Algorithmique Distribuée : TD 5, arbre couvrant en largeur d'abord

Alain Cournier Stéphane Devismes

1 Le problème

Nous nous intéressons au calcul d'un arbre couvrant en largeur enraciné en un processus R (R est appelée *racine*) d'un réseau $G = (V, E)$. Un *graphe partiel* de G est un graphe avec le même ensemble de nœuds V et dont l'ensemble d'arête E' est un sous-ensemble de E ($E' \subseteq E$). Un *arbre couvrant* est un graphe partiel de G acyclique et connexe. Un *arbre couvrant en largeur d'abord* de G, $T = (V, E_T)$, enraciné en R est un arbre couvrant de G tel que, pour tout processus p, l'unique chemin élémentaire de p à R dans T est de longueur $\|p, R\|$, où $\|p, R\|$ est la distance de p à R dans G (*i.e.*, la longueur du plus court chemin de p à R dans G). La hauteur d'un arbre enraciné est la longueur maximale d'un chemin élémentaire reliant la racine à une feuille (une feuille est un nœud de degré 1).

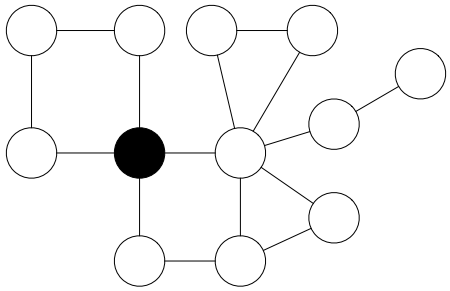


FIGURE 1 – Réseau quelconque.

Question 1. Proposez un arbre couvrant en largeur d'abord du graphe donné en figure 1. La racine étant le nœud noir. Quelle est la hauteur de votre arbre ?

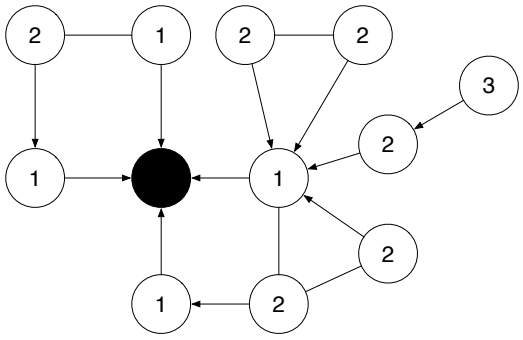


FIGURE 2 – Arbre couvrant du graphe proposé en figure 1. (Hauteur 3)

Question 2. Combien y a-t-il de d'arbres couvrants en largeur d'abord possible avec comme racine le nœud noir ?

4

Nous rappelons que le *diamètre* d'un graphe est la distance maximale entre deux nœuds, c'est-à-dire, la longueur du plus long des plus court chemins.

Question 3. Quel est le diamètre du graphe proposé en figure 1 ? Quelle est la relation entre le diamètre d'un graphe connexe et la hauteur d'un arbre couvrant en largeur d'abord de ce graphe.

Diamètre : 5. Hauteur $\leq$ Diamètre

2 L'algorithme

Nous allons écrire un algorithme (autostabilisant) *silencieux* pour le calcul d'un arbre couvrant en largeur d'abord d'un réseau bidirectionnel connexe. Cet arbre sera enraciné au processus R. Nous considérons le modèle à états sous l'hypothèse d'un démon distribué inéquitable. Nous rappelons que pour tout processus p, l'ensemble des voisins de p est noté $\mathcal{N}_p$. Nous supposons également que les processus connaissent une borne supérieure D sur le diamètre du réseau. Ensuite, nous utilisons les variables suivantes :

— Chaque processus p détient une variable d_p, dont le domaine est $\{0, \ldots, D\}$.
Dans une configuration terminale, $d_p = \|p, R\|$.

— De plus, si p n'est pas la racine, alors p a en plus un pointeur $père_p$, dont le domaine est $\mathcal{N}_p$.
Dans une configuration terminale, $père_p$ désigne le « père » de p dans l'arbre, c'est-à-dire un voisin q tel que $\|q, R\| = \|p, R\| - 1$

Ainsi, le programme de la racine R contient uniquement la règle CD suivante :

$$CD :: d_R \neq 0 \mapsto d_R \leftarrow 0$$

En revanche, tout processus nonracine p dispose de deux règles : la première, CD, pour calculer d_p, la seconde, CP, pour calculer $père_p$. La seconde règle est activable seulement si la première ne l'est pas. La première règle consiste à affecter d_p à la valeur minimum entre les variables d de ses voisins plus 1 et D, si d_p ne vaut pas déjà cette valeur.

