Calcul distribué simple et efficace de la betweenness

Pierluigi Crescenzi, Pierre Fraigniaud, Ami Paz

Calcul distribué simple et efficace de la betweenness

Pierluigi Crescenzi[1] et Pierre Fraigniaud[1] [†] et Ami Paz[2] [‡]

[1] *IRIF, Université de Paris, CNRS, France*
[2] *Faculty of Computer Science, University of Vienna, Austria*

La *betweenness* est un paramètre de graphes fréquemment utilisé avec succès pour l'analyse de réseaux. Dans le cadre des réseaux d'ordinateurs, ce paramètre a ainsi été utilisé pour répondre à plusieurs objectifs, dont le routage et la répartition de ressources. Toutefois, comme observé par Maccari et al. [INFOCOM 2018], l'utilisation pratique de la betweenness pour l'amélioration de protocols réseaux reste handicapée par l'absence d'algorithmes distribués efficaces pour le calcul de ce paramètre. Cet article résume nos travaux démontrant comment ce problème peut-être résolu, en présentant un algorithm distribué efficace pour le calcul de la betweenness. Cet algorithme peut être mis en œuvre au travers de modifications élémentaires d'algorithmes de routage à vecteur de distance tels que Bellman-Ford. Le temps de convergence de l'algorithme est proportionnel au diamètre du réseau, tout comme Bellman-Ford.

La version complète de ce résumé est disponible dans les actes d'INFOCOM 2020 [CFP20].

Mots-clefs : Algorithmes distribués, protocols réseaux, betweenness centrality.

1 Introduction

La *betweenness* [Fre97] est une mesure d'importance attribuée à tout sommet d'un graphe. La betweenness d'un nœud v est la somme, pour toutes les paires (s,t) de nœuds source-cible, du rapport entre le nombre de plus courts chemins de s à t passant par v, et le nombre total de plus courts chemins de s à t. Ainsi, un nœud de betweeness élevée appartient à un nombre relatif élevé de plus courts chemins, tandis qu'un nœud de betweenness faible appartient à peu de plus courts chemins, relativement à leur nombre total. La notion de betweenness a été appliquée avec succès à l'analyse de réseaux. En effet, dans les réseaux sociaux, un nœud avec une betweenness élevée est vraisemblablement un nœud influent, et, dans les réseaux informatiques, un nœud de betweennness élevée peut être utilisé efficacement pour stocker des ressources pertinentes, mais peut également endommager gravement les communications en cas de défaillance ou de dysfonctionnement. En conséquence, la betweenness et ses variantes ont été utilisées pour optimiser le comportement des réseaux de communication et des réseaux informatiques, que ce soit pour la conception de réseaux maillés sans fil, le routage, la détection de liens, le placement et l'allocation de ressources, le contrôle de la topologie, l'optimisation des taux de transmission ou la sécurité (voir [CFP20] et les références qu'il contient). Par exemple, la fréquence optimale à laquelle les liaisons incidentes doivent être testées en chaque nœud d'un réseau radio est connue pour être inversement proportionnelle à la racine carrée de la betweenness du nœud [MC16].

Toutefois, comme observé par Maccari et al. [MGG$^+$18], les techniques d'optimisation de réseau basées sur la betweenness souffrent de deux problèmes principaux. Premièrement, même si l'on suppose que chaque nœud a accès à des informations sur l'ensemble du réseau, ce qui est par exemple le cas dans les protocoles à état de liens, le calcul de la betweenness peut nécessiter des ressources de calcul excessives. Deuxièmement, il n'existe aucun algorithme efficace connu pour calculer la betweennness dans le

[†] Financements complémentaires au travers des projets ANR DESCARTES, and Inria GANG.
[‡] Ce travail a été mené lorsque le troisième auteur était en poste à l'IRIF comme post-doc financé par la Fondation Sciences Mathématiques de Paris.

contexte des protocoles à vecteur de distance. Les algorithmes distribués existants pour calculer la betweenness ou les plus courts chemins sont soit conçus pour des modèles trop faibles par rapport aux réseaux du monde réel, soit dédiés à des classes restreintes de réseaux (DAG, arbres), soit ils échangent entre les nœuds une quantité d'informations supérieure à la capacité des protocoles à vecteur de distance. En fait, même l'algorithme élégant et pratique de Maccari et al. [MGG+18] pour calculer une mesure semblable à la betweenness [Bra08] échange plus d'informations entre les nœuds que ce qu'il est possible d'attendre d'un protocole à vecteur de distance.

