Optimization for Non-Mathematicians

Drawing Graphs with `Graphviz`

MATLAB itself only started with release 2015b to offer functions for drawing graphs as in Figure 0.1.[1] However we continue to use the open source software `Graphviz` in this class. `Graphviz` consists of a series of executable programs (`dot`, `neato`, `fdp`, `sfdp`, `twopi` and `circo`) for the layout of graphs. These differ in the algorithms used to place the nodes.[2]

There exists a MATLAB interface for `Graphviz` named `graphViz4Matlab`, which does not offer access to all capabilities, but is sufficient for our purpose. Especially for this class `graphViz4Matlab` has been equipped with several extensions[3] by us, which meanwhile have been included in the official version. In addition, a high level interface function `Graph` has been written, which further simplifies the usage of `graphViz4Matlab`.[4]

Installation of `graphViz4Matlab` and `Graph`

For the installation the following steps are necessary (*only once*). Please login under **Linux** in the MRZ computer pool.

(a) Download the latest version of `graphViz4Matlab` by typing in the console:

```
git clone https://github.com/graphviz4matlab/graphviz4matlab.git
```

The downloaded content can then be found in the subdirectory `graphviz4matlab` (of the current directory).

(b) Download the file `Graph.m` from the class homepage and save it in the above mentioned directory `graphviz4matlab`.

(c) For the transformation of adjacency to incidence matrices and vice versa we use the two files `adj2inc.m` und `inc2adj.m` by Ondrej Sluciak. These can be obtained (as two `.zip` archives) from Matlab Central File Exchange by searching for *adjacency incidence matrix*.[5] Unzip these archives into the above mentioned directory `graphviz4matlab`. Alternatively, the files `adj2inc.m` und `inc2adj.m` can be found on the class homepage.

[1] The MATLAB native functions `graph` and `digraph` create graphs from adjacency matrices, which can be displayed using `plot`.

[2] Anyone who is interested in more details will find the documentation for `Graphviz` under `https://www.graphviz.org/Documentation.php`, see e.g., the document Using Graphviz as a library (Drawing graphs with Graphviz).

[3] These concern the possibility to adjust the line style of the edges (`-edgeLineStyle`), to attach labels to the edges (`-edgeLabels`) and to draw bidirectional edges as two separate edges, slightly offset.

[4] `Graph` enables us to draw a graph based on its incidence matrix (Definition 14.4), which is also used in the lecture. Since `graphViz4Matlab` expects an adjacency matrix (Definition 14.3), the incidence matrix is transformed first. Moverover the interface `Graph` facilitates the use of node and edge labels.

[5] Matlab Central is the place to go for MATLAB projects which users share among each other.

(d) Download the file `setup_graph.sh` from the class homepage and save it in your home directory.

Subsequently you should follow the steps below *in every session* in which you would like to use `graphViz4Matlab`:

(a) Execute the script `setup_graph.sh`, which sets several paths for `Graphviz` and `graphViz4Matlab`. To this end, type in the console:

```
source setup_graph.sh
```

The script will also query the version of `dot` (one of the programs out of `Graphviz` suite); the output should look like:

```
dot - graphviz version 2.28.0 (20130928.0220)
```

(b) Launch MATLAB, e.g., by typing `matlab`.

By calling (typing) `graphViz4Matlab` in the MATLAB window you should now see an example graph. Furthermore try the example programs `graphViz4MatlabDEMO1`, `graphViz4MatlabDEMO2` and `graphViz4MatlabDEMO3`.

Using `Graph`

You will find the following code segments on the class homepage under the name `demoGraph.m`. You may paste each code block into the command window, or use `echodemo demoGraph` for an interactive demonstration.