Question 4. Écrivez les deux règles CD et CP pour un processus nonracine p.

$$CD :: d_p \neq \min(\{d_q, q \in \mathcal{N}_p\} \cup \{D-1\}) + 1 \mapsto d_p \leftarrow \min(\{d_q, q \in \mathcal{N}_p\} \cup \{D-1\}) + 1$$

$$CP :: \min\{d_q, q \in \mathcal{N}_p\} = d_p - 1 \wedge d_{père_p} \neq d_p - 1 \mapsto père_p \leftarrow q \in \mathcal{N}_p, d_q = d_p - 1$$

Question 5. CD et CP sont-elles mutuellement exclusive pour un processus nonracine donné ? Justifiez.

Les deux règles CD et CP sont mutuellement exclusives. Effet, supposons le contraire, on a alors, en particulier, $d_p \neq \min(\{d_q, q \in \mathcal{N}_p\} \cup \{D-1\}) + 1$ et $\min\{d_q, q \in \mathcal{N}_p\} = d_p - 1$. Donc, $d_p \neq \min(\{d_p - 1\} \cup \{D-1\}) + 1$. Or, par définition, $d_p \leq D$. Donc, $d_p \neq d_p$, contradiction.

3 Preuve de l'algorithme

L'algorithme étant silencieux, la preuve de correction consiste d'abord à montrer que les variables locales définissent un arbre couvrant en largeur dans toute configuration terminale (correction partielle). Puis, nous montrons que toute exécution atteint une configuration terminale en un nombre fini de pas de calcul (terminaison).

3.1 Correction partielle

Soit γ_t une configuration terminale.

Question 6. Démontrez, par contradiction, le lemme suivant.

Lemme 1. *Pour tout processus p, on a* $d_p \geq \|p, R\|$ *dans* γ_t.

Preuve. Supposons, par contradiction, qu'il existe un processus p tel que $d_p < \|p, R\|$ dans γ_t. Soit $p_{min} \in \{p, d_p < \|p, R\| \text{ dans } \gamma_t \wedge (\forall q, d_q < \|q, R\| \text{ dans } \gamma_t, d_q \geq d_p)\}$. Tout d'abord, puisque, par définition, $R.d \geq 0 = \|r, r\|$, nous avons $p_{\min} \neq r$. Ensuite, par définition, tout voisin q de p_{min} vérifie alors l'une de ces conditions :

1. $d_q \geq d_{p_{min}}$ ou
2. $d_q \geq \|q, R\|$.

Dans le premier cas, on a clairement $d_{p_{min}} < d_q + 1$. Dans le deuxième cas, $d_{p_{min}} < \|p_{min}, R\| \leq \|q, R\| + 1 \leq d_q + 1$. Enfin, $d_{p_{min}} < \|p_{min}, R\| \leq \mathcal{D} \leq D$, *i.e*, $d_{p_{min}} < (D-1) + 1$. Ainsi, $d_{p_{min}} < \min(\{d_q, q \in \mathcal{N}_{p_{min}}\} \cup \{D-1\}) + 1$ et nous pouvons conclure que la règle CD est activable à p_{min}, contradiction. □

Question 7. Démontrez, par récurrence sur la distance des processus à la racine R, le lemme suivant.

Lemme 2. *Pour tout processus p, on a* $d_p = \|p, R\|$ *dans* γ_t.

Preuve. By induction on the distances.

Base Case : By definition, the root is the unique process at distance 0 from itself. Then, since γ_t is terminal, Action CD is disabled at R and so $d_R = 0 = \|R, R\|$ in γ_t.

Induction Hypothesis : Let $k \geq 0$. Assume that for every process p such that $\|p, R\| = k$, we have $d_p = \|p, R\|$ in γ_t.