Nos résultats. Nous décrivons dans ce résumé un algorithme distribué simple et rapide pour calculer la betweenness. Plus précisément, notre algorithme permet à chaque nœud v de calculer sa propre betweenness, notée bc_v. Notre algorithme est *simple* au sens qu'il peut être implémenté selon une forme semblable à des protocoles de vecteur de distance basés sur Bellman-Ford. Concrètement, Bellman-Ford demande à chaque nœud v d'envoyer à chacun de ses voisins une paire de valeurs (t,d) pour chaque nœud cible t, où d est la distance actuelle de v à t, telle que perçue par v. Notre algorithme demande simplement à chaque nœud d'envoyer à chaque voisin un quadruplet de valeurs (t,d,s,b) pour chaque nœud cible t, où s est l'estimation actuelle en v du nombre de plus courts chemins de v à t, et b est la contribution actuelle de t à la betweenness de v. Notre algorithme est *rapide* au sens qu'il converge en un nombre de phases proportionnelles au diamètre du réseau. De plus, la quantité amortie de calcul effectué en chaque nœud v lors de la réception d'un message d'un voisin u lié à une cible t est constante, c'est-à-dire indépendante de la taille du réseau. Nous avons effectué un ensemble de *simulations* confirmant à la fois la correction de notre algorithme et l'analyse de son efficacité. Nous avons considéré différents scénarios, y compris des réseaux pondérés et non pondérés, et diverses structures générées par des modèles synthétiques (Erdös-Rényi, attachement préférentiel, etc.) ou extraites de réseaux du monde réel.

Le message principal de cet article est que la betweenness peut être calculée efficacement en chaque nœud de manière distribuée, même dans le contexte des protocoles à vecteur de distance. Par conséquent, il n'y a pas d'obstacle à l'utilisation de la betwenness pour optimiser la fonctionnalité des réseaux, au moins en ce qui concerne le calcul de ce paramètre par le réseau lui-même durant son utilisation. Cela résout une question laissée ouverte dans les travaux de Maccari et al. [MGG+18].

2 Définitions

Soit $G=(V,E)$ un graphe connexe non-orienté, dont les arêtes sont pondérées positivement. Le poids de chaque arête $e \in E$ est noté $w(e) > 0$. Un chemin entre deux sommets distincts $s,t \in V$, ou s-t chemin, est une sequence $v_0,\ldots,v_k$ avec $k \geq 0$, $v_0 = s$, $v_k = t$, et $\{v_{i-1},v_i\} \in E$ pour tout $i = 1,\ldots,k$. La longueur d'un tel chemin est $\sum_{i=1}^{k} w(\{v_{i-1},v_i\})$. Un chemin de longueur minimum entre s et t un appelé plus court chemin entre s et t. La distance $\mathsf{dist}(s,t)$ entre deux sommets s et t est la longueur d'un plus court chemin entre s et t. Le diamètre pondéré de $G=(V,E)$ est défini comme $\max_{s,t\in V} \mathsf{dist}(s,t)$. Le diamètre pondéré ne reflète toutefois pas le temps de convergence des protocoles de routage basés sur Bellman-Ford. La complexité de ce dernier est en effet lié à une autre notion de diamètre, définie comme suit. Pour toutes paire de sommets $s \neq t$, soit $P_1,\ldots,P_\ell$ les $\ell \geq 1$ plus courts chemins entre s et t dans G. Soit $\mathsf{minhop}(s,t)$ le minimum, pour $i = 1,\ldots,\ell$, du nombre d'arêtes de P_i. La notion importante de diamètre pour Bellman-Ford est $\mathsf{diam}(G) = \max_{s,t\in V} \mathsf{minhop}(s,t)$.

