# Un algorithme distribué pour le clustering de grands graphes

Wissem Inoubli, Sabeur Aridhi, Haithem Mezni, Mondher Maddouri, Engelbert Nguifo

# Un algorithme distribué pour le clustering de grands graphes

Wissem Inoubli * **, Sabeur Aridhi**
Haithem Mezni*** Mondher Maddouri****, Engelbert Mephu Nguifo ♯

*Faculté des Sciences de Tunis, LIPAH
** Université de Lorraine, LORIA/ Inria NGE, France
*** SMART Lab, Institut supérieur de gestion de Tunis, Tunisie
**** College Of Business, University de Jeddah
♯ Université Clermont Auvergne, LIMOS

**Résumé.** Le clustering de graphes est l'une des techniques clés qui permet de comprendre les structures présentes dans les données de graphe. La détection des clusters et l'identification des ponts et des bruit sont également des tâches critiques car elles jouent un rôle important dans l'analyse des graphes. Récemment, plusieurs algorithmes de clustering de graphes ont été proposés et utilisés dans de nombreux domaines d'application. La plupart de ces algorithmes sont basés sur les algorithmes de clustering structurel. Néanmoins, ces derniers ont été conçus pour le traitement des petits graphes. D'où, leur performance peut se dégrader dans le cas des graphes larges qui imposent des défis supplémentaires. Dans cet article, nous proposons DSCAN, un algorithme distribué de clustering de graphes qui est basé sur le clustering structurel. Notre algorithme est implimenté sur la base de framework de traitement de grands graphes BLADYG. L'évaluation expérimentale de DSCAN a montré son efficacité et sa compétitivité pour le traitement de grands graphes.

## 1 Introduction

Plusieurs algorithmes de clustering de graphes sont proposés dans la littérature. À titre d'exemples, les approches basées sur la modularité. Le partitionnement de graphes, le min-cut et le clustering spectral. Ces méthodes fournissent en sortie une liste de clusters (groupes) qui ne sont pas vraiment suffisants pour comprendre le comportement du graphe étudié. Pour relever ce défi, l'algorithme de clustering structurel (SCAN) est proposé dans Xu et al. (2007). Le but de développement de ce dernier est non seulement l'identification des clusters dans le graphe, mais également la génération des informations supplémentaires telles que les ponts (sommets entre un ou plusieurs cluster) et les valeurs bruits (sommets qui n'appartiennent à aucun cluster). Le principe de fonctionnement de SCAN est basé sur la topologie ou la structure du graphe. Son principe de base consiste à regrouper des sommets qui partagent le nombre maximal de voisins. De plus, cet algorithme calcule la similarité entre tous les sommets du graphe afin de réaliser le clustering. Néanmoins, cette étape de calcul de similarité est linéaire et elle dépend du nombre d'arêtes du graphe d'entrée. Ceci dégrade la performance de SCAN en particulier dans le cas de grands graphes. En plus, le clustering structurel de graphes est

l'une des méthodes de regroupement les plus efficaces pour différencier les multiples types de sommets d'un graphe donné. Récemment, plusieurs travaux ont été proposés pour surmonter les inconvénients de clustering strucurel dans les grands graphe. Dans Shiokawa et al. (2015), les auteurs implémentent une extension de l'algorithme de base SCAN, dénotée par SCAN ++. L'algorithme suggéré a utilisé une nouvelle mesure structurelle pour calculé la similarité entre deux sommets donnés. Ainsi, SCAN ++ pourrait économiser de nombreuses opérations de calcul de similarité structurelle. De la même manière, les auteurs dans Chang et al. (2017) supposent que l'identification des sommets noyaux représente une tâche essentielle et coûteuse dans SCAN. Partant de cette hypothèse, ils décrivent une méthode d'élagage afin d'identifier les sommets noyeux et donc, d'éviter un grand nombre de calcul de similarité structurelle. D'autres travaux décrivent des implémentations parallèles de l'algorithme SCAN à savoir : Che et al. (2018) et Zhao et al. (2017). En étudiant l'ensemble des travaux susmentionnés, nous constatons qu'ils partagent deux limites majeures : (1) ils ne sont pas en mesure de traiter les grands graphes et (2) ils ne considèrent pas les graphes qui sont déjà distribués/ partitionnés. Dans le but de surmonter ces limites, nous proposons dans ce papier, un nouvel algorithme de clustering de grands graphes (centralisés ou distribués) que nous dénotons par DSCAN. Le reste du papier est organisé comme suit. Après avoir introduit le clustering structurel de graphes dans la première section, nous présentons les concepts de base qui lui sont relatifs dans la deuxième section. Dans la troisième section, nous détaillons notre algorithme proposé. Dans la quatrième section, nous exposons les résultats expérimentaux. La dernière section est finalement réservée pour la conclusion et les travaux futurs.

