

Associative Graph Data Structures Used for **Acceleration** of K Nearest Neighbor Classifiers



Adrian Horzyk
horzyk@agh.edu.pl



**AGH University of
Science and Technology**
Krakow, Poland



Krzysztof Gołdon
krzysztofgoldon@gmail.com



ICANN18

October, 2018, Rhodes, Greece

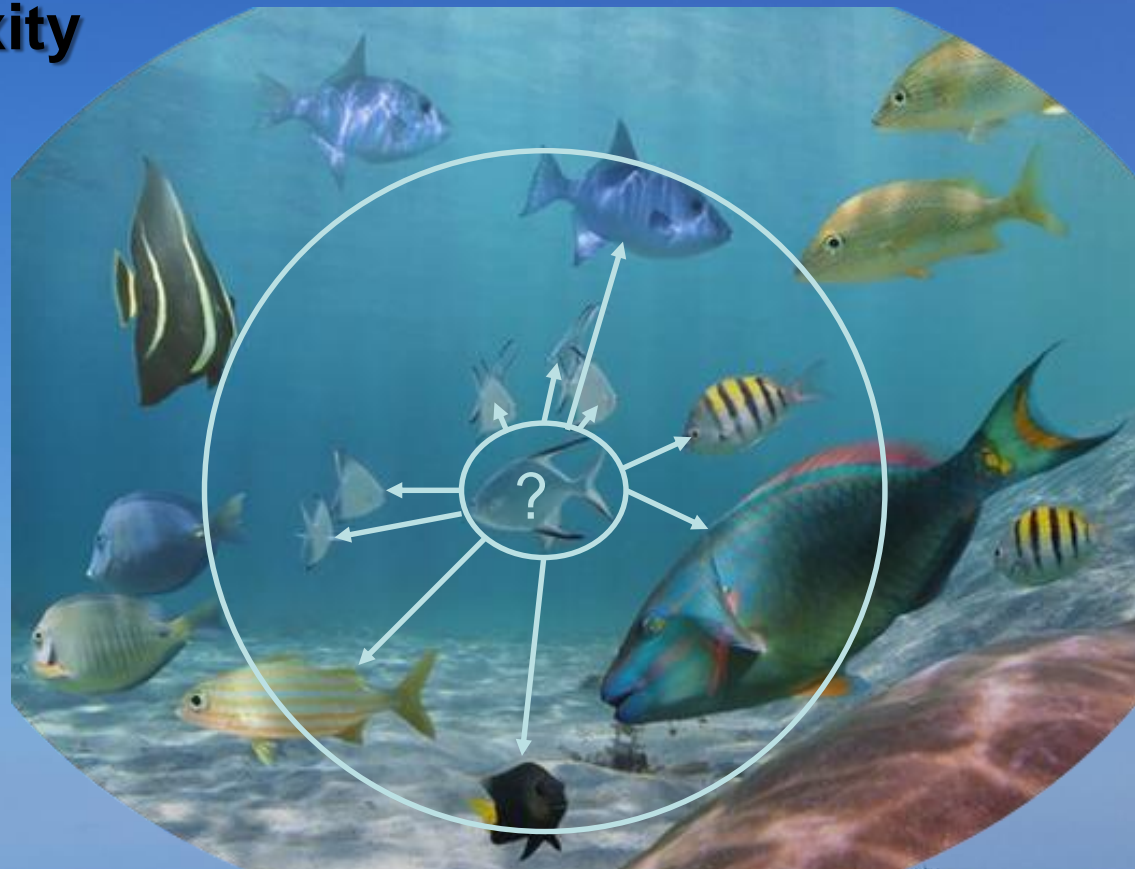
Drawbacks of KNN Classifiers

KNN classifiers are robust to noisy training data and very easy to implement, but they are:

- » **Lazy** because they do not create a computational model,
- » **High computational cost** because they require to compute the distance of each classified sample to all training data (linear computational complexity for each classified sample)

while other classifiers usually have **constant computational complexity** when classifying samples.

Therefore, KNN cannot be efficiently used to Big Data!



Why Storing Data in the Tables?

We mostly use tables to store, organize and manage data in computer science:

SAMPLE OBJECTS	ATTRIBUTES				CLASS LABEL
	SEPAL LENGTH	SEPAL WIDTH	PETAL LENGTH	PETAL WIDTH	
O1	5.4	3.0	4.5	1.5	Versicolor
O2	6.3	3.3	4.7	1.6	Versicolor
O3	6.0	2.7	5.1	1.6	Versicolor
O4	6.7	3.0	5.0	1.7	Versicolor
O5	6.0	2.2	5.0	1.5	Virginica
O6	5.9	3.2	4.8	1.8	Versicolor
O7	6.0	3.0	4.8	1.8	Virginica
O8	5.7	2.5	5.0	2.0	Virginica
O9	6.5	3.2	5.1	2.0	Virginica

However, common relationships like minima, maxima, identity, similarity, neighborhood, number of duplicates must be found in loops that search for them and evaluate various conditions.

The more data we have, the longer time requirements we face!

What can be done to achieve better efficiency?



Big Data...

Associate!

Big Problem?

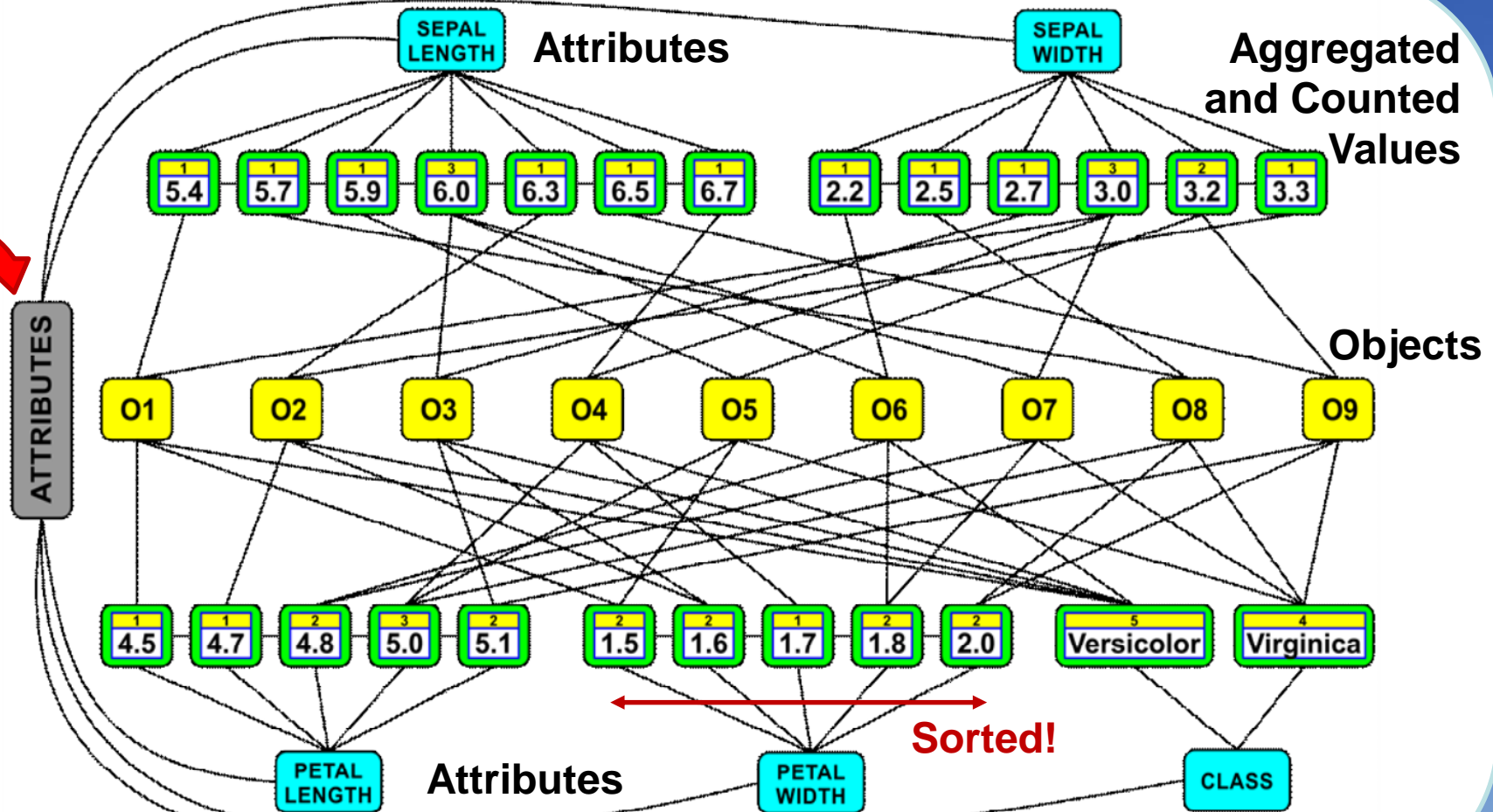
Associative Graph Data Structure (AGDS)

AGDS links related data of various kinds horizontally and vertically:



ATTRIBUTES					
SAMPLE OBJECTS	SEPAL LENGTH	SEPAL WIDTH	PETAL LENGTH	PETAL WIDTH	CLASS LABEL
O1	5.4	3.0	4.5	1.5	Versicolor
O2	6.3	3.3	4.7	1.6	Versicolor
O3	6.0	2.7	5.1	1.6	Versicolor
O4	6.7	3.0	5.0	1.7	Versicolor
O5	6.0	2.2	5.0	1.5	Virginica
O6	5.9	3.2	4.8	1.8	Versicolor
O7	6.0	3.0	4.8	1.8	Virginica
O8	5.7	2.5	5.0	2.0	Virginica
O9	6.5	3.2	5.1	2.0	Virginica

