A Tool for Pixelated Graph Representations

Thomas Bläsius · Fabian Klute · Benjamin Niedermann · Martin Nöllenburg
Karlsruhe Institute of Technology (KIT)

Pixelated Graph Representations

At GD 2013 in Bordeaux

Visibility Representation (VR)

a planar graph

its visibility representation

We want to minimize the height of a **visibility representation**.

Pixelated VR

pixelation

Boolean grid for node

We represent every line as a collection of **PIXELS**

For every object there is a **grid of Boolean variables** indicating which pixels represent the object.

ILP Constraints

We can formulate **ILP constraints** that force the pixels to form a VR.

- nodes are boxes of height 1
- edges are boxes of width 1
- nodes never share a pixel
- a node and an edge share a pixel ⇔ they are incident
- edges share pixels only at their endpoints

overlaps

Restrict the height of the VR by restricting the grid-height.

We can model **many other problems** in this way.

1-Dimensional
- pathwidth
- bandwidth
- *st*-orientation

2-Dimensional
- bar *k*-VR
- boxicity

I would like to have a **GUI** where I can input a graph, select a set of constraints and get the resulting drawing.

Uh . . . Er . . .
Aye, that would be cool . . .

Back in Karlsruhe

Can you write a program that lets the user **load a graph**, runs the **ILP** and displays the **resulting drawing**?

Sure!

The user must be able to **select constraints** out of a predefined list.

Shouldn't be too hard.

It should be **easy** for the user to **specify custom constraints**.

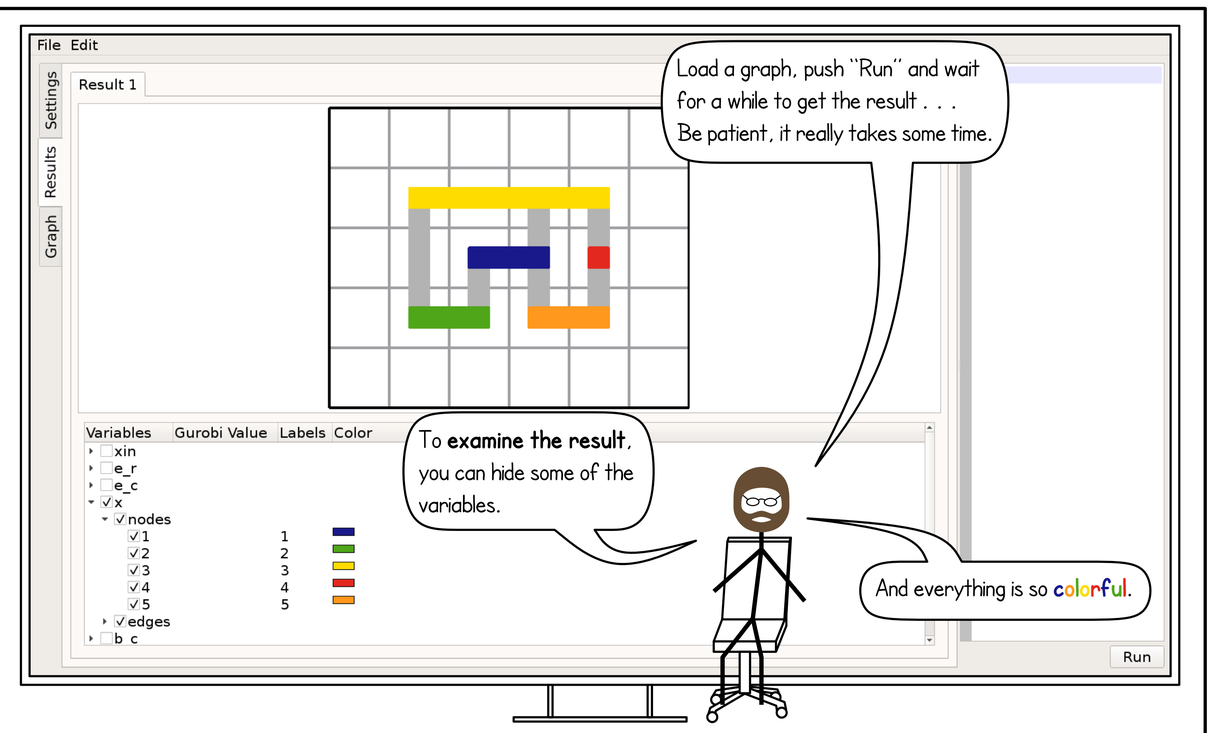
Uhh . . . I can try . . .

