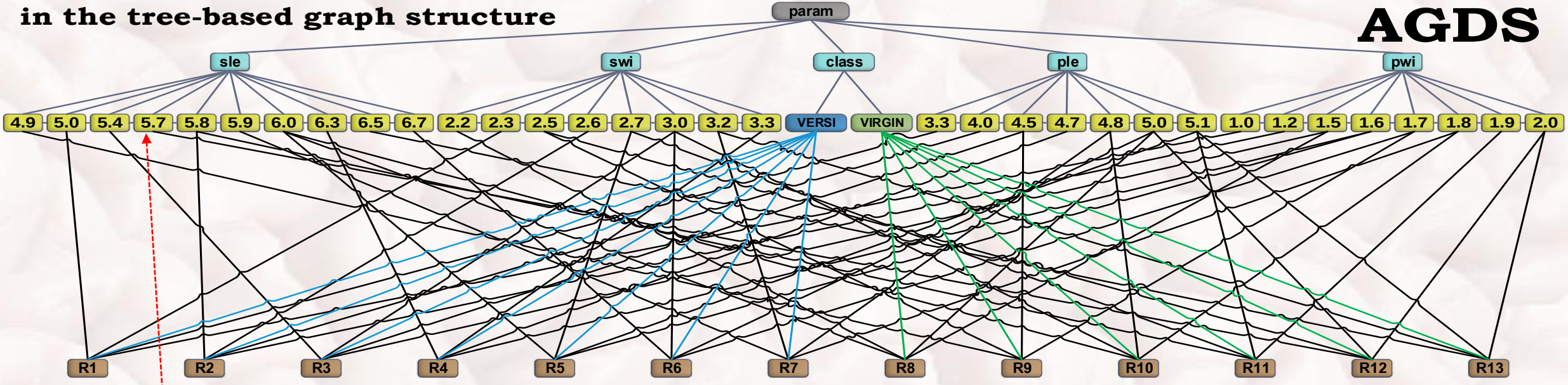






IRIS PATTERNS in the tree-based graph structure

AGDS



To represent Value Nodes, use sorted lists or tables together with the half-search algorithm or (the more ambitious solution: with the AVB+trees) which will allow you to find two closest values when the searched one will not be found. Standard hash-tables and sorted-lists have no such functionality!

WEGIHTS COMPUTATION

AGDS nodes representing neighboring (subsequent) values of each attribute a_k are connected and the weight of this connection (edge) is computed after the following formula:

$w_{v_i^{a_k},v_j^{a_k}} = 1 - \frac{|v_i^{a_k} - v_j^{a_k}|}{r^{a_k}}$ weights do not need to be stored in this structure but computed on demand when necessary. In this case, you don't need to update weights when adding new objects to this structure.

where

$v_i^{a_k}$, $v_j^{a_k}$ - are values represented by the neighboring attribute nodes, which are connected by an edge in the AGDS graph,

$r^{a_k} = v_{max}^{a_k} - v_{min}^{a_k}$ - is the current range of values of the attribute a_k.

