École Nationale Supérieure d'Électronique, Informatique, Télécommunications, Mathématique et Mécanique de Bordeaux

1 avenue du Dr Albert Schweitzer
B.P. 99 33402 Talence Cedex

RAPPORT DE PROJET PER

Déformation de Grille pour la visualisation d'information

Notice analytique

École Nationale Supérieure d'Électronique, Informatique, Télécommunications, Mathématique et Mécanique de Bordeaux
1 avenue du Dr Albert Schweitzer
B.P. 99 33402 Talence Cedex

Titre :	
Résumé :	
Mots clés :	
Caractéristiques :	1 volume - x pages - x annexes
Type de travail et durée :	
Date de publication :	

THANKS

Terminologie

Terme	Signification
xx	xx
xx	xx
xx	xx

Abstract

français

Abstract

english

CONTENTS

LIST OF FIGURES

INTRODUCTION

The article summarized in this document was written by A.Lambert, R.Bourqui and D . Auber, researchers in LaBRI in Bordeaux.

The article we are currently working on deals with how to visualize graphs containing many nodes and edges. With improvements in data acquisition comes an increase of the size and the complexity of graph and this huge amount of data generally causes visual clutter, in our case due to edges crossing.

For example, it could be interesting to visualize data in fields like biology, social sciences, data mining, or computer science and then emphasize their high-level pattern to help users perceive underlying models.

Some classes of graph, such as trees or acyclic graphs, clearly facilitate user understanding by effective representation. However, most of graphs do not belong to these classes and algorithm giving nice results in terms of time and space complexity but also in terms of aesthetic criteria for any graph do not exist yet. For example, force-directed method produces pleasant and structurally significant results but do not help users comprehension due to data complexity.

Up to now, several techniques were used to reduce this clutter, based on compound visualization or edge bundling. In a compound visualization, nodes are gathered into metanodes and inter-cluster edges are merged into metaedges. To retrieve the information, metanodes could be collapsing or expanding. Edge bundling technique routes edges into bundles. This uncover high level edge pattern and emphasize relationships.

Yet an important constraint is the impossibility for some nodes to move while avoiding edges crossing because node positions bring information: the compound visualization is consequently not suitable.

Some existing representations take into account this duty: some reducing edge clutters (Edge routing, Interactive techniques, Confluent Drawing, Node clustering, Edge clustering) and the other enhancing edge bundles visualization (Smoothing curves, Coloring edges). Edge routing use shortest-path edge routing to bound the number of edge crossings and use non-point-size to avoid node-edge overlaps. It do not highlight the underlying model. Interactive techniques remove clutter around the user's focii in a fisheye-like manner while preserving node position: this technique does not reduce the clutter of the entire representation. In Confluent Drawing, groups of crossing edges are drawn as curved overlapping lines. Node Clustering routed edge along the hierarchy tree branches. Both methods can not be applied to any graph. Edge Clustering route edges either on the outer face of the circle or in its inner faces and bundle them to optimize area utilization.

Our solution is based on an edge bundling technique coupled with a grid built from the original graph. We also used a GPU-based rendering method to highlight bundles densities without losing edge color.

Finally, our resolution allow an improvement of the clutter reduction and the performance compared with existent methods

Distribution of nodes: Graph coloring

In order to implement a parallel asynchronous version of Tutte algorithm, it is necessary to separate graph nodes into different sets. Each node must be independant with others node in the same set. This problem is similar to the famous problem of graph colorating.

The objectif of the modified Tutte algorithm is to handle graph of thousands nodes. To separate such a number of nodes, it is more performant to use a heuristic of the algorithm of graph coloring.
The greedy algorithm is a simple and good solution to separate nodes into sets fastly and efficitively.

The algorithm used in this project is :

```
G={V,E}
Y = V
color = 0
While Y is not empty
   Z = Y
   While Z is not empty
      Choose a node v from Z
      Colorate v with color
      Y = Y - v
      Z = Z - v - {neighbors of v}
   End while
   color ++
End while
```

Reference : Introduction to Algorithms (Cormen, Leiserson, Rivest, and Stein) 2001, Chapter 16 "Greedy Algorithms". =======

CHAPTER 1

THE CONTEXT

CHAPTER 2 INITIALISATION

2.1 Grid Presentation

2.2 Tutte Algorithm

CHAPTER 3

TUTTE'S ALGORITHM

This part of text comes from the article [1].

In this paper, we will use basic graph theory terminology, see for example [2]. Let $G = (V, E)$ be a planar graph. A mapping Γ of G into the plane is a function $\Gamma : V \cup E \to P(\mathbb{R}^2)$ which maps a vertex $v \in V$ to a point in $\mathbb{R}^2$ and an edge $e = uv \in E$ to the straight line segment joining $\Gamma(u)$ and $\Gamma(v)$. A mapping is an embedding if distinct vertices are mapped to distinct points, and the open segment of each edge does not intersect any other open segment of an edge or a vertex.

