IEEE Symposium Series on Computational Intelligence

December 6-9, 2019 Xiamen, China

Associative Data Model in Search for Nearest Neighbors and Similar Patterns

Adrian Horzyk

horzyk@agh.edu.pl

Google: Horzyk

AGH University of Science and Technology

Krakow, Poland

Janusz A. Starzyk

starzykj@ohio.edu

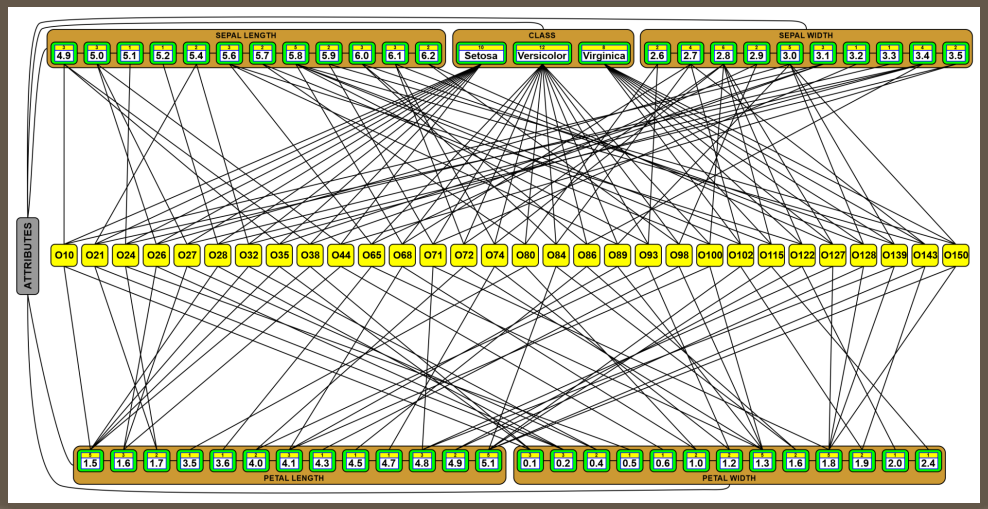
Google: Janusz Starzyk

Ohio University, Athens, Ohio, U.S.A., School of Electrical Engineering and Computer Science

University of Information Technology and Management, Rzeszow, Poland

Inspiration and Objectives

Creation of the efficient nearest neighbors and similar pattern search algorithms based on the brain-like structure and associative processes.



Disadvantages of Tabular Structures

Tabular structures do not relate objects vertically, so many relations between stored objects must be discovered during time-taking searches through them:

Let's have a table of data:

What we can say about the stored data in this table without browsing it many times in loops and evaluating many human-written conditions?

Which objects are the most similar or different?

Can we quickly order it according to various criteria?

Can we quickly point out the most similar objects to the given one, e.g. to the object "93"?

What will we have to do to find similarities, differences, minima, maxima, groups, clusters, …?

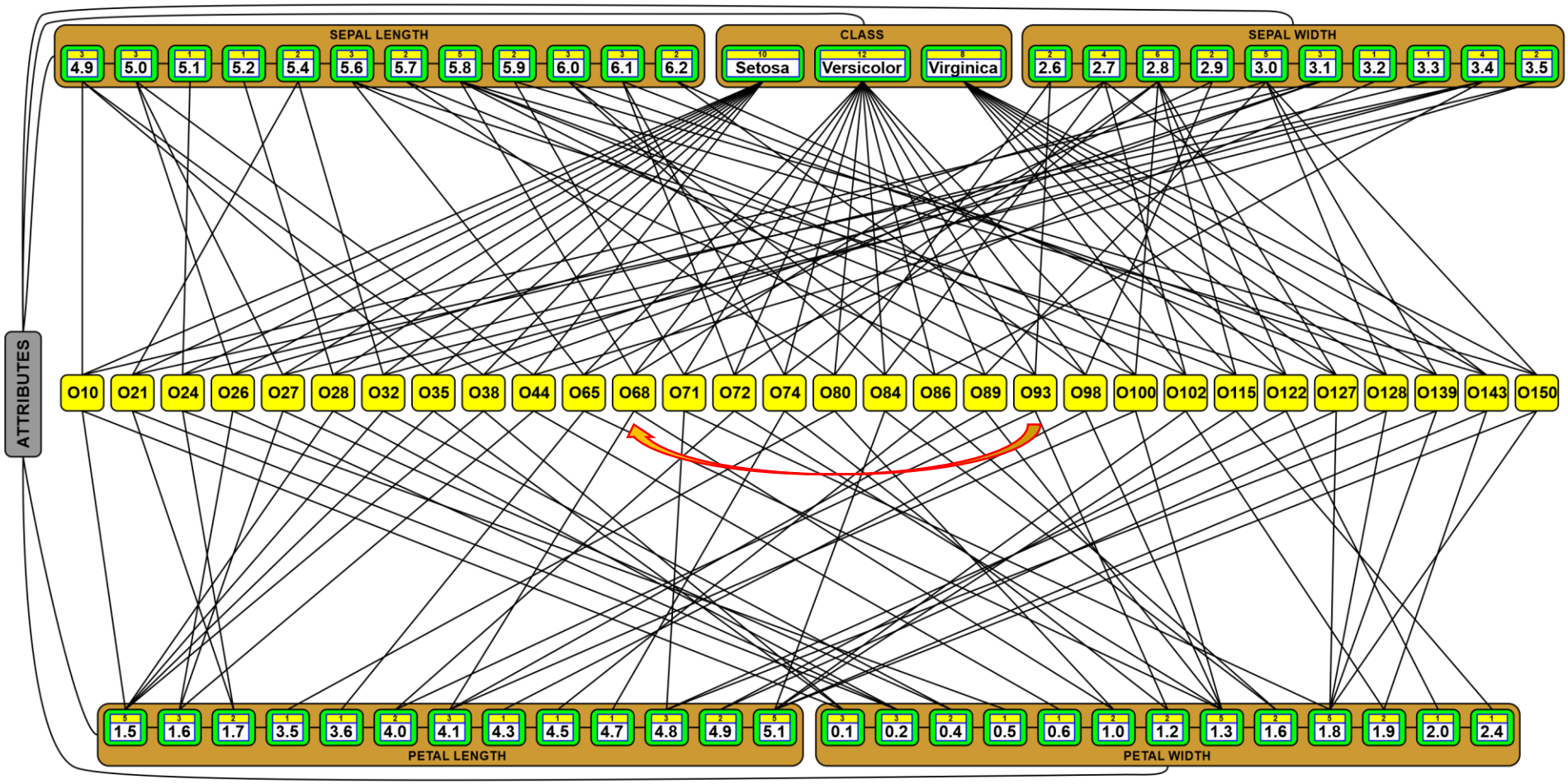
How much time it takes when we have huge amount of data stored in such tables?

10	Iris-setosa	4.9	3.1	1.5	0.1
21	Iris-setosa	5.4	3.4	1.7	0.2
24	Iris-setosa	5.1	3.3	1.7	0.5
26	Iris-setosa	5.0	3.0	1.6	0.2
27	Iris-setosa	5.0	3.4	1.6	0.4
28	Iris-setosa	5.2	3.5	1.5	0.2
32	Iris-setosa	5.4	3.4	1.5	0.4
35	Iris-setosa	4.9	3.1	1.5	0.1
38	Iris-setosa	4.9	3.1	1.5	0.1
44	Iris-setosa	5.0	3.5	1.6	0.6
65	Iris-versicolor	5.6	2.9	3.6	1.3
68	Iris-versicolor	5.8	2.7	4.1	1.0
71	Iris-versicolor	5.9	3.2	4.8	1.8
72	Iris-versicolor	6.1	2.8	4.0	1.3
74	Iris-versicolor	6.1	2.8	4.7	1.2
80	Iris-versicolor	5.7	2.6	3.5	1.0
84	Iris-versicolor	6.0	2.7	5.1	1.6
86	Iris-versicolor	6.0	3.4	4.5	1.6
89	Iris-versicolor	5.6	3.0	4.1	1.3
93	Iris-versicolor	5.8	2.6	4.0	1.2
98	Iris-versicolor	6.2	2.9	4.3	1.3
100	Iris-versicolor	5.7	2.8	4.1	1.3
102	Iris-virginica	5.8	2.7	5.1	1.9
115	Iris-virginica	5.8	2.8	5.1	2.4
122	Iris-virginica	5.6	2.8	4.9	2.0
127	Iris-virginica	6.2	2.8	4.8	1.8
128	Iris-virginica	6.1	3.0	4.9	1.8
139	Iris-virginica	6.0	3.0	4.8	1.8
143	Iris-virginica	5.8	2.7	5.1	1.9
150	Iris-virginica	5.9	3.0	5.1	1.8

?

Benefits of Associative Graph Structures

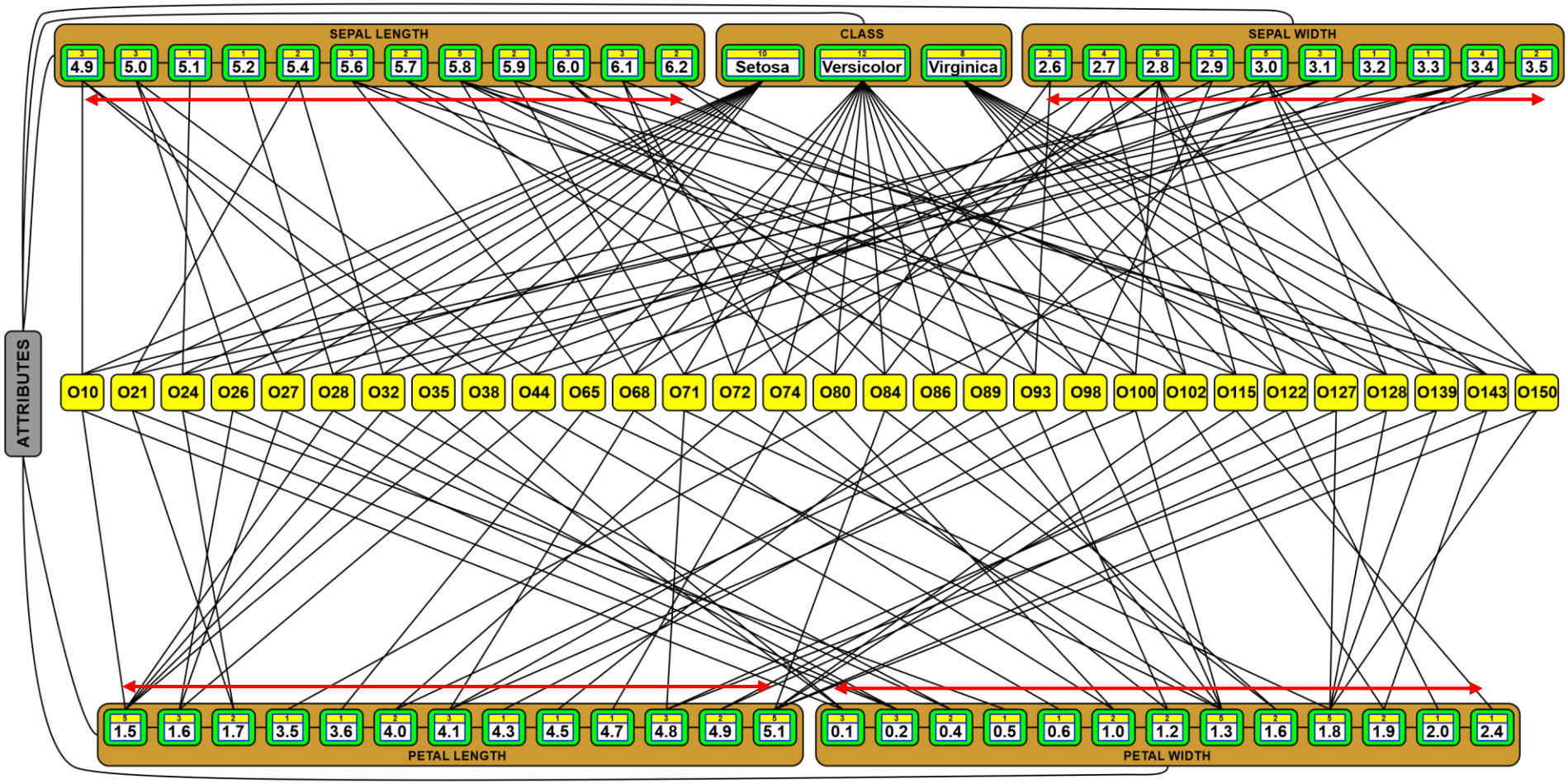
Associative graph structures cannot only relate objects horizontally and vertically, but they can also represent any kind of association between objects what simplifies and accelerates all search processes:



What is the difference and where are advantages in associative graph data representation?