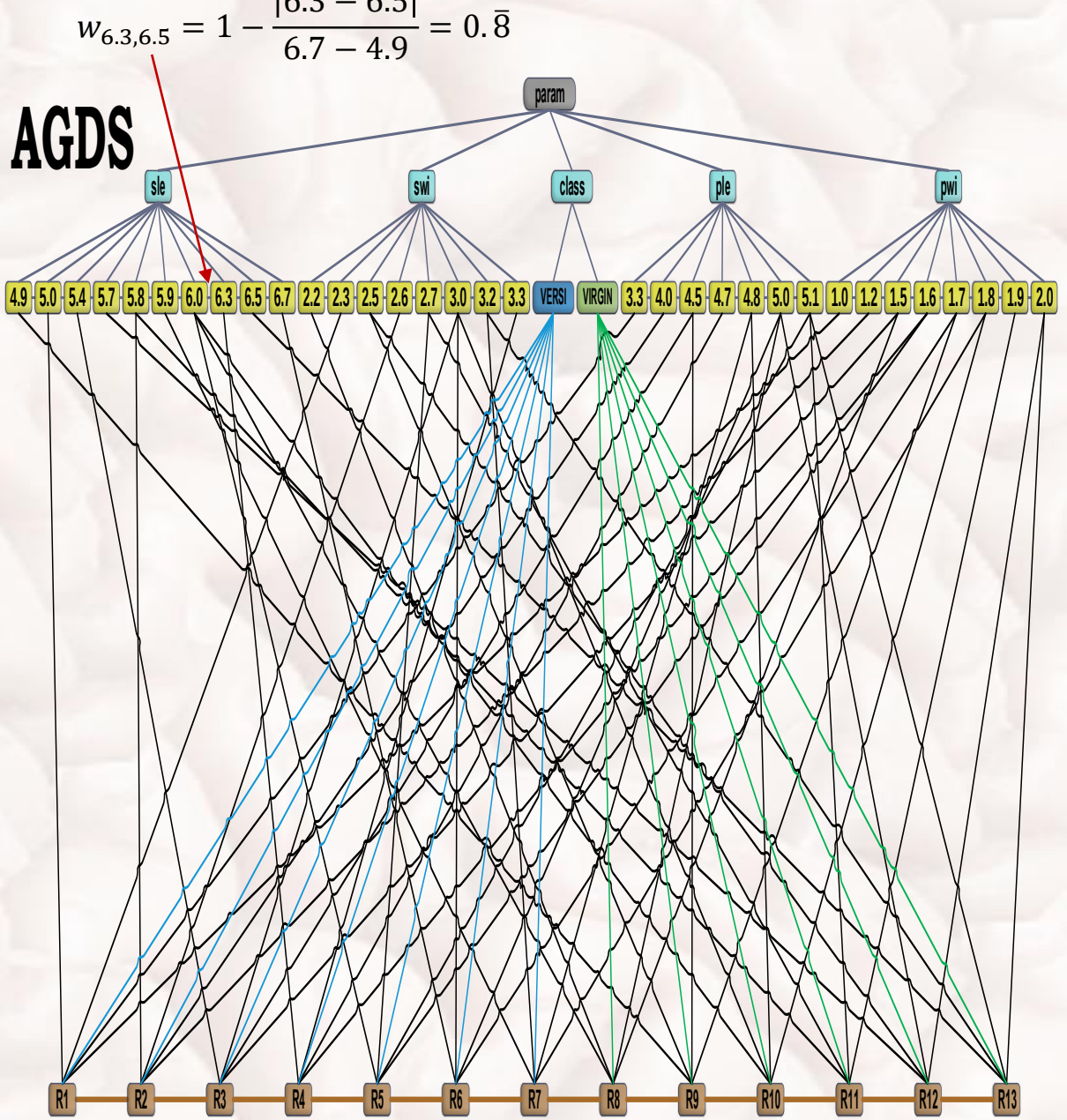
The weight of the connection from the value node $v_i^{a_k}$ of the attribute a_k to the object node R_m is determined after the number of occurrences $N_i^{a_k}$ of this value ($v_i^{a_k}$) in all objects:

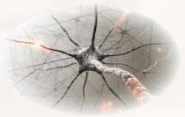
$$w_{v_i^{a_k},R_m} = \frac{1}{N_i^{a_k}} = \frac{1}{\|v_i^{a_k}\|}$$

These numbers ($N_i^{a_k} = \|v_i^{a_k}\|$) are stored in the individual value nodes of each attribute. This number is equal to the number or all connections of this value node to all object nodes if there are no duplicated objects in the table used to create the AGDS structure.

In the opposite direction, the weights of connections from the object nodes to the value nodes are always equal to one:

