HyDraw Documentation

Ricardo Euler

August 31, 2017

Contents

Please send the bugs you come across or any other annotation to: `euler@zib.de`

1 Introduction

HyDraw is a javaview-based tool for the visualization of hypergraphs. It mainly consists of some simple commands to add nodes and hyperarcs and to possibly delay their execution so you can display a hypergraph that changes over time. HyDraw uses XML-based input.
To run Hydraw you have to use the startup script

```
$ HyDraw.sh
```

For more help type

```
$ HyDraw.sh -h
```

There are three ways of getting data into HyDraw: a) loading a file, b) reading stdin, c) writing directly in the console window. The standard way of loading a file into HyDraw is

```
$ HyDraw.sh myfile.xml
```

You can also open a file via the Menu→Load File option or by piping it at the program start. Any other source that produces XML-code suitable for HyDraw can also be piped. To read from stdin, write

```
$ cat myFile.xml | HyDraw.sh -s
```

Be aware that in this last case the Project functionality and options to open and close files are not available.
Lastly, you can pass your commands directly to the integrated console window. Generally, everything that can be written inside an input file can also be understood by the console.

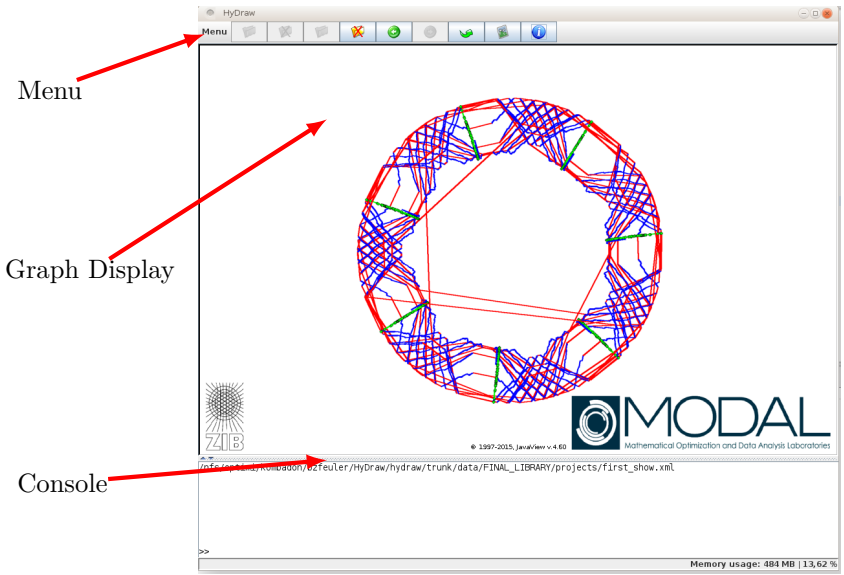


Figure 1: The HyDraw main window.

2 HyDraw Data Format

The input of HyDraw is XML-based. This means that there is a special XML schema for HyDraw.
A HyDraw graph file has to have the following structure.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<HyDraw>
<!-- Here goes all of your code-->
</HyDraw>
```

We now have a deeper look in what can be done in the body of this XML-file. Since commands are processed consecutively, you have to make sure that everything is defined at the time you refer to it.

2.1 Nodes and Hyperarcs

First of all, nodes should be added to the graph. This can be done with

```
<Node ID="0" color="#ff00ff" size="1"
      x="2" y="0" z="4"/>
```

You can also have a node in polar coordinates:

```
<Node ID="1" color="black"
      x="0" y="5" angle="3.14"/>
```

But you have to specify if you want to display the polar or Cartesian coordinates. The standard is Cartesian. See Section 2.2. Now, we define a first hyperarc by

```
<Hyperarc ID="0" color="blue" size="0.5" cost="4.5">
   <Link head="0" tail="1"/>
</Hyperarc>
```

To create a real hyperarc more nodes are needed. The missing attributes are set to standard values.

```
<Node ID="2" x="0" y="0" z="4" />
<Node ID="3" x="3" y="3" z="3" visible="false" />
<Hyperarc ID="1" cost="4.54">
   <Link head="0" tail="1"/>
   <Link head="2" tail="3"/>
</Hyperarc>
```

Note that a hyperarc has to have the same amount of head and tail nodes and that it is not allowed to contain the same node twice.
At some point you might want to remove some nodes and hyperarcs or to change some of their properties. For this purpose, you can use the following commands. Deleting a node will also delete all associated hyperarcs.

