Hierarchy Visualisator - documentation v0.1

Łukasz P. Olech
e-mail: lukasz.piotr.olech@gmail.com
https://www.researchgate.net/profile/Lukasz_Olech2

July 10, 2016

1 Introduction

The proposed hierarchy visualisator was developed to provide a simple tool to present hierarchically-structured data graphically together with some dedicated measures. This tool should help in exploration and assessment of hierarchical data (e.g. created by some algorithm [1]). By principle, it is devoted to visualisation of a structure called Object Cluster Hierarchy (former Hierarchy of Clusters, Hierarchy of Groups) [1, 2, 3]. However, it can be successfully used with any hierarchically-structured data (e.g. rooted trees), that is compliant with a format described in 5. This format assumes that the hierarchy is built using three basic elements: **nodes**, **objects** (instances), and **parent-child relations** between nodes. *Nodes* represent groups, that must have names starting with *gen.* and followed by dot-delimited numbers. These numbers indicate the *parent-child relations* (position of nodes within the hierarchy). It is worth to note that any node can have any number of children or none. Furthermore, nodes contain any number of *objects* (data instances), including 0. Example hierarchy is shown in 1. In that figure node *gen.0.0* is empty (doesn't contain objects), and other nodes have a different number of objects. Node *gen.0.1* has two instances (*G, H*), whereas *gen.0.2* has four (*I, J, K , L*). In the example, one instance belongs to only one node, but it can belong to any number of nodes, similarly to *fuzzy clustering*. The number of children can also vary between nodes, e.g. node *gen.0* has 3 children, node *gen.0.2* has one child, and nodes *gen.0.1, gen.0.0.0, gen.0.0.1, gen.0.2.0* don't have any children.

The visualisator was designed to handle hierarchies with different structures. It means that the software is capable of clearly visualising a high, narrow hierarchy as well as a short, wide one. It can also handle a large number of nodes and instances.

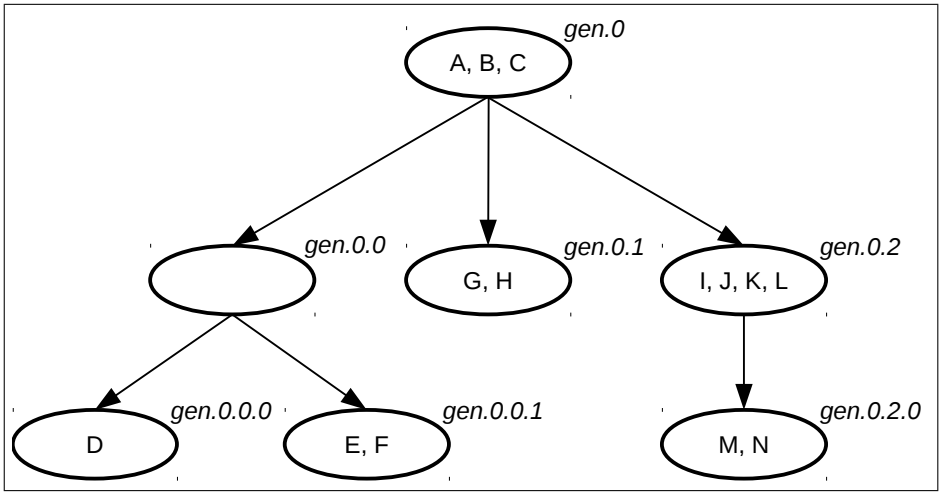

Figure 1: Example Object Cluster Hierarchy. Nodes are indicated by ellipses, node names are written in italics to the north-east of every node. Node instances are single capitalised letters written inside nodes. Arrows show the relations between nodes.

2 Obtaining The Visualisator

The source code of the visualisator can be easily cloned from GitHub repository `https://github.com/toSterr/hierarchy_visualisator` or downloaded from the web page of Department of Computational Intelligence, Faculty of Computer Science and Management, Wrocław University of Science and Technology, `http://kio.pwr.edu.pl/`. The GitHub repository contains source code, necessary `*.jar` files, and this document. Visit the GitHub to obtain the latest version of visualisator.