## 2 Préliminaires

En tant qu'un algorithme populaire, SCAN utilise la similarité structurelle entre les sommets pour effectuer le clustering. En plus, il fournit d'autres informations supplémentaires telles que les bruit et les ponts. Pour mieux assimiler ces idées, nous présentons un aperçu des graphes et de leur clustering par application de SCAN. Considérons un graphe $G = V, E$, où $V$ est un ensemble de sommets et $E$ est un ensemble d'arêtes entre ces sommets. Chacun de ces éléments peut représenter un objet/lien dans des applications réelles. Soit $u$ et $v$ deux sommets en $V$, nous notons par $(u, v)$ un bord compris entre $u$ et $v$ ; $u$ (resp. $v$) est considéré comme un voisin de $v$ (resp. $u$). Dans ce qui suit, nous étendons quelques définitions de base qui sont relatives à la classification de graphes structurels et qui seront utilisées par notre algorithme DSCAN. (Voisinage structurel) Le voisinage structurel d'un sommet $v$, dénoté par $N(v)$, représente tous les voisins d'un sommet donné $v \in V$ incluant le sommet $v$ :

$$N(v) = \{u \in V | (v, u) \in E\} \cup \{v\} \tag{1}$$

(Similarité structurelle) La similarité structurelle entre chaque paire de sommets $(u, v)$ dans $E$ représente un nombre de voisins structurels partagés entre $u$ et $v$. Il est défini par $\sigma(u, v)$.

$$\sigma(u, v) = \frac{|N(u)| \cap |N(v)|}{\sqrt{|N(u)| \,.\, |N(v)|}} \tag{2}$$

Après avoir calculé la similarité structurelle avec Eq. (2), SCAN utilise deux paramètres pour détecter les sommets noyaux d'un graphe donné $G$.

($\epsilon$-neighborhood) Chaque sommet a un ensemble de voisins structurels, comme indiqué dans la Définition 2. Pour regrouper un sommet et ses voisins dans le même cluster, ils doivent avoir un lien fort entre eux. Les liens forts sont dénotés par $\epsilon$ -vosinage. Pour SCAN, il utilise un seuil $\epsilon$ et le paramètre Eq. 3 afin de filtrer les connexions les plus fortes de chaque sommet. Le voisinage $\epsilon$ est défini comme suit :

$$N_\epsilon(u) = \{N(u)|\sigma(u,v) \geq \epsilon\} \tag{3}$$

Notons que le seuil $0 < \epsilon \leq 1$ indique dans quelle mesure deux sommets $u$ et $v$ sont connectés en fonction des voisins structurels partagés. En outre, il représente une métrique avec laquelle les sommets les plus importants, également appelés sommets *noyaux*, sont détectés.
(Noyau) La détection des sommets centraux est une étape fondamentale de l'algorithme SCAN. Cette étape consiste à trouver les sommets dominants dans un graphe $G$. Cette étape permet de construire l'ensemble de clusters ou un schéma de clustering. Un sommet noyau $v$ est un sommet qui possède un nombre suffisant de voisins fortement liés à celui-ci $N_\epsilon(v)$. Nous utilisons $\mu$ comme nombre minimum de voisins fortement connectés (voir Définition 2). Un sommet de type $noyau$ est modélisé comme suit :