$$w_{R_m,v_i^{a_k}} = 1$$

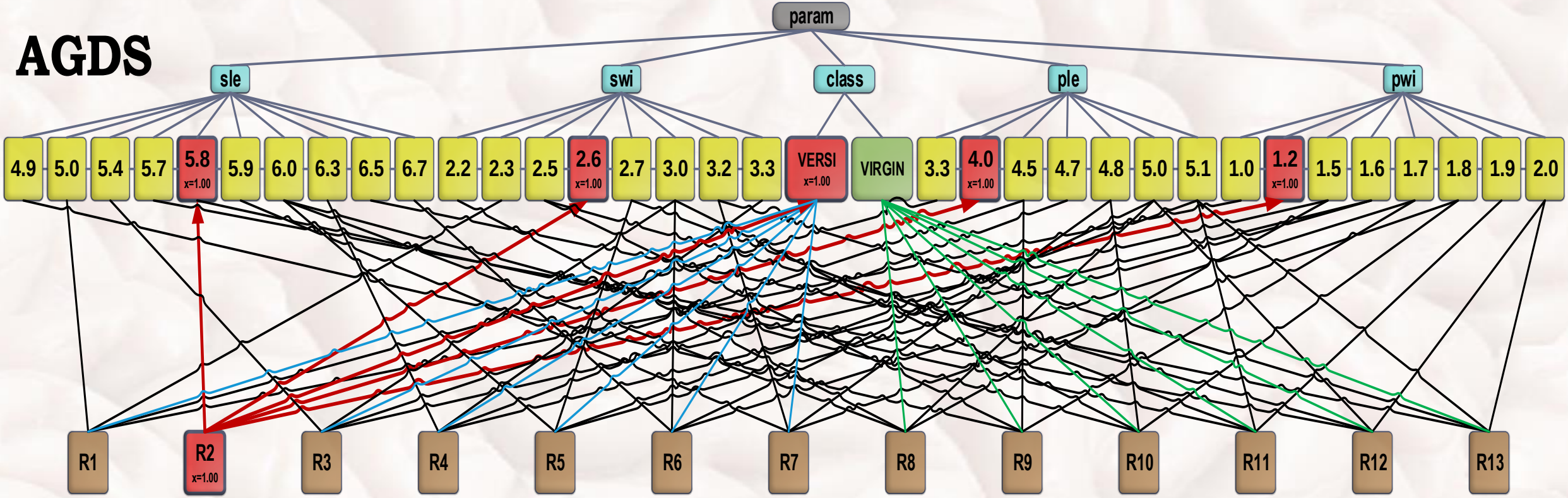




ASSOCIATIVE INFERENCE USING AGDS STRUCTURES

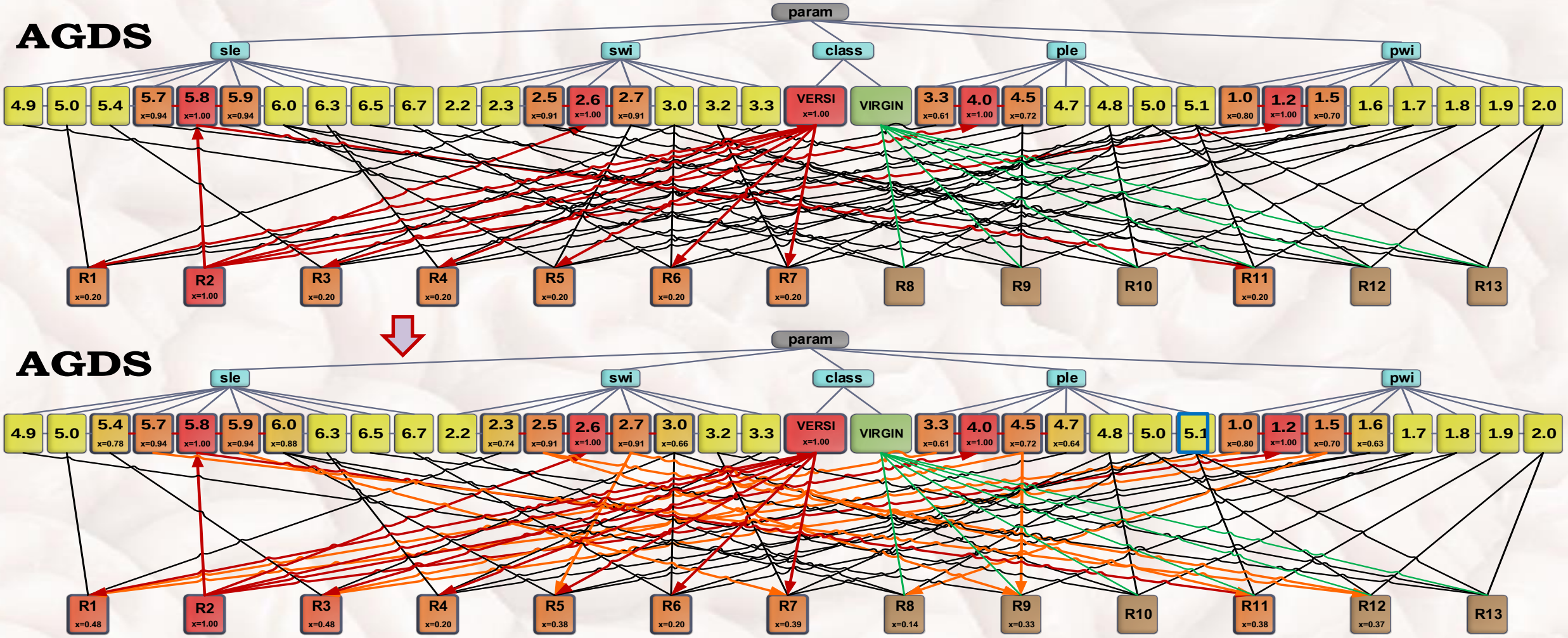
Associative data structures AGDS can be now used for associative inference, which is based on moving along the connections to the connected nodes and computing some values in these nodes on the basis of the send values multiplied by weights of these connections. In such a way we get the information about e.g. similarity of objects represented by other nodes of the same kind or about the objects that satisfy some given conditions defined by the represented attribute values. Let's use our AGDS graph created for 13 Irises for such inference looking for objects (Irises) Rx which are most similar to R2.