3 System Requirements

Visualisator was written and compiled with `Java 8 SE`, though it could be easily downgraded to `Java 6 SE`. The program uses several external libraries:

- `commons-cli-1.2.jar` - Apache Commons CLI library (version 1.2), used to handle command line with the program (`https://commons.apache.org/proper/commons-cli/`),
- `commons-lang3-3.4.jar` - Apache Commons Lang 3.0 library (version 3.4), used to handle some `ArrayUtils` operations (`https://commons.apache.org/proper/commons-lang/`),
- `commons-math3-3.6.jar` - Apache Commons Math 3.0 library (version 3.6), used to handle several mathematical operations (`https://commons.`

apache.org/proper/commons-math/),

- prefuse-beta-20071021.jar - Prefuse Visualisation Toolkit, the latest non-GitHub version downloaded from Prefuse web page, it was used in visualisation of the nodes of the hierarchy (http://prefuse.org/),
- utils.prefuse.histogram package in source code - it is an implementation of histograms in prefuse library, made by Kaitlin Duck Sherwood and Jeffrey Heer (http://blog.webfoot.com/2007/08/01/robobait-prefuse-histograms/). Their code was further modified, adjusted and bugfixed in order to suit my needs.

All the libraries are included in the GitHub project.

4 Running The Visualisator

The visualisator is compiled as runnable *.jar file, named hierarchy_visualisator.jar. This software is designed to be used by a command line. Its execution can be customised by a series of options, most of which have default values assigned. The purpose of default values is to provide a way of running the program quickly, with the necessity to specify only a minimal set of options. Additionally one should note that all of the options can be provided with both short (e.g. -h) and long (e.g. --help) form.

In order to list all command line options one should execute the hierarchy_visualisator.jar without any option or with -h (alternative --help) option, as shown below:

```
java -jar hierarchy_visualisator.jar -h
```

The output, by the time of creating that documentation will be as follows:

```
usage: java -jar hierarchy_visualisator.jar
 -b,--bins-number <number of bins>            Number of each histogram
                                              bins. Default: 100.
 -bg,--background-color <color>               Background color of every
                                              output image. Possible
                                              values: {green, black, blue,
                                              lightBlue, yellow, cyan,
                                              lightGray, gray, darkGray,
                                              magenta, orange, pink, red,
                                              white}. Default: white.
 -c,--class-attribute                         Provided input file contains
                                              also a ground truth class
                                              assignment. Class attribute
                                              will be omitted by this
                                              program. Assumed that class
                                              is in the second column
```

```
                                        (attribute) in the input
                                        file.
-cg,--child-group-color <color>         Color whit which all Child
                                        Groups (successors) will be
                                        painted on the output
                                        images. Possible values:
                                        {green, black, blue,
                                        lightBlue, yellow, cyan,
                                        lightGray, gray, darkGray,
                                        magenta, orange, pink, red,
                                        white}. Default: green.
-da,--display-all                       Display all points on the
                                        output images, so the other
                                        non-child groups (e.g.
                                        siblings and all parent
                                        groups) are also displayed.
-h,--help                               Prints this message.
-ht,--images-height <pixel number>      Height of the instances
                                        visualisation part (center
                                        image) on the output images.
                                        Provided in pixels. Default:
                                        600 px.
-i,--input <file path>                  Path to file with input
                                        data. It should be a
                                        properly formatted *.csv
                                        file.
-in,--instance-name                     Provided input file contains
                                        also a unique name of every
                                        instance, which will be
                                        omitted by this program.
                                        Assumed that instance names
                                        are in the third column
                                        (attribute) in input file
                                        when the class attribute is
                                        also provided or in the
                                        second column otherwise.
-lg,--current-level-group-color <color> Color which indicates
                                        current Level Group on the
                                        output images. Possible
                                        values: {green, black, blue,
                                        lightBlue, yellow, cyan,
                                        lightGray, gray, darkGray,
                                        magenta, orange, pink, red,
                                        white}. Default: red.
-o,--output <directory path>            Path where output *.PNG
                                        files showing every
```