$$V_c = \{v \mid N_\epsilon(v) \geq \mu\} \tag{4}$$

(Bordure) Soit $v_c$ un sommet ayant deux listes de voisins structurels : (1) les voisins faiblement connectés à $v_c$ qui sont également appelés les sommets bruit ($N(V_c) \setminus N_\epsilon(V_c)$) et (2) les voisins fortement connectés sont appelés *voisins accessibles structurés*. Dans notre travail, les voisins structurés accessibles sont appelés sommets bordures. $N_\epsilon(V_c)$ représente les sommets bordures d'un sommet noyau $V_c$. Une fois que les sommets et leurs bordures sont déterminés, nous procédons à l'étape de clustering. Pour ce faire, nous utilisons la définition suivante : (Cluster) Un cluster $C$ ($|C| \geq 1$) est un sous-ensemble non vide de sommets, où sa construction est basée sur l'ensemble des sommets noyau et leurs sommets de bordure. En comparaison avec les autres algorithmes de clustering de graphes, SCAN renvoie des informations supplémentaires telles que celles de ponts et de bruit comme défini ci-dessous :

(Pont et bruit) L'étape de clustering répartie les sommets noyaux et leurs bordures en un ensemble de clusters. Cependant, certains sommets n'ont pas de liens étroits avec un sommet noyau. D'où, la détermination des clusters qu'ils peuvent les inclure est impossible. Pour résoudre ce problème, l'algorithme SCAN classe ces sommets dans deux catégories : (1) pont et (2) bruit. Un sommet $v$, qui ne fait partie à aucun cluster et qui a au moins deux voisins dans deux clusters différents, est considéré comme un pont. Sinon, il est considéré comme un bruit.

# 3 Algorithme proposé

Dans cette section, nous présentons DSCAN, un nouvel algorithme distribué pour le clustering structurel de graphes. D'un point de vue architectural, notre approche est basée sur un modèle maître/esclaves et elle est implémentée sur la base de framework de traitement des grands graphes BLADYG Aridhi et al. (2017). Dans ce framework, les esclaves (des machines de type travailleur) sont responsables de l'exécution d'un calcul spécifique sous le contrôle de la machine maître. De même, les données d'entrée doivent être divisées en des ensembles de morceaux (sous-graphes dans notre cas). Dans ce qui suit, nous présentons les deux étapes principales de DSCAN : (1) le parittion de graphes et (2) le clustering distribué de graphes.

## 3.1 Partitionnement de graphe distribué

Dans cette étape, DSCAN commence par la division du graphe d'entrée $G$ en plusieurs partitions $P_1P_2..P_n$ tout en maintenant la cohérence des données (structure globale du graphe). Pour garantir la propriété de cohérence lors de la division du graphe d'entrée, nous devons identifier une liste d'arêtes de coupe permettant d'obtenir une vue globale de $G$. Habituellement, le problème de partitionnement des graphes est classé dans la famille des problèmes NP-difficiles, qui doivent évaluer toutes les combinaisons pour obtenir le meilleur résultat de partitionnement. Pour cette raison, nous avons proposé un algorithme de partitionnement approximatif et distribué comme étant une étape préliminaire pour notre algorithme de clustering distribué. Dans l'étape de partitionnement, la machine maître divise de manière équitable le fichier de graphe d'entrée en sous-fichiers en fonction du nombre d'arêtes. Puis, elle envoie ces derniers à tous les travailleurs. Chaque travailleur obtient une liste d'arêtes et de sommets de son sous-fichier. Il reçoit également, la liste des sommets de travailleurs voisins par la machine maître. Ceci est utile pour la détermination des sommets en frontière et donc pour l'identification des arêtes de coupe. Notons que ces dernières dénotent les arrêtes ayant au moins un sommet en frontière.

### 3.1.1 Clustering distribué

Après avoir divisé le graphe d'entrée $G$ en sous-graphes répartis entre les travailleurs, DSCAN déclenche l'étape de clustring distribué. Cette étape est divisée à son tour en deux parties fondamentales : le clustering local et l'aggrégation des résultats intermédiaires. Dans ce qui suit, nous décrivons en détail le principe de fonctionnement de ces dernières ainsi que le lien entre elles.