Benefits of Associative Graph Structures

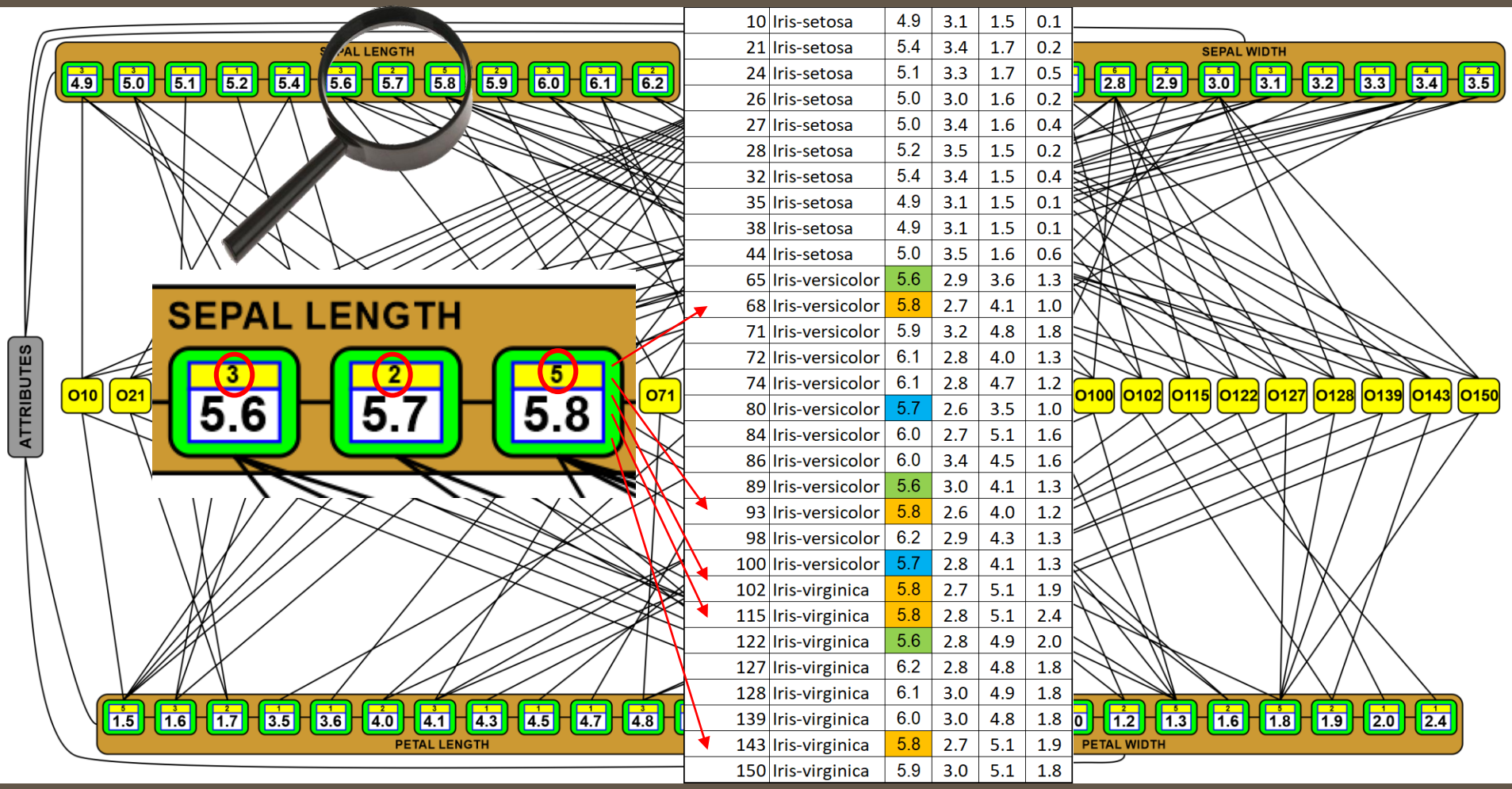
All data are sorted for all attributes simultaneously and **stored in order**!
So we don't need to sort data any more before searching through them.



What's more?

Benefits of Associative Graph Structures

All duplicates of the data of all attributes separately **are aggregated and counted**!
So we don't need to search for duplicates, number of unique values or count up how many different values and how many duplicates of each value we have.

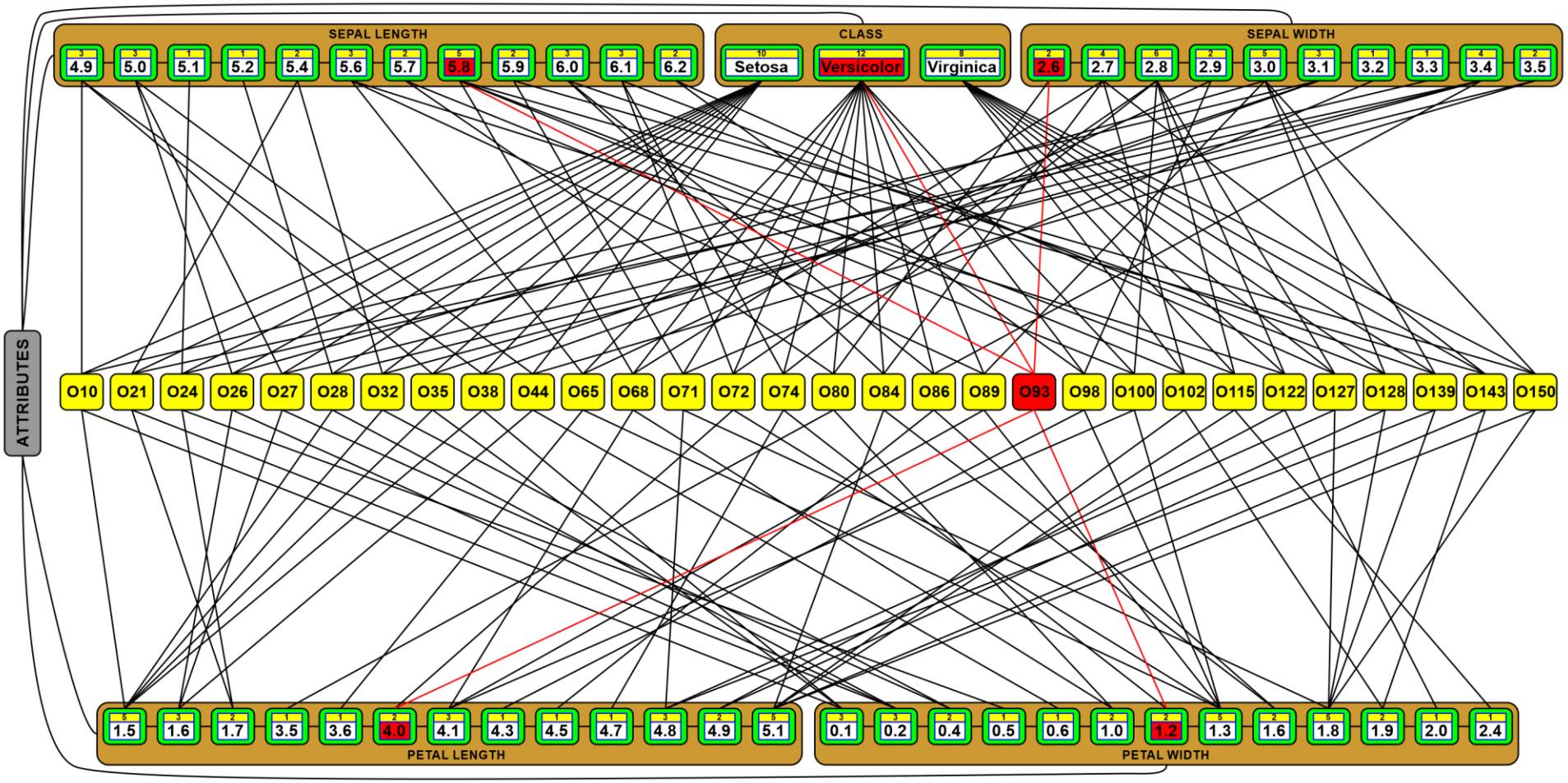


10	Iris-setosa	4.9	3.1	1.5	0.1
21	Iris-setosa	5.4	3.4	1.7	0.2
24	Iris-setosa	5.1	3.3	1.7	0.5
26	Iris-setosa	5.0	3.0	1.6	0.2
27	Iris-setosa	5.0	3.4	1.6	0.4
28	Iris-setosa	5.2	3.5	1.5	0.2
32	Iris-setosa	5.4	3.4	1.5	0.4
35	Iris-setosa	4.9	3.1	1.5	0.1
38	Iris-setosa	4.9	3.1	1.5	0.1
44	Iris-setosa	5.0	3.5	1.6	0.6
65	Iris-versicolor	5.6	2.9	3.6	1.3
68	Iris-versicolor	5.8	2.7	4.1	1.0
71	Iris-versicolor	5.9	3.2	4.8	1.8
72	Iris-versicolor	6.1	2.8	4.0	1.3
74	Iris-versicolor	6.1	2.8	4.7	1.2
80	Iris-versicolor	5.7	2.6	3.5	1.0
84	Iris-versicolor	6.0	2.7	5.1	1.6
86	Iris-versicolor	6.0	3.4	4.5	1.6
89	Iris-versicolor	5.6	3.0	4.1	1.3
93	Iris-versicolor	5.8	2.6	4.0	1.2
98	Iris-versicolor	6.2	2.9	4.3	1.3
100	Iris-versicolor	5.7	2.8	4.1	1.3
102	Iris-virginica	5.8	2.7	5.1	1.9
115	Iris-virginica	5.8	2.8	5.1	2.4
122	Iris-virginica	5.6	2.8	4.9	2.0
127	Iris-virginica	6.2	2.8	4.8	1.8
128	Iris-virginica	6.1	3.0	4.9	1.8
139	Iris-virginica	6.0	3.0	4.8	1.8
143	Iris-virginica	5.8	2.7	5.1	1.9
150	Iris-virginica	5.9	3.0	5.1	1.8

What's more?