```
<deleteHyperarc ID="0"/>
<deleteNode ID="0"/>
```

If you want to delete all nodes and hyperarcs, respectively, you can use the `resetGraph` command.

```
<resetGraph/>
```

If you just want to change properties of nodes and hyperarcs you can again just invoke the `node` and `hyperarc` command, respectively.

```
<Node ID="2" color="#00000f"/>
<Node ID="0" size="0"/>
<Hyperarc ID="1" value="0" color="#ff00ff" size="3"/>
```

The IDs and the Links contained in a hyperarc cannot be changed.
If the color or size of a node or hyperarc was not set, HyDraw will use a global value that will be one of values found in Section 9 by default. If you want to change the defaults, you can use the `global` tag in the following way.

```
<global nodecolor="#00ff00" hyperarccolor="#00ff00" nodesize="1"
    hyperarcsize="1"/>
```

However, this does not affect nodes and hyperarcs with specified values and to this point there is no way to bring them back to take the global value.
You can reset the global values to their defaults by invoking

```
<restoreDefaultSettings/>
```

2.2 Coordinate System

We have already seen how to create nodes and how to specify their Cartesian coordinates. However, HyDraw allows you to employ more coordinate systems. Those that are available can be found in the table below. A node will only appear in a certain coordinate system if all necessary coordinates have been specified. The coordinate system can be changed by

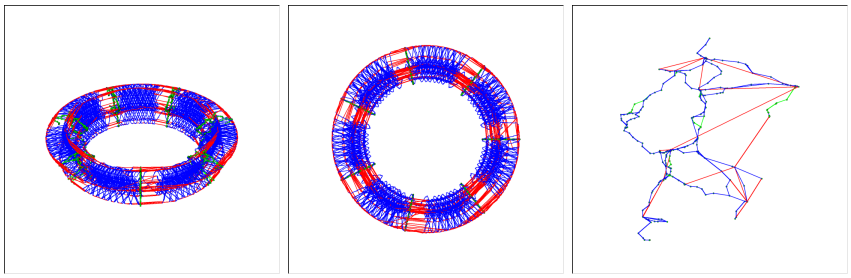
```
<CoordinateSystem type="xyz"/>
```

Naturally, a hyperarc will only be drawn if all his nodes have a specified location in the current coordinate system. The tree coordinates are an exception where no user-defined hyperarcs are drawn. Instead an arc is drawn for every parent-child-relation. In this case the concrete locations of the nodes will be calculated by HyDraw. In the following example a node is created with coordinates for every coordinate system.

```
<!--Creating the node without any coordinates.-->
<Node id="1"/>
<!-- Setting coordinates for the two and three dimensional
    cartesian coordinate systems.-->
```

xyz	3-dimensional Cartesian coordinates in x, y, z
xy	2-dimensional Cartesian coordinates in x, y
xz	2-dimensional Cartesian coordinates in x, z
yz	2-dimensional Cartesian coordinates in y, z
polar2d	polar coordinates in x and ϕ
polar3d	polar coordinates in x, ϕ and ρ
cylindric	cylindric coordinates in x, ϕ, z
tree	coordinates specified by parent-child relations

Figure 2: the different coordinate systems



(a) cylindric (b) polar (c) xz

Figure 3: The same data in different coordinate systems.

```
<Node id="1" x="1" y="1" z="1"/>
<!-- Setting polar coordinates. -->
<Node id="1" x="1" phi="1"/>
<!--The cylindric coordinate system uses the polar coordinates
    and adds a z-axis to them. -->
<Node id="1" z="1"/>
<!--Extending the polar coordinates to the third dimension. -->
<Node id="1" rho="1"/>
<!--Setting a node as its own parent will make it a root node in
     the tree. -->
<Node id="1" parent="1"/>
<!--A second node that has the previous node as its parent. -->
<Node id="2" parent="1"/>
```

When using polar or cylindric coordinates the x-coordinate is used as radius. Of course all those values can also be set at once:

```
<Node id="1" x="1" y="1" z="1" phi="1" rho="1" parent="1"/>
```

The coordinate system tag contains the `xShift` and `hStretch` attributes. Both do only affect polar or cylindric coordinate systems, respectively. The `xShift` is a constant value that is added to the radius of each node and the `hStretch` stretches the hypergraph along the z-axis in the cylindric coordinate system. The value for `hStretch` has to be greater than zero. Here is an example that does increase the radius by one and stretches at the Z-axis by a factor of two.