La version distribuée standard de Bellman-Ford permet à chaque nœud v de calculer $\mathsf{dist}(v,t)$ pour tout $t \in V$. Cet algorithme converge en $\mathsf{diam}(G)$ phases, où une phase est définie comme le temps requis par chaque nœud v pour envoyer tous les paires $(t,D[t])$, $t \in V$ à tous ses voisins (où $D[t]$ est l'estimation actuelle de $\mathsf{dist}(v,t)$ en v), recevoir tous les messages (t,d), $t \in V$, envoyés par chacun de ses voisins, et effetuer les mises à jour relatives à la réception de ces message. En effet, une simple induction sur $h \geq 0$ permet de montrer que, pour tout $t \in V$, chaque sommet v avec $\mathsf{minhop}(v,t) \leq h$ calcule $\mathsf{dist}(v,t)$ correctement après h phases par Bellman-Ford.

Pour tous sommets $s \in V$ et $t \in V$, soit $\sigma_{s,t}$ le nombre de plus courts chemins de s à t dans G (avec $\sigma_{s,s} = 1$), et soit $\sigma_{s,t}(v)$ le nombre de plus courts chemins de s à t passant par le sommet v (avec $\sigma_{s,s}(s) = 1$).

Définition 1 *La* betweenness [Fre97] *d'un nœud v est* $\mathsf{bc}_v = \frac{1}{(n-1)(n-2)} \sum_{s\neq v, t\neq v} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$.

3 Calcul distribué de la betweenness

Notre algorithme distribué de calcul de la betweenness est essentiellement une implementation de l'algorithme séquentiel de Brandes [Bra01] par programmation dynamique. Pour faciliter la presentation, nous considérons le graphe non-orienté $G = (V,E)$ comme un graphe orienté où chaque arêtes $\{u,v\}$ est remplacée par deux arcs symétriques (u,v) et (v,u), chacun avec le même poids que l'arête $\{u,v\}$. Egalement, nous étendons la definition de $\sigma_{s,t}(v)$ aux arcs, en définissant $\sigma_{s,t}(u,v)$ comme le nombre de plus courts chemins de s à t traversant l'arc (u,v). Les deux faits ci-après découlent directement des définitions.

Fait 1 *Si $\sigma_{s,t}(v) \neq 0$, i.e., si v appartient à un plus court s-t chemin, alors $\sigma_{s,t}(v) = \sigma_{s,v} \cdot \sigma_{v,t}$. De même, si l'arc (u,v) est sur un plus court s-t chemin, alors $\sigma_{s,t}(u,v) = \sigma_{s,u} \cdot \sigma_{v,t}$.*

Pour tout $v \in V$ et tout $t \in V$, soit $\mathsf{NH}_v(t)$ l'ensemble des voisins de v qui appartiennent à un plus court chemin de v à t, et, pour tout $s \in V$, soit $\mathsf{PH}_v(s)$ l'ensemble des voisins u de v tels que v appartient à un plus court chemin de s à u. (NH et PH sont pour "next hop" et "previous hop", respectivement).

Fait 2 *Pour tout $t \neq v$, $\sigma_{v,t} = \sum_{u \in \mathsf{NH}_v(t)} \sigma_{u,t}$. Si v est sur un plus court s-t chemin, alors $\sigma_{v,t} = \sum_{u \in \mathsf{PH}_v(s)} \sigma_{v,t}(v,u)$.*

Enfin, pour tout $s \in V$, soit $\mathsf{bc}_v(s)$ la contribution de la source s à bc_v, c'est-à-dire, $\mathsf{bc}_v(s) = \sum_{t \neq v} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$, et donc $\mathsf{bc}_v = \frac{1}{(n-1)(n-2)} \sum_{s \neq v} \mathsf{bc}_v(s)$ pour tout sommet v. Le résultat ci-dessous est crucial pour notre implémentation distribuée de l'algorithme de Brandes.

