Technical Report

# Efficient, Proximity-Preserving Node Overlap Removal


Claire Pennarun
Tatiana Rocher

Bordeaux 1
Projet d'Etude et de Recherche



January 11, 2014

# Contents

# Chapter 1

# Subject presentation and state of the art

A graph is a data structure encoding information with the use of nodes and edges (which are binary relations between nodes).

Graph drawing aims to represent given information as a graph, generally through a ”node-link” layout, letting nodes and edges be displayed.

Most of the layout algorithms consider nodes as points, but some need to let appear additional information as labels. For example, London subway maps would be useless without the indication of the stations on the lines.

This could lead to an overlap of some nodes. That must be avoided, as it clearly confuses the understanding of the graph.

Many approachs are generally considered ; the easiest to apply is to ”scale” the layout until no overlaps occur. This method has the advantage to preserve the global shape of the layout, but the area of the graph can become very inconvenient. That is why a compromise between the preservation of the shape of the graph and a minimization of the total area has to be found.

Different algorithms have been devised to answer the problem.

Our project consists in understanding the algorithm PRISM proposed by Gansner and Hu in [**?**] and in analyzing the feasability of its implementation as a plugin for the Tulip software.

# Chapter 2

# PRISM algorithm

## 2.1 Description of the algorithm

The PRISM algorithm focuses on two main constraints for the final layout of the graph. First, the area taken by the layout must be minimal. The second constraint is to preserve the global "shape" of the original layout by maintaining all proximity relations between the nodes.

**Algorithm 1**: PRISM

**Input**: $p_i^0$ : coordinates of each vertex
width $w_i$ and height $h_i$ of each vertex $(i = 1, 2, ..., |V|)$

**repeat**
    $G_{DT}$ : proximity graph of $G$ by Delaunay triangulation
    **for** *all edges of* $G_{DT}$ **do**
        Compute the overlap factor
    $\{p_i\}$ : solution of the proximity stress model
    $p_i^0 = p_i$
**until** no more overlaps along edges of $G_{DT}$;
**repeat**
    $G_{DT}$ : proximity graph of $G$ by Delaunay triangulation
    Find overlaps in $G$ through a scan-line algorithm
    Add the overlapping edges to $G_{DT}$
    **for** *all edges of* $G_{DT}$ **do**
        Compute the overlap factor
    $\{p_i\}$ : solution of the proximity stress model
    $p_i^0 = p_i$
**until** no more overlaps found by the scan-line algorithm;

## 2.2 Complexity

# Chapter 3

# Implementation within Tulip

## 3.1 Tulip node structure

## 3.2 Resolution of the stress model

## 3.3 Scan-line algorithm

# Chapter 4

# Tests and results

## 4.1 Use of GraphViz

# Chapter 5

# Conclusion