Jolies décompositions

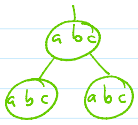
1 décembre 2021 20:32

En anglais: nice decomposition

Une décompo. $T=(V_T, E_T)$ d'un graphe G est jolie si:

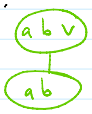
1) Chaque B_i a 0, 1, ou 2 enfants

2) Si B_i a 2 enfants B_e et B_r, on a $B_i = B_e = B_r$
On appelle B_i un noeud de jointure



3) Si B_i a 1 enfant B_j, il y a 2 cas possibles:

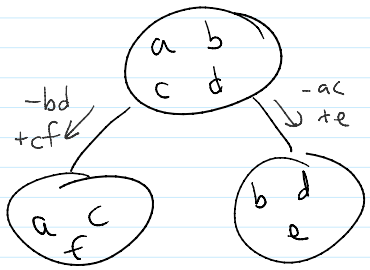
- $B_i = B_j \cup \{v\}$ pour un $v \in V(G)$
  B_i est un noeud d'introduction

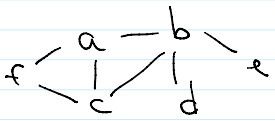


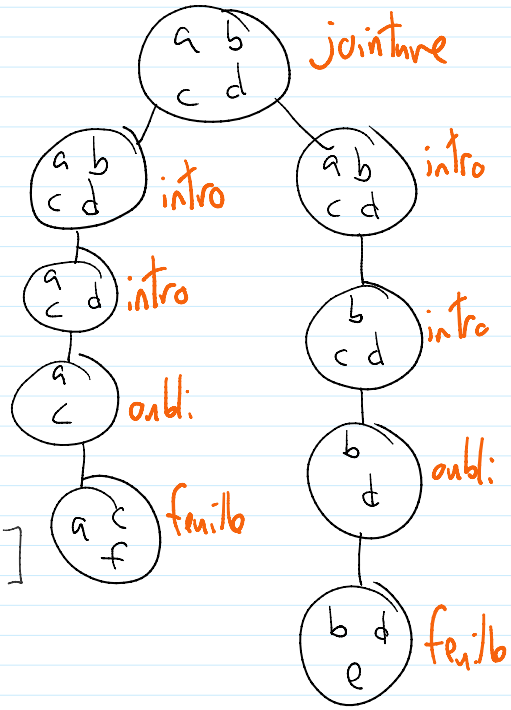
- $B_i = B_j \setminus \{v\}$ pour un $v \in V(G)$
  B_i est un noeud d'oubli

Thm: on peut transformer une décompo de G en une jolie décompo $T=(V_T, E_T)$ t.q. $|V_T| \in O(n)$ en temps polynomial.

ex:

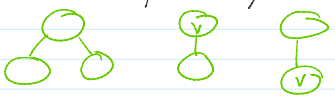






Avantage: récurrence + simple, calcul ++ rapide

Inconvénient: 3 cas à gérer [jointure, intro, oubli]



MAX-INDSET

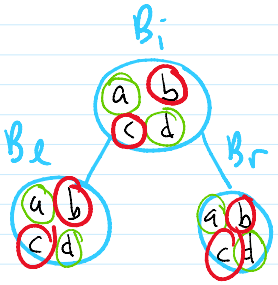
$M[B_i, c_i]$ = taille max d'un ens indép. I de $G(B_i)$
sachant que $I \cap B_i = c_i^{vert}$

si B_i = feuille $\Rightarrow$ comme avant

$$M[B_i, c_i] = \begin{cases} -\infty & \text{si } c_i^{vert} \text{ pas indep.} \\ |c_i^{vert}| & \text{sinon} \end{cases}$$

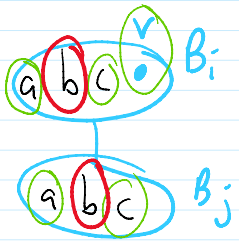
$M[B_i, c_i] = \begin{cases} \text{si } c_i \text{ pas inadp.} \\ |c_i^{vert}| \text{ sinon} \end{cases}$

1) si B_i est un noeud de jointure
Soient B_ℓ, B_r ses enfants
Puisque $B_i = B_\ell = B_r$, on doit garder le même coloriage c_i