Induction Step : Let q be any process such that $\|q, R\| = k+1$. Since $k+1 > 0$, $q \neq R$ and there exists a neighbor q' of q such that $\|q', R\| = \|q, R\| - 1 = k$. By induction hypothesis, $d_{q'} = k$ in γ_t. Then, for every neighbor q'' of q, we have $\|q'', r\| \in \{k, k+1, k+2\}$ and, by Lemma 1, we have $d_{q''} \geq k$ in γ_t. Finally, since $\mathcal{D} \geq \|q, r\| = k+1$, we have $k \leq \mathcal{D} - 1 \leq D - 1$. Thus, since γ_t is terminal, $d_q = \min(\{q''.d : q'' \in \mathcal{N}_q\} \cup \{D-1\}) + 1 = k+1 = \|q, r\|$ in γ_t.

□

Corollaire 1. *Pour tout processus* $p \neq R$, *on a* $\min\{d_q, q \in \mathcal{N}_p\} = d_p - 1$ *dans* γ_t.

Preuve. Soit p un processus différent de R. Par définition, il existe un voisin q de p tel que $\|q, R\| = \|p, R\| - 1$. De plus, dans γ_t, $d_p = \|p, R\|$ et pour tout voisin q' de p, on a $d_{q'} = \|q', R\| \in \{\|p, R\| - 1, \|p, R\|, \|p, R\| + 1\}$ avec en particulier $d_q = \|p, R\| - 1$, d'après le lemme 2. Donc, $d_p - 1 = \|p, R\| - 1 = \min\{d_q, q \in \mathcal{N}_p\}$ dans γ_t. □

D'après le corollaire 1 et le fait que la garde de la règle CP qui est inactivable pour tout processus nonracine dans γ_t, on a :

Corollaire 2. *Pour tout processus* $p \neq R$, *on a* $d_{père_p} = d_p - 1$ *dans* γ_t.

Soit $T = (V, E_T)$, où $E_T = \{\{p, q\} \in E, q \neq R \wedge père_q = p \text{ dans } \gamma_t\}$.

Lemme 3 (Fermeture et correction)**.** T *est un arbre couvrant en largeur d'abord.*

Preuve. Nous démontrons tout d'abord que T un arbre couvrant en utilisant l'équivalence : « Un graphe de n nœuds est un arbre si et seulement s'il contient $n-1$ arêtes et est acyclique ».

T est sans cycle : Supposons, par contradiction, que T contient un cycle $p_0, \ldots, p_k, p_0$. Tout d'abord, par définition, R ne fait pas partie de ce cycle. Ensuite, supposons, sans perte de généralité, que pour tout $i > 0$, $père_{p_i} = p_{i-1}$ et $père_{p_0} = p_k$. D'après le corollaire 2, on a $d_{p_{i-1}} < d_{p_i}$ et par transitivité, $d_{p_0} < d_{p_k}$. Or, puisque $père_{p_0} = p_k$, on a $d_{p_k} < d_{p_0}$ d'après le corollaire 2, contradiction.

E_T contient $n-1$ arêtes : Suivant le même raisonnement que précédement, chaque arête de E_T est pointée par le pointeur $père$ d'un unique processus nonracine (deux processus adjacents ne peuvent désigner la même arête) : Supposons le contraire, soit deux voisins p et q tels que (1) $père_p = q$ et (2) $père_q = p$ D'après le corollaire 2, (1) implique $d_q < d_p$ et (2) implique $d_p < d_q$, contradiction.

Ainsi, puisqu'il y a $n-1$ processus nonracines, on a $|E_T| = n - 1$.

Nous montrons maintenant que T est en largeur d'abord. Soit $p_0 = R, \ldots, p_k$ l'unique chemin de R à p_k dans l'arbre. Par définition, pour tout $i > 0$, $père_{p_i} = p_{i-1}$. D'après le corollaire 2, on a $d_{p_{i-1}} = d_{p_i} - 1$ et par transitivité, on a $d_{p_0} = d_{p_k} - k$. De plus, $d_{p_0} = d_R = 0$. Donc, $d_{p_k} - k = 0$: d_{p_k} est égale à la longueur k du chemin entre R et p_k. Or, d'après le lemme 2, d_{p_k} est aussi égale à $\|R, p_k\|$. D'où la longueur k de l'unique chemin de p_k vers R dans T est égale à la distance de R vers p_k dans G. □

3.2 Terminaison

Soit $D_i = \{p \in V, d_p \leq i\}$, pour tout $i \in [0..D]$.

