Layout in Visual Modelling

Gitte Bluekens

S0110627

Inspired by D. Dubé, Graph Layout for domain-specific modeling (2006)

Table of contents

Layered drawing technique

- Phase 1: layer assignment
  - greedy cycle removal algorithm
  - layering algorithm
    - BFS layering
    - Longest-path layering
    - Minimum width layering
- Phase 2: crossing reduction

Greedy cycle removal

Algorithm 1 Greedy cycle removal

Input: A graph G=(V,E)

Output: An acyclic and topologically sorted graph G

```
V ← topological sort of V
for all v in V do
    for all child in v.getChildren() do
        if Π(child) < Π(v) then
            reverse edge between v and child
        end if
    end for
end for
```

BFS layering

1. Find the root vertices
2. Recursively label discovered vertices the next layer

- Optimal height
- Unbounded width

Longest-path layering

1. Layering leaf vertices at layer 1
2. Vertices are added to successive layers if all their children are in layers below them

- Optimal height
- Unbounded width

Minimum width layering

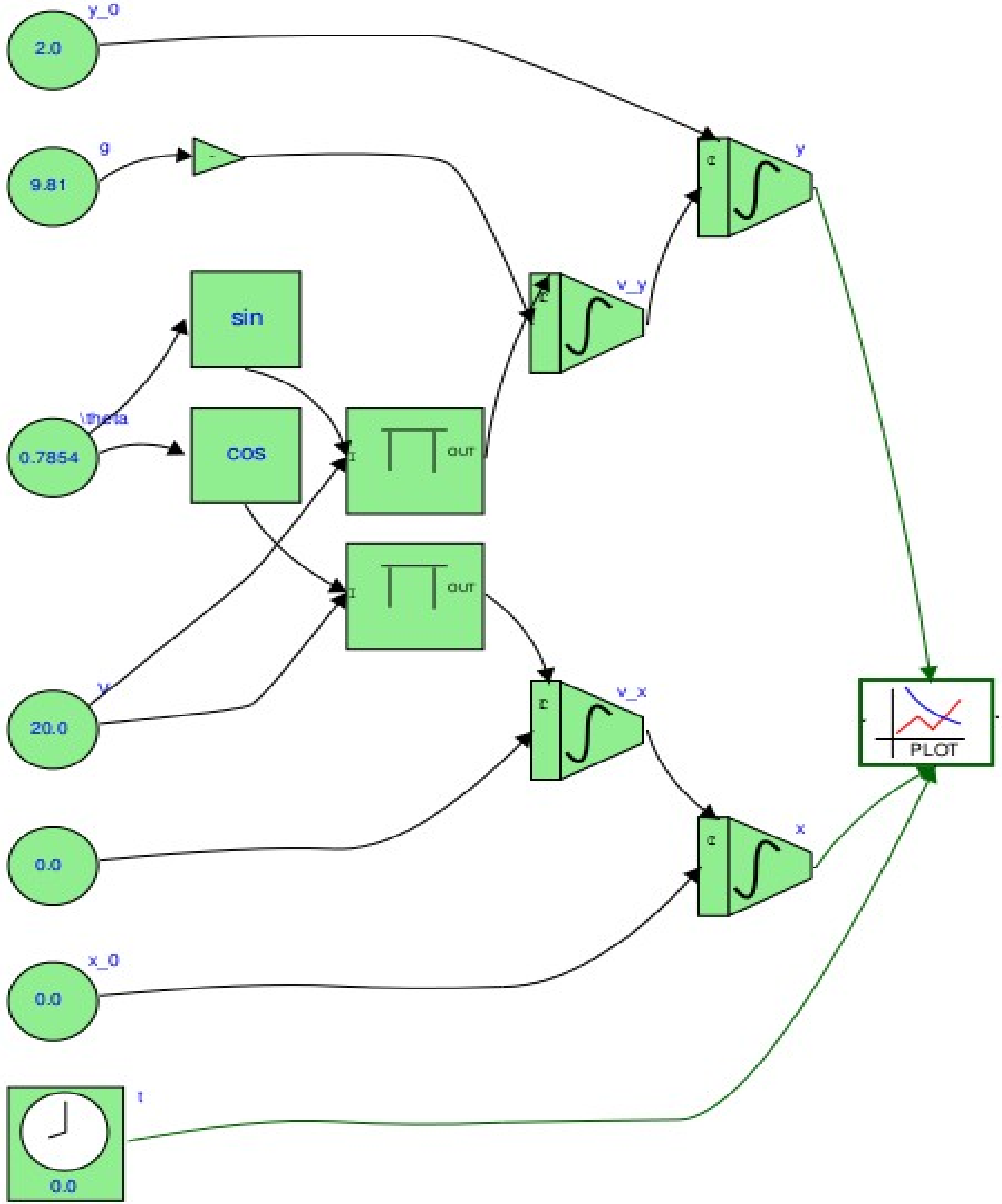
1. Longest path algorithm
2. Connect all unconnected vertices

