ALGO2 – Largeur arborescente

François Schwarzentruber

May 8, 2023

1 Fixed-parameter tractable

Définition 1 (paramétrisation) Une *paramétrisation* est une fonction[1] κ qui à toute instance associe un entier.

Définition 2 (problème paramétrée) Un *problème paramétrée* a la forme suivante :
- entrée : une instance x ;
- paramétrisation : κ ;
- sortie : oui si x est une instance positive, non sinon.

Définition 3 (fpt-algorithme) Un algorithme est un fpt-algorithme par rapport à κ s'il existe une fonction calculable f et un polynôme *poly* tel que pour tout entrée x, $A(x)$ s'exécute en temps au plus $f(\kappa(x)) \times poly(|x|)$.

Définition 4 (fixed-parameter tractable) Un problème de paramétrisation κ s'il est décidé par un fpt-algorithme par rapport à κ.

2 Motivation

2.1 Sur les arbres = facile

Beaucoup de problèmes sont faciles sur des arbres avec programmation dynamique

Définition 5 **Ensemble indépendant maximum**

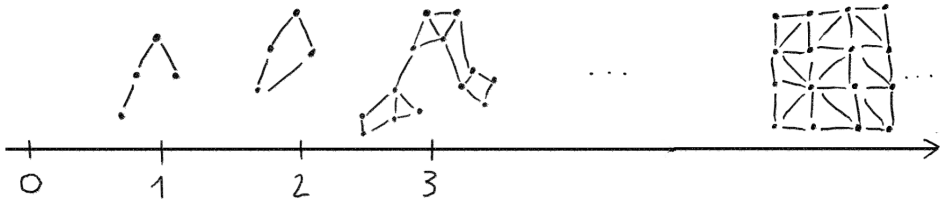
entrée : graphe $G = (V, E)$ non orienté
sortie : un ensemble $S \subseteq V$ de sommets deux-à-deux non adjacents.

Proposition 6 Ensemble indépendant maximum restreint aux arbres est dans P.

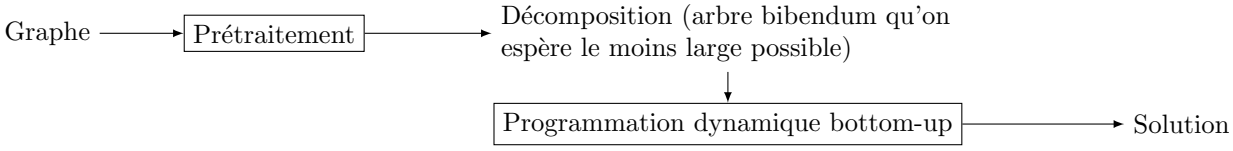
2.2 Paramétrisation qui mesure l'arbritude

En pratique, beaucoup de graphes ressemblent à des arbres.

Quid d'une paramétrisation qui mesure de combien un graphe ressemble à un arbre ?



2.3 Principe pour un algorithme efficace



[1] Dans [FG06] p. 4, on la suppose calculable en temps polynomial.

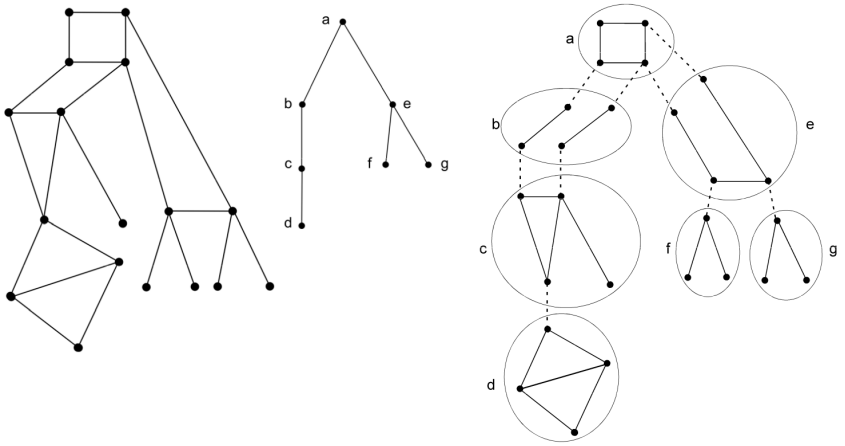
3 Définitions

Mettre un graphe dans un arbre bibendum.

Définition 7 (décomposition d'un graphe) Soit $G = (V, E)$ un graphe non orienté. Une *décomposition* A de G est un arbre où :

1. les nœuds de A sont (étiquetés par) des *sacs*, i.e. des sous-ensembles de sommets de G ;
2. pour tout sommet x dans G, l'ensemble des nœuds dont les sacs contiennent x forment un sous-arbre connexe de A ;
3. toute arête $\{x, y\}$ de G est incluse dans au moins un sac.