1. **We start in the node R2 which assumes the similarity value x=1.0, because this node is 100% similar to itself.**
2. **Next, we assign values x of the connected nodes representing the following values: 5.8, 2.6, VERSI, 4.0, and 1.2 by multiplying the value coming from the node R2 with the connection weights, which are equal 1.0. So, as a result, we achieve x=1.0 for all these connected nodes.**



ASSOCIATIVE INFERENCE USING AGDS STRUCTURES

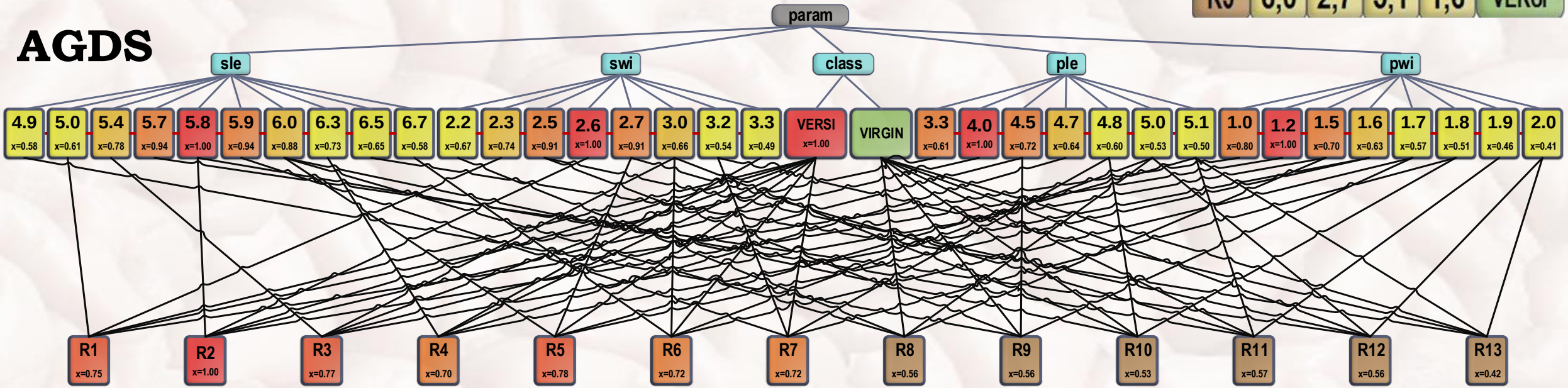
3. Subsequently, the values computed for these nodes are multiplied by next connection weights and send to the neighbor connected value nodes, for which we also compute their similarity values x.
4. Similarly, we compute the similarity values x for connected object nodes with regards to the necessity to add the passed weighted values to the sums already stored in these nodes, e.g. for the node R3 we compute x = 1.0 * 0.2 + 0.72 * 0.2 + 0.7 * 0.2 = 0.48