Benefits of Associative Graph Structures

Defining or defined objects can be quickly found thanks to direct connections between defining and defined objects, as well as other interrelated objects connected by the indirect connections.

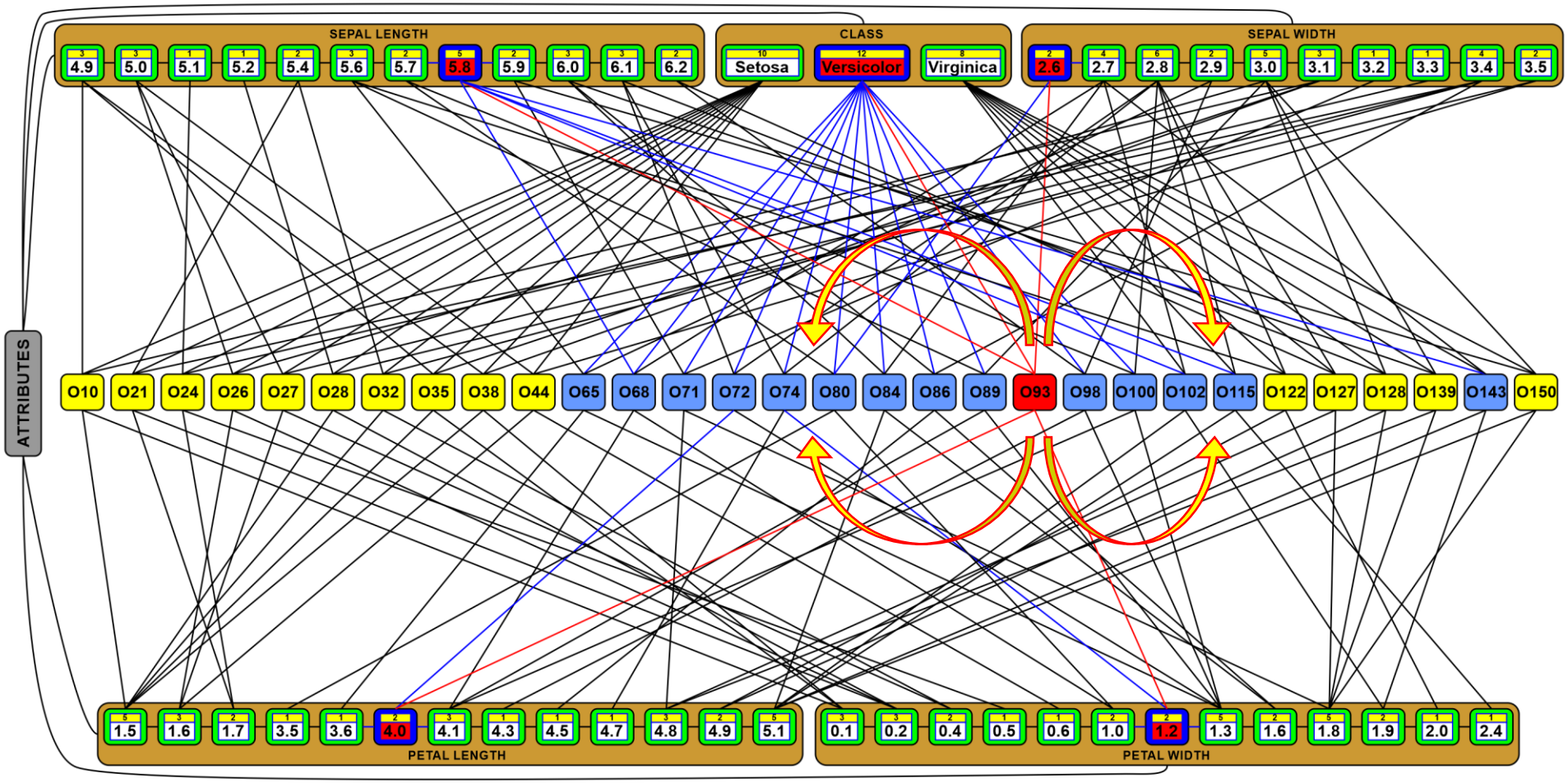


What's more?

Benefits of Associative Graph Structures

Other objects defined <u>by the same values</u> or objects can also be quickly found

thanks to the aggregated representation of the same values and objects in these associative graph structures.

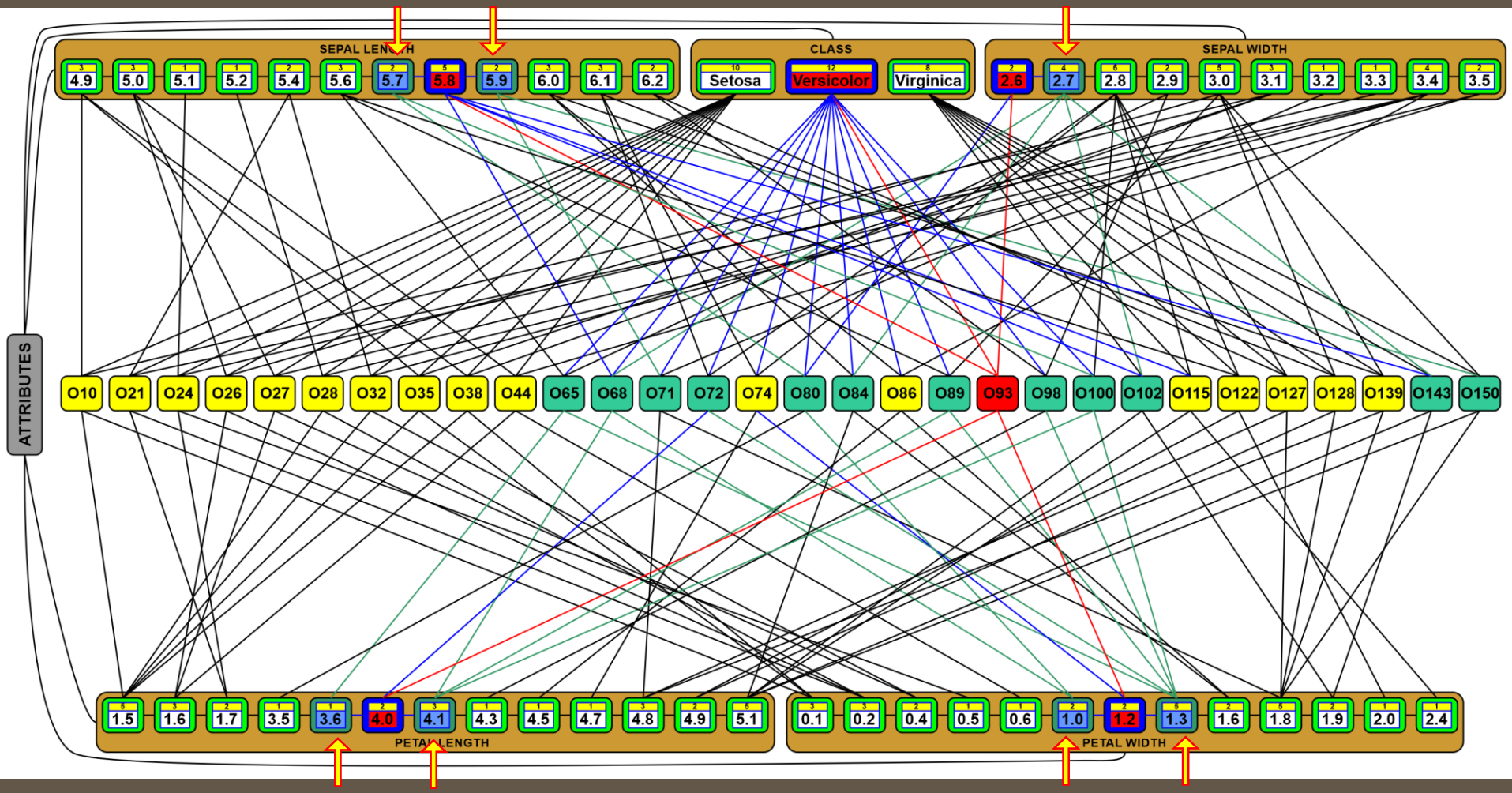


What's more?

Benefits of Associative Graph Structures

Other similar objects defined <u>by the similar values</u> can be also quickly found

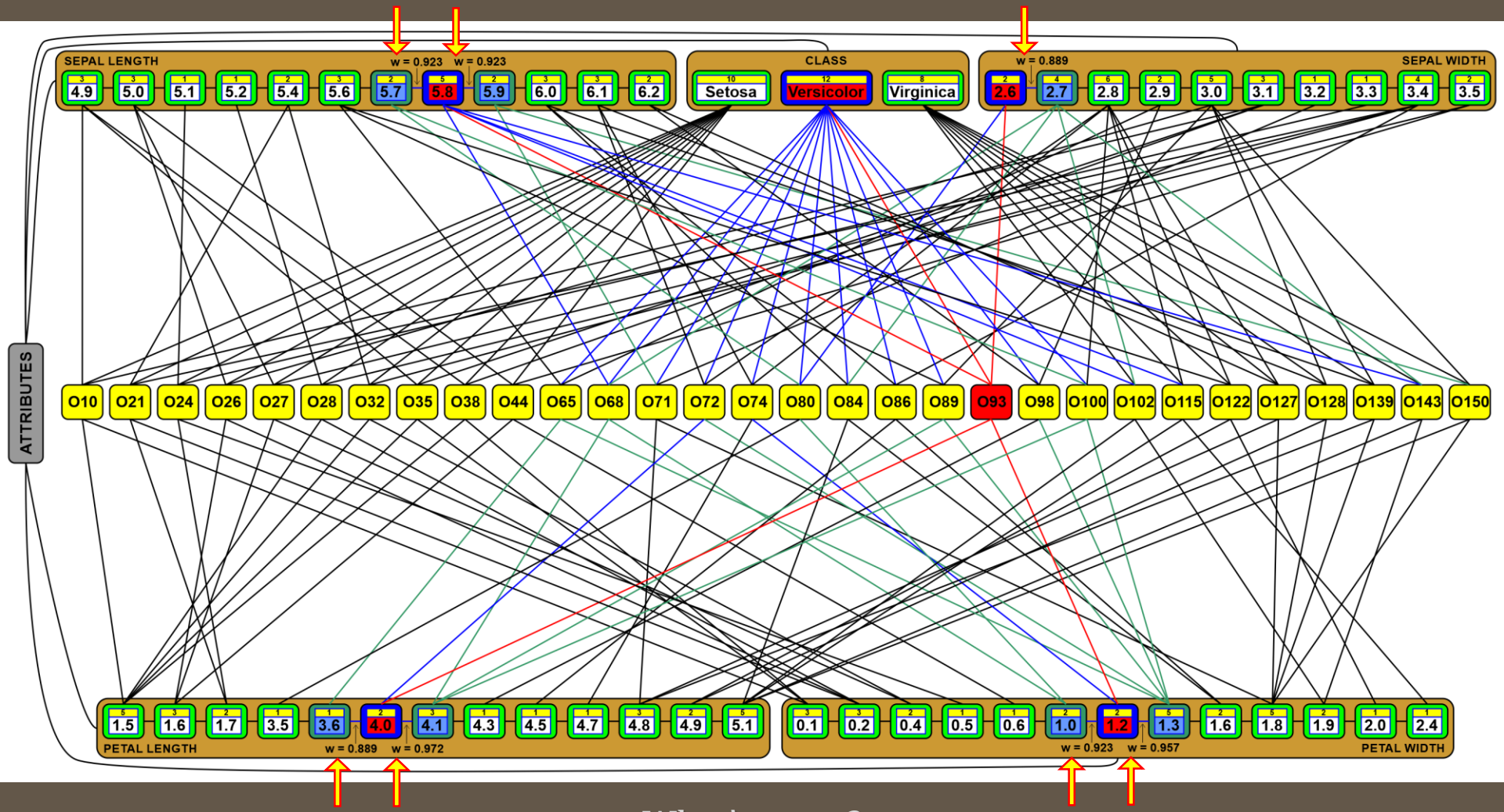
thanks to the storing all attribute data in sorted order and
the connections to the nearest (neighbour) values in these orders.