**Etape 1 : clustering local**. Comme il est présenté dans la figure 1, le graphe d'entrée $G$ est fragmenté en plusieurs sous-graphes/partitions ($\mathbb{P}$), chacun d'eux est affecté à une machine esclave (travailleur). L'étape de partitionnement décrite dans la section 3.1, est effectuée conformément au paramètre $\alpha$ qui fait référence au nombre de machines esclaves .

Pour éviter la perte d'informations au cours de l'étape de partitionnement (les arêtes reliant des sommets dans les différentes partitions), les sommets de frontière sont dupliqués dans les partitions voisines. Ensuite, pour chaque partition $P_i$, le clustering local est effectué sur les différentes partitions des machines esclaves. Nous considérons les cas où plusieurs conflis entre deux partitions $\mathbb{P}1$ et $\mathbb{P}2$ ont eu lieu. Par exemple, le sommet $v \in V_{P1}^{f}$ est un sommet noyau dans $\mathbb{P}1$ et un sommet de type bruit dans $\mathbb{P}2$. Ces conflits doivent être évités lors de l'agrégation des résultats locaux.

**Etape 2 : fusion**. En camparaison avec l'algorithme de référence SCAN, la distribution du calcul de similarité et l'étape de clustering local peuvent réduire le temps de réponse de DSCAN. Cependant, nous devons prendre en compte l'exactitude des résultats renvoyés. Pour garantir les mêmes résultats que SCAN, nous définissons un ensemble de scénarios pour assurer l'étape de fusion. Ces scénarios seront appliqués sur toutes les partitions de $G$ . Ils sont répétés jusqu'à ce que toutes les partitions soient combinées. Pour chaque paire de partitions $P_i$ et $P_j$, une fonction de fusion doit être exécutée pour combiner les résultats locaux de $P_i$ et $P_j$ . Ces résultats sont stockés dans des variables globales au niveau de la machine maître. L'algorithme présente également plusieurs scénarios (Lemmas 3.1.1, 3.1.1 et 3.1.1) pour résoudre les conflits rencontrés.

**Entree :** Graphe $G$ , paramètres $(\mu, \epsilon, \alpha)$
**Sortie :** $\mathbb{C}, \mathbb{P}, \mathbb{B}$
/* C liste de clusters, P liste des ponts et B : liste des bruits */
/* Etape 1: Clustering local (d'une manière parrallèle) */
**pour chaque** $Worker \in \mathbb{W}$ **faire**
  *Calculer la similarité structurelle pour toute la partition* $P_i$;
  *Déterminer les sommets de type noyau dans* $P_i$ ;
  *Construire les clusters locaux dans* $P_i$;
  *Déterminer les sommets de types pont et bruit dans* $P_i$;
**fin**
/* Etape 2: Fusion */
**pour chaque** $Worker \in \mathbb{W}$ **faire**
  **si** *(*$C_1 \cap C_2 \cap .. \cap C_\alpha = \mathbb{V}$; *and* $\exists V_i \in \mathbb{N}oyau$ *Suivant 3.1.1)* **alors** ;
  $C \leftarrow$ **Fusionner**($C_1, C_2, ..C_\alpha$);
  Enoyer $C$ à la machine maître;
  **sinon**
    Envoyer les clusters locaux à la machine maître;
  **fin**
  **pour** $V_i$ ***Dans*** $\mathbb{B}ruits$ **faire**
    **si** *(*$V_i \in \mathbb{N}oyaux \cup \mathbb{B}ordures \cup \mathbb{P}onts$ *suivant 3.1.1 )* **alors** ;
    ***Supprimer*** $V_i$ *de la liste de bruit* $\mathbb{B}ruit$;
  **fin**
  **pour** $V_i$ ***Dans*** $\mathbb{B}ruits$ **faire**
    $Nb_{Connections} \leftarrow 0$;
    **pour** $C_i$ ***Dans*** $\mathbb{C}lusters$ **faire**
      **si** $(N(V_i) \cap C \neq \varnothing)$ **alors** ;
      $Nb_{Connections} + +$;
    **fin**
    **si** ( $Nb_{Connections} \geq 2$ *suivant 2 )* **alors** ;
    ***Ajouter*** $V_i$ *dans* $\mathbb{P}onts$ ;
    ***Supprimer*** $V_i$ *de* $\mathbb{B}ruits$;
  **fin**
**fin**
*Envoyer C,P,B à la machine maître*;