	hierarchy level and a *.csv file with hierarchystatistics will be stored.
-og,--other-group-color <color>	Color with which all Other Groups (e.g. siblings) will be painted on the output images. Possible values: {green, black, blue, lightBlue, yellow, cyan, lightGray, gray, darkGray, magenta, orange, pink, red, white}. Default: lightGray. To display these points, the -da flag must be set.
-pa,--parent-ancestors-group-color <color>	Color with which current Parent group all ancestors will be painted on the output images. Possible values: {green, black, blue, lightBlue, yellow, cyan, lightGray, gray, darkGray, magenta, orange, pink, red, white}. Default: lightBlue. To display these points, the -da flag must be set.
-pg,--parent-group-color <color>	Color with which direct Parent Group (immediate ancestor) will be painted on the output images. Possible values: {green, black, blue, lightBlue, yellow, cyan, lightGray, gray, darkGray, magenta, orange, pink, red, white}. Default: blue. To display these points, the -da flag must be set.
-ps,--point-scale <real number>	Scaling factor (floating point number) of points drawn on images. Default: 1.0 (no scaling).
-sv,--skip-visualisation	Program will skip printing the output visualisations (images). Only hierarchy statistics file will be produced.

`-w,--images-width <pixel number>`	`Width of the instances visualisation part (center image) on the outputimages. Provided in pixels. Default: 800 px.`

So, given the above description, the simplest visualisator invocation would be:

```
java -jar hierarchy_visualisator.jar -i input/file/path.csv -o output/folder/path
```

The program is highly customizable, so one can change:

- width and/or height of center part of output images (that part is showing the instances in feature space), in order to have 100px width and 200px of height:

  ```
  java -jar hierarchy_visualisator.jar -i input/file/path.csv -o
  output/folder/path -w 100 -ht 200
  ```

- colors of both: nodes and points in images, in order to change all possible colors:

  ```
  java -jar hierarchy_visualisator.jar -i input/file/path.csv -o
  output/folder/path -cg orange -lg black -og pink -pa cyan -pg
  yellow -da
  ```

  remember to add `-da` flag for `-pg`, `-pa`, `-og` colors to be effective

- instances point size on output image, if one would like to have it twice bigger than default:

  ```
  java -jar hierarchy_visualisator.jar -i input/file/path.csv -o
  output/folder/path -ps 2
  ```

- number of histogram bins, in order to have 200 bins:

  ```
  java -jar hierarchy_visualisator.jar -i input/file/path.csv -o
  output/folder/path -b 200
  ```

- or combine options

  ```
  java -jar hierarchy_visualisator.jar -i input/file/path.csv -o
  output/folder/path -w 100 -ht 200 -b 200 -pa cyan -pg yellow -da
  ```

  There are more options that will be covered in the following sections.

5 Input File Format

The input file should be a properly formatted `*.csv` file. The comma separated file (`*.csv`) uses a semicolon (';') as a separator and does not require any header to be provided beforehand. Every row is assumed to represent one data instance, where first column has to indicate group to which the instance belongs, let's call it *assign-class column*. In the minimal input file, the *assign-class column* should be followed by at least **two** feature values. There is no upper limit on the number of feature values, but once decided, all rows should have the same number of dimensions.

The *assign-class column* should be in the form of `gen.X`, where the `X` are the integer numbers delimited by a dot ("."). These numbers indicate the position of a node within the hierarchy, where `gen.0` is the root node, `gen.0.0` is its left-most child, and `gen.0.1` it the root's child that is on the right of `gen.0.0`. See the example in 1. The groups in the *assign-class column* should be provided in DSF[1] (*depth first search*) order.

Optionally, the *assign-class column* can be followed by another class assignment - the *ground-truth-class column*. This assignment may indicate the external true assignment of instances to groups and the *assign-class column* is the assignment produced by some external classifier. The format of these group names should be the same as for the *assign-class column*, with the exception that the *ground-truth-class column* values do not need to be provided in DFS order (because this assignment depends on the instance not on the *assign-class column*). This data can be used in order to produce some external validation measures, e.g. [3]. In the current release this column is not used. In order to accurately parse input file with *assign-class column* one should run the program with `-c` flag:

```
java -jar hierarchy_visualisator.jar -i input/file/path.csv -o output/folder/path
-c
```

Furthermore, the *the ground-truth-class column* (or *assign-class column* if `-c` flag is not specified) can be followed by *instance-name* column. The purpose of this column is to give a name to every instance. Currently it is not used by the visualisator. This column's values can be any text. To properly parse the *instance-name* column, the `-in` flag should be provided:

```
java -jar hierarchy_visualisator.jar -i input/file/path.csv -o output/folder/path
-c -in
```

The summary of invocation flags and sample input file content is provided in 1.

