Arbre et diffusion

Master 2 Informatique - UFR S.A.T

Pr. Ousmane THIARE

ousmane.thiare@ugb.edu.sn
http://www.ousmanethiare.com/

16 avril 2020

Chapitre 3 : Arbre et diffusion

Dans ce chapitre nous donnons des algorithmes élémentaires pour la diffusion, la concentration, et la construction d'arbre de diffusion.

Il s'agit de transmettre un message M à tous les sommets du graphe depuis un sommet distingué que l'on notera r_0. On peut vérifier que :

Proposition

Tout algorithme distribué de diffusion A sur un graphe G à n sommets vérifie (en mode synchrone ou asynchrone) :

- $Message(A, G) \geq n - 1$.

Il s'agit de transmettre un message M à tous les sommets du graphe depuis un sommet distingué que l'on notera r_0. On peut vérifier que :

Proposition

Tout algorithme distribué de diffusion A sur un graphe G à n sommets vérifie (en mode synchrone ou asynchrone) :

- $Message(A, G) \geq n - 1$.
- $Temps(A, G) \geq exc(r_0)$.

On rappelle que $exc(r_0)$, l'excentricité de r_0, est la profondeur minimum d'un arbre couvrant G et de racine r_0. C'est donc la profondeur d'un arbre couvrant en largeur d'abord qui a pour racine r_0. Notons en passant que le diamètre de G, noté diam(G), est la valeur maximum de l'excentricité.

Preuve Proposition : Il faut informer n-1 sommets distincts du graphe. L'envoi d'un message sur une arête ne peut informer qu'un seul sommet (celui situé à l'extrémité de l'arête). Donc il faut que le message circule sur au moins n-1 arêtes différentes de G pour résoudre le problème. Donc $Message(A, G) \geq n - 1$.

Dans G il existe un sommet u à distance au $t = exc(r_0)$ de r_0. Dans le scénario synchrone, chaque message traversant une arête prend un temps 1. Il faut donc un temps au moins t pour que u soit informé. Donc $Temps(A, G) \geq t$

Une stratégie courante pour réaliser une diffusion est d'utiliser un arbre couvrant enraciné en r_0, disons T. L'intérêt est qu'on ne diffuse M que sur les arêtes de T. Donc, ici on va supposer que chaque sommet u de G connaît T par la donnée de son père Père(u) et de l'ensemble de ses fils Fils(u). L'algorithme ci-dessous est spécifique pour chaque arbre T.

Algorithme $Cast_T(r_0)$ (code du sommet u)

1. Si $u \neq r_0$, attendre de recevoir un message (M, v)
2. Pour tout $v \in Fils(u)$, Send(M, v)

Proposition

En mode synchrone ou asynchrone :

- $Message(Cast_T(r_0), G) = |E(T)| = n - 1.$

Notons que la hauteur de T vaut $exc(r_0)$ si T est un arbre en largeur d'abord. Dans ce cas, l'algorithme $Cast_T$ est optimal en temps et en nombre de message.

Proposition

En mode synchrone ou asynchrone :

- $Message(Cast_T(r_0), G) = |E(T)| = n - 1.$
- $Temps(Cast_T(r_0), G) = hauteur(T).$

Notons que la hauteur de T vaut $exc(r_0)$ si T est un arbre en largeur d'abord. Dans ce cas, l'algorithme $Cast_T$ est optimal en temps et en nombre de message.

On utilise cette technique (flooding en Anglais) en l'absence de structure pré-calculée comme un arbre de diffusion. Un sommet u ne connaît pas Père(u) et Fils(u).

Algorithme $Flood(r_0)$ (code du sommet u)

1. Si $u = r_0$, alors Send(M, v) pour tout voisin v de u.
2. Sinon,
   (a) Attendre de recevoir (M, v)
   (b) Send(M, w) pour tout voisin $w \neq v$ de u.

Proposition

En mode synchrone ou asynchrone.

- $Message(Flood(r_0), G) = 2|E(G)| - (n-1)$.

Proposition

En mode synchrone ou asynchrone.

- $Message(Flood(r_0), G) = 2|E(G)| - (n-1)$.
- $Temps(Flood(r_0), G) = exc(r_0)$.

