

Visualisation de Graphes Séries-Parallèles

Présentation de PSTL

Morane Gruenpeter Tanguy Retail

24 mai 2016

Université Pierre et Marie Curie

Encadrants : Matthieu Dien et Antoine Genitrini

Plan de la présentation

- 1 Introduction
- 2 Visualisation des arbres - Treedisplay
- 3 Les graphes séries-parallèles (GSP)
- 4 Algorithme de visualisation sur les GSP
 - Contraintes d'affichage
 - Déroulement de l'algorithme
- 5 Implémentation
- 6 Conclusion

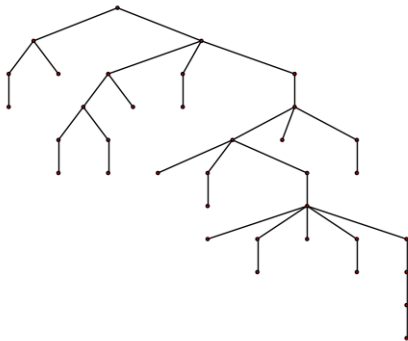
Objectif du projet

Étendre l'algorithme de visualisation d'arbre à la classe des graphes séries-parallèles.

Déroulement du projet

- étudier l'algorithme de visualisation des arbres.
- étudier les points communs entre graphes séries-parallèles et arbres.
- créer un algorithme adapté aux graphes séries-parallèles.

Visualisation des arbres - Treedisplay



Cet algorithme calcule les coordonnées des nœuds telles que :

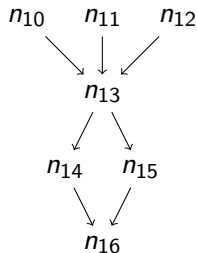
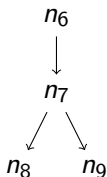
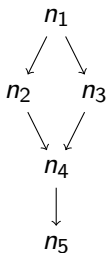
- Les arêtes de l'arbre ne s'intersectent pas.
- Les nœuds de même profondeur : même axe horizontal.
- L'ordre des nœuds est respecté.
- Un nœud parent est centré vis à vis de ses enfants.
- Les nœuds sont espacés de manière homogène.

Un algorithme linéaire en deux passes

- Une passe pour déterminer l'ordonnée d'un nœud et retenir le décalage.
- La deuxième passe pour centrer un nœud par rapport à ses enfants en appliquant le décalage.

Définition

Un graphe série-parallel (GSP) est un graphe orienté défini récursivement.