```
<CoordinateSystem xShift="1" hStretch="2"/>
```

You can use the different coordinate systems to show the same data in different perspectives. In the following example the x,z and *phi* coordinates were set.

2.3 Inhibit and continue drawing

HyDraw processes every command separately. Especially, this means that nearly every command will invoke a costly repaint of the graph display. If you draw many nodes and hyperarcs at once, this may cause a great slowdown. To avoid this, you can use

```
<inhibitDrawing/>
<!-- many commands -->
```

```
<continueDrawing/>
```

to inhibit repaints of the canvas, process many commands and then allow drawing again when you're done. Of course, you can also use this if you just want to make multiple changes to the graph model visible at once.

2.4 Fitting

It may happen that you specify coordinates that lie outside the visible part of the graph display. In this case you can hit the f key to scale and center your graph in the middle of the display. You can also do this with an XML command. If you do not want to to take care of if this yourself you can tell HyDraw to automatically fit the graph in the right position.

```
<!--center and scale graph in display area -->
<fit/>
<!--activate automatic fitting -->
 <autofiton/>
 <!-- deactivate automatic fitting -->
 <autofitoff/>
```

Fitting can also be controlled via Menu→Load File.

2.5 Show functionality

You might want to use HyDraw not just to draw your hypergraph, but to show some kind of steps that led to it in real time. For this purpose, you can use the `sleep`-tag. It will pause the processing of the input for the given amount of time. The following code snippet draws a node, waits for five seconds and then changes its color.

```
<Node id="1" x="1" y="1" z="1"/>
<sleep milliseconds="5000"/>
<Node id="1" size="3"/>
```

A similar effect can be obtained by the `block` command.

```
 <block/>
```

Here, the processing of your input is stopped until you press the space-bar with focus in the graph display.
You can also directly interrupt and continue the visualization by pressing backspace and space, respectively.

2.6 SCIP's Output for vbcTool

In addition to the standard input format, HyDraw is also able to process the output `SCIP` generates to visualize its branch-and-bound tree with `vbcTool`. For information on this format we refer to [2] and [3]. Be aware, however, that not the whole `vbcTool`-input can be read but just the part of it that is used by `SCIP`. HyDraw adds a search functionality that can be used to highlight, for example, some variables in the search tree. When reading files HyDraw automatically detects which input it is given, but when reading from stdin you have to set the -v flag. For example,

```
$ cat myvbc.dat | HyDraw.sh -v -s
```

3 Including Images

Imagine the nodes of your graph are all the cities in Germany. You may want to include a map of Germany(i.e. an image) to the graphs display to emphasize it. See Figure 4 for an example. The image is drawn as a texture on a quadrilateral in the canvas. If the image should be drawn in 3D space you have to give the coordinates of the quadrilaterals upper left, lower left and lower right corner since a hyperplane is defined by three points. For 2D it is enough to give the upper left and lower right corner. Specifying the lower left corner will have no effect. The code can look as follows

```
<image name="my_img" path="folder/file.xml" ulx="0" uly="1" ulz=
    "0" llx="0" uly="0" ulz="0" lrx="1" lry="0" lrz="0"/> <!--
    xyz coordinates-->
```

or

```
<image path="folder/file.xml" ulx="0" ulphi="1" ulz="0" llx="0"
    ulphi="0" ulz="0" lrx="1" lrphi="0" lrz="0"/> <!-- cylindric
    coordinates-->
```

The `name` attribute is optional. If it is not set, the path will be used as name. You can make an image visible or invisible, respectively, with the following code.

```
<image name="my_img" visible="bool"/> <!-- bool can be false/
    true or 0/1-->
```

It is not possible to change the coordinates of an already created image. The attempt will result in an error message. Images can be removed with

```
<removeImage name="my_img"/>
<resetImages/> <!-- deletes all images-->
```

In addition, it is possible to set a background image, using the command

```
<backgroundImage path="back3.png"/>
```

4 Node Information

You can pass your own informations to nodes in HyDraw. They will be visible if you click at a node after starting the pick-vertex-mode(You can choose it after right-clicking in the graph display). Additionally, the nodes ID, its size and color and the hyperarcs it is in will be displayed.
To set the information contained in a node invoke

```
<Node id="1" info="some information"/>
```

This overwrites all previously stored information. If you want to add information you have to use