ASSOCIATIVE INFERENCE USING AGDS STRUCTURES

5. Finally, when we go through all the connected (associated) values nodes computing theirs values of similarities by multiplying the sender similarity values by connection weights. We also computed weighted sums for all object nodes, where these weights are here equal w = 1/5 = 0.2. The computed similarity values for the nodes Rx can be used to compare and designate the most similar objects to the object R2: R5 (78%), R3 (77%), R1 (75%), …

R1	5,0	2,3	3,3	1,0	VERSI
R2	5,8	2,6	4,0	1,2	VERSI
R3	5,4	3,0	4,5	1,5	VERSI
R4	6,3	3,3	4,7	1,6	VERSI
R5	6,0	2,7	5,1	1,6	VERSI



It is also worth noting that AGDS graphs are not neural structures, so we are not obligated to multiply the nodes similarity values by connection weights, but we can also use another formulas, e.g. we can subtract the complement of the connection weight value from the similarity value represented by the sender: x' = x - (1 – w).

Consequently, we get another measure of similarity represented by the value nodes and object nodes.

We can also used DASNG graph formulas to calculate weights between value nodes and object nodes to emphasize rarity of the value using the frequency of connections coming out from value nodes: w = 1 / the number of outgoing connections.

Implementation of knowledge-based inferences and similarity computations on AGDS structures

The **charging level x** of the internally stimulated node is defined as a weighted sum (as in the 2nd ANN generation):

$$x_n = \sum_{k=1}^{s_n} x_k \cdot w_k$$

Reciprocal edges are created between value nodes $V_i^{a^k}$ and $V_j^{a^k}$ representing similar values $v_i^{a^k}$ and $v_j^{a^k}$ of the same attribute a^k and forward stimuli in both directions with the same **weight**:

$$w_{v_i^{a^k}, v_j^{a^k}} = 1 - \frac{\left| v_i^{a^k} - v_j^{a^k} \right|}{r^{a^k}}$$