Brain inspired
associative
AGDS

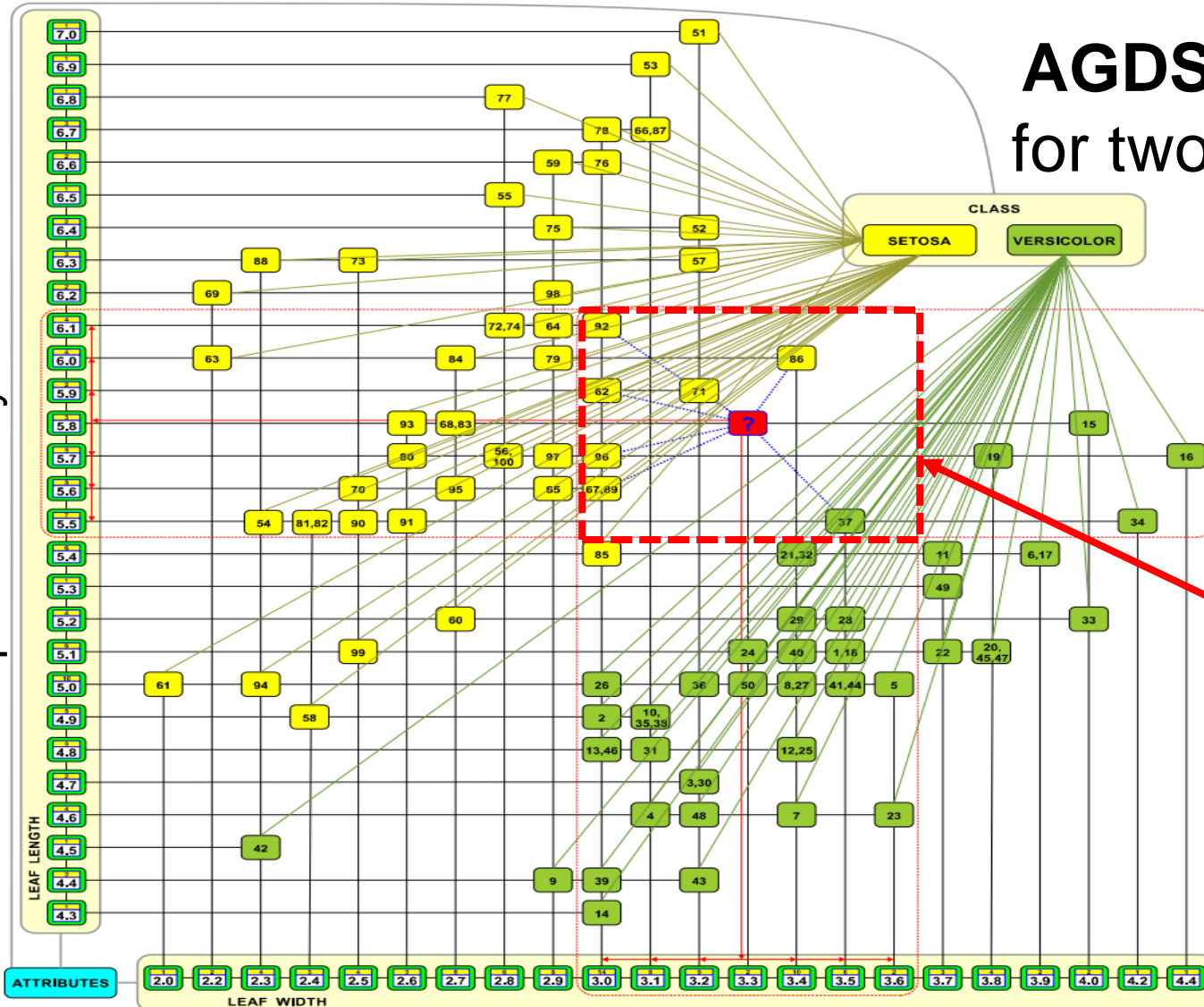


The connections represent various relations between AGDS elements like similarity, proximity, neighborhood, order, definition etc.

K Nearest Neighbors using AGDS Structures

The search is limited to a small region where neighbors are found:

100 values represented by 28 value nodes!



AGDS structure created for two selected attributes and 100 training samples of Iris data

K Nearest Neighbors

are searched locally in the neighborhood of the classified sample

We can save a lot of computational time using created associations in the AGDS!

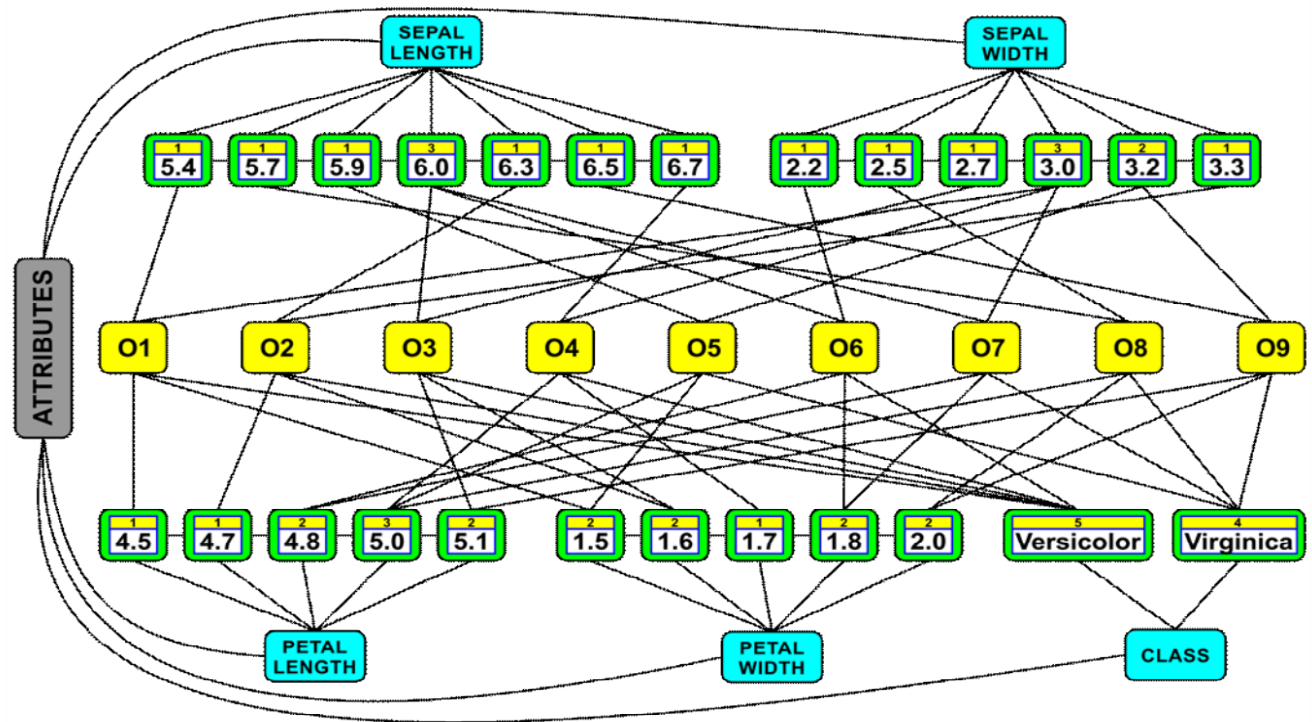
100 values represented by 22 value nodes!

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.		
2.		
3.		



1. Create an empty k-row rank table that will consist of the pointers to the k nearest neighbors and their distances to the classified object.

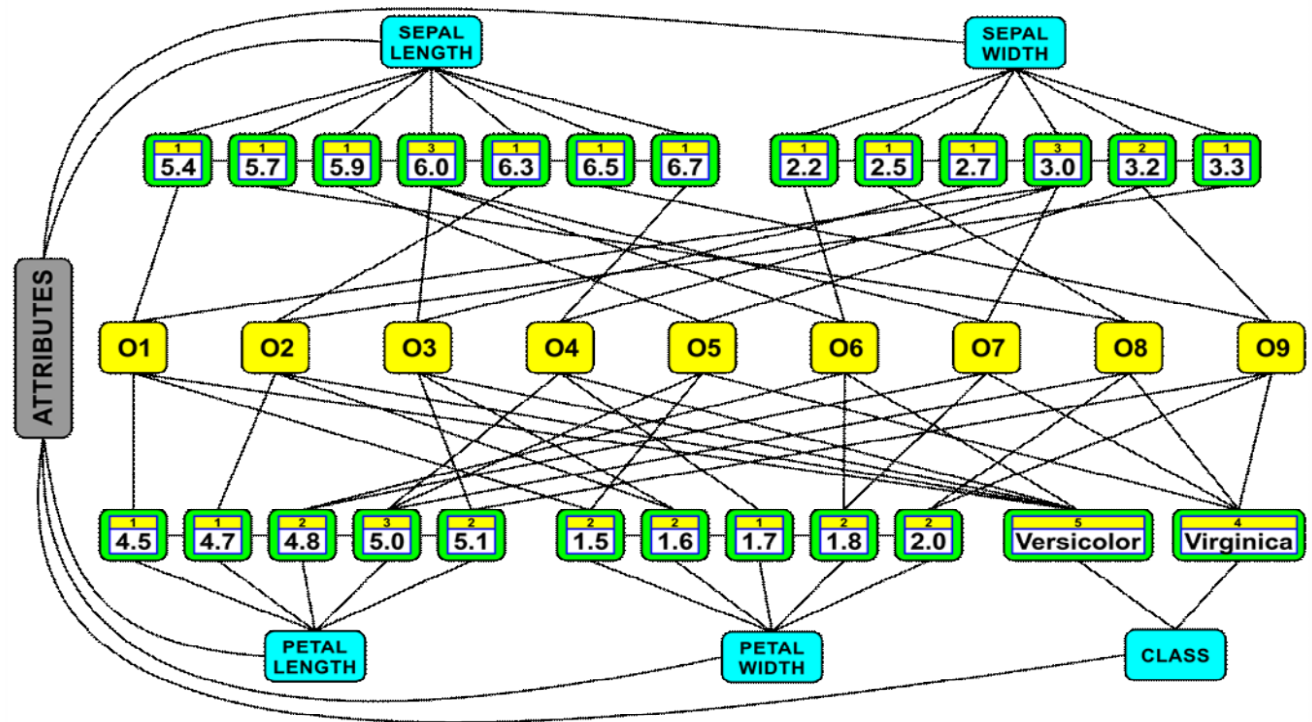
Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.		
2.		
3.		

Classify [5.7; 2.5; 4.8; 1.6]



2. For the first attribute value of the classified object, find the closest attribute value in the constructed AGDS structure.