   Place them in the first or last layer

- Yields drawings with good aspect ratios
- Creates longer edges that traverse multiple layers

Crossing minimization

- Layer-by-layer sweep
  - Stores current best ordering of vertices, as compared to the best number of crossings seen so far.
  - Stopping condition:
    - Hard limit on iterations
    - Crossing reduction doesn't change
    - No crossings remain



y_0
2.0
g
9.81
sin
\theta
0.7854
cos
OUT
OUT
v
20.0
0.0
x_0
0.0
t
0.0
v_y
y
v_x
x
PLOT

Spring-embedder algorithm

- Edges = springs
- Vertices = rings
- Pre-processing step recommended
  → improve convergence speed and quality

Spring-embedder algorithm

1. Acquire center coordinates of vertices
2. Set 2D force vectors to zero
3. Set repulsion charges to diagonal lenght of vertex
4. Loop: calculate forces acting on vertices

Repulsion algorithm

1. Calculate Manhattan and Euclidean distances
2. Calculate scalar force
3. Multiply force by 2D Manhattan distance vector

- avoid vertex overlaps
- generate large repulsive forces if overlap

Attraction algorithm

- Treats edges as physical springs

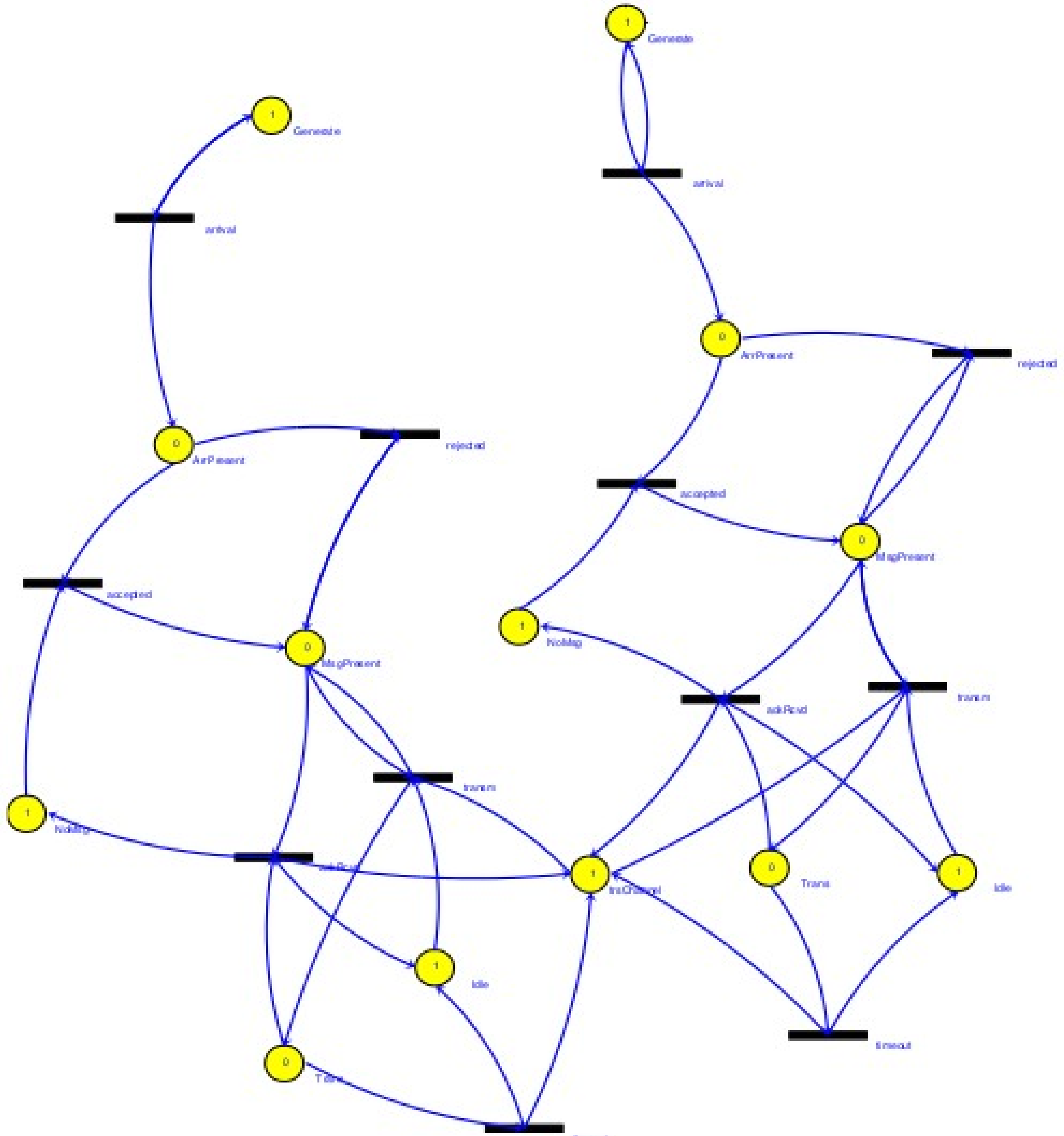
1. Calculate Manhattan and Euclidean distances
2. Calculate spring force
3. Multiply force by 2D Manhattan distance vector