What's more?

Benefits of Associative Graph Structures

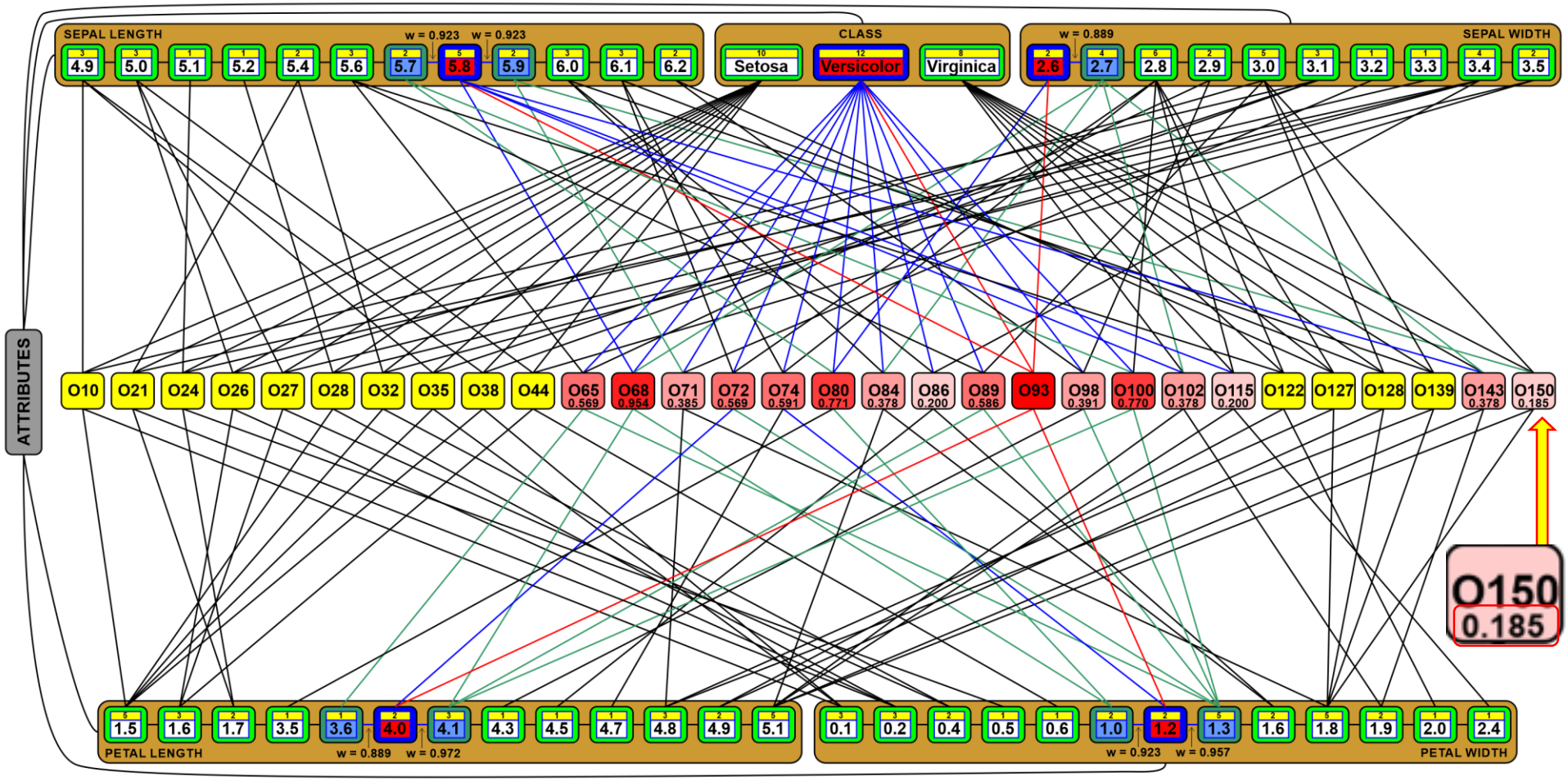
All connections are weighted, so there is not only a binary representation of relations but also the ability to express **the different strengths of associations** between represented data and objects, evaluating values of data relations.



What's more?

Benefits of Associative Graph Structures

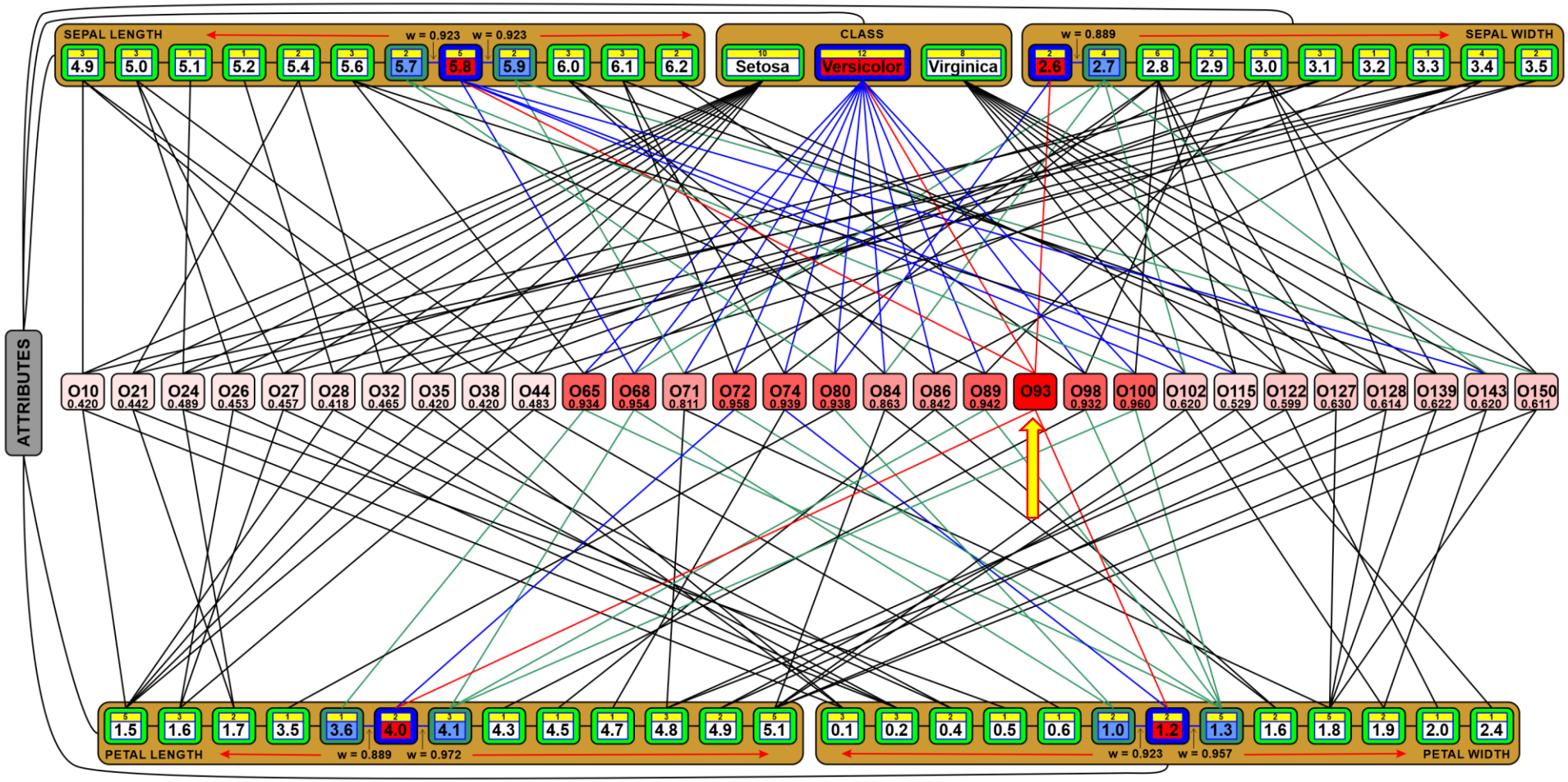
Associative strengths between represented objects can be quickly computed
thanks to the weighted connections between nodes representing defining data and objects on all closest paths between such objects.



We can search for the strongest associated objects only in the close surroundings.