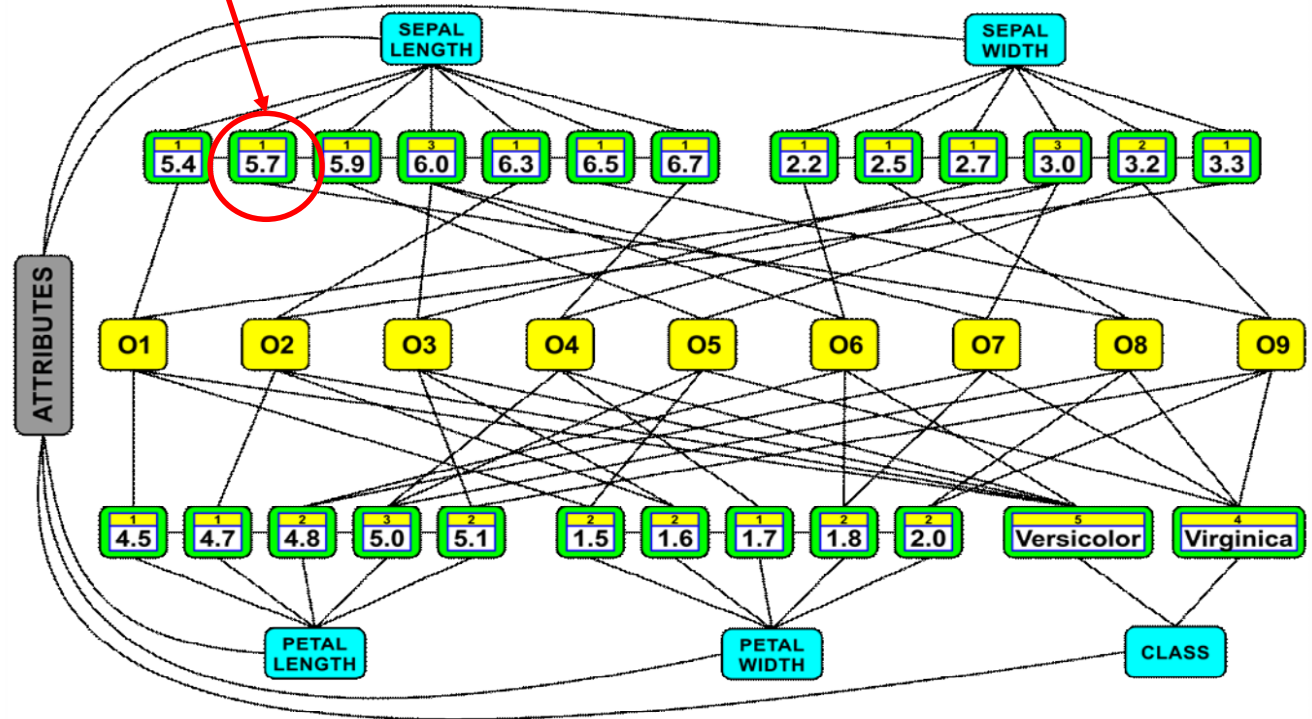
Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.		
2.		
3.		

[5.7; 2.5; 4.8; 1.6]



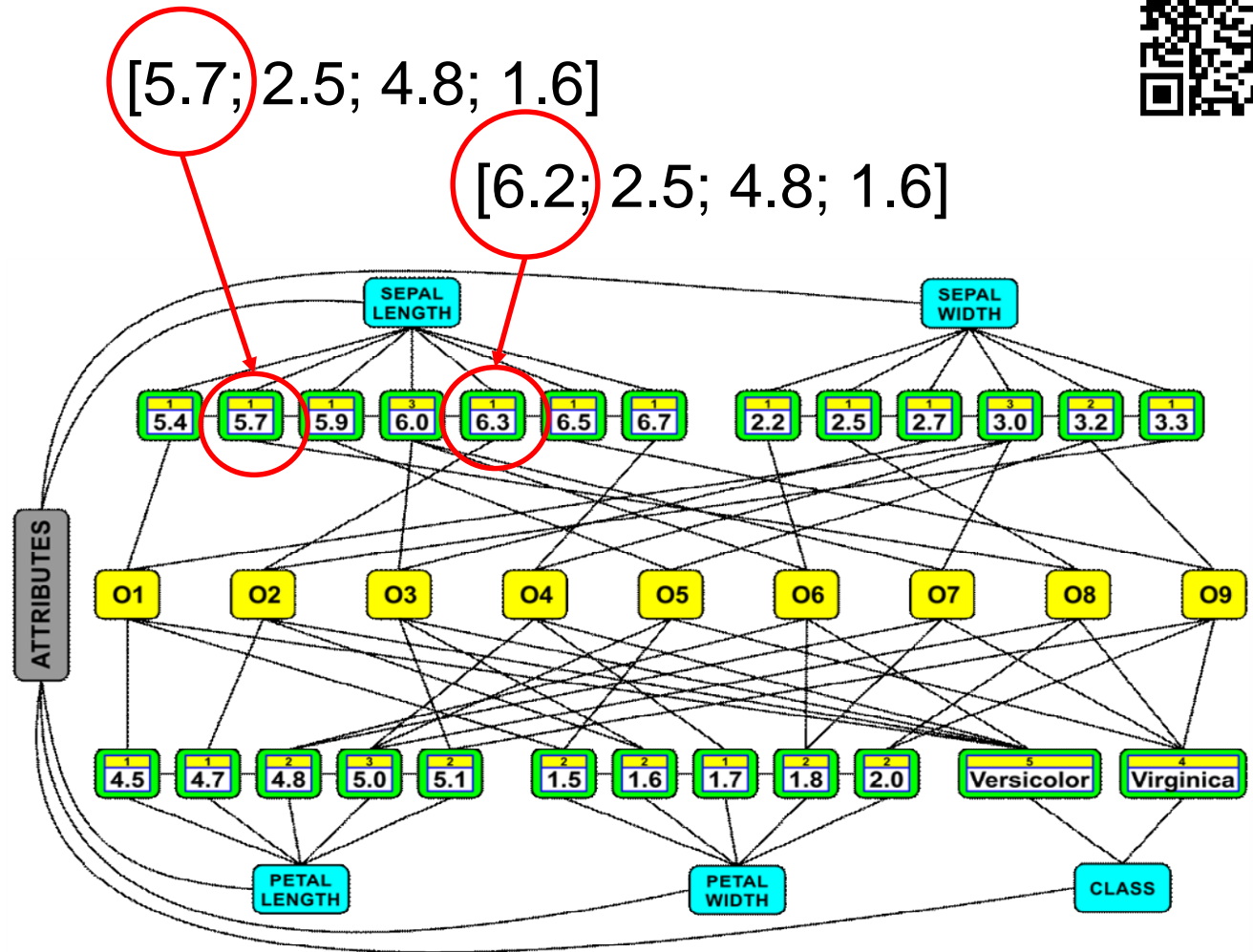
3. When the first attribute value of the classified object is represented by an existing value node of the AGDS structure, go to step 5, else go to step 4.

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.		
2.		
3.		



4. When the first attribute value of the classified object is not represented by any value node of this first attribute, then the closest value is represented by the value node representing the nearest lower or the nearest bigger value or both. Choose the nearest value or one of the nearest values and go to step 5.

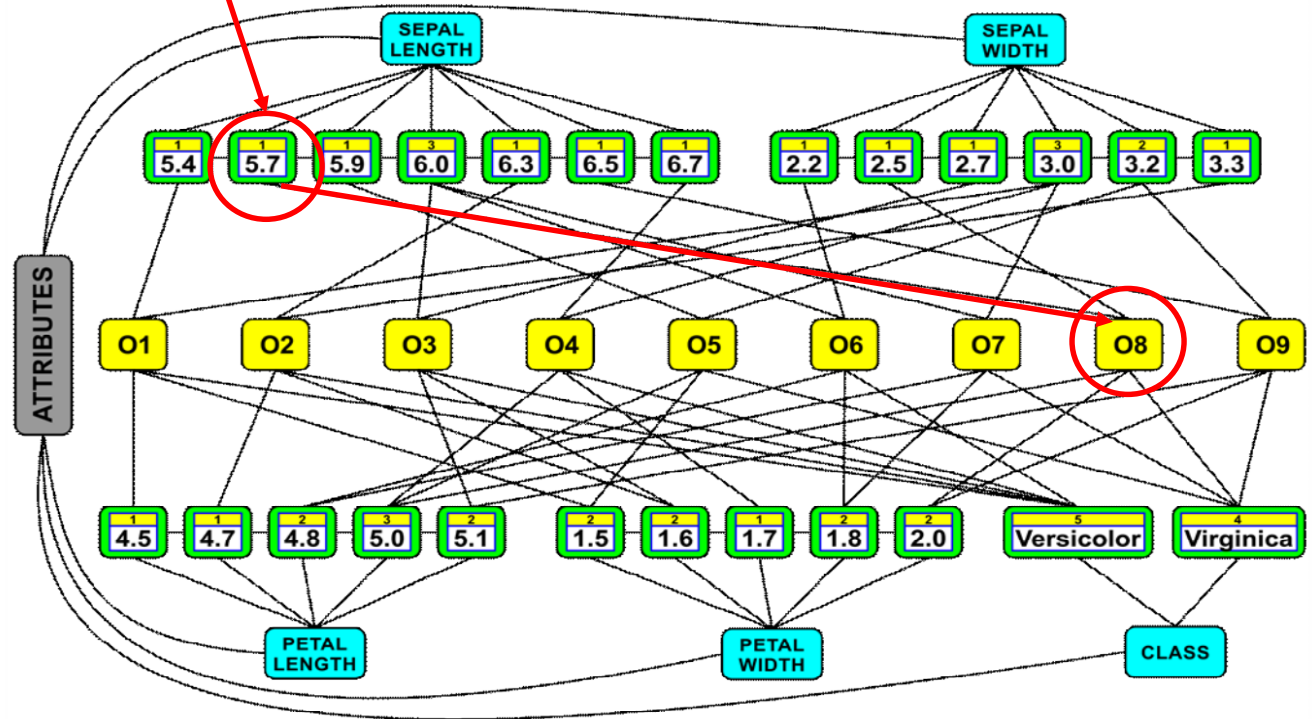
Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.		
2.		
3.		

[5.7; 2.5; 4.8; 1.6]



5. Go along all edges of the selected value node to all connected object nodes and perform step 6 for all these object nodes.

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.		
2.		
3.		

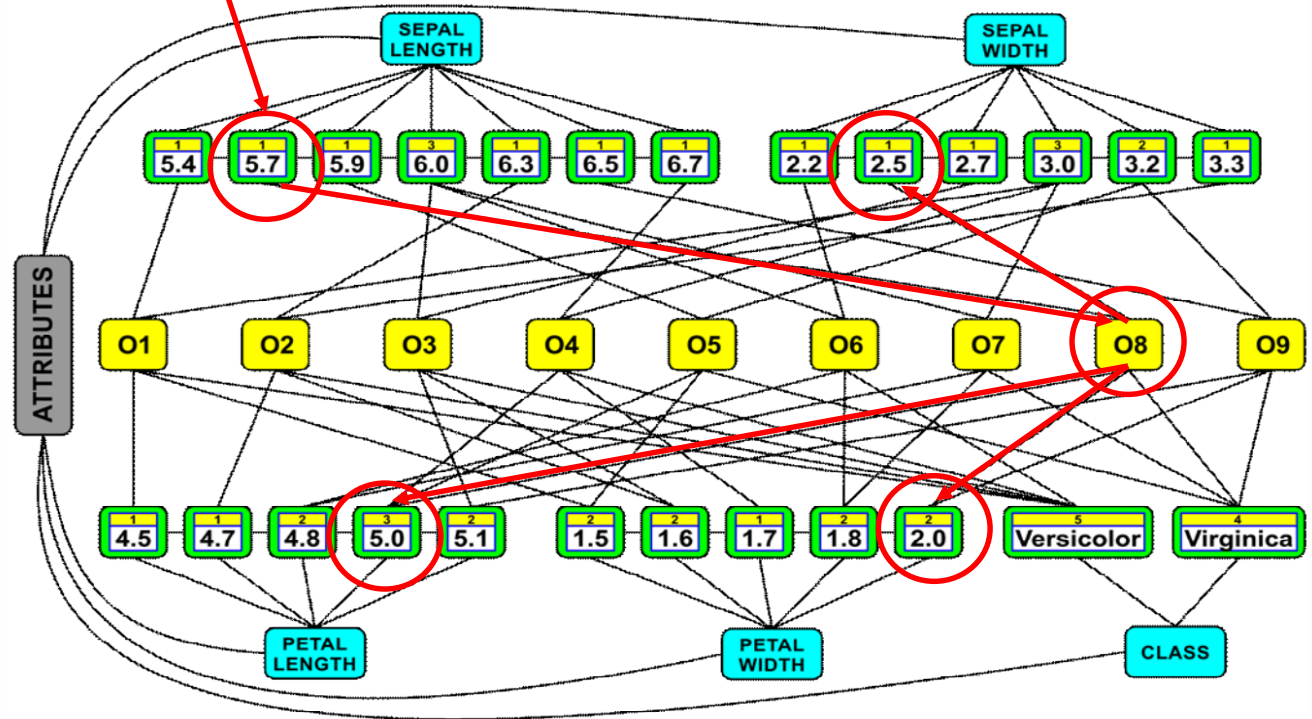
[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$

$$d_e(x, y) = 0.45$$

$$d_m(x, y) = 0.60$$



6. For the reached object node, go to all connected value nodes, except the value node from which this object node was reached, and **compute the distance** according to (1) or (2). Next, try to insert this object node to the rank table in step 7.