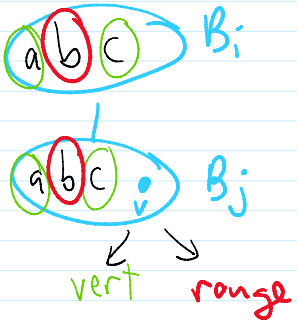
$$M[B_i, c_i] = M[B_\ell, c_i] + M[B_r, c_i] - |c_i^{vert}| \qquad O(1)$$

2) si B_i est un noeud d'introduction
Soit B_j son enfant. Puisque $B_j \subseteq B_i$, il n'y a qu'un seul coloriage c_j^* compatible avec c_i.



$$M[B_i, c_i] = M[B_j, c_j^*] + \begin{cases} 1 & \text{si } v \text{ est vert} \\ 0 & \text{sinon} \end{cases} \qquad O(1)$$

3) si B_i est un noeud d'oubli
Soit B_j son enfant. Il y a 2 colorianges possibles : c_i avec v = vert
c_i avec v = rouge
Prendre le meilleur.



$$M[B_i, c_i] = \max\left(M[B_j, c_i + v = vert], M[B_i, c_i + v = rouge]\right) \qquad O(1)$$

À la fin, on retourne $\max_{c_r}(M[B_r, c_r])$ où B_r = racine

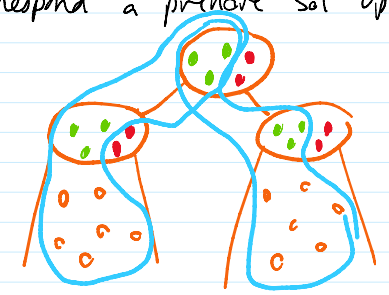
Complexité : # d'entrées $M[B_i, c_i]$ à calculer $\in O(n 2^{tw(G)})$
Chaque entrée en $O(1)$
$\Rightarrow$ Temps $O(n 2^{tw(G)})$.

Pourquoi ça marche ?

1) Jointure

$M[B_i, c_i] = M[B_\ell, c_i] + M[B_r, c_i] - |C_i^{vert}|$

Correspond à prendre sol opt de chaque côté, puis l'union



L'union peut peut pas créer d'arête uv accidentelle. Si c'était le cas, on aurait

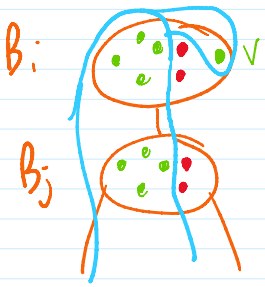
$u \in G(B_\ell) \setminus B_\ell$

$v \in G(B_r) \setminus B_r$

Mais la prop ② serait impossible à satisfaire car on ne peut pas avoir de sac avec u et v sans contredire ③.

2) Intro

$$M[B_i, c_i] = M[B_j, c_j^*] + \begin{cases} 1 & \text{si } v \text{ est vert} \\ 0 & \text{sinon} \end{cases}$$

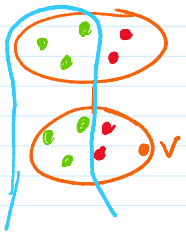


Fonctionne car on reprend la même sol que sur $G(B_j)$.

L'ajout de v ne peut pas créer d'arête accidentelle car c'est la première apparition de v et il n'a pas de voisin sans B_j.

3) Oubli

$$M[B_i, c_i] = \max(M[B_j, c_{j1}], M[B_j, c_{j2}])$$



Fonctionne car on fait juste reprendre une même solution qu'en $G(B_j)$ sans la modifier.

MAX-CUT

Entrée : graphe $G=(V,E)$
Param : $tw(G)$
Sortie : bipartition (V_1, V_2) de G qui maximise $|E(V_1, V_2)|$

Coloriage : $c_i : B_i \to \{1, 2\}$

$c_i(v) = 1 \Rightarrow$ mettre v dans V_1
$c_i(v) = 2 \Rightarrow$ mettre v dans V_2

$C_i^1 = \{v \in B_i : c_i(v) = 1\}$ $\quad$ $C_i^2 = \{v \in B_i : c_i(v) = 2\}$

$M[B_i, c_i]$ = # max d'arêtes traversantes d'une bipartition (V_1, V_2) de $G(B_i)$ sachant que
$\forall v \in C_i^1, v \in V_1$
$\forall v \in C_i^2, v \in V_2$