The "Pixelated Graphs" Tool

You can now select a subset of **predefined constraints** (macros).

I selected the constraints required for a visibility representation.

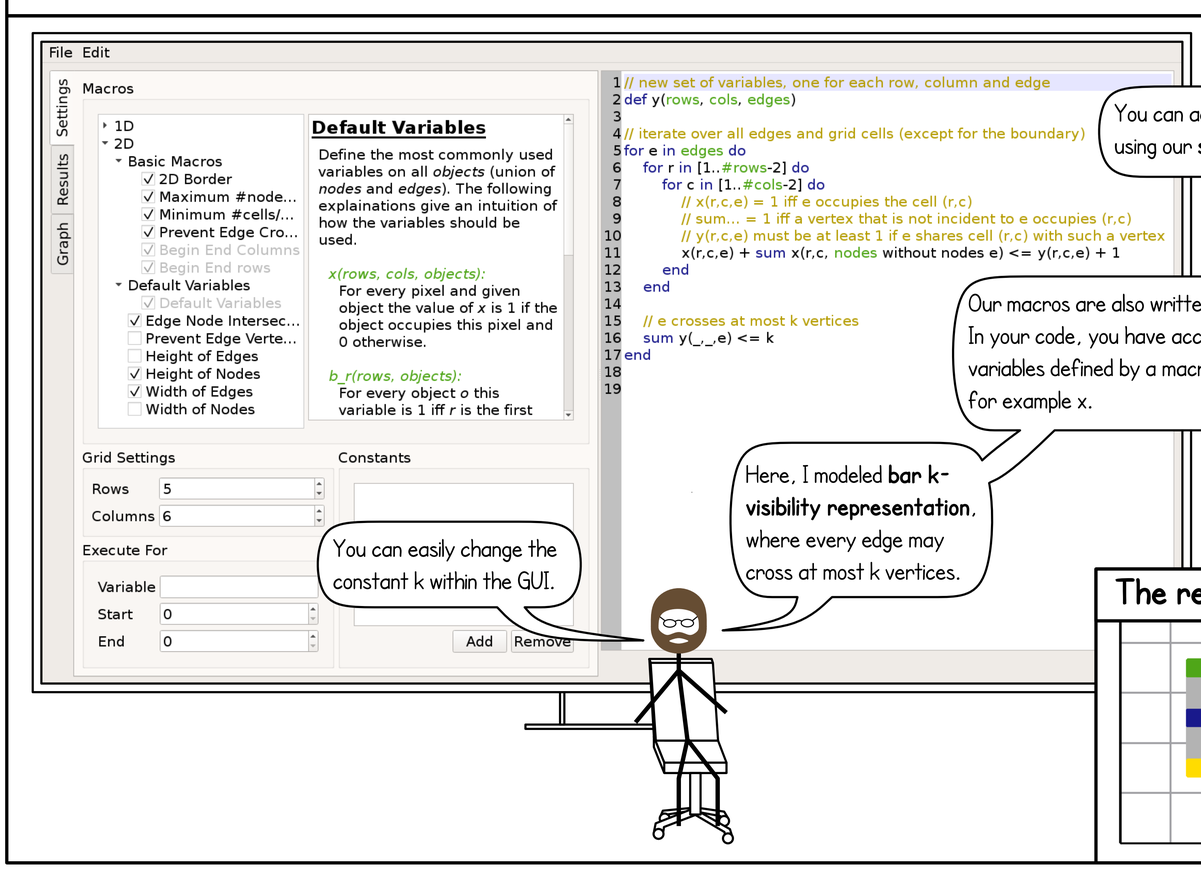
You also have to set the height and width of the grid.



Load a graph, push "Run" and wait for a while to get the result . . .
Be patient, it really takes some time.

To **examine the result**, you can hide some of the variables.

And everything is so **colorful**.