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.		
2.		
3.		

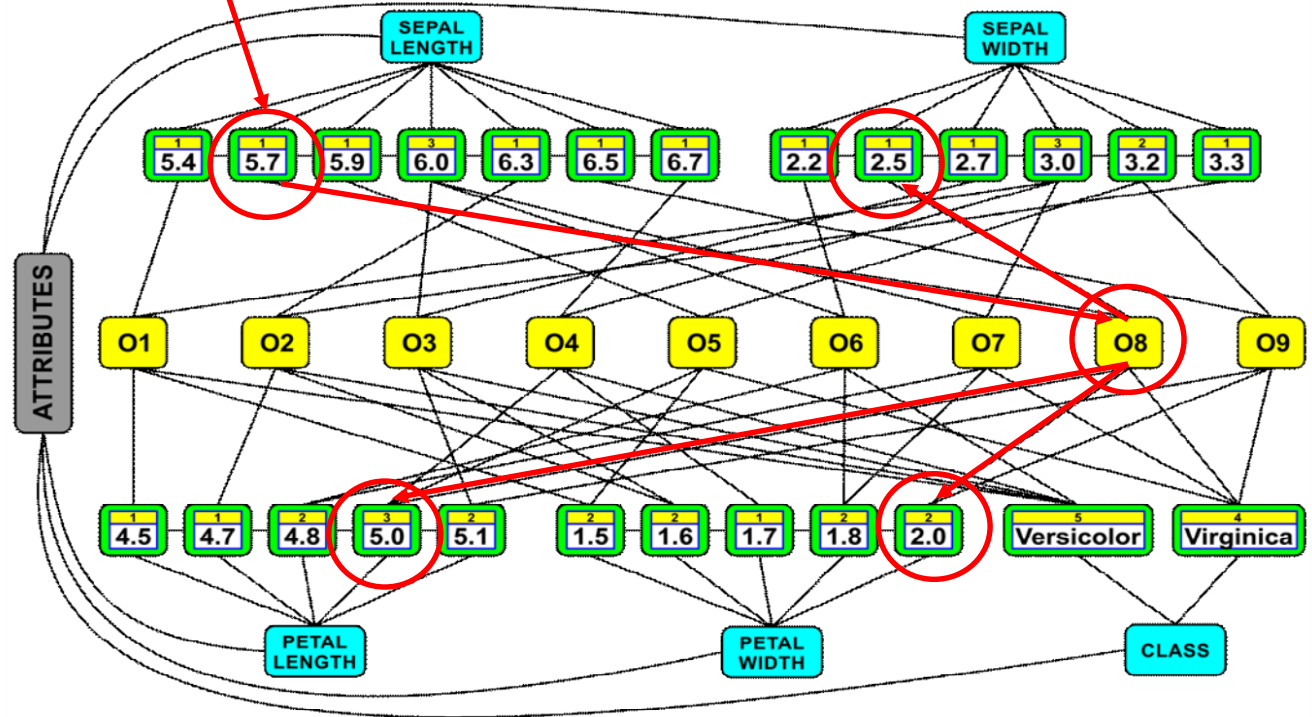
[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$

$$d_e(x, y) = 0.45$$

$$d_m(x, y) = 0.60$$



7. If the k-th row of the rank table is empty or the computed distance is shorter than the distance to the object node stored in the last (k-th) row of the rank table, go to step 8, else go to step 9.

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.	O8	0.45
2.		
3.		

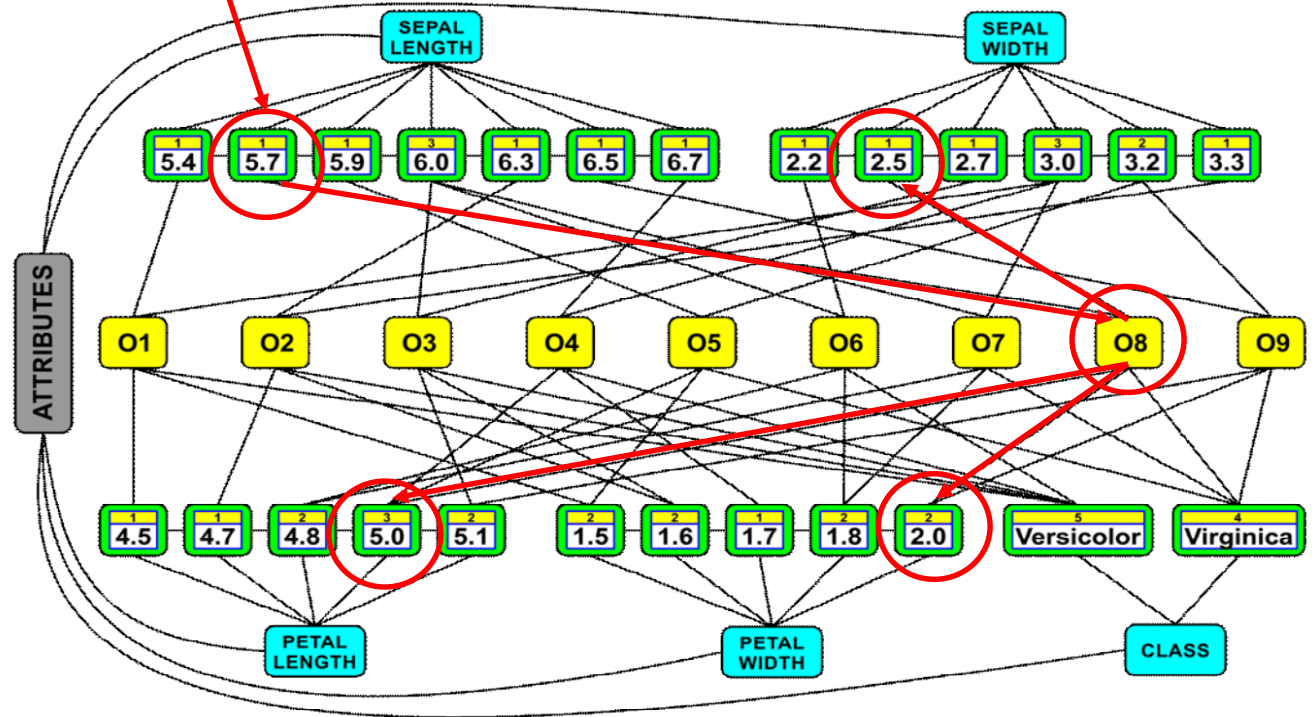
[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$

$$d_e(x, y) = 0.45$$

$$d_m(x, y) = 0.60$$



8. Insert this node and its distance to the rank table in the ascendant order (using (half) insertion sort algorithm), and if necessary (if the table is overfilled) remove the last (i.e. the most distant) object node together with its distance from this table.

Acceleration Associative Algorithm for KNN+ AGDS classifiers



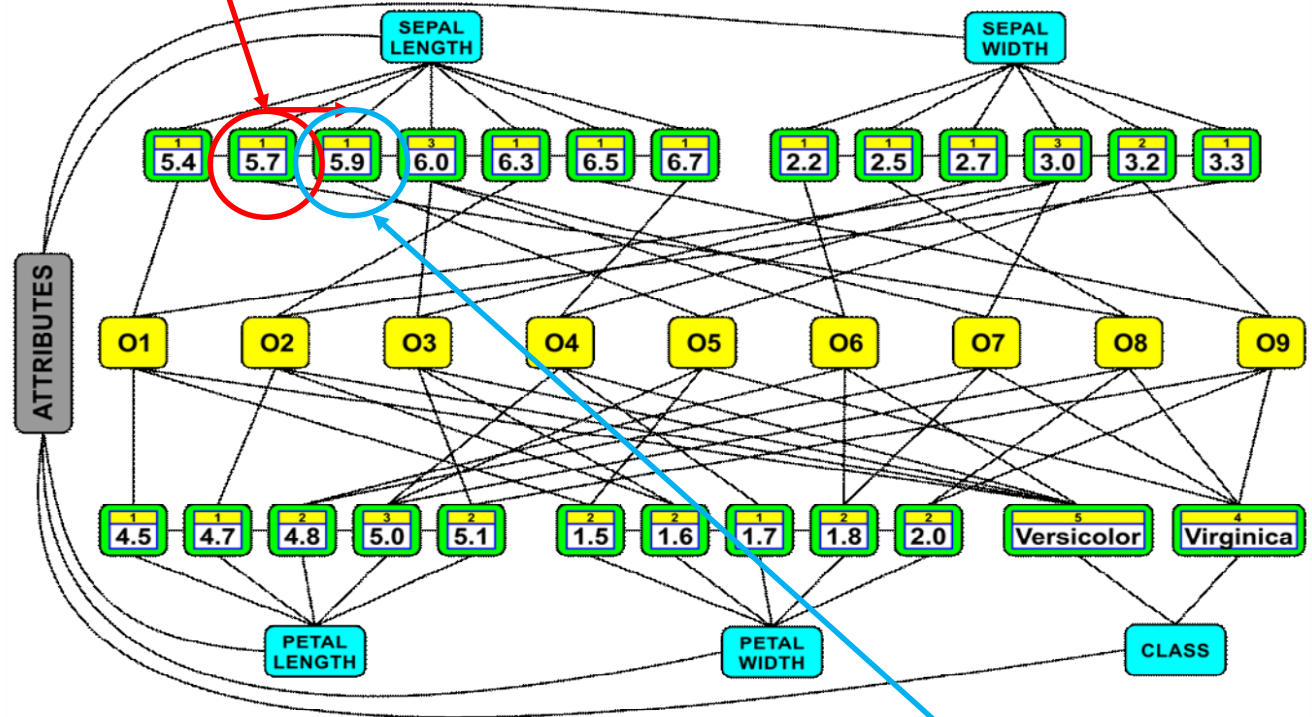
Rank table

1.	O8	0.45
2.		
3.		

[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to the **next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.	O8	0.45
2.		
3.		