Exemple 8



Notation 9 (sac en un nœud) Étant donné un nœud t d'une décomposition, on note $\partial_t \subseteq V$ le sac du nœud t.

Notation 10 (ensemble de sommets d'un sous-arbre) Étant donné un nœud t d'une décomposition, l'ensemble de sommets du sous-arbre enraciné en t est $G_t = \cup_{u \text{ descendant de } t} \partial_u$.

Définition 11 (largeur) La largeur d'une décomposition est le cardinal du plus gros sac moins 1.

Définition 12 (largeur arborescente − tree-width) La tree-width de G, notée $tw(G)$, est la largeur d'une décomposition la moins large.

4 Classes de graphes

Proposition 13 Soit G un graphe connexe. G est un arbre ssi $tw(G) = 1$.

Proposition 14 La largeur arborescente d'un cycle est 2.

Proposition 15 La largeur arborescente d'une grille $n \times n$ est n.

Proposition 16 La largeur arborescente du graphe complet K_n est $n - 1$.

Proposition 17 Si $tw(G) = k$, alors tout sous-graphe H de G vérifie $tw(H) \leq k$.

Proposition 18 Si G est planaire avec n sommets, alors $tw(G) = O(\sqrt{n})$.

5 Taille d'une décomposition

Définition 19 Une décomposition d'un graphe G est petite si pour tout nœud x, y, si $x \neq y$ alors $\partial_x \not\subseteq \partial_y$.

Proposition 20 Une petite décomposition d'un graphe $G = (V, E)$ possède au plus $|V|$ nœuds.

Proposition 21 Tout graphe G admet une petite décomposition de largeur $tw(G)$. De plus, toute décomposition peut être transformée en temps linéaire en petite décomposition de même largeur.

Proposition 22 Dans tout graphe G, il existe un sommet de degré $\leq tw(G)$.

6 Algorithmes pour décomposition

Définition 23 (problème de décision du calcul de la largeur arborescente) **tree-width**
entrée : un graphe G, un entier k
sortie : oui, si la tree-width de G est k, non sinon.

Théorème 24 (admis) Le calcul de la tree-width est NP-complet.

Théorème 25 (de Bodlaender, admis) Il existe un algorithme qui prend en entrée un graphe G qui retourne une décomposition optimale de largeur $k = tw(G)$ en temps $O(|G| \times 2^{poly(k)})$.

Théorème 26 (démontré en annexe) Il existe un algorithme prenant en entrée un graphe G qui retourne une décomposition de largeur $\leq 4tw(G) + 1$ en temps $O(|G| \times 2^{O(tw(G))})$.

7 Programmation dynamique

Proposition 27 Le problème Ensemble indépendant admet un algorithme de complexité $O(|G| \times f(tw(G)))$.

Proposition 28 Le problème de 3-coloration admet un algorithme de complexité $O(|G| \times f(tw(G)))$.

8 Théorème de Courcelle

Caractérisation logique d'une classe où l'approche programmation dynamique sur une décomposition fonctionne

Définition 29 (syntaxe de MSOL) La syntaxe de la *logique monadique du second ordre* sur les graphes est définie par la grammaire suivante :

$$\varphi \ ::= \ u{=}v \ \mid \ u{\in}X \ \mid \ uRv \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \forall x\varphi \ \mid \ \forall X\varphi$$

Définition 30 (syntaxe de MSOL) La syntaxe de la *logique monadique du second ordre* sur les graphes est définie par la grammaire suivante :

$$\varphi \ ::= \ u{=}v \ \mid \ u{\in}X \ \mid \ uRv \ \mid \ inc(u,e)\neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \forall x \in V\varphi \ \mid \ \forall e \in E\varphi \ \mid \ \forall X \subseteq V\varphi \ \mid \ \forall X \subseteq E\varphi$$

Définition 31 **Problème de Courcelle**
entrée : une formule φ de MSOL, une décomposition de G de largeur k
sortie : oui, si $G \models \varphi$, non sinon.

Définition 32 **Problème d'optimisation de Courcelle**
entrée : une formule $\varphi(X)$ de MSOL, une décomposition de G de largeur k
sortie : A de cardinal max/min tel que $G, [X := A] \models \varphi$, non sinon.

Théorème 33 (de Courcelle, admis) Le problème de Courcelle admet un algorithme en temps $O(|G| \times f(|\varphi|, k))$. Le problème d'optimisation de Courcelle admet un algorithme en temps $O(|G| \times f(|\varphi|, k))$.

