

# Visualisation d'arbres de grandes tailles<sup>1</sup>

## Présentation de PSTL

Érika Baëna

erika.baena@etu.upmc.fr Diana Malabard


diana.malabard@etu.upmc.fr Antoine Genitrini (encadrant)

antoine.genitrini@lip6.fr

Université Pierre et Marie Curie

12 mai 2014

---

1. Disponible sur: [https://github.com/BErika/PSTL\\_TreeDisplay](https://github.com/BErika/PSTL_TreeDisplay) 

# Plan de la présentation

## 1 Introduction

## 2 État des lieux

- Principes à respecter pour un affichage élégant
- Algorithmes existants

## 3 Choix d'implémentation

- Fonctionnement général
- Parsing
- Algorithme de calcul des coordonnées
- Génération de la sortie

## 4 Étude de performances

- Les parsers
- Calcul des coordonnées
- Génération de la sortie
- GraphViz vs TreeDisplay

## 5 Comparaison des rendus

- Arbre de grande taille sans labels
- Arbre de petite taille avec labels

## 6 Conclusion

# Introduction

Slide sur graphviz : exemple de rendu avec GraphViz avec minimisation de largeur

On ne peut pas représenter des arbres de recherche.

# Introduction

- Objectif : Affichage élégant et efficace de tout type d'arbre
- Problèmes
  - Qu'est-ce qu'un affichage élégant ?
  - Comment optimiser le calcul de la mise en page ?

# Introduction

Entrées prises en compte :

- Mots bien parenthésés (.txt)
- JSON (.xml)
- DOT (.dot) avec l'hypothèse que le graphe représenté est un arbre

Sorties prises en compte :

- TikZ (module  $\text{\LaTeX}$ )
- Asymptote (alternative à TikZ)
- NetworkX + Matplotlib (génère une image ou un pdf)

## Plan de la présentation

- 1 Introduction
- 2 État des lieux
  - Principes à respecter pour un affichage élégant
  - Algorithmes existants
- 3 Choix d'implémentation
  - Fonctionnement général
  - Parsing
  - Algorithme de calcul des coordonnées
  - Génération de la sortie
- 4 Étude de performances
  - Les parsers
  - Calcul des coordonnées
  - Génération de la sortie
  - GraphViz vs TreeDisplay
- 5 Comparaison des rendus
  - Arbre de grande taille sans labels
  - Arbre de petite taille avec labels
- 6 Conclusion

## Principes à respecter pour un affichage élégant

*Idée : dessiner un arbre élégant et montrer ce qui le rend élégant. Entourer les noeuds au même niveau pour le principe 2, etc etc. Mais ne pas cumuler. Faire une pseudo-animation pour montrer un principe à chaque étape de l'animation, l'un après l'autre*

- ❶ Les arêtes de l'arbre ne doivent pas s'intersecter.
- ❷ Les nœuds de même profondeur doivent être dessinés sur la même ligne horizontale.
- ❸ Les arbres doivent être dessinés de la manière la plus compacte possible.
- ❹ Un nœud parent doit être centré par rapport à ses fils.
- ❺ Un sous-arbre doit être dessiné de la même façon, peu importe où il est placé dans l'arbre.
- ❻ Les nœuds fils d'un nœud père doivent être espacés de manière homogène.

# Knuth

image exemple site

Décrit une idée de "slot disponible"

Inconvénient

- Ne respecte que les principes 1 et 2



## Algorithmes de Charles Wetherell et Alfred Shannon

image exemple site

Approche Bottom Up pour centrer le père sur ses fils

Introduction d'un tableau de slots

Inconvénient

- Ne respecte pas les principes 4 et 5

# The Mods and the Rockers

image exemple site

Traitement en deux passes

Avantage

- Respecte tous les principes

## Optimisations possible

- Utilisation du contour de l'arbre pour ne pas parcourir l'arbre en profondeur
- Utilisation Threads (mini def à l'oral)

# Plan de la présentation

- 1 Introduction
- 2 État des lieux
  - Principes à respecter pour un affichage élégant
  - Algorithmes existants
- 3 **Choix d'implémentation**
  - Fonctionnement général
  - Parsing
  - Algorithme de calcul des coordonnées
  - Génération de la sortie
- 4 Étude de performances
  - Les parsers
  - Calcul des coordonnées
  - Génération de la sortie
  - GraphViz vs TreeDisplay
- 5 Comparaison des rendus
  - Arbre de grande taille sans labels
  - Arbre de petite taille avec labels
- 6 Conclusion

# Mots bien parenthésés

## Grammaire respectée

ARBRE : '(' LABEL NOEUDS ')',

NOEUDS : ARBRE NOEUDS | e

LABEL : [a-zA-Z1-9]\* | e

## Exemple

L'arbre IMAGE est représenté par  $((()((()))))$ .

# Plan de la présentation

- 1 Introduction
- 2 État des lieux
  - Principes à respecter pour un affichage élégant
  - Algorithmes existants
- 3 Choix d'implémentation
  - Fonctionnement général
  - Parsing
  - Algorithme de calcul des coordonnées
  - Génération de la sortie
- 4 Étude de performances
  - Les parsers
  - Calcul des coordonnées
  - Génération de la sortie
  - GraphViz vs TreeDisplay
- 5 Comparaison des rendus
  - Arbre de grande taille sans labels
  - Arbre de petite taille avec labels
- 6 Conclusion

# Plan de la présentation

- 1 Introduction
- 2 État des lieux
  - Principes à respecter pour un affichage élégant
  - Algorithmes existants
- 3 Choix d'implémentation
  - Fonctionnement général
  - Parsing
  - Algorithme de calcul des coordonnées
  - Génération de la sortie
- 4 Étude de performances
  - Les parsers
  - Calcul des coordonnées
  - Génération de la sortie
  - GraphViz vs TreeDisplay
- 5 Comparaison des rendus
  - Arbre de grande taille sans labels
  - Arbre de petite taille avec labels
- 6 Conclusion

## Plan de la présentation

- 1 Introduction
- 2 État des lieux
  - Principes à respecter pour un affichage élégant
  - Algorithmes existants
- 3 Choix d'implémentation
  - Fonctionnement général
  - Parsing
  - Algorithme de calcul des coordonnées
  - Génération de la sortie
- 4 Étude de performances
  - Les parsers
  - Calcul des coordonnées
  - Génération de la sortie
  - GraphViz vs TreeDisplay
- 5 Comparaison des rendus
  - Arbre de grande taille sans labels
  - Arbre de petite taille avec labels
- 6 Conclusion



## Bilan

Pour la suite