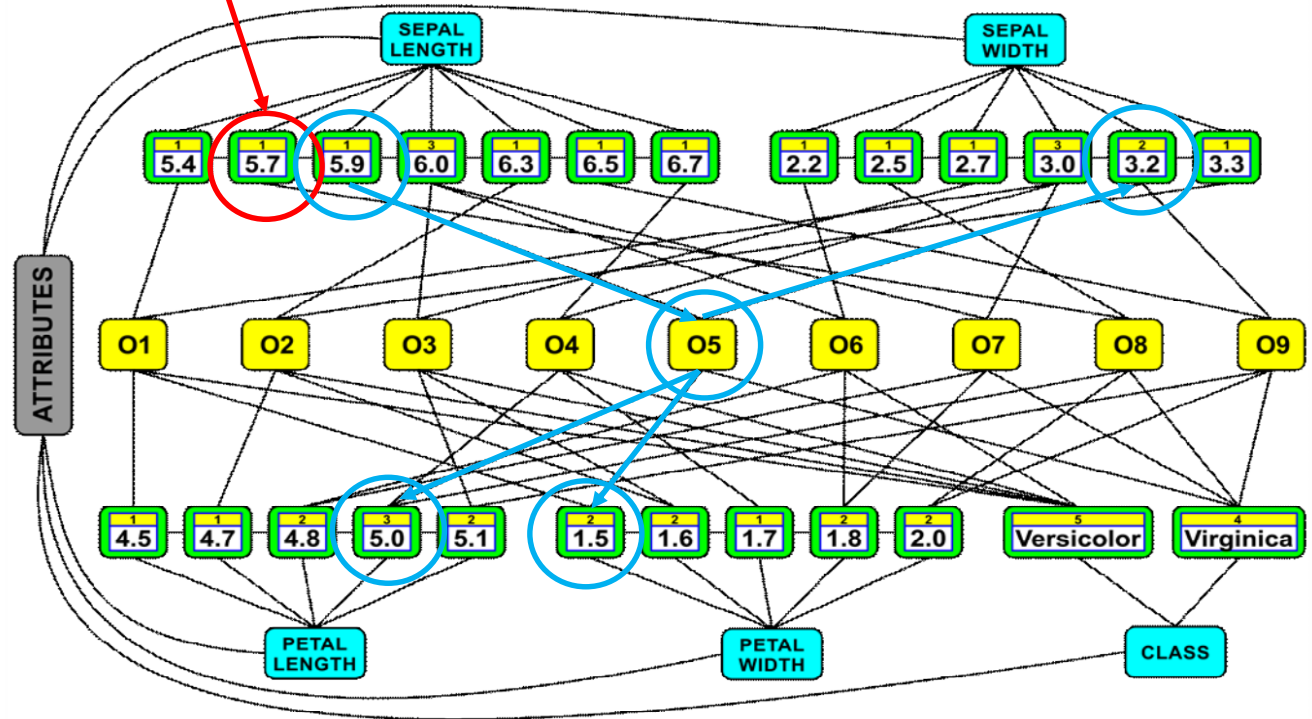
[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$

$$d_e(x, y) = 0.48$$

$$d_m(x, y) = 0.90$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to the **next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.	O8	0.45
2.	O5	0.48
3.		

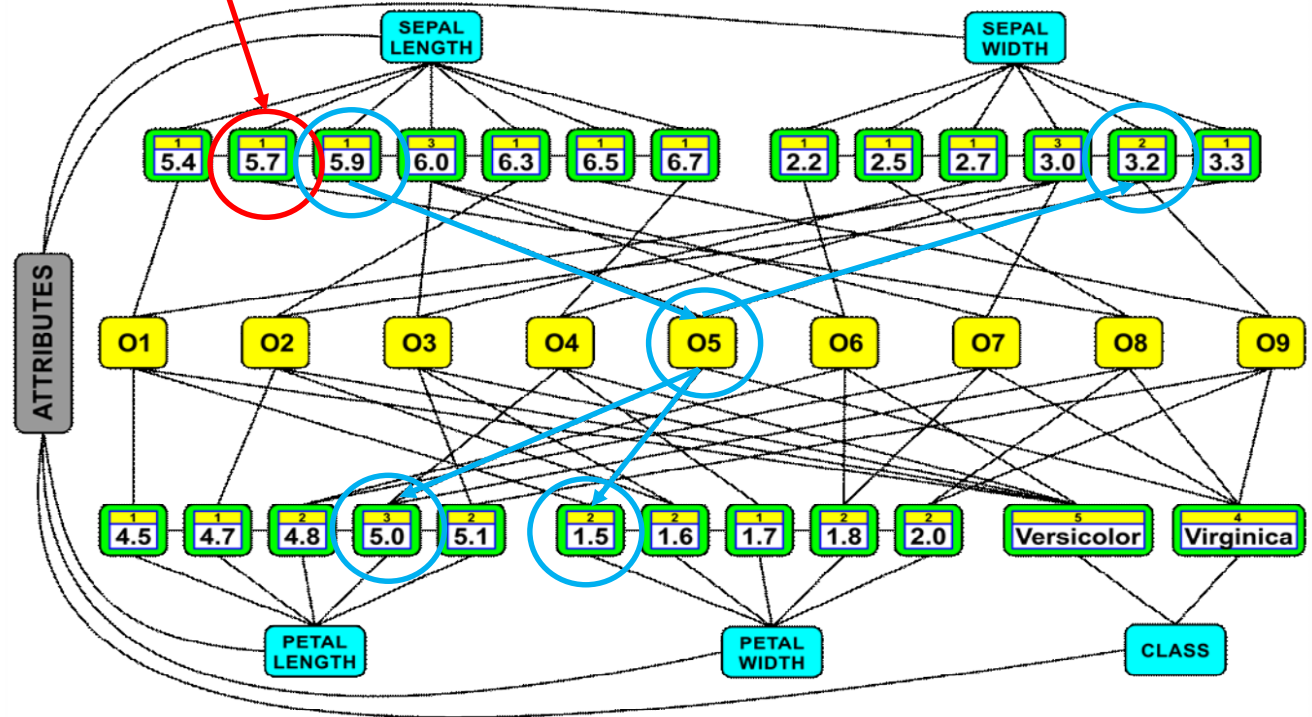
[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$

$$d_e(x, y) = 0.48$$

$$d_m(x, y) = 0.90$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to the **next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers



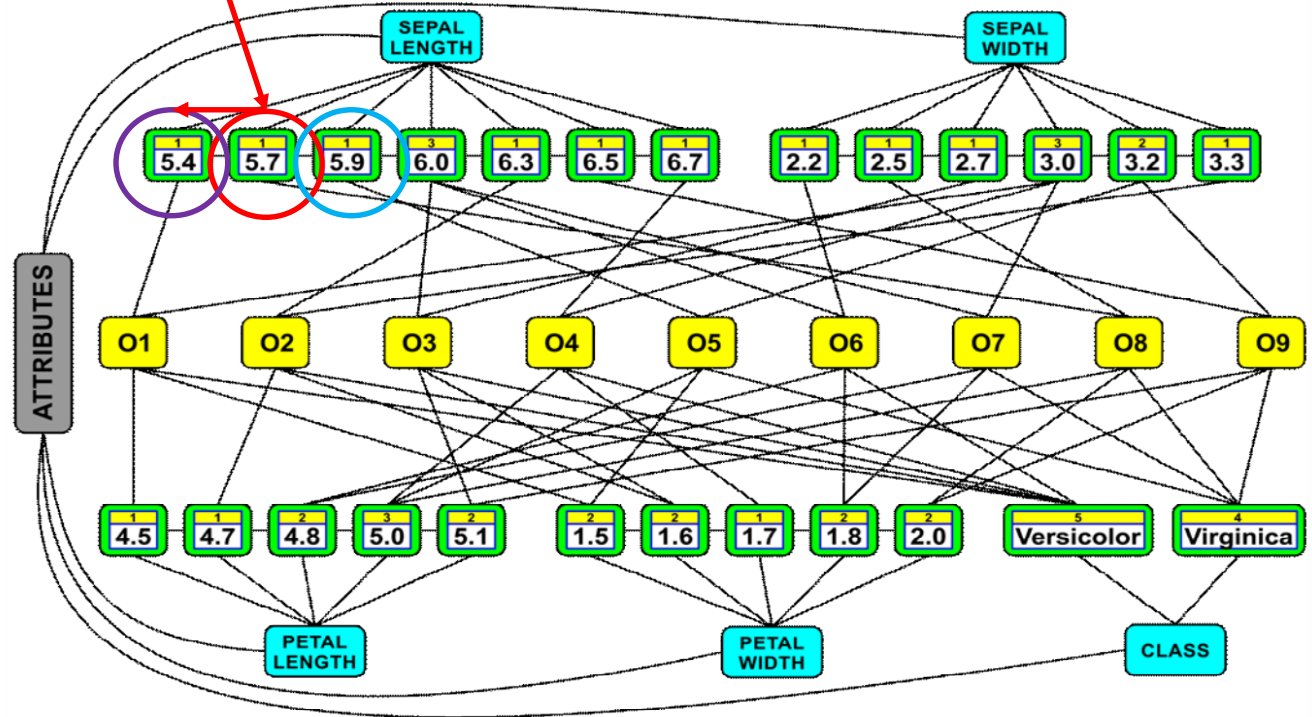
Rank table

1.	O8	0.45
2.	O5	0.48
3.		

[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to the **next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.	O8	0.45
2.	O5	0.48
3.	O1	0.66