Lemme 1 (Brandes [Bra01]) *Pour tout $s \neq v$, $\mathsf{bc}_v(s) = \sigma_{s,v} \sum_{u \in \mathsf{PH}_v(s)} \frac{\mathsf{bc}_u(s)+1}{\sigma_{s,u}}$.*

Notre calcul distribué de la betweenness est explicité dans l'algorithme 1. $N(v)$ denote l'ensemble des voisins $v \in V$ dans $G = (V,E)$, c'est-à-dire $N(v) = \{u \in V : \{u,v\} \in E\}$, et $N[v] = N(v) \cup \{v\}$ denote le voisinage fermé du nœud v. Notre algorithme requiert que chaque sommet $v \in V$ calcule et maintienne les ensembles $\mathsf{NH}_v(t)$ et $\mathsf{PH}_v(t)$ pour tout $t \in V$, selon les règles spécifiées dans les faits 1 et 2, et le lemme 1. Notre résultat principal est le théorème ci-dessous (voir la preuve dans [CFP20]).

Théorème 1 *L'algorithme 1 permet à chaque nœud de calculer sa betweenness dans tout graphe G, en $2\,\mathsf{diam}(G)+1$ phases.*

Algorithm 1 Calcul de la betweenness au nœud v

1: **function** INIT
2: **for all** $t \in V$ **do**
3: $D[t] \leftarrow +\infty$ ▷ *D is a distance vector*
4: $\mathrm{NH}[t] \leftarrow \emptyset$ ▷ *next-hop vector*
5: $\mathrm{PH}[t] \leftarrow \emptyset$ ▷ *previous-hop vector*
6: **for all** $u \in N[v]$ **do**
7: $B[u,t] \leftarrow 0$ ▷ *eventually $\mathsf{bc}_u(t)$*
8: $S[u,t] \leftarrow 0$ ▷ *eventually $\sigma_{u,t}$*
9: $S[v,v] \leftarrow 1$ ▷ $\sigma_{v,v} = 1$
10: $D[v] \leftarrow 0$ ▷ $\mathsf{dist}(v,v) = 0$

11: **function** SEND
12: **loop** ▷ *periodic updates are sent to neighbors*
13: **for all** $t \in V$ **do**
14: send $(t, D[t], S[v,t], B[v,t])$ to all $u \in N(v)$

15: **function** RECEIVE(message (t,d,s,b) from $u \in N(v)$)
16: **if** $t \neq v$ **then** $C \leftarrow C - B[v,t]$
17: **if** $u \in \mathrm{NH}[t]$ **then**
18: $\mathrm{NH}[t] \leftarrow \mathrm{NH}[t] \setminus \{u\}$
19: **if** $t \neq v$ **then** $S[v,t] \leftarrow S[v,t] - S[u,t]$
20: **if** $u \in \mathrm{PH}[t]$ **then**
21: $\mathrm{PH}[t] \leftarrow \mathrm{PH}[t] \setminus \{u\}$
22: $B[v,t] \leftarrow B[v,t] - A[u,t]$
23: $S[u,t] \leftarrow s$
24: $B[u,t] \leftarrow b$
25: **if** $d + w(\{u,v\}) < D[t]$ **then**
26: $D[t] \leftarrow d + w(\{u,v\})$
27: **else if** $d + w(\{u,v\}) = D[t]$ **then**
28: $\mathrm{NH}[t] \leftarrow \mathrm{NH}[t] \cup \{u\}$
29: **if** $t \neq v$ **then** $S[v,t] \leftarrow S[v,t] + S[u,t]$
30: **else if** $d - w(\{u,v\}) = D[t]$ **then**
31: $\mathrm{PH}[t] \leftarrow \mathrm{PH}[t] \cup \{u\}$
32: **if** $S[u,t] \neq 0$ **then**
33: $A[u,t] \leftarrow S[v,t] \cdot \frac{B[u,t]+1}{S[u,t]}$
34: **else**
35: $A[u,t] \leftarrow 0$
36: $B[v,t] \leftarrow B[v,t] + A[u,t]$
37: **if** $t \neq v$ **then** $C \leftarrow C + B[v,t]$