FIG. 1 – *L'algorithme DSCAN*

**(Fusion de clusters locaux)** Soit $\mathbb{C}_1$ et $\mathbb{C}_2$ deux ensembles de clusters locaux dans différentes partitions $P_1$ et $P_2$, respectivement. $\exists c_1 \in \mathbb{C}_1$ et $\exists c_2 \in \mathbb{C}_2$, $Noyau(c_1) \cap Noyau(c_2) \neq \emptyset$. Soit $C_i$ un cluster qui regroupe un ensemble de sommets de type bordure ou noyau. Si $C_i$ partage au moins un sommet noyau $c$ avec un autre cluster $C_j$, alors $c$ a un ensemble de bordures dans $C_i$ et $C_j$ . De même, tous les sommets dans $C_i$ et $C_j$ peuvent être accessibles à partir de $c$. D'où, $C_i$ et $C_j$ devraient être fusionnés en un même cluster.

**(De bruit à pont)** soit $\mathbb{C}_1$ et $\mathbb{C}_2$ deux ensembles de clusters locaux dans les deux partitions $P_1$ et $P_2$, respectivement.
$\exists$ $C_1$ et $C_2$ deux clusters qui appartiennent aux deux ensembles de clusters différents $\mathbb{C}_1$ et $\mathbb{C}_2$. De plus, $\exists$ un sommet de type bruit dans les deux partitions $P_1$ et $P_2$ et $N(o) \cap c_1 \neq \emptyset$ et $N(o) \cap c_2 \neq \emptyset$

Si $C_i$ et $C_j$ $(i \neq j)$ partagent un sommet de type bruit $o$, cela signifie que $o$ est faiblement connecté aux deux clusters $C_i$ et $C_j$. Conformément à la Définition 2, $o$ devrait être changé a un sommet de type pont.

**(De Pont à bruit)** soit $C_1$ et $C_2$ deux clusters locaux respectivement dans les deux partitions $P_1$ et $P_2$.
$\exists$ un pont $b$ en fonction uniquement de deux clusters $c_1$ et $c_2$. Lorsque $c_1$ et $c_2$ seront fusionnés en un seul cluster (pendant l'étape de fusion), $b$ devrait être changé à un sommet de type bruit.

Soit $C_i$ et $C_j$ deux clusters qui ont un ensemble de sommets (bordures ou noyaux) et un ensemble de ponts avec d'autres clusters locaux et $\exists$ $b$ un sommet de pont uniquement en fonction des deux clusters $C_i$ et $C_j$ où $i \neq j$. Dans l'étape de fusion et selon le lemme 3.1.1, si un ou plusieurs clusters partagent au moins un sommet noyau, nous les fusionnons en un même cluster. Dans ce cas ($C_i$, $C_j$) $\Rightarrow C$ ce qui rend $b$ faibelement connecté à un seul cluster ($C$). Ainsi, selon Définition 2, $b$ devrait changer son statut de pont à bruit.

# 4 Etude expérimentale

Dans cette section, nous présentons une étude expérimentale où nous évaluons l'efficacité de notre algorithme pour le clustering structurel de grands graphes distribués.