[1] `https://en.wikipedia.org/wiki/Depth-first_search`

Execution flags	(none)	`-c`	`-c -in`	`-in`
File content	gen.0;0.5;0.7 gen.0.1;1.2;4.0 gen.0.3;1.2;4.0	gen.0;gen.1.2;0.5;0.7 gen.0.1;gen.0.1;1.2;4.0 gen.0.3;gen.0.3.1;1.2;4.0	gen.0;gen.1.2;first1;0.5;0.7 gen.0.1;gen.0.1;second2;1.2;4.0 gen.0.3;gen.0.3.1;third3;1.2;4.0	gen.0;first1;0.5;0.7 gen.0.1;second2;1.2;4.0 gen.0.3;third3;1.2;4.0

Table 1: Exemple input file content, depending on the input flags

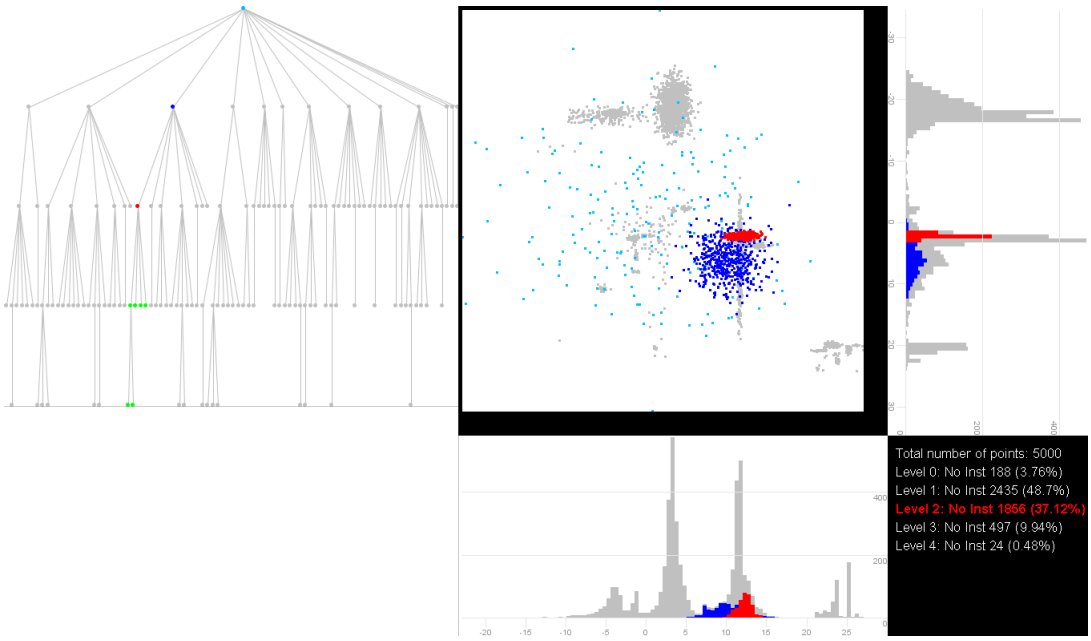

Figure 2: Examole output form visualisator.

6 Program Output

The hierarchy visualisator puts its results into the output folder. The results consist of `*.png` images and `*.csv` file. The number of produced images equals the number of groups in the input file because for every group one image is created. The exa mple image is shown in 2. The output image is divided into four sections. On the left, there is the visualisation of hierarchy structure, as described in 6.1. On the right of the hierarchy structure is visualisation of distribution of instances in a feature space 6.2. On the right side and below the visualisation of distribution of instances are histograms of those instances 6.3. On the bottom right, the simple level statistics are printed.

6.1 Hierarchy Structure

Hierarchy structure is drawn in a way to visualise all nodes, because of this the size of nodes is calculated dynamically. Every node has a color regarding its position and the image that is being analysed.

Using the default parameters provided in 4:

- Currently investigated node is in red color,
- Parent of current color is in blue color,
- All predecessors of the parent node described in the previous bullet are light blue,
- All children (direct and indirect) are green.

This view helps with navigation over images and shows how the hierarchy is structured.

6.2 Instances

In the centre of every image, a visualisation of instances is presented. **Only the first two features are displayed**. The color of instances is similar to these listed in 6.1. The size of every point can be altered using `-s` option.

6.3 Histograms

Based on the distribution of points in the feature space, the two histograms are drawn - each for corresponding dimension in 6.2. The number of bins can be changed by the `-b` flag. The purpose of these histograms and instances visualisation described in 6.2 is to give better intuition about generated points.