Gravity algorithm

1. Impart on each vector a velocity towards the gravitational field source

2. Calculate force vector

- Increase area usage efficiently

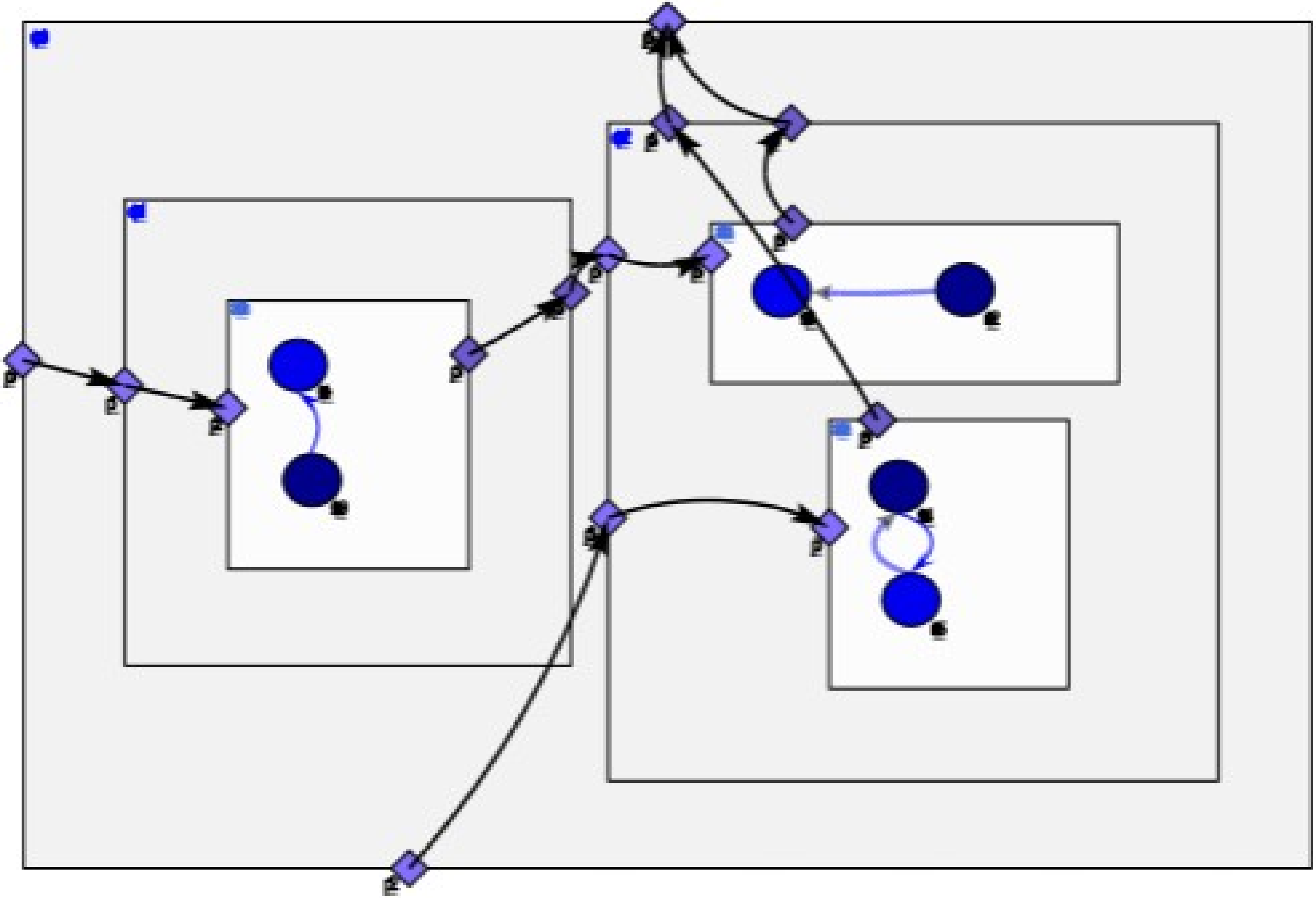


1
Generate
arrival
0
ArrPresent
rejected
accepted
0
MsgPresent
transm
1
NoMsg
1
1
Idle
0
timeout
1
Generate
arrival
0
ArrPresent
rejected
accepted
0
MsgPresent
1
NoMsg
ackRcvd
transm
1
0
Trans
1
Idle
timeout

Force-tranfer layout algorithm

- Initialization phase
  - Set forces acting on each vertex to zero
  - Set position of vertex to its center coordinate
- Simulation phase
  - Each vertex exerts forces on overlapping neighboring vertices

    1. Calculate Manhattan and Euclidean distances
    2. Compute scalar force magnitude
- Termination
  - No more overlap
  - Fixed number of iterations