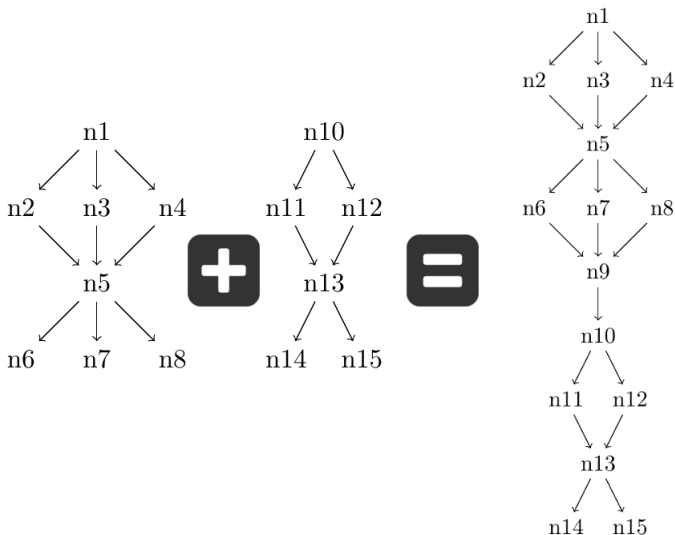


Définition - l'ensemble vide

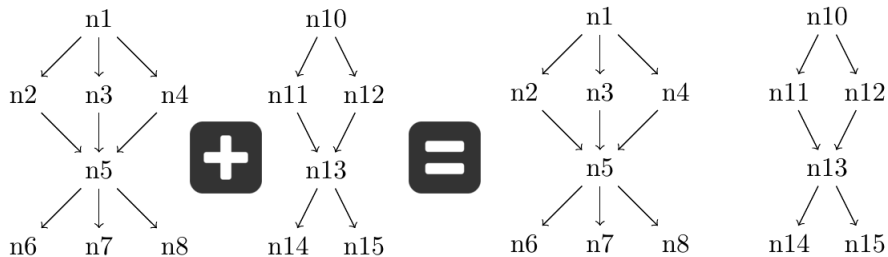
Définition - un nœud

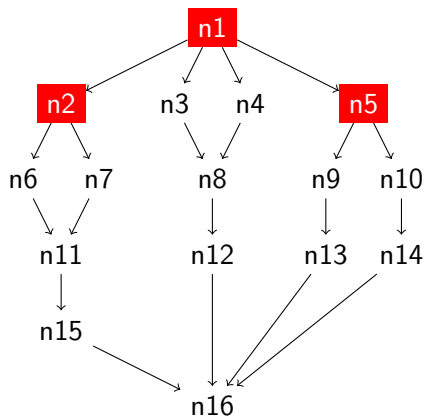
n_1

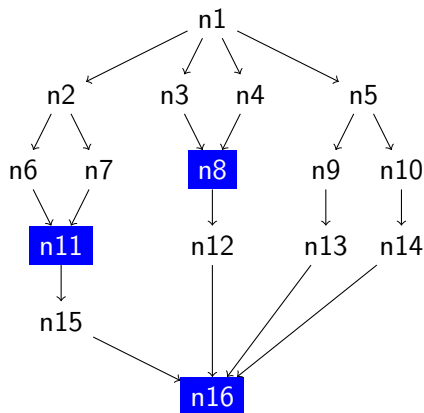
Définition - Composition en série



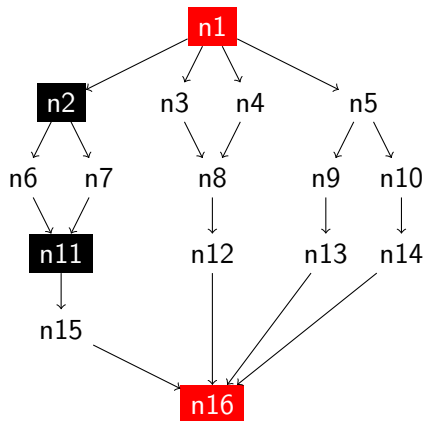
Définition - Composition en parallèle



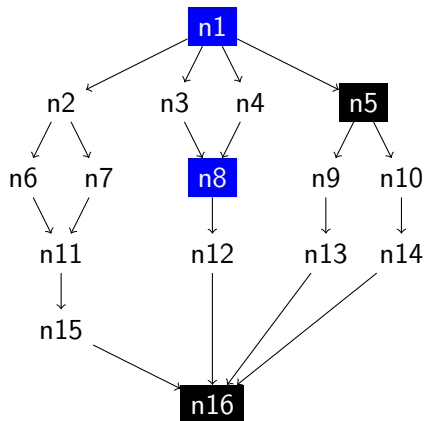




Autres définitions - diamant complet

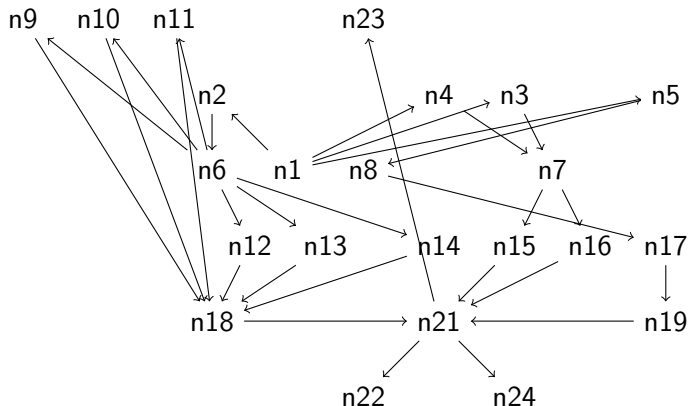


Autres définitions - diamant incomplet



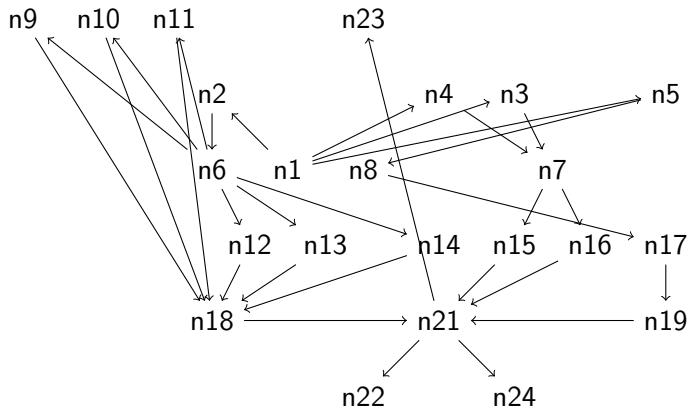