Question 8. Combien de fois D_0 peut être modifié ?

n fois. R peut entrer s'il n'y est pas initialement mais ne peut pas en sortir. Tout processus nonracine peut sortir s'il y est initialement mais ne peut pas y rentrer.

Question 9. Supposons que le contenu de D_k ne peut être modifié qu'un nombre fini de fois. Qu'en est-il alors de l'ensemble D_{k+1}, s'il existe ? (Justifiez votre réponse par une preuve)

D_{k+1} ne peut être modifié qu'un nombre fini de fois Supposons que D_{k+1} existe. Par hypothèse, il existe une configuration γ dans l'exécution à partir de laquelle, le contenu de D_k n'est plus jamais modifié. Si $k + 1 = D$, alors plus rien ne se passe, car si D_{k+1} est modifié alors un D_k est modifié aussi. Si $k + 1 < D$, alors à partir de γ tout processus peut entrer (resp. sortir) au plus une fois dans D_{k+1} (en exécutant CD). Par suite, le contenu de D_{k+1} ne peut plus être modifié qu'un nombre fini de fois.

Question 10. D'après la réponse à la question précédente, que concluez-vous au sujet de la règle CD ?

Le nombre d'exécution de la règle CD est fini.

Question 11. Si le nombre d'exécution de la règle CD est finie dans toute l'exécution, que peut-on conclure sur le nombre d'exécution de la règle CP dans l'exécution ? (Justifiez)

Chaque processus peut exécuter la règle CP parce ce qu'elle est activable initialement ou après qu'un ensemble D_i ait été modifié. Par suite, le nombre d'exécution de la règle CP est aussi fini.

D'après les réponses aux questions 7 à 10, nous pouvons conclure :

Lemme 4. *À partir d'une configuration quelconque, le système atteint une configuration terminale en un nombre fini de pas de calcul.*

À partir d'une configuration quelconque, le système atteint une configuration terminale en un nombre fini de pas de calcul d'après le lemme 4, cette dernière étant légitime d'après le lemme 3. Ainsi, nous pouvons conclure :

Théorème 1. *L'algorithme constitué des règles CD et CP est autostabilisant et silencieux pour le calcul d'un arbre couvrant en largeur sous l'hypothèse d'un démon distribué inéquitable.*

4 Temps de stabilisation en rondes de l'algorithme

Question 12. Considérons le réseau de $\mathcal{D} + 2$ processus $p_0, \ldots p_{\mathcal{D}+1}$ donné dans la figure 3. Initialement :
- La variable d de tous les processus nonracines p_i ($i \in [0..\mathcal{D} + 1]$) est égale à D.
- La variable $père_p$ de chaque processus nonracine p_i ($i \in [1..\mathcal{D} + 1]$) pointe sur p_{i-1}.

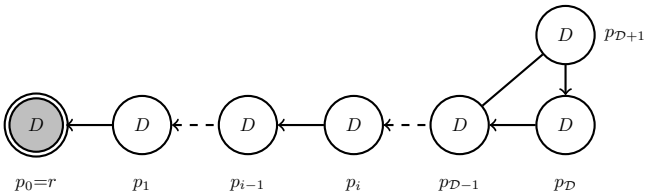
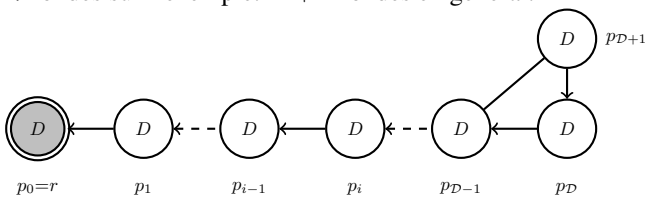


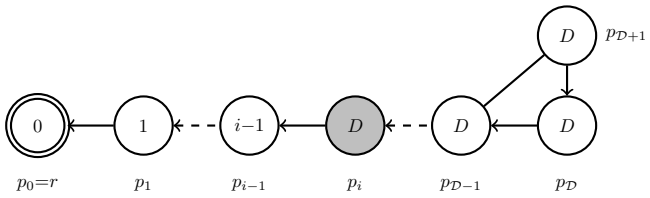
FIGURE 3 – Configuration initiale.