Tree-like layout algorithm

1. Find root vertices of graph

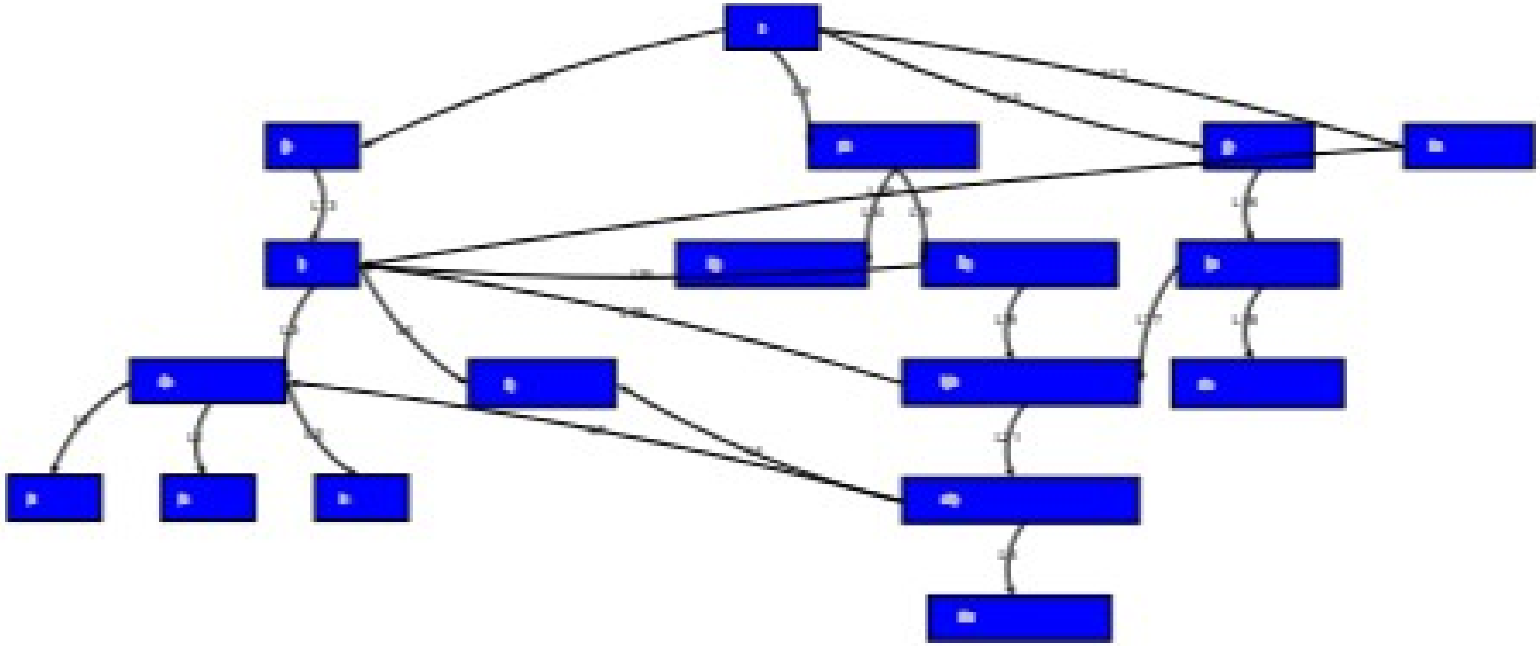
2. Recursively assign coordinates to children of root before root itself.

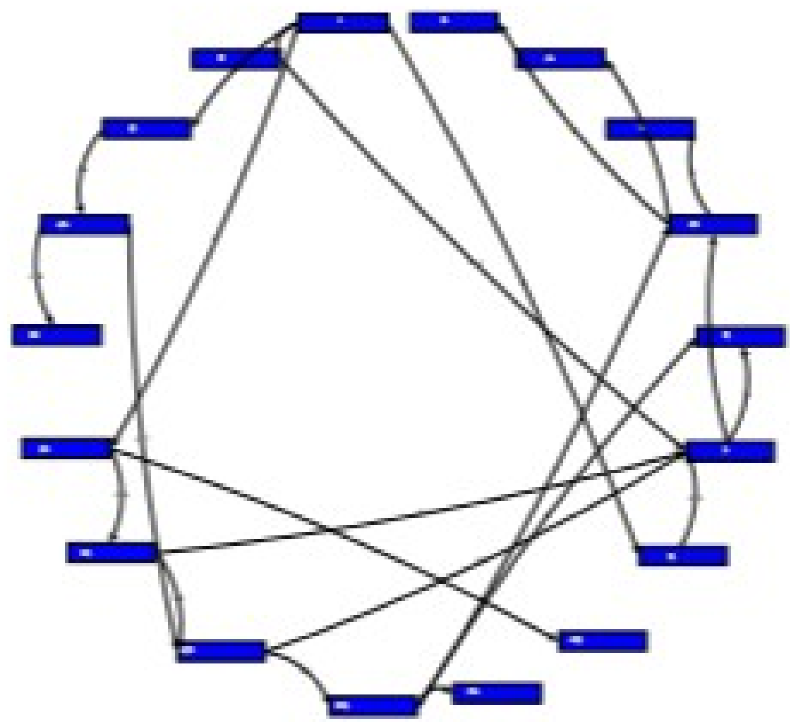
- Graph structures that are really trees

Circle layout algorithm

1. Sort vertices topologically
2. Calculate perimeter of circle
3. Calculate interval fraction

- Subgraphs or small graphs
- Preprocessing step for force directed method





Implementation project

- Effectiveness of linear constraints in AtoMPM
  - Integration of linear constraints with AtoMPM
  - Dealing with visual icons
- Layered technique in appropriate language