In 1963, Tutte [3] gave a way to build embeddings of any planar, 3-connected graph $G = (V, E)$. Let C be a cycle whose vertices are the vertices of a face of G in some (not necessarily straight-line) embedding of G. Let Γ be a mapping of G into the plane, satisfying the conditions:

- the set Ve of the vertices of the cycle C is mapped to the vertices of a strictly convex polygon Q, in such a way that the order of the points is respected;

- each vertex in $V_i = V\ V_e$ is a barycenter with positive coefficients of its adjacent vertices (Tutte assumed all coefficients to be equal to 1, but the proof extends without changes to this case). In other words, the images v of the vertices v under Γ are obtained by solving a linear system (S): for each $u \in V_{i,v|uv \in E} \lambda_{uv}(u - v) = 0$, where the λ_{uv} are positive reals. It can be shown that the system (S) admits a unique solution.

Theroem 1 *(Tutte's Theorem)* Γ *is an embedding of* G *into the plane, with strictly convex interior faces.*

CHAPTER 4

IMPLEMENTATIONS ISSUES

4.1 Naive implementation

4.1.1 Data Structure

4.1.2 Tutte Algorithm

4.2 First Optimization

4.3 Second Optimization

4.4 third Optimization

Dans l'implémention de l'algorithme nous avons défini notre propre structure de données sur laquelle nous avons deroulé l'algorithme. Des moules nous permettent de passer d'un graphe tulip à notre structure et aussi de recuperer des informations de notre structure pour les insérer dans un graphe tulip de sorte à ce que notre structure soit uns structure temporaire de stockage d'informations sur les noeuds d'un graphe passé en paramètre de notre programme.

4.5 Aim

Il est très couteux de manipuler les graphes tulip et surtout nous n'avons pas besoin de toutes les propriétés des noeuds pour le déroulement de l'algo de tute. En plus il fallait une structure légère et adaptée à l'algorithme de tutte danas la mesure où nous sommes en quête de performance. Ci-dessous quelques raisons qui nous ont conduit à la mise en place d'une nouvelle structure de données.

1. La propriétés nous nous avons seulement bésoin de savoir si un sommet est au bord pour le considérer comme fixe pendant le déroulement de l'alo de tutte.
2.
3.
4.

4.6 Structure's contents

Dans l'implémentation de notre structure pour éviter les charges en mémoire et faciliter sa anipuler l'information la plus minimale possible pour l'exécution de l'algorithme de tutte sur un graphe. Pour ce faire nous définissons une classe qui va contenir les diifértentes dont nous avons bésoins sur un noeud (les attibuts) et toutes les opérations dont nous aurons bésoin d'effectuer sur un noeud(les méthodes).

```
class MyNode {
 private:
  node n;
  bool mobile;
  Coord coord;
  vector<MyNode *> voisin;

 public:
  MyNode();
  MyNode(const node n, const Coord coord);
  MyNode(const node n, const bool mob, const Coord coord);
  ~MyNode();

  const node getNode() const;
  bool getMobile() const;
  void setMobile(const bool b);
  const Coord getCoord() const;
  void setCoord(const Coord &);
  vector<MyNode *> * getVoisin();
  vector<MyNode *> getVoisin() const;
};
```

4.6.1 The vertex's attributs needed

4.6.2 The operations on a vertex

Distribution of nodes: Graph coloring

In order to implement a parallel asynchronous version of Tutte algorithm, it is necessary to separate graph nodes into different sets. Each node must be independant with others node in the same set. This problem is similar to the famous problem of graph colorating.

The objectif of the modified Tutte algorithm is to handle graph of thousands nodes. To separate such a number of nodes, it is more performant to use a heuristic of the algorithm of graph coloring.
The greedy algorithm is a simple and good solution to separate nodes into sets fastly and effictively.

The algorithm used in this project is :

```
G={V,E}
Y = V
color = 0
While Y is not empty
   Z = Y
   While Z is not empty
      Choose a node v from Z
      Colorate v with color
      Y = Y - v
      Z = Z - v - {neighbors of v}
   End while
   color ++
End while
```

Reference : Introduction to Algorithms (Cormen, Leiserson, Rivest, and Stein) 2001, Chapter 16 "Greedy Algorithms".

CHAPTER 5 LEVEL OF REDUCTION

5.1 Improvements of our implementation

5.2 Results on graphs

CONCLUSION

BIBLIOGRAPHY

[1] E. Colin de Verdière, M. Pocchiola, and G. Vegter. Tutte's Barycenter Method applied to Isotopies. *Computational Geometry: Theory and Applications, 26*, 81–97, 2003.

[2] B. Bollob's. Modern graph theory, *volume 184 of Graduate Texts in Mathematics.* Springer-Verlag, 1998.

[3] W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–768, 1963.

Part I

Annexes

APPENDIX A
THE APPENDIX'S NAME