where

$$r^{a^k} = v_{max}^{a^k} - v_{min}^{a^k}$$

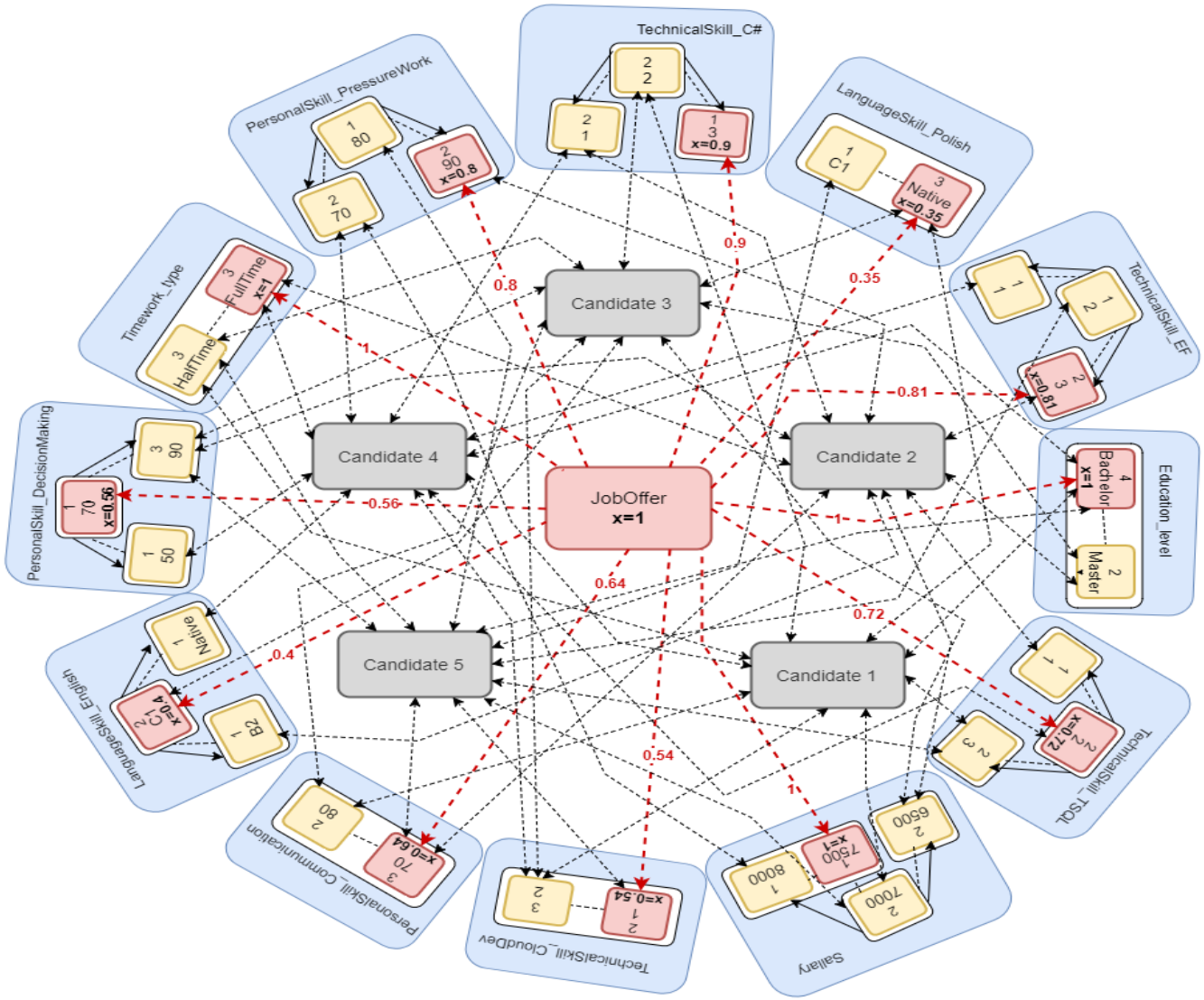
is a variation range of values of the attribute a^k.

The **weight** of the edge for the signal passing from the value node $V_i^{a^k}$ to the object node O_n can be calculated after:

$$w_{O_m, O_n} = \frac{1}{\theta_n}$$

The stimuli passing through the edge in the opposite direction

$$w_{O_n, v_i^{a^k}} = 1, w_{O_n, O_m} = 1$$



where the **threshold θ_n** is the number of values and objects that define the object nodes O_n and activate this node.

Bibliography and References

http://home.agh.edu.pl/~horzyk/lectures/ahdydci.php

ACADEMIC WEBSITE - ADRIAN HORZYK, PhD, Prof.

AGH University of Science and Technology in Cracow, Poland
Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering
Department of Automatics and Biomedical Engineering, Field of Biocybernetics

AGH

LECTURES

- Introduction to Artificial, Cognitive and Computational Intelligence
- Introduction to Neural Networks
- Models of Neurons
- Construction and Learning Strategies
- Overview of Neural Network Models
- Multilayer Perceptron MLP
- Radial Basis Function Networks RBF
- Support Vector Machine SVM
- Reinforcement Learning
- Deep Learning DL
- Unsupervised Training and Networks
- Self Organizing Maps SOM
- Recurrent Neural Networks
- Associative and Semantic Memories
- Associative Neural Systems

COMPUTATIONAL INTELLIGENCE

is already under construction and will be available at the spring semester in 2016/2017.

This course will include 28 lectures, 14 laboratory classes and 14 project classes.

What is this course about?

This course is intended to give students a broad overview and deep knowledge about popular solutions and efficient neural network models as well as to learn how to construct and train intelligent learning systems in order to use them in everyday life and work. During the course we will deal with the popular and most efficient models and methods of neural networks, fuzzy systems and other learning systems that enable us to find specific highly generalizing models solving difficult tasks. We will also tackle with various CI and AI problems and work with various data and try to model their structures in such a way to optimize operations on them throughout making data available without necessity to search for them. This is a unique feature of associative structures and systems. These models and methods will allow us to form and represent knowledge in a modern and very efficient way which will enable us to mine it and automatically draw conclusions. You will be also able to understand solutions associated with various tasks of motivated learning and cognitive intelligence.