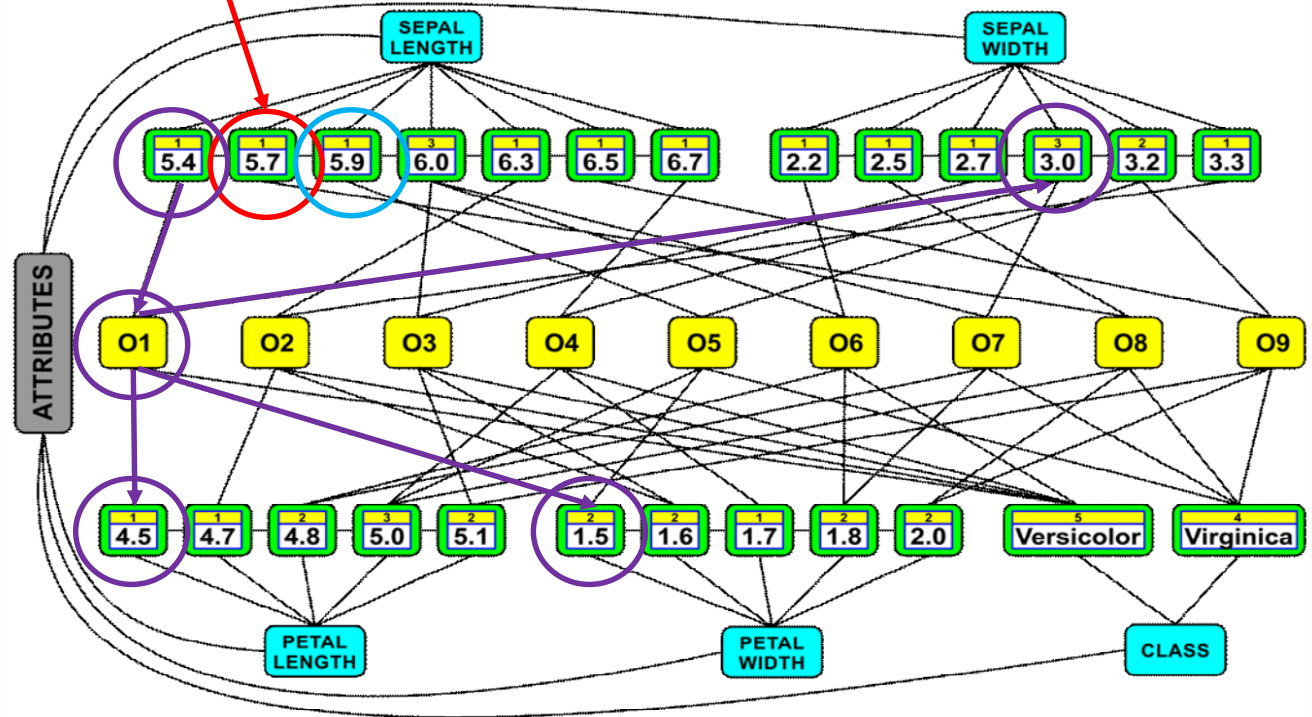
[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$

$$d_e(x, y) = 0.66$$

$$d_m(x, y) = 1.2$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to the next closest value node (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers



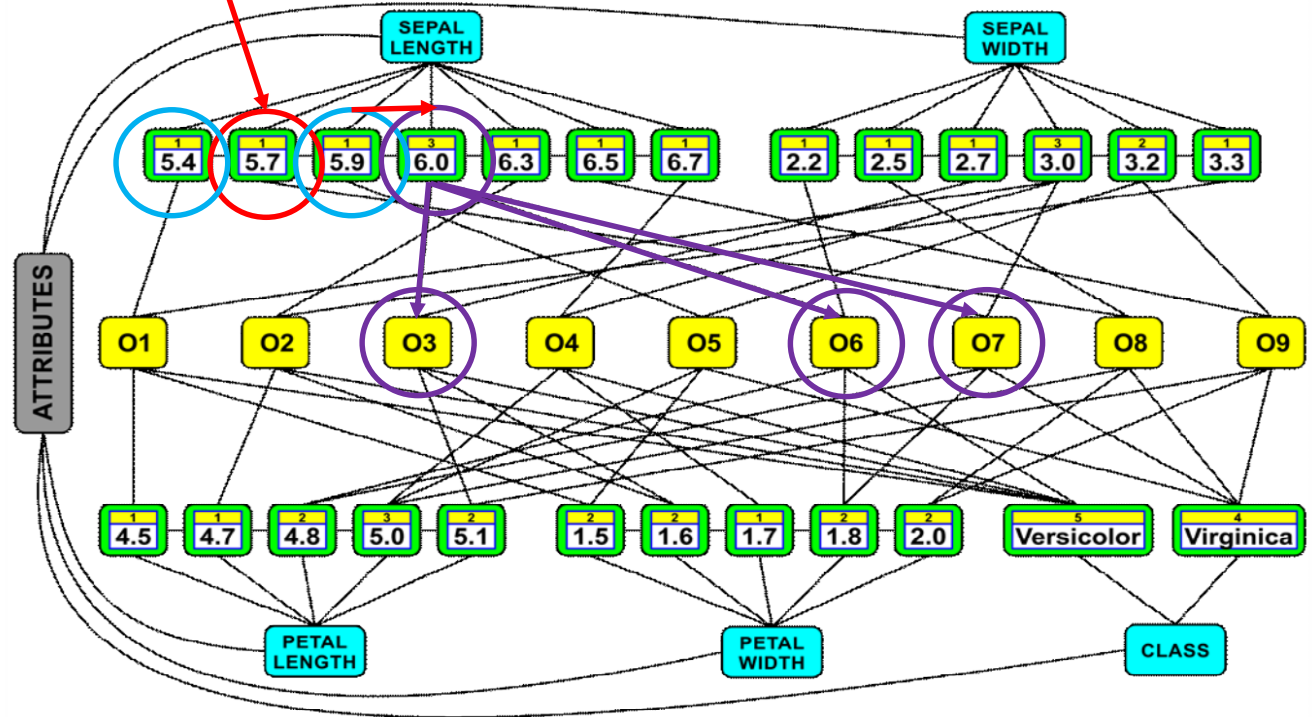
Rank table

1.	O8	0.45
2.	O5	0.48
3.	O1	0.66

[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to the **next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.	O8	0.45
2.	O3	0.47
3.	O5	0.48
	O1	0.66

[5.7; 2.5; 4.8; 1.6]

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$

O3 $d_e(x, y) = 0.47$

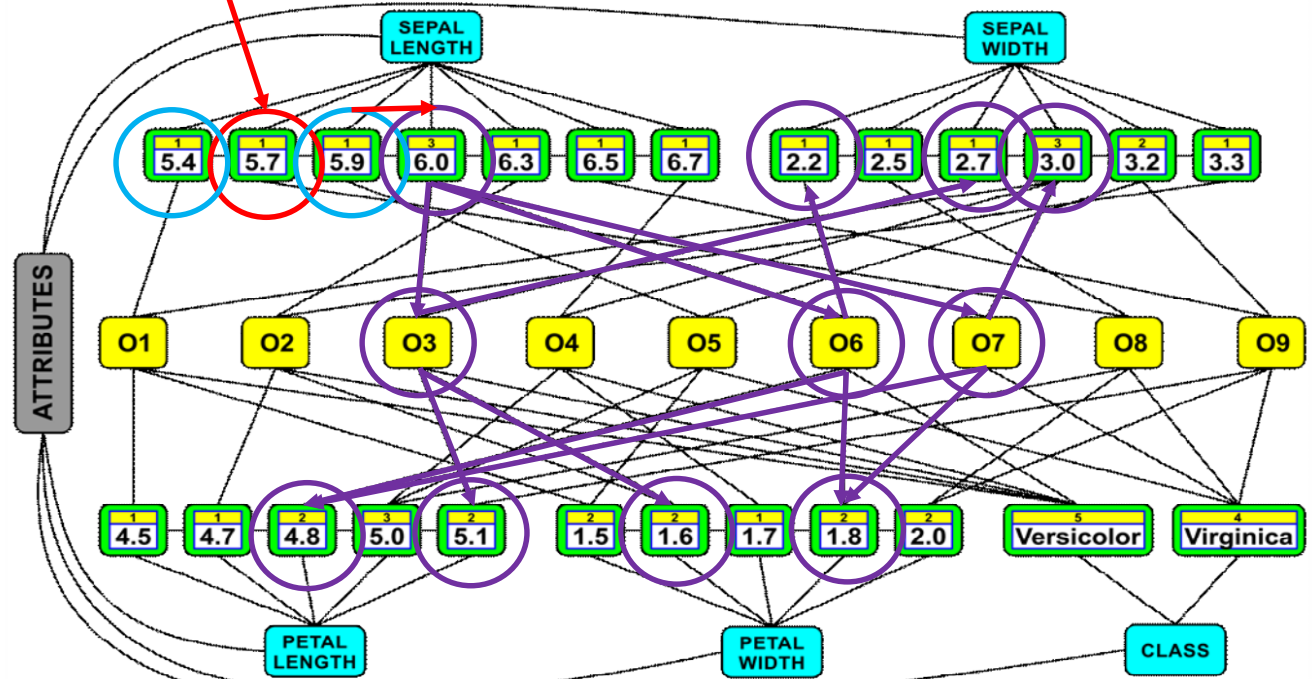
$d_m(x, y) = 0.8$

O6 $d_e(x, y) = 0.75$

$d_m(x, y) = 1.1$

O7 $d_e(x, y) = 0.62$

$d_m(x, y) = 1.0$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to the **next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers



Rank table

1.	O8	0.45
2.	O3	0.47
3.	O5	0.48

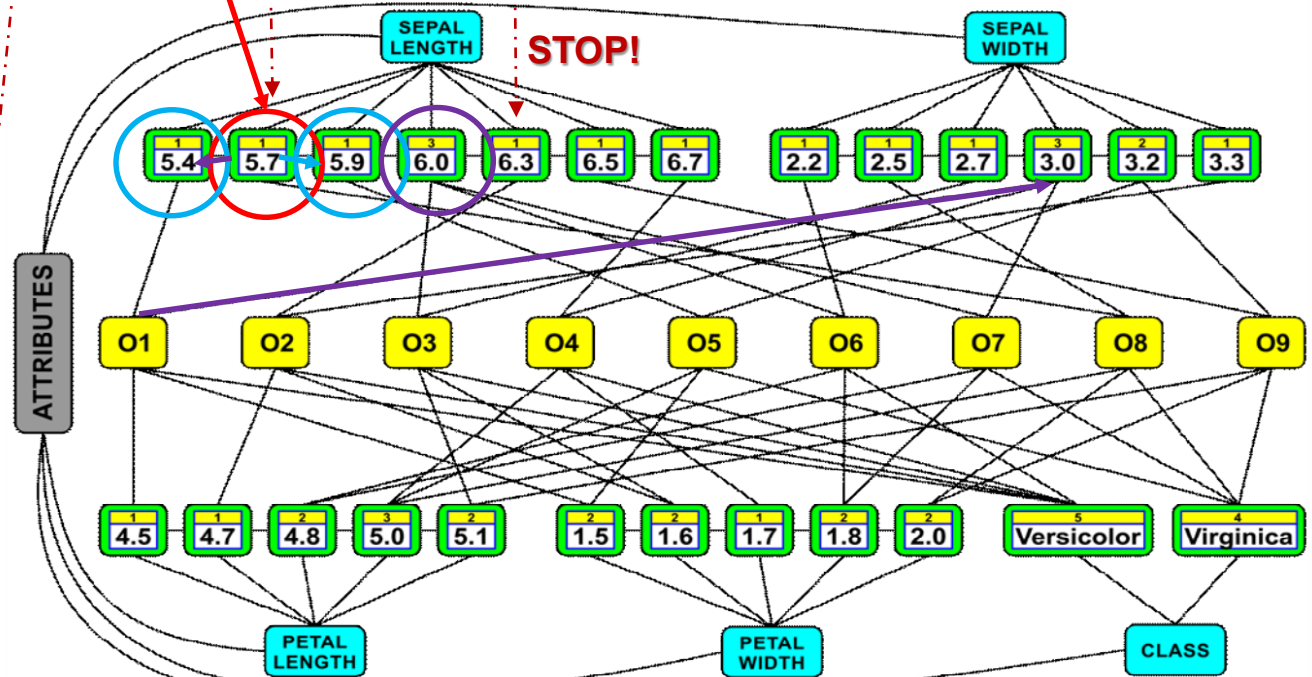
K=3 Nearest Neighbors

[5.7; 2.5; 4.8; 1.6]

$$0.48 < 0.6 = 6.5 - 5.7$$

$$d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2)$$



STOP CONDITON

10. Next, go to step 5 if the difference between these two values is less than the distance of the last object node stored in the rank table; else finish the algorithm because the rank table already contains k nearest neighbors together with their distances to the classified object.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table