6.4 Measures

Measures contain a variety of data statistics. There are two places where the measures are printed. The concise form is printed directly on the output images (the bottom right corner). The longer (complete) version is in `NAME_hierayStatistics.csv` file that is located in output folder. The `NAME` is the name of used input file. If one is interested in the statistics only, there is a possibility to skip generation of images by using the `-sv` flag:

```
java -jar hierarchy_visualisator.jar -i input/file/path.csv -o output/folder/path
-sv
```

Using this flag the program execution time is much shorter.

The `NAME_hierayStatistics.csv` consists of following measures (provided in order of occurrence):

- `Total num of instances` – the overall number of instances in the whole hierarchy,
- `Avg num of children per node` – the number of children divided by the number of nodes,
- `Sample stdev num of children per node` – standard (sample) deviation of the `Avg num of children per node`,
- `Avg num of children per INTERNAL node` – similarly as for `Avg num of children per node`, but calculated only for nodes that have at least one child,
- `Sample stdev num of children per INTERNAL node` – similar as `Sample stdev num of children per node`,
- `Avg num of children per INTERNAL node with MIN BRANCHING FACTOR 2` – similarly as for `Avg num of children per node`, but calculated only for nodes that have at least **two** children,
- `Sample stdev num of children per INTERNAL node with MIN BRANCHING FACTOR 2` – similar as `Sample stdev num of children per node`,
- `Avg num of instances per node` – the number of instances divided by the number of nodes,
- `Sample stdev num of instances per node` – sample standard deviation of above-mentioned statistic,
- `Hierarchy height` – the longest path from root to leaf node,
- `Avg hierarchy width` – the number of nodes on every hierarchy level, divided by the number of levels,
- `Sample stdev hierarchy width` – sample standard deviation of above-mentioned statistic,
- `Number of nodes` – total number of nodes,
- `Number of INTERNAL nodes` – total number of nodes that have at least one child node,
- `Number of INTERNAL nodes with MIN BRANCHING FACTOR 2` – total number of nodes that have at least **two** children,
- `Number of leaves` – total number of leaves in the hierarchy,
- `Avg path length` – averaged length of paths from root to every leaf,
- `Sample stdev path length` – sample standard deviation of above-mentioned statistic,

- `Total number of points` – repetition of this statistic, the purpose is to ease the interpretation of the histogram mentioned below.
- Next there are several statistics presented as histogram, they should be read in columns:
  - `Level` – level number (histogram's bin),
  - `No Inst` – the number of instances on a particular level,
  - `% Inst` – the percentage value of the previous statistic,
  - `Avg. No of Children per node` – similar to `Avg num of children per node` but divided further into hierarchy levels,
  - `Stdev` – sample standard deviation of above-mentioned statistic,
  - `Hierarchy width` – number of nodes on a particular level,
  - `No of leaves` – number of leaves on a particular level (each level nodes count),
- `Branching factor histogram` – this is a histogram represented in the next two rows, where first row indicates the number of children for a node, and the second row counts nodes having a specific number of children.
- Each group's empirically computed parameters of Gauss distribution:
  - `Node` – name of group,
  - `Mean vector` – centroid of that group,
  - `Covariance matrix` – empirically computed covariance matrix.

References

[1] Łukasz P. Olech and Mariusz Paradowski. *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015*, chapter Hierarchical Gaussian Mixture Model with Objects Attached to Terminal and Non-terminal Dendrogram Nodes, pages 191–201. Springer International Publishing, Cham, 2016.

[2] Michal Spytkowski and Halina Kwasnicka. Hierarchical clustering through bayesian inference. In Ngoc Thanh Nguyen, Kiem Hoang, and Piotr Jedrzejowicz, editors, *ICCCI (1)*, volume 7653 of *Lecture Notes in Computer Science*, pages 515–524. Springer, 2012.

[3] Michal Spytkowski, P. Olech, Lukasz, and Halina Kwasnicka. Hierarchy of groups evaluation using different f-score variants. In Ngoc Thanh Nguyen, Bogdan Trawiski, H. Fujita, and T.-P Hong, editors, *Intelligent Information and Database Systems*, volume 9621 of *Lecture Notes in Computer Science*, pages –. Springer-Verlag Berlin Heidelber, 2016.