Algorithme de visualisation sur les GSP

Contraintes d'affichage



Algorithme de visualisation sur les GSP

Contraintes d'affichage

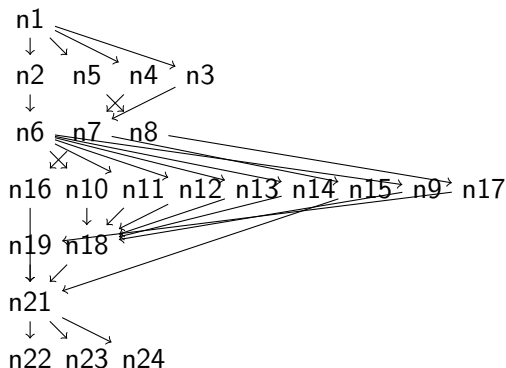


1ère contrainte

Dessiner les nœuds sur leur profondeur maximale.

Algorithme de visualisation sur les GSP

Contraintes d'affichage

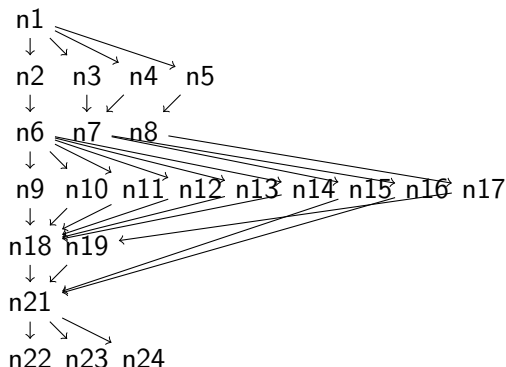


2ème contrainte

Éviter les croisements.

Algorithme de visualisation sur les GSP

Contraintes d'affichage

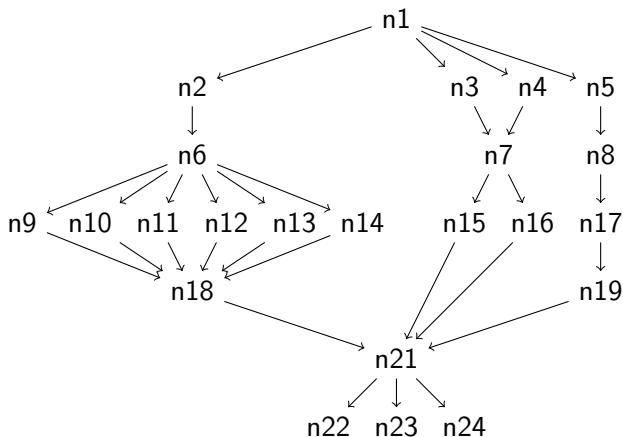


3ème contrainte

Centrer les nœuds entre leur(s) enfant(s), et leur(s) parent(s).