Notes bibliographiques

Ce cours est adapté du chapitre 11 de [FG06]. Le théorème de Courcelle reste vraie pour MSOL sur les *hypergraphes*, où les variables du premier (deuxième) ordre peuvent dénoter aussi des (ensembles d') arêtes. L'application de programmation dynamique pour ensemble indépendant dans les arbres est présenté dans [DPV08]. Il existe quelques expérimentations récentes sur la tree-width et théorie des bases de données [MSJ19].

References

[DPV08] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms*. McGraw-Hill, 2008.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[MSJ19] Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In Pablo Barceló and Marco Calautti, editors, *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, volume 127 of *LIPIcs*, pages 12:1–12:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

A Algorithme d'approximation pour décomposition

A.1 Séparation balancée

Définition 34 (W-séparateur balancé) Soit $G = (V, E)$ un graphe et $W \subseteq V$. Un W-séparateur balancé est un sous-ensemble $S \subseteq V$ tel que toute composante connexe de $G \setminus S$ contienne moins de la moitié des éléments de W.

Proposition 35 Soit $G = (V, E)$ un graphe de largeur arborescente au plus k et $W \subseteq V$. Alors il existe un W-séparateur de G de cardinal au plus $k + 1$.

A.2 Séparation

Définition 36 (séparateur) Soit $G = (V, E)$ un graphe. Soit $S, X, Y \subseteq V$. On dit que S sépare X de Y si tout chemin d'un sommet de X à un sommet de Y contient un sommet de S.

Remarque 37 Tout sur-ensemble de X sépare X de Y. Tout sur-ensemble de Y sépare X de Y.

Proposition 38 Le problème suivant admet un algorithme en temps $O(poly(k, G))$:

Séparateur petit

entrée : un graphe $G = (V, E)$, deux ensembles $X, Y \subseteq V$, un entier k
sortie : un ensemble $S \subseteq V$ de cardinal $\leq k$ qui sépare X de Y, s'il en existe un, 'non' sinon.

Corollaire 39 Le problème suivant admet un algorithme en temps $O(poly(k, G))$:

Séparateur petit'

entrée : un graphe $G = (V, E)$, deux ensembles $X, Y, Z \subseteq V$, un entier k
sortie : un ensemble $S \subseteq V \setminus (X \cup Y)$ de cardinal $\leq k$ et $Z \subseteq S$, qui sépare X de Y, s'il en existe un, 'non' sinon.

A.3 Séparation faiblement balancé

Définition 40 (W-séparateur faiblement balancé) Soit $G = (V, E)$ un graphe et $W \subseteq V$. Un W-séparateur faiblement balancé est un $S \subseteq V$ tel qu'il existe $X, Y \subseteq W$ avec :

1. $W = X \sqcup (S \cap W) \sqcup Y$;
2. S sépare X de Y
3. $0 < |X| \leq 2|W|/3$, $0 < |Y| \leq 2|W|/3$.

Proposition 41 Il y a un algorithme en temps $O(3^{3k} poly(k, G))$ pour le problème suivant :

Séparateur faiblement balancé petit

entrée : un graphe $G = (V, E)$, un ensemble $W \subseteq V$ avec $|W| = 3k + 1$, un entier k
sortie : un ensemble $S \subseteq V$ W-séparateur faiblement balancé de cardinal $\leq k + 1$, s'il existe un, 'non' sinon.

Proposition 42 Soit $G = (V, E)$ avec $tw(G) = k$. Soit $W \subseteq V$ avec $|W| \leq 3k + 1$. Alors il y a un W-séparateur faiblement balancé de cardinal au plus $k + 1$.

A.4 Algorithme calculant une décomposition

entrée : k un entier, $G = (V, E)$ un graphe, W un sous-ensemble de sommets avec $|W| \leq 3k + 1$
sortie : une décomposition arborescente de G de largeur $\leq 4k + 1$ où le sac à la racine inclut W, ou échec s'il n'y a pas de telle décomposition

fonction décomposition(k, G, $[W = \emptyset]$)
  si $|V| \leq 4k + 2$ **alors**
    renvoyer une arbre-racine où le sac contient tout V
  sinon
    $W :=$ un sur-ensemble de W de $3k + 1$ éléments
    $S :=$ un W-séparateur balancée faible (s'il en a pas échec)
    $C_1, \ldots, C_m :=$ les composantes connexes de $G \setminus S$
    pour $i := 1$ à m **faire**
      $A_i :=$ décomposition(k, $G[C_i \cup S]$, $(W \cap C_i) \cup S$)
    renvoyer

Proposition 43 L'algorithme termine et sa spécification est bien respectée.