```
<Node id="1" addInfo="some additional information"/>
```

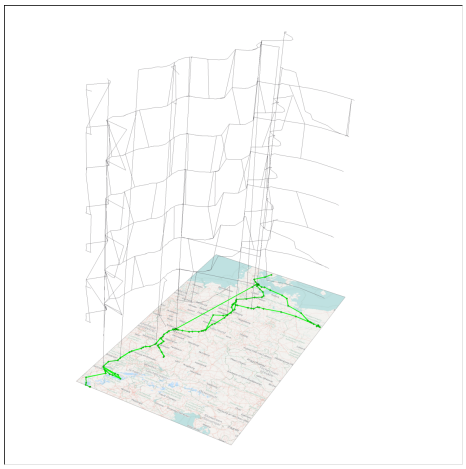


Figure 4: A graph that lies on a map of Germany. The graphs shows time-expanded train rotations through the depicted train network. Map data was taken from http://www.openstreetmap.org - Published under ODbL.

You can use the following escape characters to format your information text:

$\backslash n$	new line
$\backslash t$	tabulator
$\backslash b$	backspace

It is also possible to set labels to nodes which are visible in the graph display. You can do this by

```
<Node id="1" label="someText"/>
```

Labels can be activated and deactivated, respectively, with the following two tags.

```
<showNodeLabels/>
<hideNodeLabels/>
```

5 Projects

HyDraw allows you to wrap up a collection of input files in a project. This way you can assemble different HyDraw visualizations to a presentation.
You can load a project file by selecting “Open Project” in the menu bar. Using the left and right arrow keys or the respective buttons the input files can be browsed through.
Project files require the following syntax:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<HydrawProject>
```

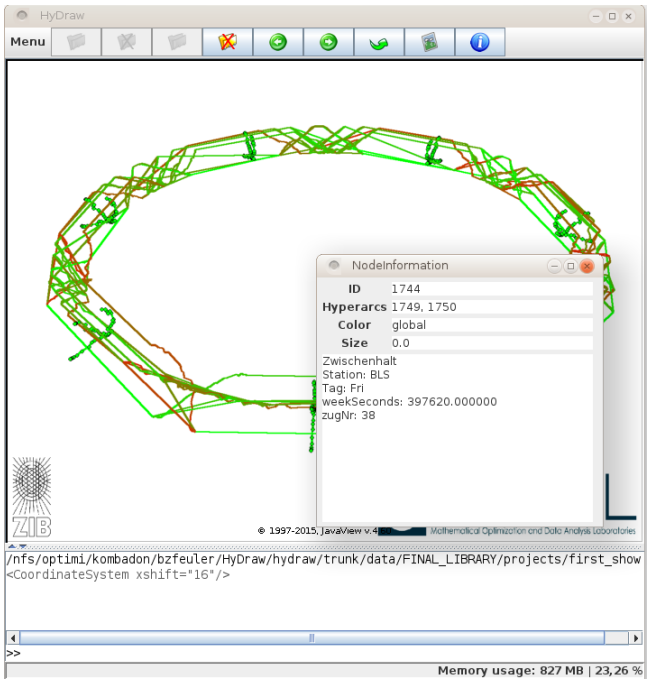


Figure 5: The node information window

```
<hydrawFile path="hydrawFile_One.xml"/>

<hydrawFile path="hydrawFile_Two.xml"/>

<hydrawFile path="some_dir/hydrawFile_Three.xml"/>

</HydrawProject>
```

HyDraw will look for those files in your current working directory and in `trunk` and `trunk/data`. You can add your own directories to the search path by using:

```
<base path="/my/base/path/"/>
```

6 Data Export

6.1 Png Export

It is possible to save the graph display as .png image. Go to Menu→PNG. You can then choose the resolution of your picture. Changing the resolution does affect the appearance of the graph in terms of the thickness of nodes and hyperarcs. If you want to save .png's in a high resolution you have to make sure during program start, that the VM allocates enough memory. This can be done by starting HyDraw with

```
$ java -jar -Xms<size> HyDraw_1.0.jar
```

There is also a XML-tag that does the same. Write

```
<takeImage path="your/path"/>
```

to make a .png and store it under path given as attribute. You can use this to make a series of multiple pictures to document changes in your hypergraph. So far, choosing the resolution is not supported.

6.2 Amira Export

You can export the hypergraph model of HyDraw to a file that can be understood by Amira, see [5]. Amira offers far more enhanced ways to visualize things than HyDraw does. Just go to Menu→Amira Export and choose your target directory. This export functionality is still in a test phase and only guaranteed to work for the cylindric and polar coordinates.

7 List of tags and attributes

The following list collects all tags and attributes with a short description.