1.	O8	0.45	Virginica
2.	O3	0.47	Versicolor
3.	O5	0.48	Virginica

K Nearest Neighbors

[5.7; 2.5; 4.8; 1.6]

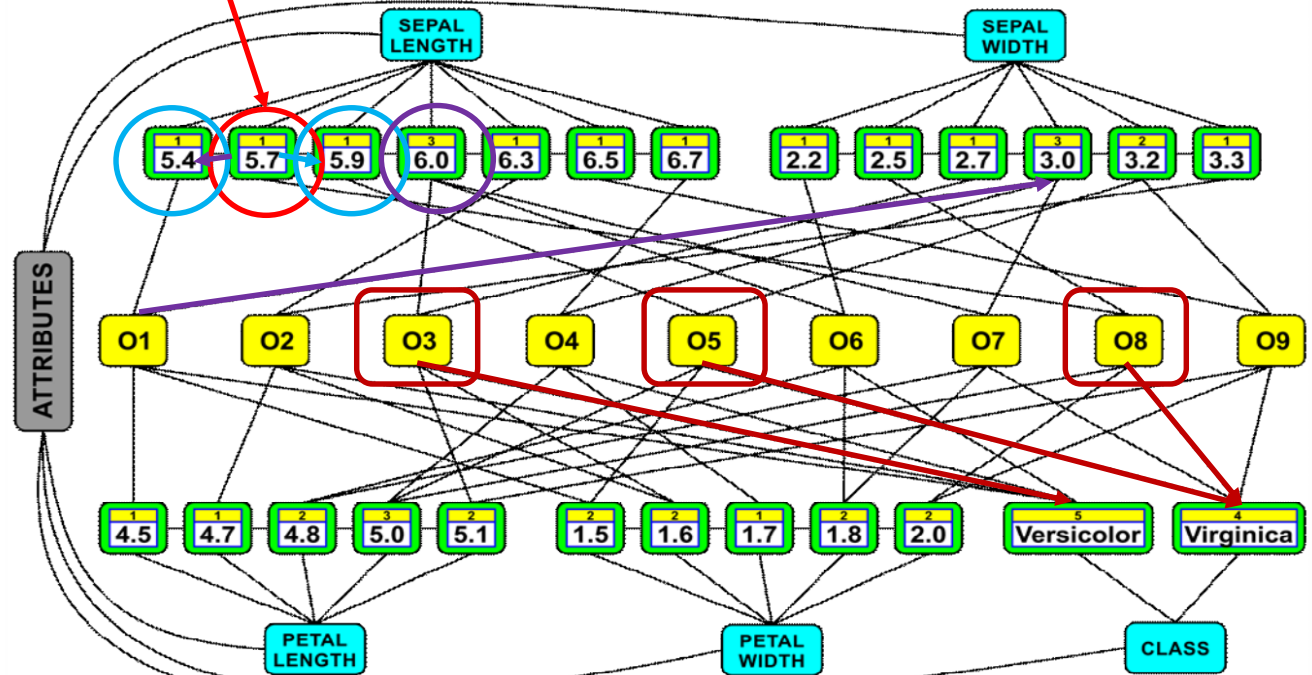
Votes

2 x Virginica
1 x Versicolor

WINNING CLASS



SAMPLE OBJECTS	ATTRIBUTES				CLASS LABEL
	SEPAL LENGTH	SEPAL WIDTH	PETAL LENGTH	PETAL WIDTH	
O1	5.4	3.0	4.5	1.5	Versicolor
O2	6.3	3.3	4.7	1.6	Versicolor
O3	6.0	2.7	5.1	1.6	Versicolor
O4	6.7	3.0	5.0	1.7	Versicolor
O5	6.0	2.2	5.0	1.5	Virginica
O6	5.9	3.2	4.8	1.8	Versicolor
O7	6.0	3.0	4.8	1.8	Virginica
O8	5.7	2.5	5.0	2.0	Virginica
O9	6.5	3.2	5.1	2.0	Virginica



11. Count votes and check which class has the most votes to establish the winning class to get the KNN classification.

Comparison of Results and Efficiencies

Time Efficiency:

Table 2. Comparison of classification time using kNN and kNN+AGDS.

Dataset	Number of instances	Number of attributes	kNN classification time [ms]	kNN+AGDS classification time [ms]	kNN+AGDS construction time [ms]
Iris	150	4	0.10	0.08	1
Banknote	1372	4	0.29	0.09	5
HTRU2	17898	8	3.14	0.09	134
Shuttle	43500	9	7.67	1.06	278
Credit Card	30000	23	8.69	1.07	499
Skin	245057	3	26.87	1.10	683
Drive	58509	48	46.15	1.24	2224
HEPMASS	1048576	28	362.32	1.41	31214

Classification time for KNN+AGDS is almost constant regardless of the size of the used training data sets.

Comparison of Results and Efficiencies

Time Efficiency as a function of the number of instances and attributes:

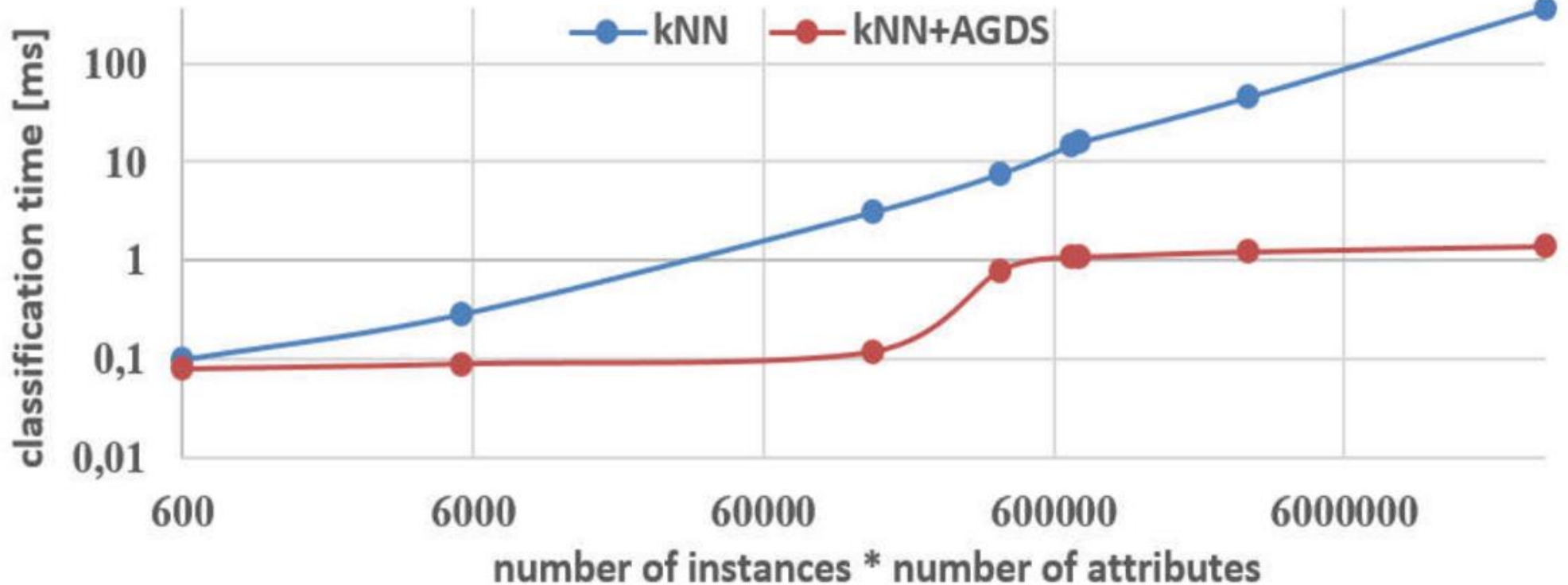


Fig. 2. Classification time as a function of the number of the instances multiplied by the number of attributes, i.e. the number of data stored in the training data tables.

The size of training data and the number of attributes do not influence KNN+AGDS efficiency as is in the classic KNN classifiers.

Comparison of Results and Efficiencies

Memory Efficiency:

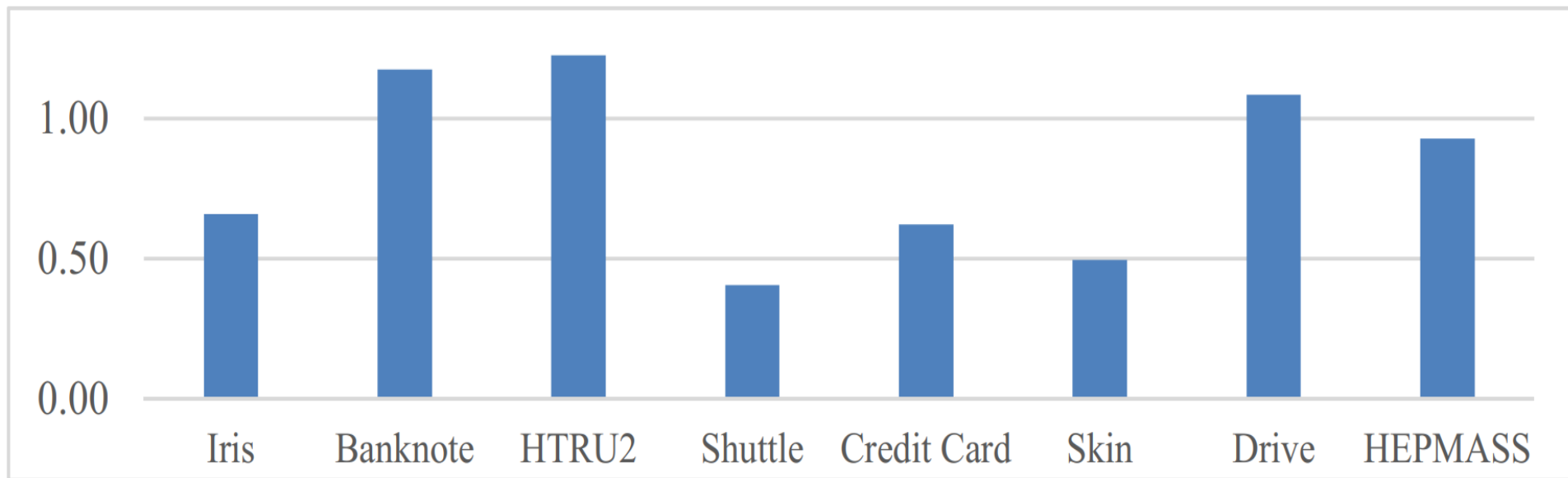


Fig. 3. Memory usage ratio of using AGDS structures to arrays for various training data.