## 4.1 Protocole expérimental

Nous avons comparé DSCAN à quatre fameux algorithmes de clustering structurels de graphes : SCAN basique, pSCAN, AnyScan, et ppSCAN. Les algorithmes comparés sont divisés en deux catégories : (1) algorithmes centralisés tels que SCAN et pSCAN et (2) algorithmes parallèles tels que AnyScan et ppSCAN. Pour exécuter des algorithmes centralisés et parallèles, nous avons utilisé une machine virtuelle *T3.2xlarge* sur Amazon EC2. Pour évaluer DSCAN, nous avons utilisé un cluster de 10 VMs *t3.large*. Pour tous les cas de test, nous avons utilisé un ensemble de graphes du monde réel (California road network (G1), com-Youtube (G2), com-Orkut(G3), com-LiveJournal (G4) et com-Friendster (G5)) obtenus à partir du projet d'analyse de réseau de Stanford (SNAP). Pour plus de détails voir Inoubli et al. (2019).

## 4.2 Résultats expérimentaux

**Speedup**. Nous évaluons l'accélération de DSCAN par rapport au SCAN ainsi qu'à un ensemble de ses variantes. Comme illustré dans la figure 2, notre approche est plus lente dans le cas de petits graphes (G1, G2, G3). En fait, un très grand écart est remarqué entre DSCAN et les autres algorithmes, en particulier avec pSCAN et ppSCAN. Cet écart se réduit lorsque la taille du graphe augmente (cas du jeu de données G4). Il est important de mentionner que DSCAN est une implémentation distribuée de SCAN et que les autres algorithmes étudiés sont centralisés. Cela entraîne des coûts supplémentaires pour la distribution, la synchronisation et la communication. De plus, Fig 2 montre qu'avec les configurations matérielles modestes, seul DSCAN peut passer à l'échelle en traitant de grands graphes comme dans le cas du jeu de données G5.

**Passage à l'échelle**. L'objectif principal de cette expérience est d'évaluer le passage à l'échelle horizontal de notre algorithme. Nous utilisons deux graphes (LiveJournal et le réseau routier californien). La figure 3 montre clairement que notre algorithme est scalable horizontalement, ce qui n'était pas garanti avec les autres algorithmes, comme indiqué dans la section de l'état de l'art. La figure 3 montre également que la durée d'exécution diminuera en fonction

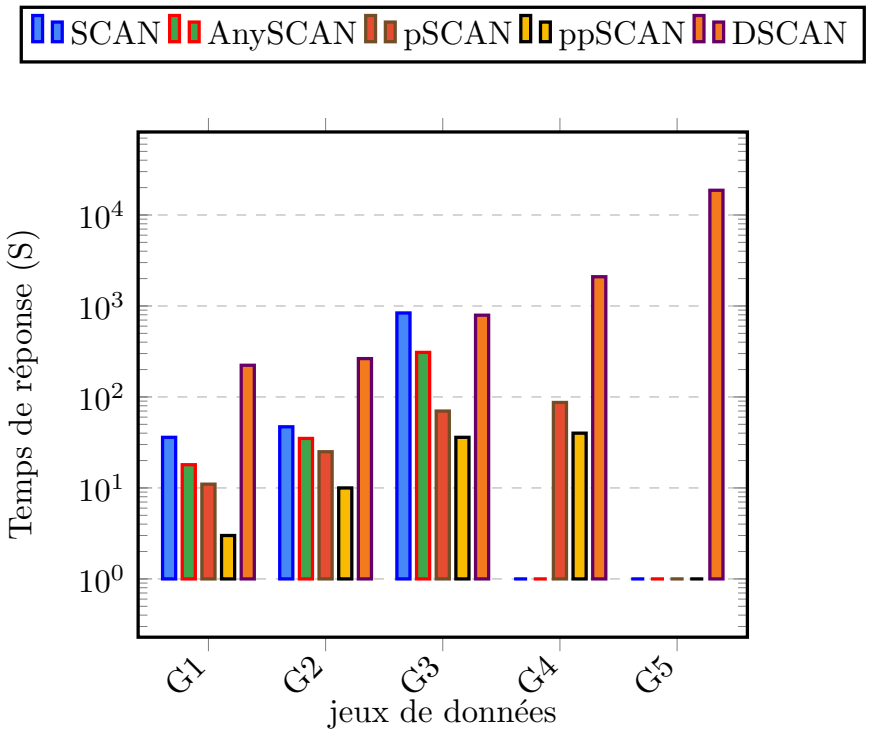


FIG. 2 – *Impact de la taille du graphe sur le temps de réponse*

du nombre de machines dans le cluster. Ainsi, lorsque nous ajoutons une nouvelle machine, la durée d'exécution diminue.