Benefits of Associative Graph Structures

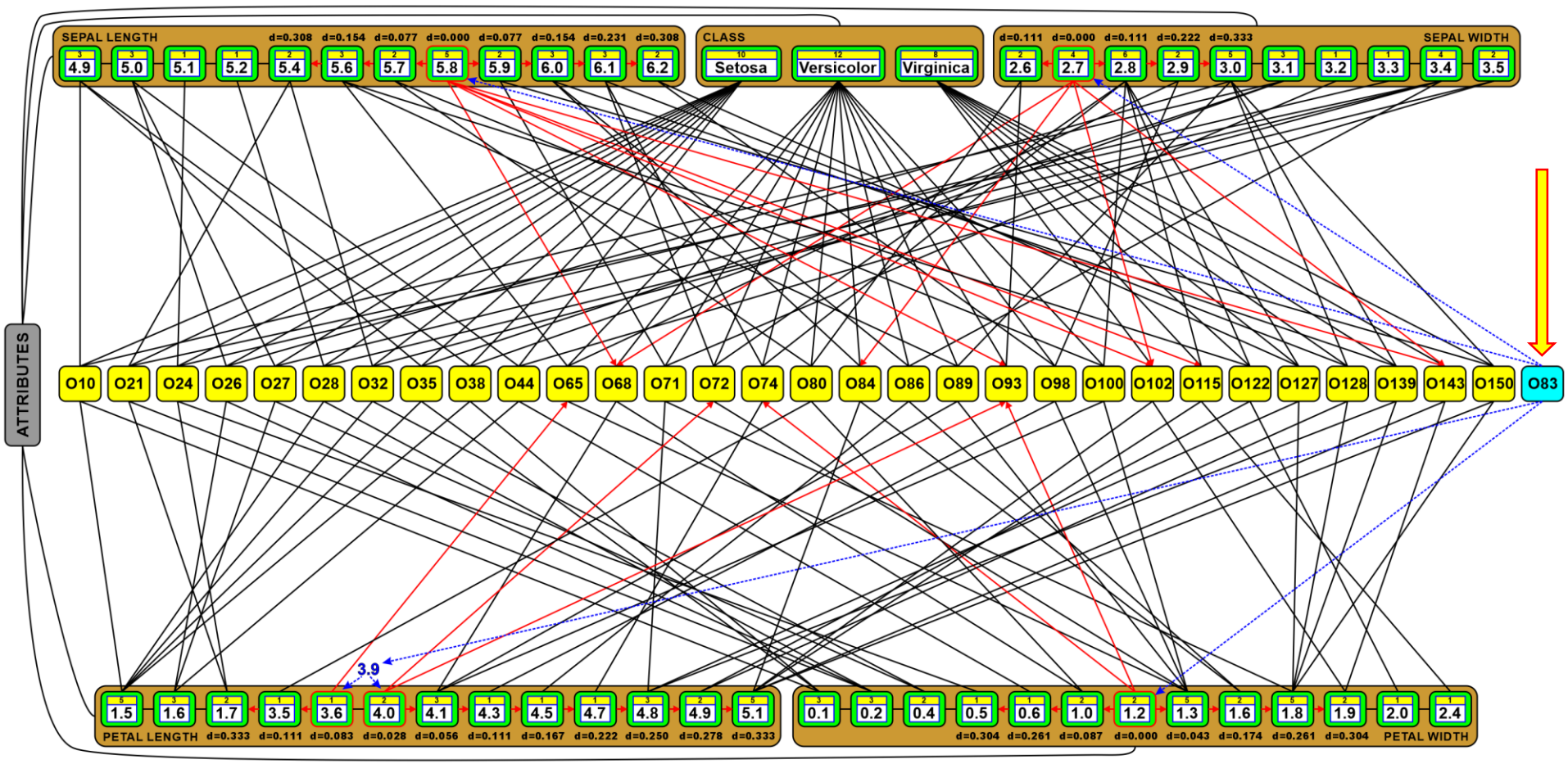
Associative strengths (here similarity) between all objects connected to the given one (here 93) can also be computed using weighted connections between nodes representing defining data and objects on all paths between such objects.



We can also search for the associated objects in the whole surroundings (in all data).

Search for the Nearest Neighbors

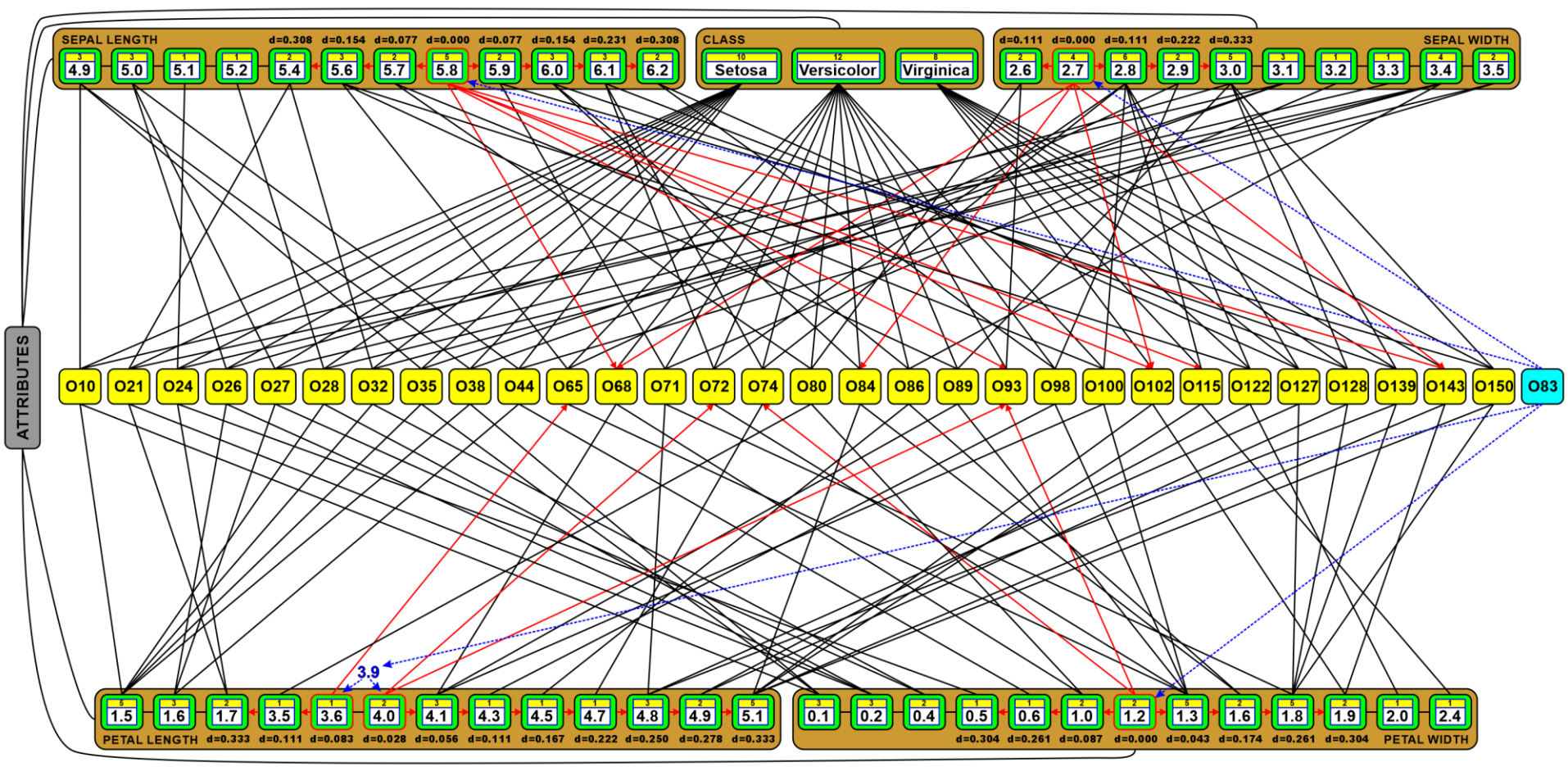
Associative Graph Data Structures (AGDS) can be easily adapted to quickly compute **Euclidean, Manhattan, or Sebestyen distances** between close objects indirectly connected by the strong enough weights in the closest surroundings until we find k nearest neighbors and are sure that all others are more distant.



In this concept, we don't need to look through all data as in the KNN search algorithm.

Associative Search for Nearest Neighbors

The three proposed algorithms start from the nodes representing values that have the smallest, normalized distances to the values defining the classified object. Next, they go along the edges to the connected object nodes to **compute object distances**. The computed distances are used to **sort objects in a rank table or a rank list**.

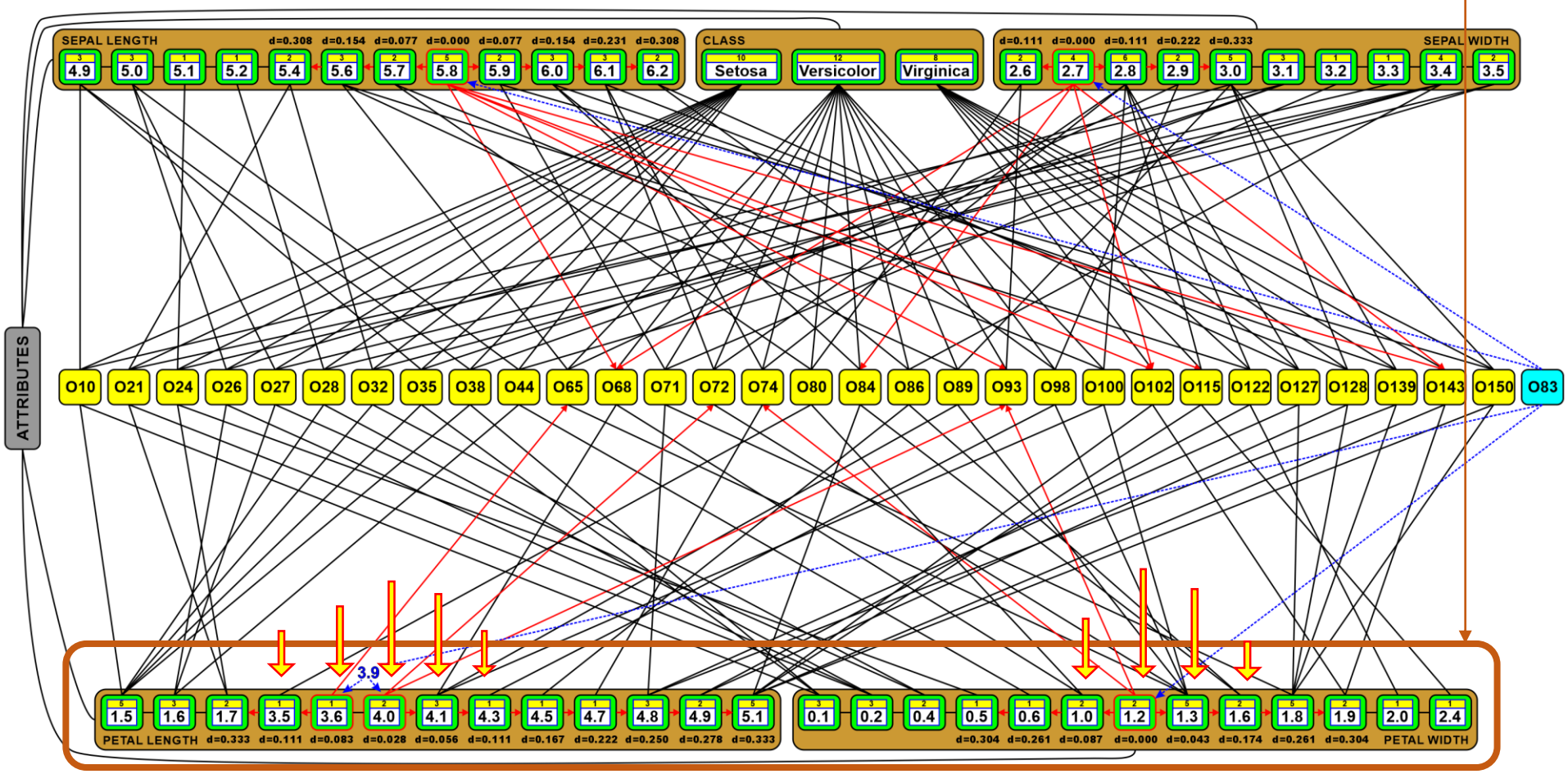


According to the proposed algorithm, the distances are computed at once or partially.

Associative Search for Nearest Neighbors

We take into account a single or a few **most variant attributes**, e.g. petal length (PL) and/or petal width (PW) in the example below because both are defined by **13 unique values**.

The closest values are: PW.1.2 (d=0.000), PL.4.0 (d=0.028), PW.1.3 (d=0.043), PL.4.1 (d=0.056), PL.3.6 (d=0.083), PW.1.0 (d=0.087), PL.3.5 (d=0.111), PL.4.3 (d=0.111), PW.1.6 (d=0.174), etc.



From the closest values, we get into the objects O74, O93, O72, … to compute object distances.