First we generate the incidence matrix of a graph which we would like to draw — as seen in Figure 0.1 top left. Then we call our `Graph` interface of `graphViz4Matlab`:

demoGraph.m

```
% Generate the incidence matrix for a transportation
% network with m = 9 nodes and n = 11 edges
A = ...
[ -1 -1  0  0  0  0  0  0  0  0  0; ...
   0  0 -1 -1  0  0  0  0  0  0  0; ...
   0  0  0  0 -1 -1  0  0  0  0  0; ...
   1  0  1  0  1  0 -1 -1 -1  0  0; ...
   0  1  0  1  0  1  1  0  0 -1 -1; ...
   0  0  0  0  0  0  0  1  0  0  0; ...
   0  0  0  0  0  0  0  0  1  0  0; ...
   0  0  0  0  0  0  0  0  0  1  0; ...
   0  0  0  0  0  0  0  0  0  0  1];

% Draw the graph
h = Graph(A);
```

The return value `h` is a handle for the generated `graphViz4Matlab` object.[6] Unfortunately the automatic has worsened compared to previous versions. However we may access and modify the node positions through the handle `h`: demoGraph.m

```
XY = h.getNodePositions();
XY = XY([1 4 5 2 3 6 7 8 9],:);
h.setNodePositions(XY);
```

Next we show how to change the standard labels $1, 2, \ldots, m$ of the nodes to individual labels (see Figure 0.1 top right): demoGraph.m

```
% Set the node labels for the production sites,
% storage sites and points of sale
nodelabels = {'P1','P2','P3','Z1','Z2','V1','V2','V3','↪
   V4'};

% Redraw the graph
h = Graph(A,h,nodelabels);
XY = h.getNodePositions();
XY = XY([1 4 5 2 3 6 7 8 9],:);
h.setNodePositions(XY);
```

The edges can be labeled in a similar way (Figure 0.1 center left): demoGraph.m

```
% Set additional edge labels
edgelabels = [0.8; 2.0; 2.5; 1.0; 1.2; 2.0; 1.0; 1.0; ↪
   1.0; 1.0; 1.0];
h = Graph(A,h,nodelabels,edgelabels);
XY = h.getNodePositions();
XY = XY([1 4 5 2 3 6 7 8 9],:);
h.setNodePositions(XY);
```

Now we rotate the generated graph by 135 degree and scale it down slightly, so that it fits completely into the picture (Figure 0.1 center right): demoGraph.m

```
% Set node positions XY manually
% Here: rotate graph about 135 degree
alpha = 135 * pi / 180;
R = [cos(alpha), -sin(alpha); sin(alpha), cos(alpha)];
XY = h.getNodePositions();
XY = XY - 0.5;              % shift to origin
XY = (R * XY')';            % rotate
XY = XY * 0.8;              % scale on 80%
XY(:,1) = XY(:,1) + 0.5;    % shift back
XY(:,2) = XY(:,2) + 0.4;    % shift back
h.setNodePositions(XY);
```

[6] We use the handle, among other things, to be able to overwrite the graph described by `h` in its window. This prevents a new window to be opened upon every call of `Graph`.

parameter -layout in graphViz4Matlab	Graphviz routine
Gvizlayout	neato
Treelayout	dot
Radiallayout	twopi
Circularlayout	circo
Springlayout	fdp
Circlelayout	—
Gridlayout	—
Randlayout	—

Table 0.1: available layouts in graphViz4Matlab and their counterparts in Graphviz

Finally we show the usage of two different layouts (Figure 0.1 bottom left and right), here again without edge labels.[7]

demoGraph.m

```
% Redraw the graph with tree layout
h = Graph(A,h,nodelabels,[],'-layout',Treelayout);
```

demoGraph.m

```
% Redraw the graph with circle layout
h = Graph(A,h,nodelabels,[],'-layout',Circlelayout);
```

You can find out about available layouts by `h.layouts` or in Table 0.1.

Your next steps should be:

(a) Read `help Graph`.

(b) Try the different buttons which you can see in the plot window of the graph. Also move some nodes in the graph and query the new node positions.

(c) Read `help graphViz4Matlab`.

(d) Experiment with different layouts for the graph by using some options of `graphViz4Matlab` which have not been mentioned above, e.g., the coloring of edges and / or nodes.

[7] All arguments starting from the fifths are passed on from our interface `Graph` to `graphViz4Matlab`, see also `help Graph` and `help graphViz4Matlab`.