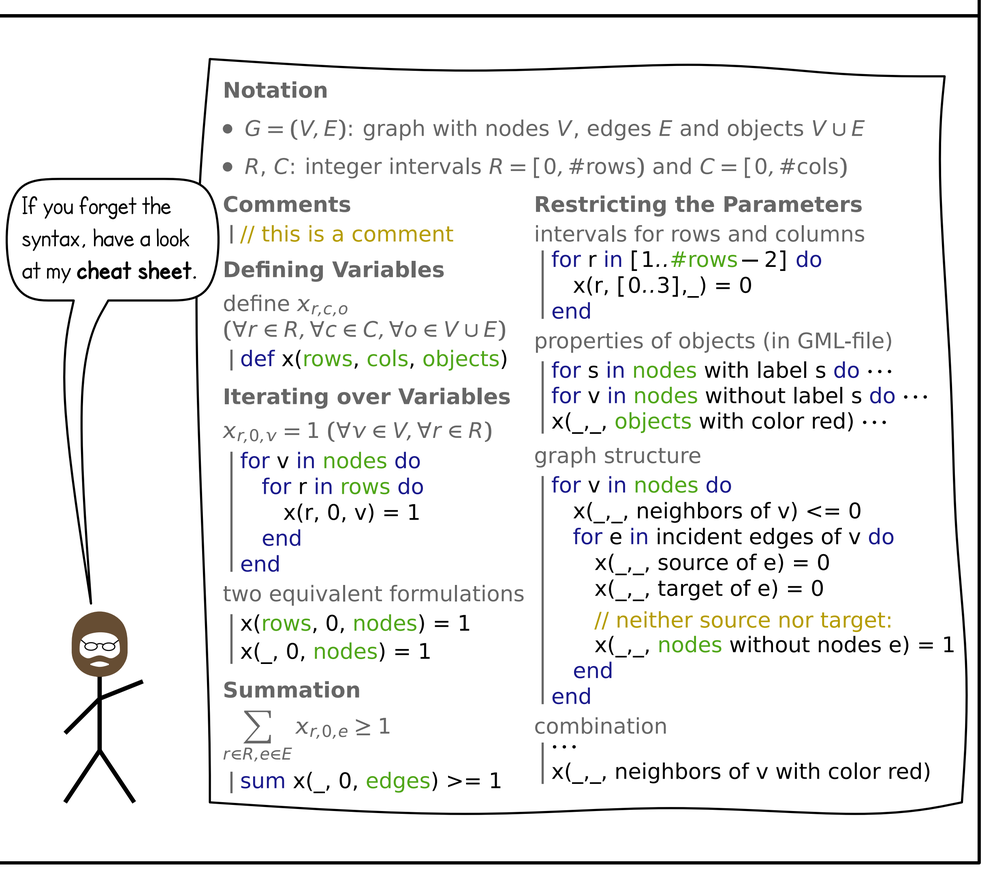
You can add your own constraints using our **simple language PGL**.

Our macros are also written in PGL. In your code, you have access to variables defined by a macro like for example x.

Here, I modeled **bar k-visibility representation**, where every edge may cross at most k vertices.

You can easily change the constant k within the GUI.

The resulting drawing



If you forget the syntax, have a look at my **cheat sheet**.

Notation

- $G = (V, E)$: graph with nodes V, edges E and objects $V \cup E$
- R, C: integer intervals $R = [0, \#\text{rows})$ and $C = [0, \#\text{cols})$

Comments

```
// this is a comment
```

Defining Variables

define $x_{r,c,o}$ $(\forall r \in R, \forall c \in C, \forall o \in V \cup E)$

```
def x(rows, cols, objects)
```

Iterating over Variables

$x_{r,0,v} = 1$ $(\forall v \in V, \forall r \in R)$

```
for v in nodes do
  for r in rows do
    x(r, 0, v) = 1
  end
end
```

two equivalent formulations

```
x(rows, 0, nodes) = 1
x(_, 0, nodes) = 1
```

Summation

$\sum_{r \in R, e \in E} x_{r,0,e} \geq 1$

```
sum x(_, 0, edges) >= 1
```

Restricting the Parameters

intervals for rows and columns

```
for r in [1..#rows − 2] do
  x(r, [0..3],_) = 0
end
```

properties of objects (in GML-file)

```
for s in nodes with label s do ···
for v in nodes without label s do ···
x(_,_, objects with color red) ···
```

graph structure

```
for v in nodes do
  x(_,_, neighbors of v) <= 0
  for e in incident edges of v do
    x(_,_, source of e) = 0
    x(_,_, target of e) = 0
    // neither source nor target:
    x(_,_, nodes without nodes e) = 1
  end
end
```

combination

```
···
x(_,_, neighbors of v with color red)
```

Based on: *Using ILP/SAT to Determine Pathwidth, Visibility Representations, and other Grid-Based Graph Drawings*
Therese C. Biedl · Thomas Bläsius · Benjamin Niedermann · Martin Nöllenburg · Roman Prutkin · Ignaz Rutter
Graph Drawing 2013, Pages 460–471, LNCS 8242, Springer International Publishing