Considérons maintenant le problème de construire de manière distribué un arbre couvrant de racine r_0. Des variantes de ce problème seront examinées plus amplement au chapitre 4. Plus précisément on souhaite qu'à la fin de l'exécution de l'algorithme chaque sommet u de G fixe une variable Père(u) correspondant à son père, et un ensemble Fils(u) correspondant à l'ensemble de ses fils.

Proposition

Le problème de la diffusion à partir d'un sommet r_0 a une complexité équivalente en temps et en nombre de messages au problème de la construction d'un arbre couvrant enraciné en r_0.

Preuve Proposition

Si un algorithme peut construire un arbre T de racine r_0, arbre donné en chaque sommet par son père et ses fils, alors il suffit d'appliquer l'algorithme $Cast_T(r_0)$ ce qui donne un algorithme de diffusion.

Inversement, si l'on connaît un algorithme A de diffusion à partir de r_0, il suffit, pendant l'exécution de A de définir le père de u comme le voisin v d'où u a reçu M en premier. La flèche du temps interdit la création d'un cycle.

Si l'on souhaite récupérer les fils de u, il faut émettre vers son père un message particulier et collecter ces messages. Cela ajoute n-1 messages supplémentaires, et une unité à la complexité en temps. Cela ne modifie pas les complexités en temps et en nombre de messages (n-1 est une borne inférieure pour la diffusion (Propo 1)).

En combinant les deux dernières propositions, on obtient directement :

Corollaire

Il existe un algorithme distribué noté SpanTree, basé sur Flood, permettant de construire un arbre couvrant de racine r_0 sur tout graphe G et sommet r_0 (en synchrone ou asynchrone) avec $Temps(SpanTree, G) = O(D)$ et $Message(SpanTree, G) = O(m)$.

Il est a noté que les arbres ainsi construits diffèrent beaucoup s'ils s'exécutent en mode synchrone ou asynchrone. Par, exemple, dans le cas où G est une clique (graphe complet), l'arbre généré par SpanTree en synchrone est toujours une étoile, alors qu'il peut s'agir de n'importe quel arbre couvrant en mode asynchrone.

Un problème potentiel des algorithmes précédant est qu'un sommet ne sait pas si la diffusion est terminée. C'est en fait un problème général en algorithmique distribuée. Cela peut être très gênant en asynchrone lorsque plusieurs algorithmes doivent être enchaînés. Le risque est que des messages des deux algorithmes se mélangent et aboutissent à des erreurs.
On considère donc la variante de la diffusion où en plus de réaliser une diffusion à proprement parlée, on souhaite que chaque sommet u possède une variable booléenne Fin(u), qui lorsqu'elle passe à VRAI indique à u que tous les sommets de G ont bien reçus le message M.

La solution consiste à modifier l'algorithme Flood en envoyant un message (« done ») à son père lorsque qu'on est sûr que tous les sommets de son propre sous-arbre ont été informés. Puis, le sommet r_0 diffuse un message de fin vers tous les sommets de G. Pour cela on se sert de l'arbre que l'on construit en même temps que Flood (message « père »).

Algorithme $Flood\&Echo(r_0)$ (code du sommet u)
1. $Fils(u) := \emptyset$ et $Fin(u) := FAUX$.
2. Si $u = r_0$, alors Send(« data »+M, v) pour tout voisin v de u, poser $c := deg(u)$.
3. Sinon,
(a) Attendre de recevoir un message (M, v).
(b) Send(« père », v), Send(« data »+M, w) pour tout voisin $w \neq v$ de u.
(c) Poser Père(u) :=v et c :=deg(u)-1.

4. Tant que $c > 0$:
   (a) Attendre de recevoir un message (M', v).
   (b) Si M' de type « data », Send(« ack »,v).
   (c) Si M' de type « père », $Fils(u) := Fils(u) \cup \{v\}$.
   (d) Si M' de type«ack» ou «done», poser c :=c-1.
5. Si $u \neq r_0$, Send(« done », Père(u)) et attendre un message « fin ».
6. Fin(u) :=VRAI, Send(« fin », v) pour tout $v \in Fils(u)$.

Dans la suite on notera D=diam(G) le diamètre de G, et m=|E(G)| le nombre d'arêtes de G.

Proposition

En mode synchrone :

- $Message(Flood\&Echo(r_0), G) = O(m)$.