## 5 Conclusion

Dans cet article, nous avons proposé DSCAN, un algorithme distribué pour le clustering de grands graphes en se basant sur la similarité structurelle. DSCAN est basé sur une architecture distribuée et une autre maître/esclaves qui le rend scalable et fonctinnable sur les machines modestes. L'algorithme proposé est capable d'exécuter n'importe quelle taille du graphe et peut être parallèlement mis à l'échelle sur un grand nombre de machines. Nous avons évalué la performance de DSCAN par rapport à d'autres algorithmes. Les expériences ont montré qu'il garantie une scalabilité horizontale ce qui n'est pas le cas avec les autres algorithmes.

## Références

Aridhi, S., A. Montresor, et Y. Velegrakis (2017). Bladyg : A graph processing framework for large dynamic graphs. *Big Data Research 9*, 9–17.

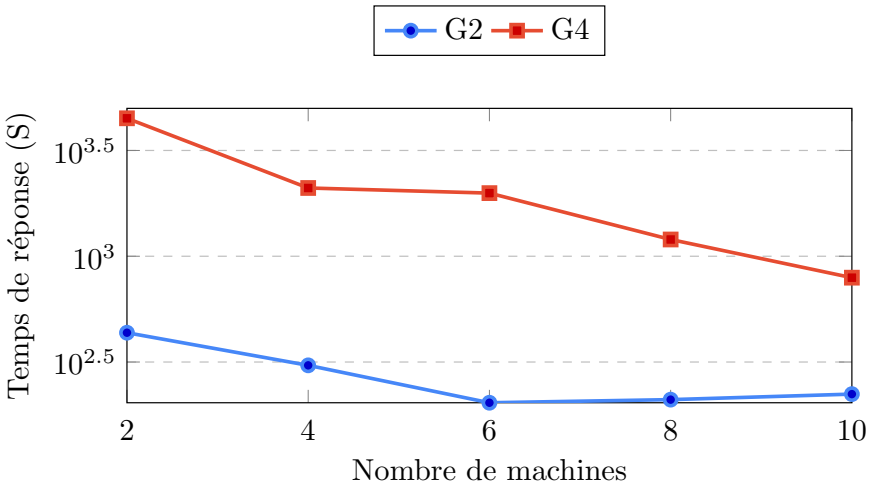


FIG. 3 – *Impact du nombre de machines sur le temps de réponse (avec $\epsilon = 0.5$, $\mu = 3$).*

Chang, L., W. Li, L. Qin, W. Zhang, et S. Yang (2017). pscan : Fast and exact structural graph clustering. *IEEE Transactions on Knowledge and Data Engineering 29*(2), 387–401.

Che, Y., S. Sun, et Q. Luo (2018). Parallelizing pruning-based graph structural clustering. In *Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018, Eugene, OR, USA, August 13-16, 2018*, pp. 77 :1–77 :10.

Inoubli, W., S. Aridhi, H. Mezni, M. Mondher, et E. Nguifo (2019). A distributed algorithm for large-scale graph clustering, hal-02190913v2.

Shiokawa, H., Y. Fujiwara, et M. Onizuka (2015). Scan++ : efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. *Proceedings of the VLDB Endowment 8*(11), 1178–1189.

Xu, X., N. Yuruk, Z. Feng, et T. A. Schweiger (2007). Scan : a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 824–833. ACM.

Zhao, W., G. Chen, et X. Xu (2017). Anyscan : An efficient anytime framework with active learning for large-scale network clustering. In *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 665–674. IEEE.

## Summary

Graph clustering is one of the key techniques to understand the structures present in the graph data. In addition to clusters detection, the identification of hubs and outliers is also a critical task as it plays an important role in the analysis of graph data. In this paper, we propose DSCAN, a novel distributed graph clustering algorithm based on the structural clustering algorithms.