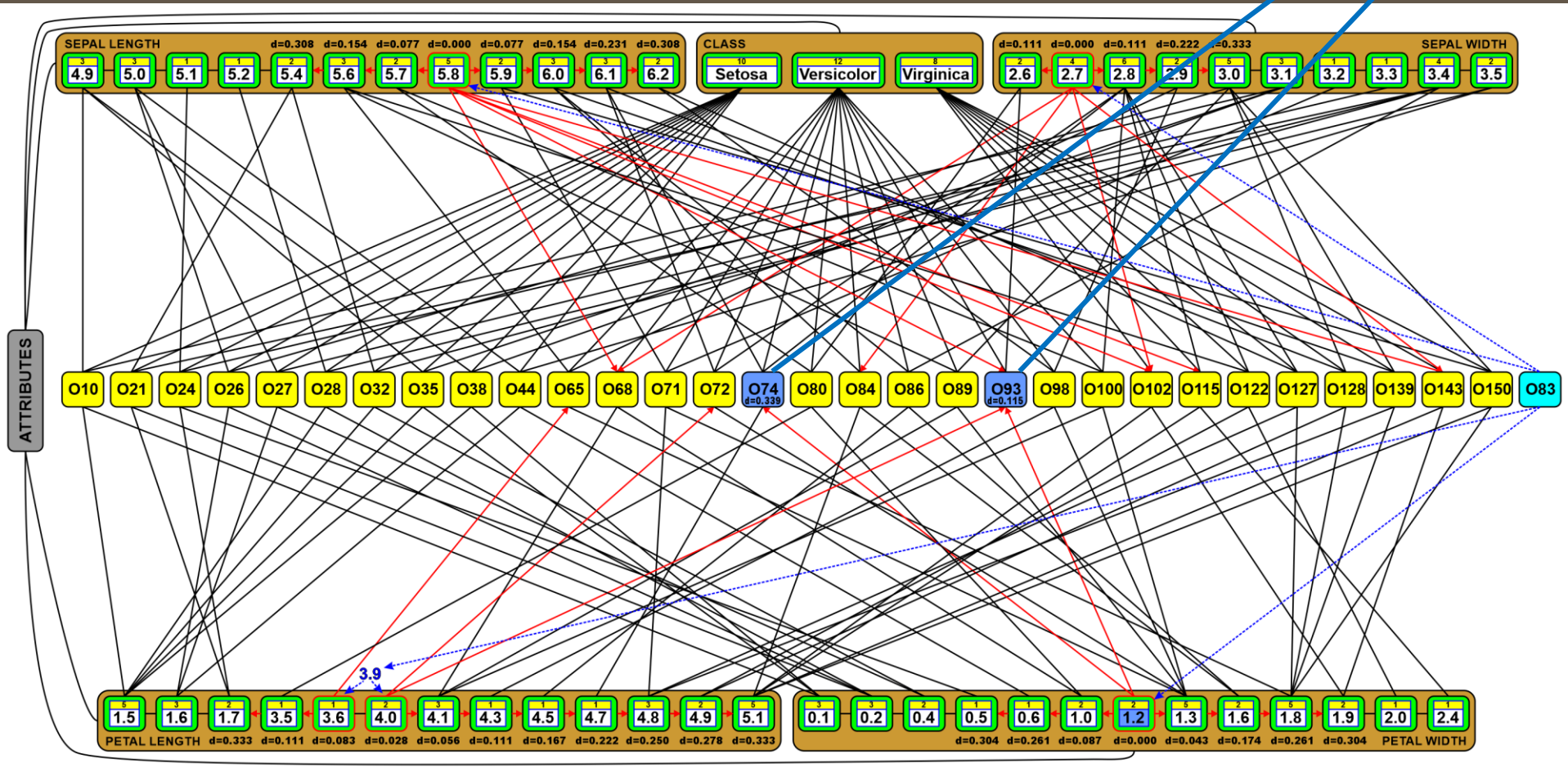
Associative Search for Nearest Neighbors

Let's look for 3 nearest neighbors (the most similar objects) to the O83 = [5.8, 2.7, 3.9, 1.2].

1. Start from the closest value PW.1.2 (d=0.000) and compute Euclidean distances for connected objects O74 (d=0.339) and O93 (d=0.115), and insert them into the rank list that was empty because we just started this algorithm, so the values are added in sorted order:

This rank list is 3-element long because we are searching for 3 nearest neighbors. When this list will be full the elements will be replaced to leave there the nearest.

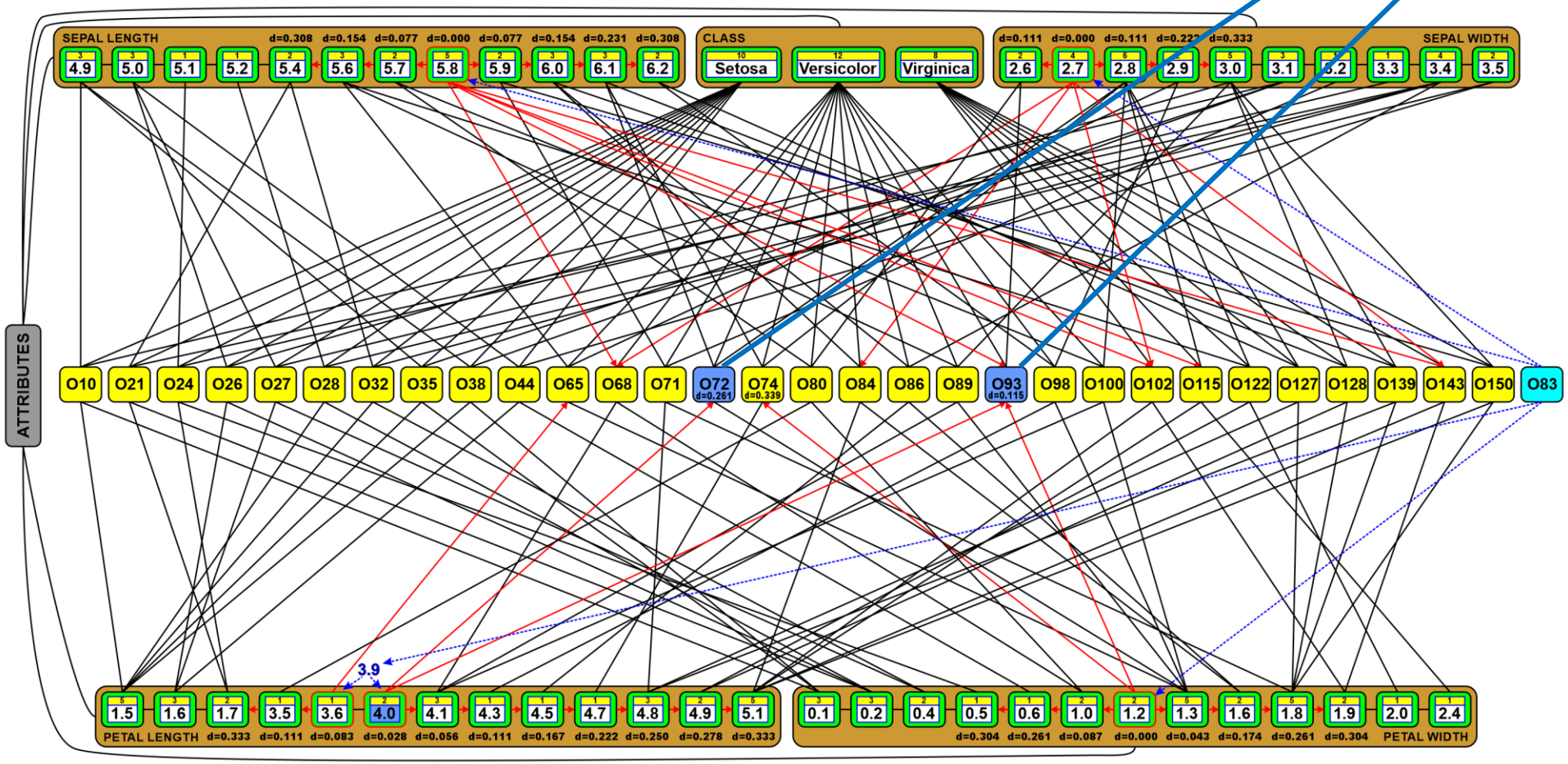
Rank List	
0.115	O93
0.339	O74



Associative Search for Nearest Neighbors

2. Move to the next closest value from the selected subset of the most variant attributes PL.4.0 (d=0.028) and compute Euclidean distances for connected objects O72 (d=0.261) to which distances were not yet computed (for O93 the distance was already computed), and insert them into the rank list in sorted order. After this step, the rank list has 3-element but the algorithms will continue the search for nearest neighbors until the distance to the next value is longer than the most distant object of this rank list.

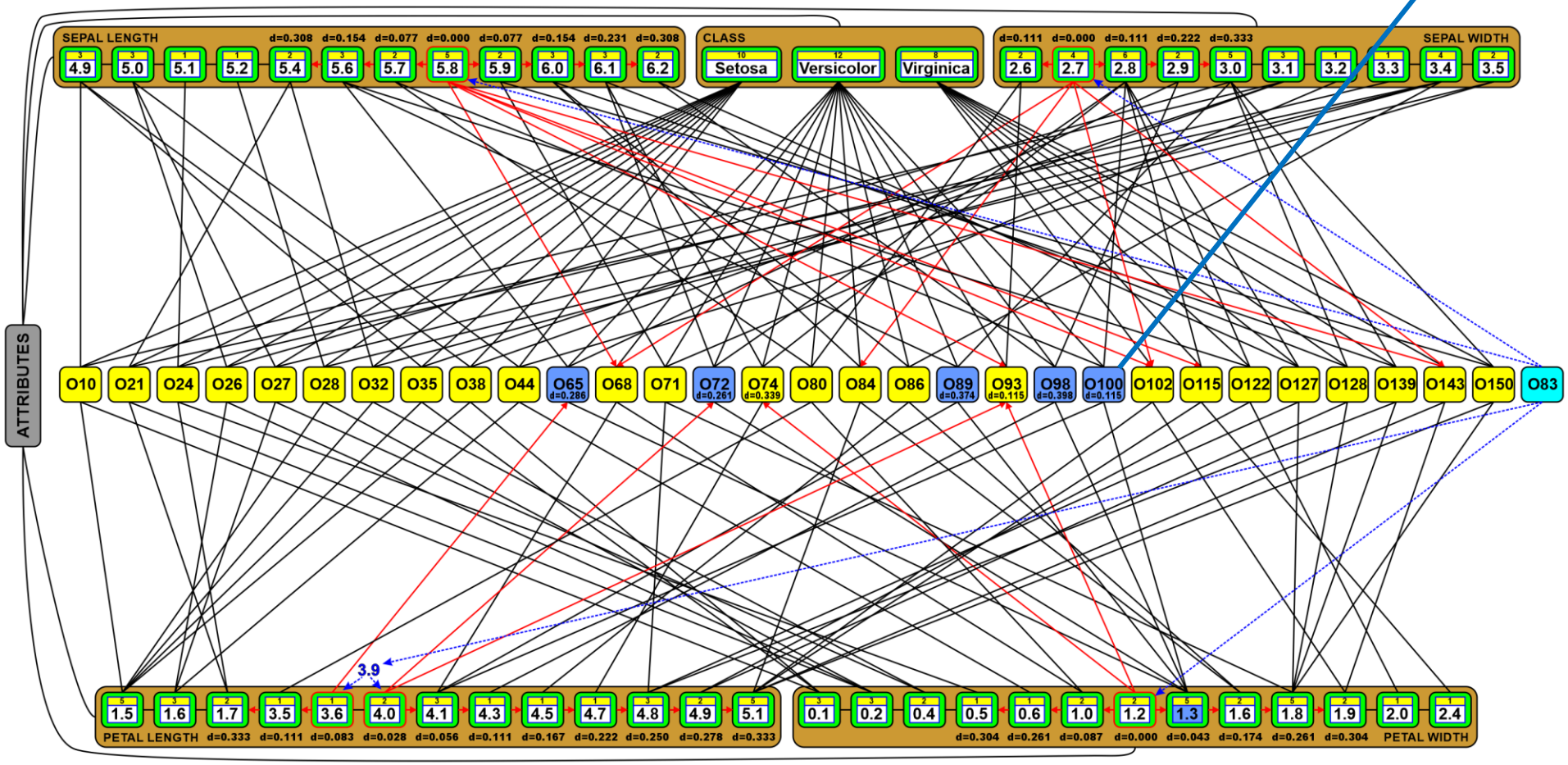
Rank List	
0.115	O93
0.261	O72
0.339	O74