HyDraw		Root tag for HyDraw files
CoordinateSystem		Options on the coordinate system.
type	String	Choose the coordinate system.
hStretch	double	Set horizontal stretch.
rShift	double	Set radius shift.
node		Create or change node.
id	int	Pass an integer as ID of this node.
color	Hex	Set this nodes color. You can pass a hexadecimal color code.
size	double	Set the size of the node to the passed double.
x	double	Set the nodes x-coordinate to the passed double.
y	double	Set the nodes y-coordinate to the passed double.
z	double	Set the nodes z-coordinate to the passed double.
phi	double	Set the angle for polar coordinates.
info	String	Set the string as information to the node.
addInfo	String	Add the string to the nodes information.
label	String	Set a label to be displayed next to the node.
hyperarc		Create or change hyperarc.
ID	String	Give this hyperarc an integer as identifier.
color	Hex	Set this nodes color. You can pass a hexadecimal color code.
size	double	Set the size of the hyperarc to the passed double.
link		Define a link.
tail	int	Set a node as tail of this link by passing its ID.
head	int	Set a node as head of this link by passing its ID.
sleep		Let the processing of the input sleep for a while.
milliseconds	int	Stop processing of the file for the passed amount of seconds.
block		Stop processing until space was pressed.
deleteNode		Delete the node.
id	int	Specify the node for deletion.
deleteArc		Delete the hyperarc.
id		Specify the hyperarc for deletion.
global		Change properties of nodes and hyperarcs globally.

nodecolor	Hex	Set the global node color.
nodesize	double	Set the global node size.
arccolor	Hex	Set the global hyperarc color.
arcsize	double	Set the global hyperarc size.
label	bool	Set visibility of node labels to true(1) or false(0).
resetGraph		Delete the whole graph.
reset		Delete the whole graph and all images.
inhibitDrawing		Inhibit drawing.
continueDrawing		Continue drawing.
autofiton		Graph is scaled and centered to fit to the display after every command.
autofitoff		Disable autofit.
fit		Scale and center graph in the middle of the display.
backgroundImage		Use a png image as background.
path	String	If empty a white background is used.
image		Place a png image freely in the canvas.
ulx	double	Set x coordinate of upper left corner.
uly	double	Set y coordinate of upper left corner.
ulz	double	Set z coordinate of upper left corner.
ulphi	double	Set phi coordinate of upper left corner.
llx	double	Set x coordinate of lower left corner.
lly	double	Set y coordinate of lower left corner.
llz	double	Set z coordinate of lower left corner.
llphi	double	Set phi coordinate of lower left corner.
lrx	double	Set x coordinate of lower right corner.
lry	double	Set y coordinate of lower right corner.
lrz	double	Set z coordinate of lower right corner.
lrphi	double	Set phi coordinate of lower right corner.
visible	bool	Set visibility to true(1) or false(0)
name	String	Set an intern name for the image.
path	String	Path to the image file.
removeImage		Deletes image.
name	String	Name of the image that should be deleted.
removeImages		Deletes all images.
takeImage		Make a picture of the graph display.
path	String	Stores picture in the given file.
restoreDefaultSettings		Restores default settings.
filename		You can optionally pass have pass the name of the file you loaded.

8 Shortcuts

The following shortcuts are available.

Left Arrow	previous project file
Right Arrow	Next project file
Space	Continue visualization
Backspace	Pause visualization
o	Rotate graph
v	Rotate graph with fixed vertical axis
s	Scale graph
t	Translate graph parallel to display
z	Translate graph back and forth
c	Center camera on graph
f	Fit graph into display
r	Reset viewer to default

9 Standard values

Most of the XML-attributes are not mandatory. If you do not use them, one of the following standard values will be used. There are also commands to change those values, see Section 7.

property	**value**
node size	1.0
node color	#ff0000
hyperarc size	1.0
hyperarc color	#000000
coordinate system	xyz
xShift	0
hStretch	0
autofit	off
nodelabels	hidden

References

[1] *DragonConsole*. `https://github.com/bbuck/DragonConsole`.

[2] Gerald Gamrath et al. *The SCIP Optimization Suite 3.2*. eng. Tech. rep. 15-60. Takustr.7, 14195 Berlin: ZIB, 2016.

[3] Sebastian Leipert. *vbcTool*. `http://www.informatik.uni-koeln.de/ls_juenger/vbctool/`. 2000–2004.

[4] Konrad Polthier. *JavaView*. `http://www.javaview.de/`. 1997–2016.

[5] Detlev Stalling, Malte Westerhoff, and Hans-Christian Hege. "Amira: a Highly Interactive System for Visual Data Analysis". In: *The Visualization Handbook*. Elsevier, 2005, pp. 749–767.