En supposant que $D > \mathcal{D}$, donnez la trace de l'exécution synchrone de l'algorithme à partir de cette configuration initiale pour $n = 7$ processus. Quel est le temps de stabilisation en rondes sur cet exemple ? Généralisez à tout réseau de cette forme.

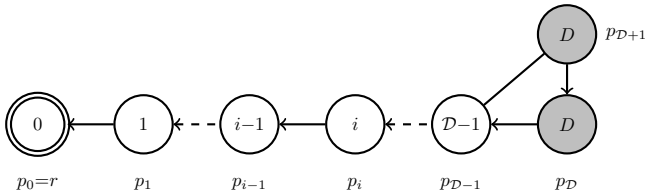
7 rondes sur l'exemple. $\mathcal{D}+2$ rondes en général.



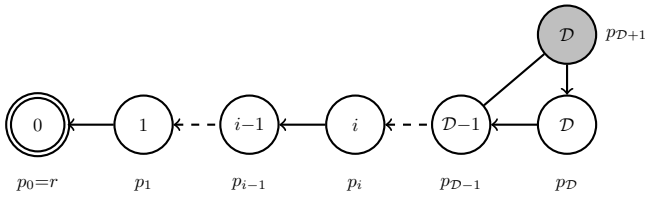
Initial Configuration



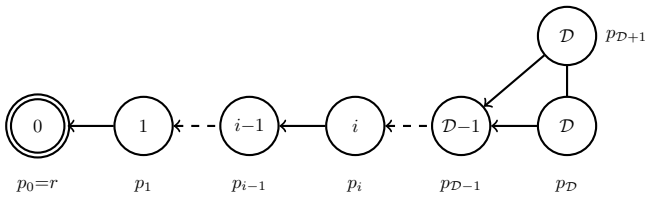
After Round i ($i > 0$).



After Round $\mathcal{D}$.



After Round $\mathcal{D}+1$.



After Round $\mathcal{D}+2$ (terminal configuration).

Question 13. Reprenez l'exemple précédent avec $D = \mathcal{D}$. Que concluez-vous ?

La complexité devient $\mathcal{D}+1$ rondes.

Pour les questions suivantes, nous rappelons que nous considérons un démon dustribué inéquitable.

Question 14. Après une ronde, que peut-on dire de l'état de la racine ? de l'état des processus nonracine ?

Après une ronde, $D_R = 0$ pour toujours. Pour tout processus $p \neq R$, on a $D_p > 0$ pour toujours.

Question 15. Que peut-on déduire de l'état des processus après $k > 0$ rondes ? après $\mathcal{D} + 1$ rondes ?

Après au plus $k > 0$ rondes, pour tout processus p on a pour toujours : si $\|p, R\| < k$, alors $d_p = \|p, R\|$, sinon $d_p > k$.

D'où, si on applique le résultat avec $k = \mathcal{D} + 1$, où $\mathcal{D}$ est le diamètre de G, on obtient pour tout processus p, d_p est constant pour toujours.

Question 16. Quel est le temps de stabilisation en rondes de l'algorithme (justifiez).

$\mathcal{D} + 2$ rondes

Au début de la $\mathcal{D} + 2^{\text{ème}}$ ronde, la valeur du pointeur $père_p$ de tout processus nonracine p dépend uniquement des variables d de p et ses voisins dont les valeurs sont fixées pour toujours, d'après la réponse de la question 12. Ainsi, la règle CP, qui attribue la valeur de $père_p$, est inactivable, alors elle l'est pour toujours. Sinon, elle est continûment activable. Dans ce dernier cas, elle est exécutée au cours de la ronde $\mathcal{D} + 2$. De plus, par définition il existe un voisin q de p tel que $\|q, R\| = \|p, R\| - 1$. D'après la réponse à la question 12 $d_q = d_p - 1$. Ainsi, l'exécution par p de la règle CP rend sa garde fausse pour toujours. D'où :

Lemme 5. *Après au plus $\mathcal{D} + 2$ rondes, la règle CP n'est plus jamais activable.*

Corollaire 3. *Le temps de stabilisation de l'algorithme est au plus de $\mathcal{D} + 2$ rondes, où $\mathcal{D}$ est le diamètre du réseau.*