Associative Search for Nearest Neighbors

3. Move to the next closest value from the selected subset of the most variant attributes PW.1.3 (d=0.043) and compute Euclidean distances for connected objects O65 (d=0.286), O89 (d=0.374), O98 (d=0.398), and O100 (d=0.115) to which distances were not yet computed, insert them into the rank list in sorted order, and remove the most distant objects. The most distant object O72 has its distance still bigger than the distance to the next closest attribute value, so the search for 3 nearest neighbors is continued.

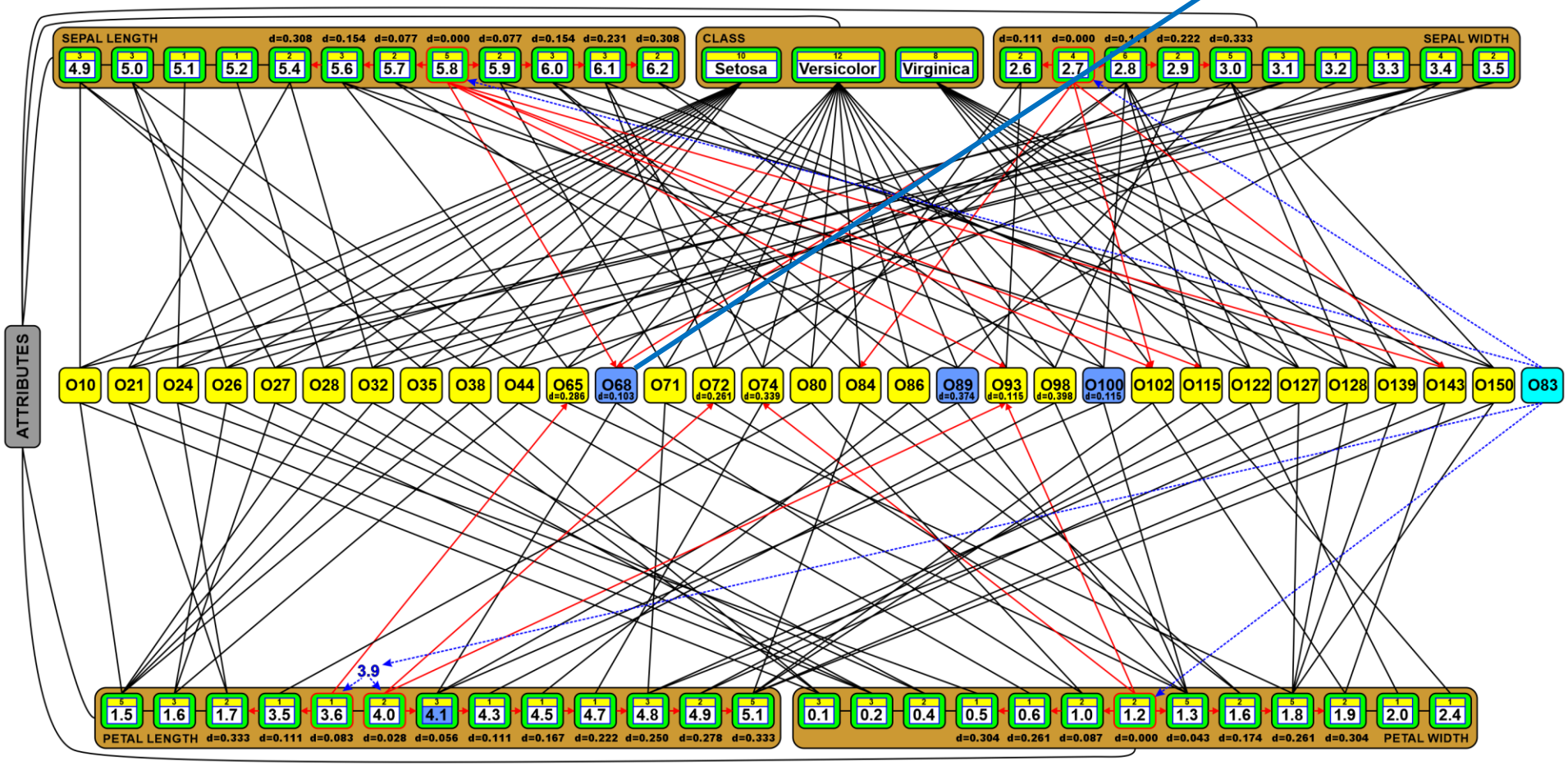
Rank List	
0.115	O93
0.115	O100
0.261	O72



Associative Search for Nearest Neighbors

4. Move to the next closest value from the selected subset of the most variant attributes PL.4.1 (d=0.056) and compute Euclidean distances for connected objects O68 (d=0.103) to which the distance were not yet computed, and insert them into the rank list in sorted order. After this step, we already have 3 nearest neighbors, but the algorithm must run for a few steps more because the stop condition is not yet satisfied, so theoretically there might be some other objects which could be closer.

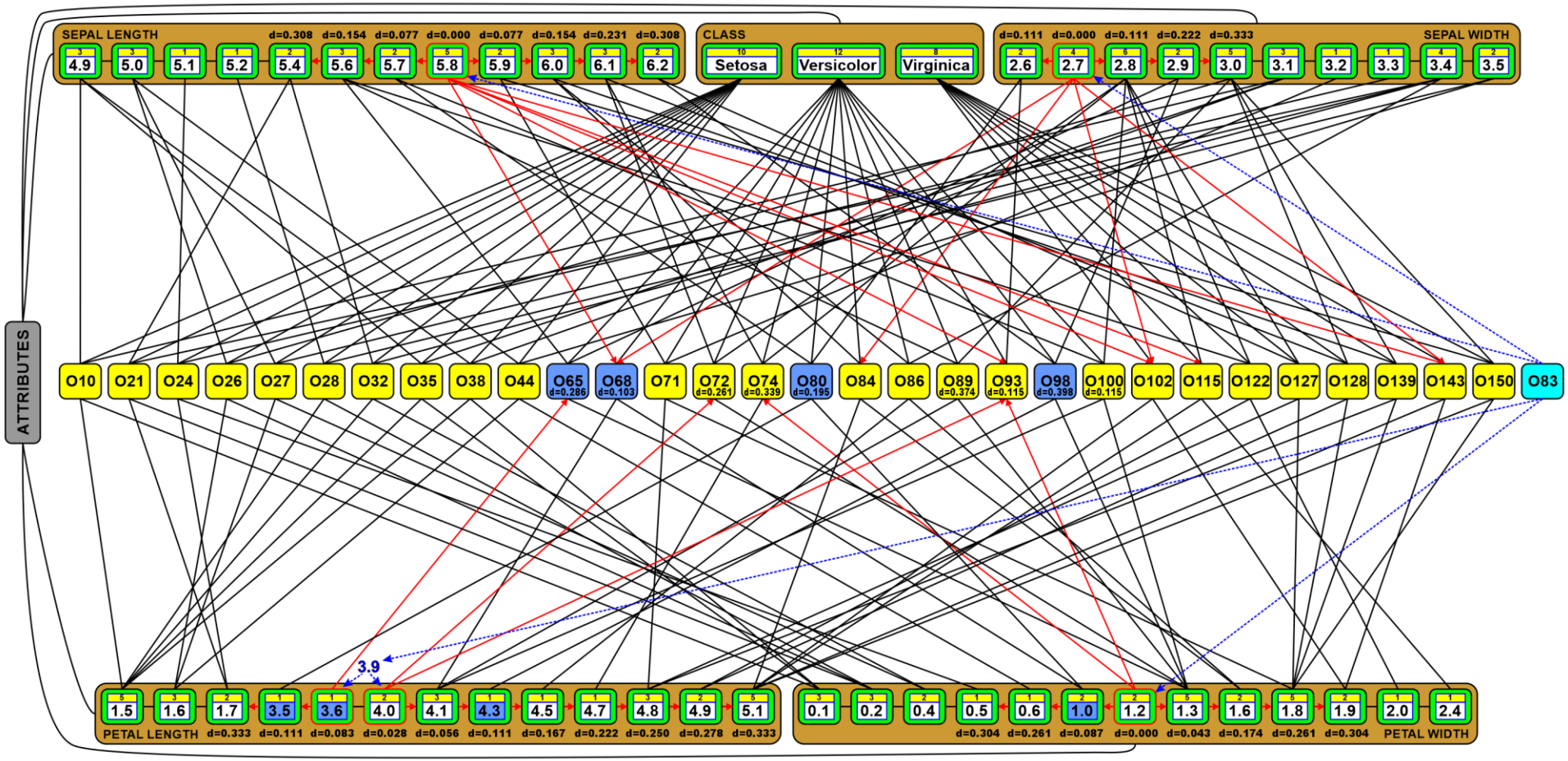
Rank List	
0.103	O68
0.115	O93
0.115	O100