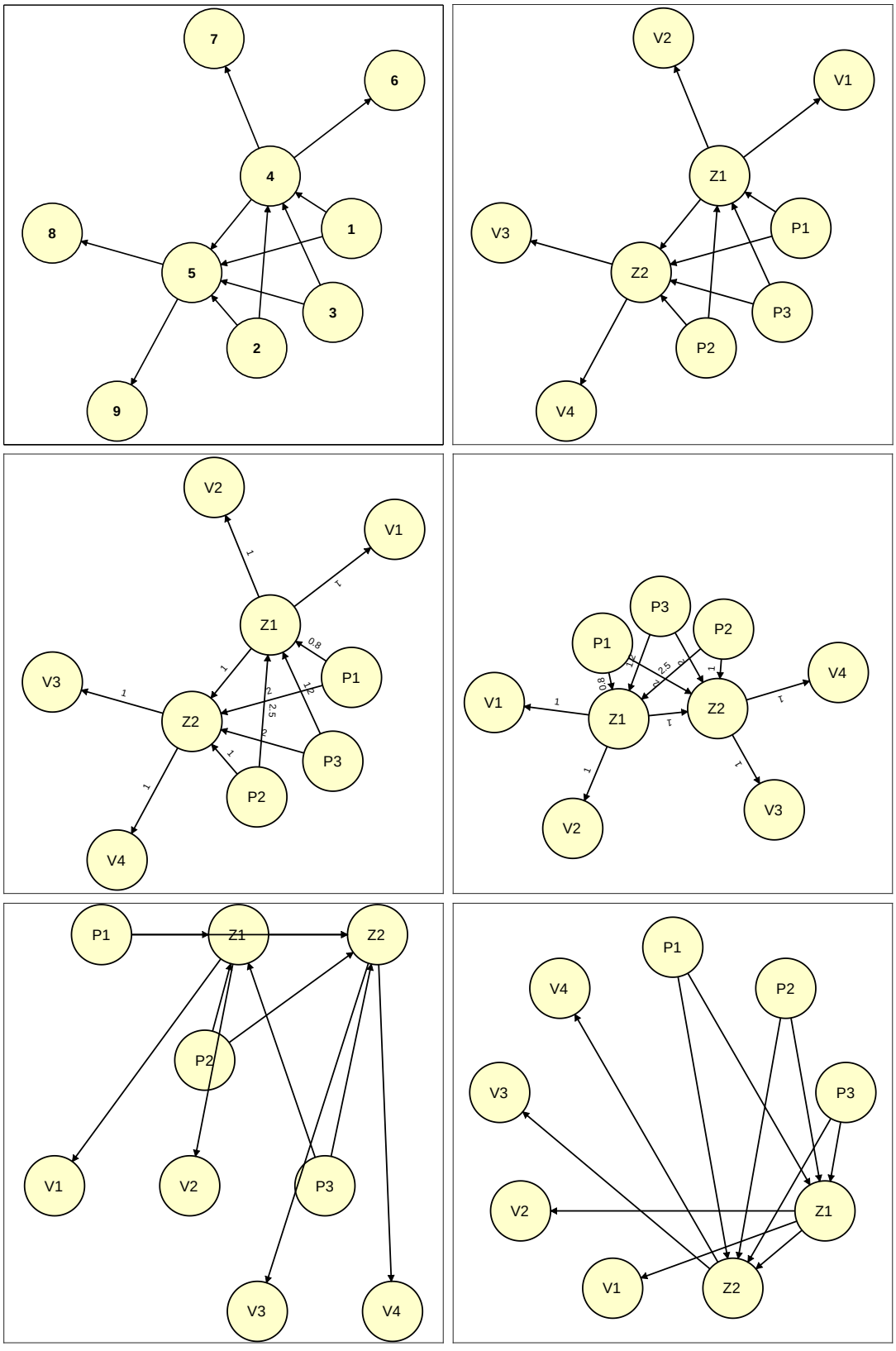

Figure 0.1: In the top left figure you see the standard layout of the graph. Top right: node labels have been assigned. Center left: edge labels have been set in addition. Center right: the node positions have been changed manually. The two figures on the bottom show the graph in the `Treelayout` and `Circlelayout`, respectively.