4 Résultats expérimentaux

Nous avons implémenté un simulateur Java afin d'analyser les temps de convergence de notre algorithme dans différents types de réseaux, générés par des modèles de graphes aléatoires ou issus du monde réel. Le simulateur utilise une horloge virtuelle et, pour chaque nœud, déclenche un événement d'envoi par unité de temps. Contrairement à [MGG+18], aucune gigue aléatoire n'a cependant été ajoutée aux événements issus aux nœuds, de sorte que le simulateur analyse l'algorithme dans le cas d'une synchronisation parfaite. Chaque simulation se termine lorsque tous les nœuds convergent vers l'état stationnaire, c'est-à-dire lorsque les valeurs de betweenness calculées dans l'algorithme 1 ne changent plus. La figure 1 illustre quelques résultats obtenus par simulation pour un réseau de systèmes autonomes de 3011 nœuds et diamètre 9. A gauche, la figure représente, en fonction du nombre de phases, l'erreur globale définie comme $\|\mathsf{bc} - C\|_2/\|\mathsf{bc}\|_2 = \sqrt{\sum_{v\in V}(\mathsf{bc}_v - C[v])^2}/\sqrt{\sum_{v\in V}(\mathsf{bc}_v)^2}$. On constate que l'algorithme converge en le nombre de phases spécifié dans le théorème 1. Plus intéressant est le fait qu'après $\mathsf{diam}(G)$ phases, soit la moitié du temps de convergence, l'erreur globale n'est plus que d'environ 10 à 15%. Les nœuds obtiennent donc rapidement une bonne approximation de leur betweenness. A droite, la figure représente, en fonction du nombre de phases, le nombre de nœuds dont la valeur de betweenness calculée est égale à la valeur exacte. On constate qu'après la moitié du temps de convergence global, une fraction significative des nœuds ont déjà obtenu leur betweenness correcte (les nœuds de betweenness zero obtiennent immédiatement leur betweenness exacte, et ne sont pas indiqués sur la figure). Et après 3/4 du temps, une très grande majorité des nœuds ont obtenu leur betweenness exacte. Seules une petite fraction des nœuds nécessitent d'exécuter $2\,\mathsf{diam}(G)$ phases pour converger.

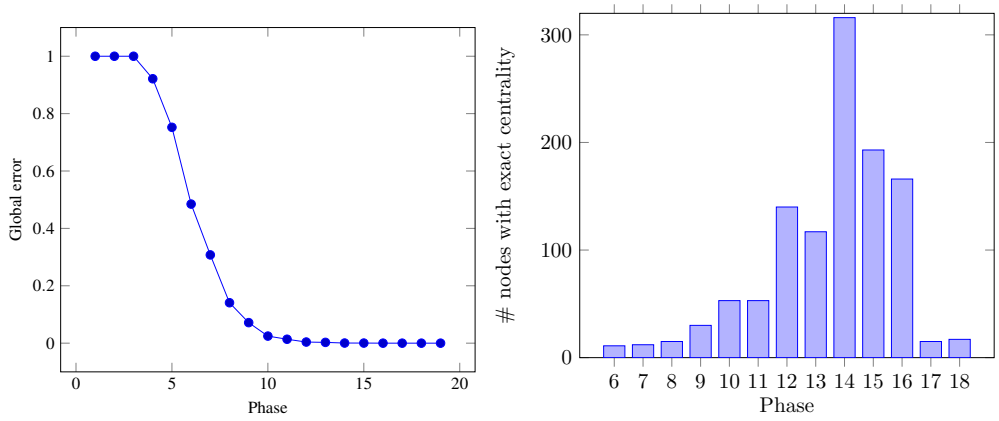

FIGURE 1: Erreur globale et histogramme de convergence

Références

[Bra01] U. Brandes. A faster algorithm for betweenness centrality. *J. of Mathematical Sociology*, 25(2) :163–177, 2001.

[Bra08] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2) :136–145, 2008.

[CFP20] P. Crescenzi, P. Fraigniaud, and A. Paz. Simple and fast distributed computation of betweenness centrality. In *39th IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2020.

[Fre97] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1) :35–41, 1997.

[MC16] L. Maccari and R. Lo Cigno. Pop-routing : Centrality-based tuning of control messages for faster route convergence. In *35th IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1–9, 2016.

[MGG+18] L. Maccari, L. Ghiro, A. Guerrieri, A. Montresor, and R. Lo Cigno. On the distributed computation of load centrality and its application to DV routing. In *37th IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 2582–2590, 2018.