Associative Search for Nearest Neighbors

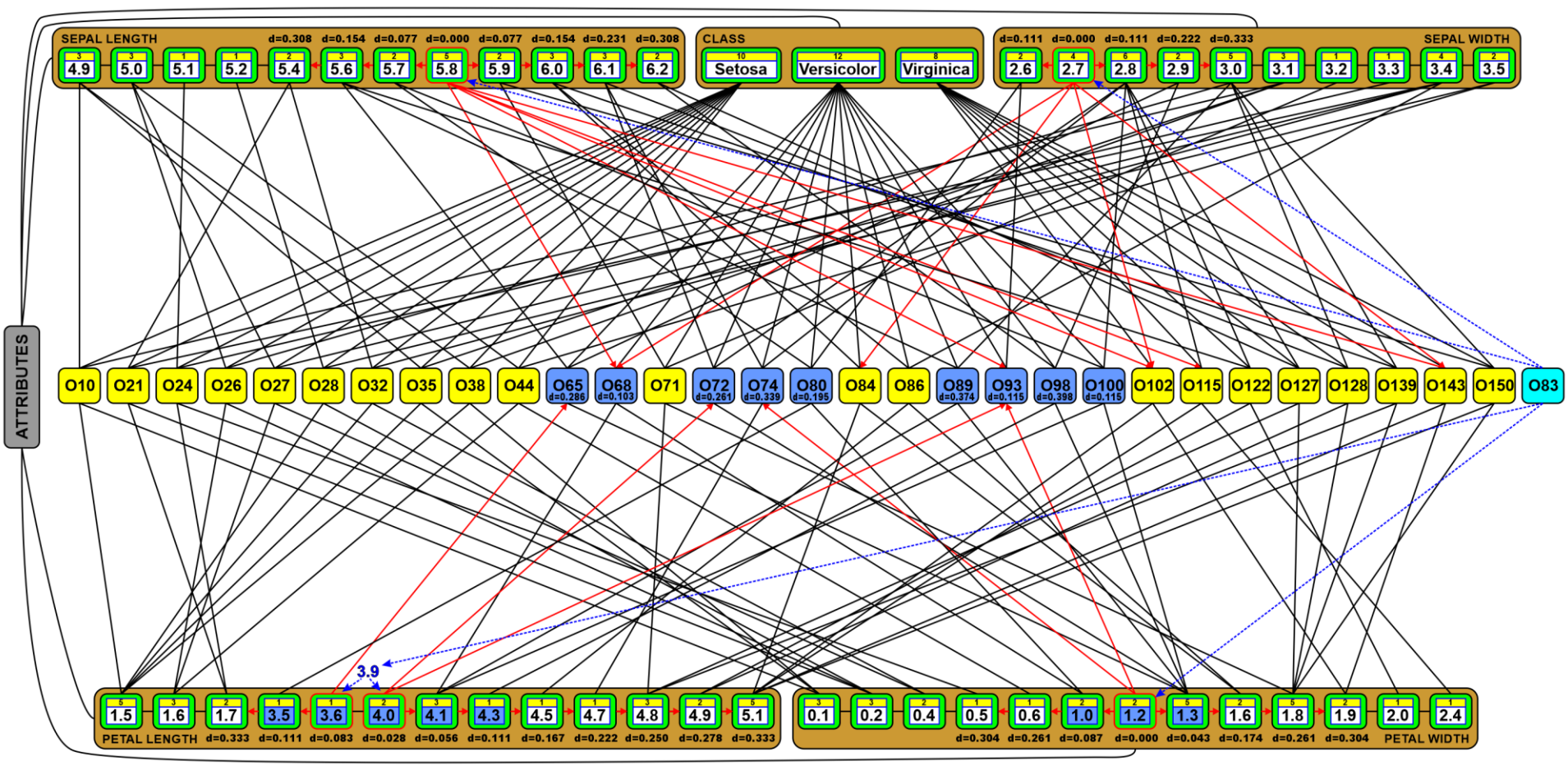
5. Move to the next closest values from the selected subset of the most variant attributes PL.3.6 (d=0.083), PW.1.0 (d=0.087), PL.3.5 (d=0.111), PL.4.3 (d=0.111) and compute Euclidean distances for connected objects O80 (d=0.195) to which the distance were not yet computed, and insert them into the rank list in sorted order. The rank list does not change during these four steps any more. Finally, the stop condition has been satisfied and the algorithm stops finding 3 nearest neighbors: O68, O93, and O100.

Rank List	
0.103	O68
0.115	O93
0.115	O100



Associative Search for Nearest Neighbors

Conclusion. As could be noticed, the algorithm required to go through 8 closest attribute values connected to 9 objects from 30 objects in this object set. So we have saved time for the computation of 21 Euclidean distances thanks to the use of the associative graph structures. This simple example is not representative for huge datasets, where the savings are much bigger, but it illustrates the use of one of the associative search algorithm presented in the accompanying paper that describe three sophisticated algorithms optimizing various aspects of such searches.



Experimental Results and Comparisons

Datasets				Create	Time Efficiency of the Algorithms				Speed comparisons		
Name of dataset	Data Volume			AGDS	KNN	AKNN-1	AKNN-L	AKNN-T	AKNN-1	AKNN-L	AKNN-T
	Samples	Attributes	k	ms	ticks	ticks	ticks	ticks	to KNN	to KNN	to KNN
Immunotherapy	90	7	2	1	146	145	113	68	1.01	1.29	2.15
Immunotherapy	90	7	3	1	164	153	127	72	1.07	1.29	2.28
Immunotherapy	90	7	5	1	171	161	133	84	1.06	1.29	2.04
Immunotherapy	90	7	10	1	199	166	173	93	1.20	1.15	2.14
Immunotherapy	90	7	20	1	211	177	167	105	1.19	1.26	2.01
Iris	150	4	2	1	154	48	29	19	3.21	5.31	8.11
Iris	150	4	3	1	175	52	32	23	3.37	5.47	7.61
Iris	150	4	5	1	194	66	44	27	2.94	4.41	7.19
Iris	150	4	10	1	220	83	57	45	2.65	3.86	4.89
Iris	150	4	20	1	238	97	84	61	2.45	2.83	3.90
Banknote	1372	4	2	46	1349	161	240	106	8.38	5.62	12.73
Banknote	1372	4	3	46	1434	201	264	156	7.13	5.43	9.19
Banknote	1372	4	5	46	1469	289	376	187	5.08	3.91	7.86
Banknote	1372	4	10	46	1526	368	498	260	4.15	3.06	5.87
Banknote	1372	4	20	46	1569	497	680	340	3.16	2.31	4.61
Wine Quality Red	1599	11	2	69	4073	2372	1063	730	1.72	3.83	5.58
Wine Quality Red	1599	11	3	69	4137	2724	1229	837	1.52	3.37	4.94
Wine Quality Red	1599	11	5	69	4219	3265	1377	903	1.29	3.06	4.67
Wine Quality Red	1599	11	10	69	4296	3275	1646	1058	1.31	2.61	4.06
Wine Quality Red	1599	11	20	69	4422	3564	1794	1172	1.24	2.46	3.77
Skin Data	245057	3	2	114198	42952	743	467	439	57.81	91.97	97.84
Skin Data	245057	3	3	114198	43076	825	527	510	52.21	81.74	84.46
Skin Data	245057	3	5	114198	43856	934	573	617	46.96	76.54	71.08
Skin Data	245057	3	10	114198	44401	1332	761	826	33.33	58.35	53.75
Skin Data	245057	3	20	114198	45477	1682	1001	1064	27.04	45.43	42.74
Eye	14980	14	2	3142	56531	61733	10293	8982	0.92	5.49	6.29
Eye	14980	14	3	3142	56599	61778	10312	9015	0.92	5.49	6.28
Eye	14980	14	5	3142	57372	62312	11407	9720	0.92	5.03	5.90
Eye	14980	14	10	3142	57898	64150	12376	10411	0.90	4.68	5.56
Eye	14980	14	20	3142	58280	65844	13422	11633	0.89	4.34	5.01

How much faster it is ?

Experimental Results and Comparisons

Datasets			Comparison Distances of the Best Classifications of AKNN-T				
Name of dataset	Data Volume		Euclidean		Sebestyen		Weights of the Attributes for Sebestyen Measure
	Samples	Attributes	Correct	k	Correct	k	
Immunotherapy	90	7	77.78%	3	100,00%	3	[2.674, 1.256, 0.649, 3.776, 1.912, 2.806, 3.436]
Iris	150	4	93.33%	3	100,00%	3	[0.651, 0.2353, 3.113, 0.825]
Banknote	1372	4	100.00%	3	100,00%	3	weighting unnecessary (all weights equal to 1)
Wine Quality Red	1599	11	58.75%	12	65.63%	28	[1.815, 2.451, 1.372, 0.817, 2.094, 2.810, 3.120, 2.820, 1.821, 0.871, 3.628]
Skin Data	245057	3	99.96%	3	99.96%	3	weighting unnecessary (all weights equal to 1)
Eye	14980	14	84.18%	3	85.85%	5	[1.172, 3.820, 2.254, 1.210, 2.885, 1.052, 0.878, 4.121, 2.581, 2.115, 2.528, 3.354, 0.206, 3.490]

Improvement of the classification performance

Conclusion:

Taking various attributes with various priorities improves the total performance of the classifier.

Conclusion and Remarks

- ✓ **Associative Graph Data Structures (AGDS) together with the associative search approaches allowed to find k Nearest Neighbors faster than the classic algorithm.**
- ✓ **Associative Graph Structures represent many more data relationships, which makes some algorithms working faster, simplifying search effort and saving time.**

Future Research of Associative Graphs

- ✓ **Representation of any data and relationships in one efficiently managed graph.**
- ✓ **Fast processing of vectorized data of any type: classic, sequential or structured.**
- ✓ **Representation of semantic memories and various data dependencies.**
- ✓ **Representation of knowledge in the cognitive systems.**
- ✓ **Combining with other deep neural network architectures to learn and represent more complex dependencies and use them in various computational tasks.**

Questions or Remarks?

Adrian Horzyk

horzyk@agh.edu.pl

Google: Horzyk

AGH

University of Science and Technology in Krakow, Poland

1. Basawaraj, **J. A. Starzyk, A. Horzyk**, Episodic Memory in Minicolumn Associative Knowledge Graphs, IEEE Transactions on Neural Networks and Learning Systems, Vol. 30, Issue 11, 2019, pp. 3505-3516, DOI: 10.1109/TNNLS.2019.2927106 (TNNLS-2018-P-9932).
2. **J. A. Starzyk**, Ł. Maciura**, A. Horzyk**, Associative Memories with Synaptic Delays, IEEE Transactions on Neural Networks and Learning Systems, DOI: 10.1109/TNNLS.2019.2921143 (TNNLS-2018-P-9188).
3. **A. Horzyk**, K. Gołdon, and **J.A. Starzyk**, Temporal Coding of Neural Stimuli, In: 28th International Conference on Artificial Neural Networks (ICANN 2019), Springer-Verlag, LNCS 11731, pp. 607-621, 2019, DOI: 10.1007/978-3-030-30493-5_56A. **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.
4. **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.
5. **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.
6. **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.
7. **A. Horzyk**, **J. A. Starzyk** and Basawaraj, ***Emergent creativity in declarative memories***, IEEE Xplore, In: 2016 IEEE Symposium Series on Computational Intelligence, Greece, Athens: Institute of Electrical and Electronics Engineers, Curran Associates, Inc. 57 Morehouse Lane Red Hook, NY 12571 USA, 2016, ISBN 978-1-5090-4239-5, pp. 1 - 8, DOI: 10.1109/SSCI.2016.7850029.
8. **Horzyk, A.**, ***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.
9. **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.
10. **A. Horzyk**, ***Human-Like Knowledge Engineering, Generalization and Creativity in Artificial Neural Associative Systems***, Springer-Verlag, AISC 11156, ISSN 2194-5357, ISBN 978-3-319-19089-1, ISBN 978-3-319-19090-7 (eBook), Springer, Switzerland, 2016, pp. 39 – 51, DOI 10.1007/978-3-319-19090-7.

Janusz A. Starzyk

starzykj@ohio.edu

Google: Janusz Starzyk

1804

OHIO UNIVERSITY

Athens, OH, U.S.A.

UNIVERSITY of INFORMATION TECHNOLOGY and MANAGEMENT in Rzeszow, POLAND