En mode asynchrone :

Proposition

En mode synchrone :

- $Message(Flood\&Echo(r_0), G) = O(m)$.
- $Temps(Flood\&Echo(r_0), G) = O(exc(r_0)) = O(D)$.

En mode asynchrone :

Proposition

En mode synchrone :

- $Message(Flood\&Echo(r_0), G) = O(m)$.
- $Temps(Flood\&Echo(r_0), G) = O(exc(r_0)) = O(D)$.

En mode asynchrone :

- $Message(Flood\&Echo(r_0), G) = O(m)$.

Proposition

En mode synchrone :

- $Message(Flood\&Echo(r_0), G) = O(m)$.
- $Temps(Flood\&Echo(r_0), G) = O(exc(r_0)) = O(D)$.

En mode asynchrone :

- $Message(Flood\&Echo(r_0), G) = O(m)$.
- $Temps(Flood\&Echo(r_0), G) = O(n)$.

Preuve :

En ce qui concerne le nombre de messages, les étapes 2, 3 envoient $\sum_u deg(u) = 2m$ messages, qui sont reçus à l'étape 4. Les étapes 5, 6 concentrent et diffusent le long d'un arbre, donc consomment 2(n-1) messages. En tout, cela fait $O(m+n) = O(m)$ messages puisque G est connexe (et donc $m \geq n-1$).

En ce qui concerne le temps, les étapes 2, 3, 4 prennent un temps $O(D)$. La concentration puis la diffusion aux étapes 5, 6 prennent un temps $O(D)$ en synchrone. Malheureusement, en mode asynchrone, on ne maîtrise pas la hauteur de l'arbre, qui peut être de n-1. La complexité en temps asynchrone est donc $O(D+n) = O(n)$ (car $D < n$).

La concentration est un problème similaire à celui de la diffusion. Il s'agit pour chaque sommet u du graphe d'envoyer un message personnel M_u vers un sommet distingués r_0. C'est en quelques sortes le problème inverse, sauf que les messages ne sont pas tous identiques comme dans le cas d'une diffusion.
Voici une solution, lorsqu'un arbre T de racine r_0 est disponible, c'est-à-dire connu de chaque sommet u par les variables Père(u) et Fils(u).

Diffusion
Mesures de complexité
Arbre de diffusion
inondation
Arbre couvrant
Détection de la terminaison
Concentration

Algorithme $Converge_T(r_0)$ (code du sommet u)

1. Poser c :=|Fils(u)|.
2. Si c=0 et $u \neq r_0$, $Send(M_u, Pre(u))$
3. Tant que c>0 :
   (a) Attendre de recevoir (M, v)
   (b) c :=c-1.
   (c) Si $u \neq r_0$, Send(M, Père(u)).

Si aucun arbre n'est disponible, le plus simple est alors d'en calculer un, par exemple grâce à SpanTree, puis d'appliquer $Converge_T$. La mauvaise solution étant que chacun des sommets réalise indépendamment une diffusion : on est donc sûr que r_0 reçoive une copie de chaque message. Cela n'est pas efficace pour le nombre de messages.

Proposition

Il existe un algorithme distribué Converge qui réalise pour tout graphe G et sommet r_0 la concentration vers r_0 avec :
En mode synchrone :

- $Message(Converge, G) = O(m)$

En mode asynchrone :

Proposition

Il existe un algorithme distribué Converge qui réalise pour tout graphe G et sommet r_0 la concentration vers r_0 avec :
En mode synchrone :

- $Message(Converge, G) = O(m)$
- $Temps(Converge, G) = O(D)$.

En mode asynchrone :

Proposition

Il existe un algorithme distribué Converge qui réalise pour tout graphe G et sommet r_0 la concentration vers r_0 avec :
En mode synchrone :

- $Message(Converge, G) = O(m)$
- $Temps(Converge, G) = O(D)$.

En mode asynchrone :

- $Message(Converge, G) = O(m)$

Proposition

Il existe un algorithme distribué Converge qui réalise pour tout graphe G et sommet r_0 la concentration vers r_0 avec :
En mode synchrone :

- $Message(Converge, G) = O(m)$
- $Temps(Converge, G) = O(D)$.

En mode asynchrone :

- $Message(Converge, G) = O(m)$
- $Temps(Converge, G) = O(n)$.