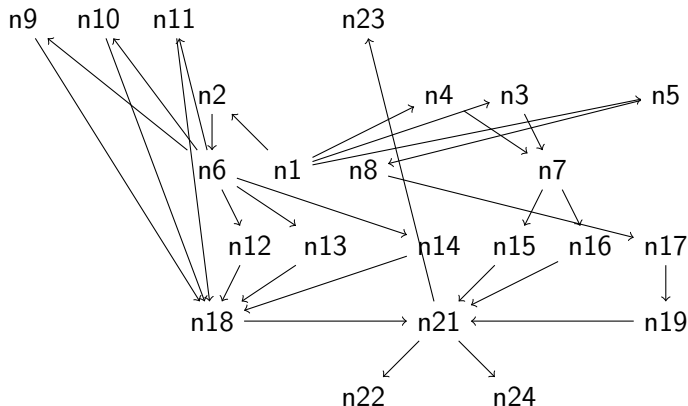
Algorithme de visualisation sur les GSP

Contraintes d'affichage



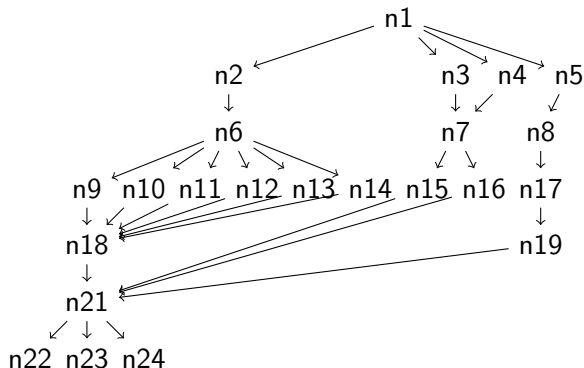
Algorithme de visualisation sur les GSP

Déroulement de l'algorithme



Algorithme de visualisation sur les GSP

Déroulement de l'algorithme

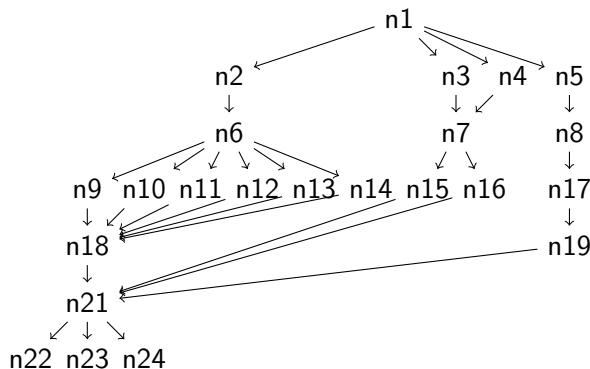


1ère & 2ème passes

Déterminer l'ordonnée des nœuds, essayer de les centrer vis-à-vis de leurs enfants, et retenir le décalage sinon.

Algorithme de visualisation sur les GSP

Déroulement de l'algorithme

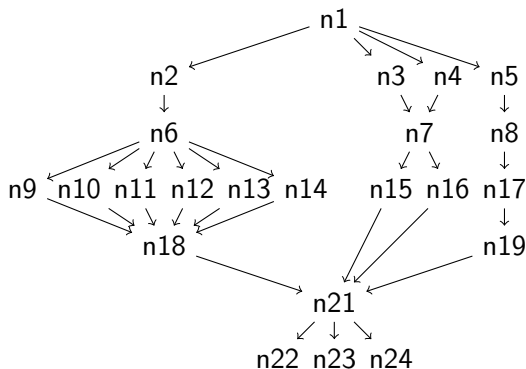


3ème passe

Appliquer le décalage et identifier les diamants.

Algorithme de visualisation sur les GSP

Déroulement de l'algorithme

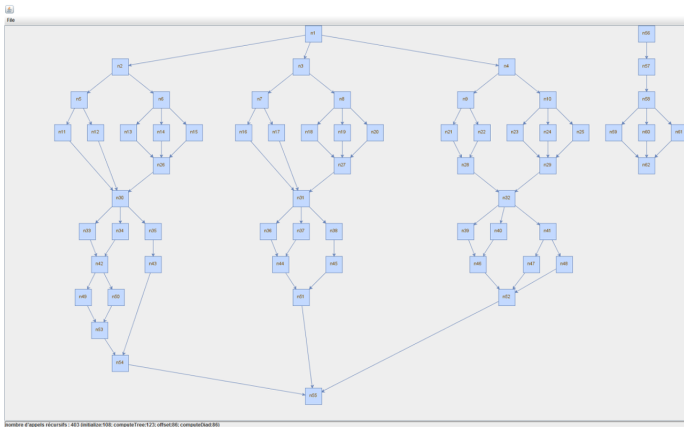


4ème passe

Aligner les diamants et leurs sous-arbres.

Implémentation en JAVA

- Un parseur DOT (Alexander Merz).
- Un générateur PNG, JPEG, BMP et TIKZ.
- Une interface graphique avec la bibliothèque SWING et MxGraph.



conclusion :

- Contraintes respectées.
- Complexité linéaire.
- Fonctionne aussi sur les arbres.

conclusion :

- Contraintes respectées.
- Complexité linéaire.
- Fonctionne aussi sur les arbres.

Pour la suite :

- Extension à une classe de graphes plus large.

Github : <https://github.com/fyrthis/PSTL>