KNN+AGDS classifiers also usually **use less memory** than KNN classifiers due to the number of duplicated values in training data.



Conclusions and Important Contributions

- ✓ AGDS structures provide high-speed access to neighbor values and similar objects because of the aggregations and ordering of all values simultaneously for all attributes.
- ✓ AGDS stores data together with the most common vertical and horizontal relations, so there is no need to loop and search for these relations wasting resources.
- ✓ Typical operations on the AGDS structures take logarithmic time of the number of unique attribute values that are operated, but the expected complexity on real data containing many duplicates is usually constant.
- ✓ The efficiency of the presented classification algorithm using AGDS structures grows with the amount of the training data.

Algorithm, pseudocode ... can be found in the paper



Associative Graph Data Structures Used for Acceleration of K Nearest Neighbor Classifiers

Adrian Horzyk^(✉)  and Krzysztof Gołdon 

AGH University of Science and Technology, Krakow, Poland
horzyk@agh.edu.pl, krzysztofgoldon@gmail.com

Abstract. This paper introduces a new associative approach for significant acceleration of k Nearest Neighbor classifiers (kNN). The kNN classifier is a lazy method, i.e. it does not create a computational model, so it is inefficient during classification using big training data sets because it requires going through all training patterns when classifying each sample. In this paper, we propose to use Associative Graph Data Structures (AGDS) as an efficient model for storing training patterns and their relations, allowing for fast access to nearest neighbors during classification made by kNNs. Hence, the AGDS significantly accelerates the classification made by kNNs, especially for large and huge training datasets. In this paper, we introduce an Associative Acceleration Algorithm and demonstrate how it works on this associative structure substantially reducing the number of checked patterns and quickly selecting k nearest neighbors for kNNs. The presented approach was compared to classic kNN approaches successfully.

```
FindNextClosest(val)
if (valueNodeLessClosest == null) and (valueNodeGreaterClosest == null)
then
    valueNodeLessClosest = BinSearchEqualOrLess(val)
    if (valueNodeLessClosest == null)
    then valueNodeGreaterClosest = First
        return valueNodeGreaterClosest
    else if (valueNodeLessClosest.Val == val)
    then valueNodeGreaterClosest = valueNodeLessClosest
        return valueNodeGreaterClosest
    else if (valueNodeLessClosest.IsNotMax)
    then valueNodeGreaterClosest = valueNodeLessClosest.Next
        if (val - valueNodeLessClosest.Val < valueNodeGreaterClosest.Val -
val)
        then return valueNodeLessClosest
            else return valueNodeGreaterClosest
        else valueNodeGreaterClosest = null
            return valueNodeLessClosest
    else if (val - valueNodeLessClosest.Val < valueNodeGreaterClosest.Val - val)
    then if (valueNodeLessClosest.IsNotMin)
        then valueNodeLessClosest = valueNodeLessClosest.Prev
            else if (valueNodeGreaterClosest.IsNotMax)
            then valueNodeGreaterClosest = valueNodeGreaterClosest.Next
                else return null
            else if (valueNodeGreaterClosest.IsNotMax)
            then valueNodeGreaterClosest = valueNodeGreaterClosest.Next
                else if (valueNodeLessClosest.IsNotMin)
                then valueNodeLessClosest = valueNodeLessClosest.Prev
                    else return null
            if (val - valueNodeLessClosest.Val < valueNodeGreaterClosest.Val - val)
            then return valueNodeLessClosest
                else return valueNodeGreaterClosest
```

A. Horzyk and K. Gołdon, Associative Graph Data Structures Used for Acceleration of K Nearest Neighbor Classifiers, In: 27th International Conference on Artificial Neural Networks (ICANN 2018), Springer-Verlag, LNCS 11139, pp. 648-658, 2018.

Questions or Remarks?

1. A. Horzyk, J. A. Starzyk, J. Graham, Integration of Semantic and Episodic Memories, IEEE Transactions on Neural Networks and Learning Systems, Vol. 28, Issue 12, Dec. 2017, pp. 3084 - 3095, 2017, DOI: 10.1109/TNNLS.2017.2728203.
2. A. Horzyk, J.A. Starzyk, Multi-Class and Multi-Label Classification Using Associative Pulsing Neural Networks, IEEE Xplore, In: 2018 IEEE World Congress on Computational Intelligence (WCCI IJCNN 2018), 2018, (in print).
3. A. Horzyk, J.A. Starzyk, Fast Neural Network Adaptation with Associative Pulsing Neurons, IEEE Xplore, In: 2017 IEEE Symposium Series on Computational Intelligence, pp. 339 -346, 2017, DOI: 10.1109/SSCI.2017.8285369.
4. A. Horzyk, Deep Associative Semantic Neural Graphs for Knowledge Representation and Fast Data Exploration, Proc. of KEOD 2017, SCITEPRESS Digital Library, pp. 67 - 79, 2017, DOI: 10.13140/RG.2.2.30881.92005.
5. A. Horzyk, Neurons Can Sort Data Efficiently, Proc. of ICAISC 2017, Springer-Verlag, LNAI, 2017, pp. 64 - 74, ICAISC BEST PAPER AWARD 2017 sponsored by Springer.
6. A. Horzyk, How Does Generalization and Creativity Come into Being in Neural Associative Systems and How Does It Form Human-Like Knowledge?, Elsevier, Neurocomputing, Vol. 144, 2014, pp. 238 - 257, DOI: 10.1016/j.neucom.2014.04.046.
7. A. Horzyk, Innovative Types and Abilities of Neural Networks Based on Associative Mechanisms and a New Associative Model of Neurons - Invited talk at ICAISC 2015, Springer-Verlag, LNAI 9119, 2015, pp. 26 - 38, DOI 10.1007/978-3-319-19324-3_3.
8. A. Horzyk, Associative Graph Data Structures with an Efficient Access via AVB+trees, In: 2018 11th International Conference on Human System Interaction (HSI), 2018, IEEE Xplore, pp. 169 - 175.
9. A. Horzyk and K. Goldon, Associative Graph Data Structures Used for Acceleration of K Nearest Neighbor Classifiers, In: 27th International Conference on Artificial Neural Networks (ICANN 2018), Springer-Verlag, LNCS 11139, pp. 648-658, 2018.



AGH University of
Science and Technology
Krakow, Poland



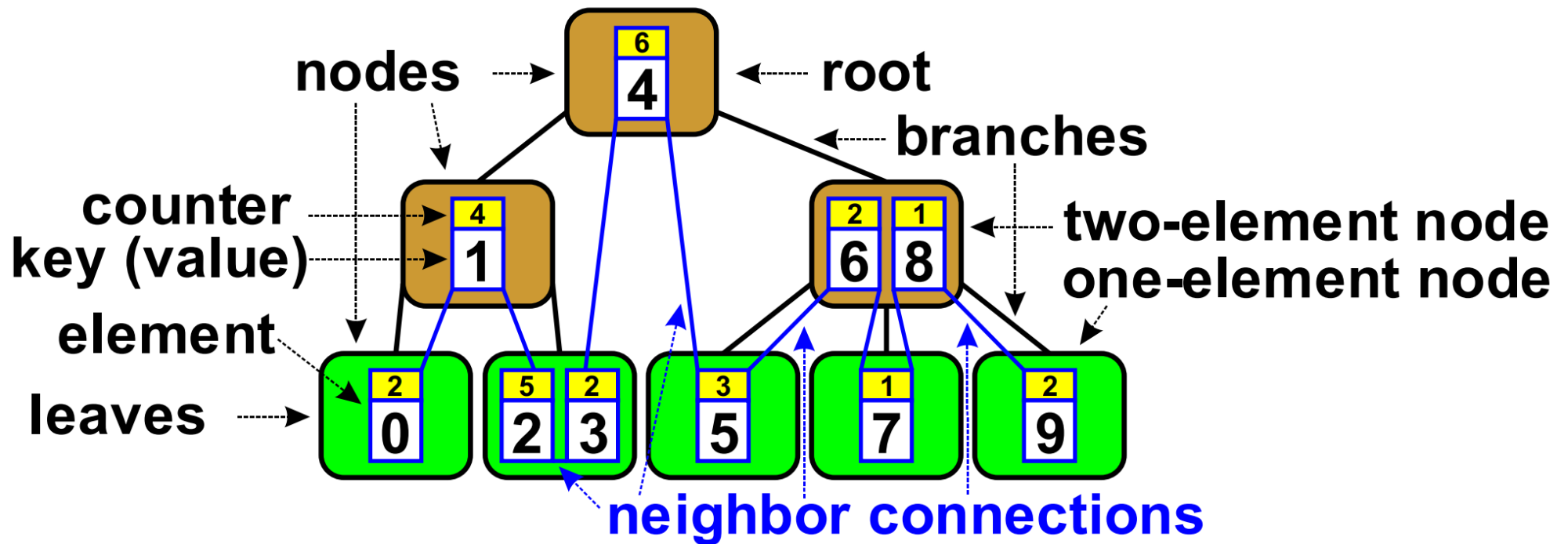
Adrian Horzyk
horzyk@agh.edu.pl



Krzysztof Goldon
krzysztofgoldon@gmail.com

AVB+TREES can additionally accelerate kNN+AGDS

AVB+tree is a hybrid structure that represents the sorted list of aggregated elements which are quickly accessible via self-balancing B-tree structure. Elements aggregate and count up all duplicates of represented values.



AVB+trees are typically much smaller in size and height than B-trees and B+trees thanks to the aggregations of duplicates and not using any extra internal nodes as signposts as used in B+trees:

A. Horzyk, *Associative Graph Data Structures with an Efficient Access via AVB+trees*, In: 2018 11th International Conference on Human System Interaction (HSI), 2018, IEEE Xplore, pp. 169 – 175.