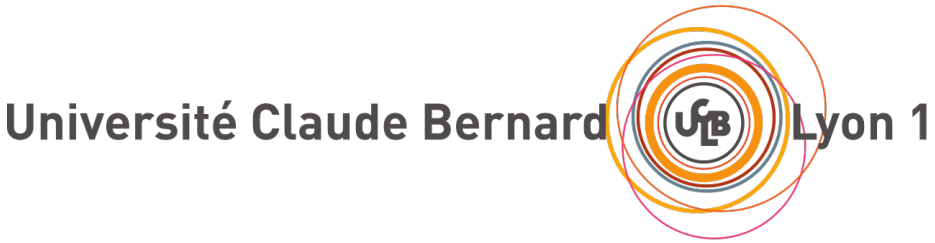
Compression de graphes par partitionnement structurel

Mohammed Haddad, François Pitois, Hamida Seba

LIRIS - Laboratoire d'InfoRmatique en Image et Systèmes d'information

16 - 18 novembre 2020



La compression de graphes

Définition (Compression sans perte de graphes)

Une fonction de *compression sans perte* $f : \mathcal{G} \to \{0,1\}^*$ est une fonction telle qu'il existe une fonction de décompression $g : \{0,1\}^* \to \mathcal{G}$ vérifiant, $g(f(G)) = G$ pour tout $G \in \mathcal{G}$.

Définition (La recherche structurelle)

La *recherche structurelle* consiste à trouver des structures dans un graphe. Une structure est un sous-graphe assez simple et facilement compressible. Par exemple, on peut considérer les structures suivantes : clique, quasi-clique, stable, quasi-stable, biparti, presque-biparti, cycle, etc.

État de l'art

Certains sont basés sur de la recherche structurelle :

Algorithme	**Caractéristiques**
ConDeNSe [Liu et al., 2018]	Encode la matrice d'adjacence
[Maneth and Peternek, 2018]	Encode via une grammaire
[Chatterjee et al., 2016]	Encode la matrice d'adjacence
[Fan et al., 2012]	Peut effectuer des requêtes

D'autres non :

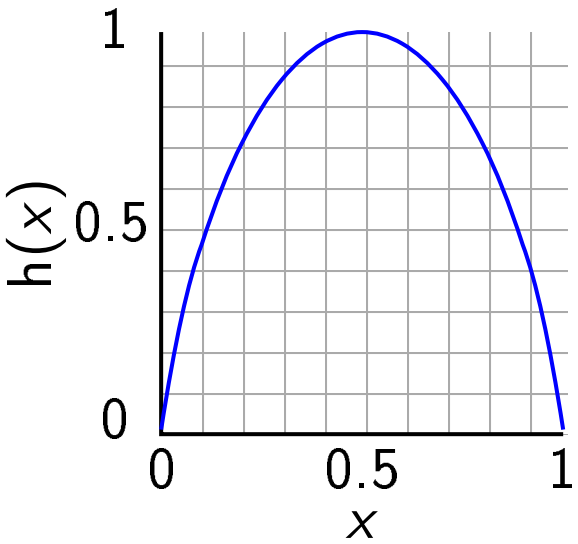
Algorithme	**Caractéristiques**
SlashBurn [Kang et al., 2011]	Basé sur de l'ordonnancement

Le codage arithmétique

Théorème

Le codage arithmétique *permet d'encoder asymptotiquement une suite de n bits de densité d en n h(d) bits.*

On rappel que $d = \#(1)/n$ et que h représente l'entropie de Shannon de la suite de bits, c'est-à-dire que $h(x) = -x\log(x) - (1-x)\log(1-x)$.

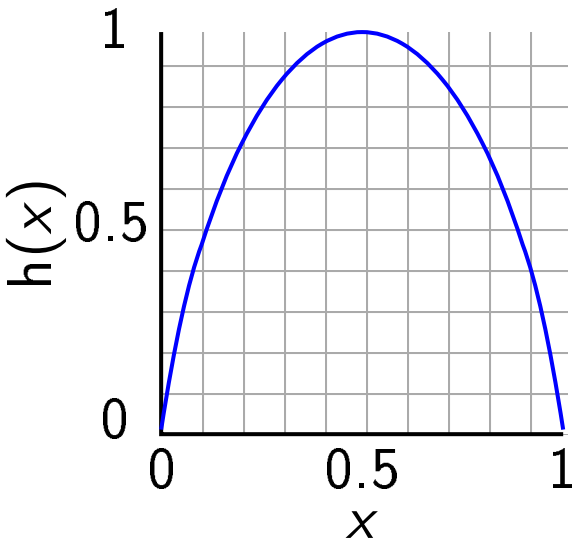


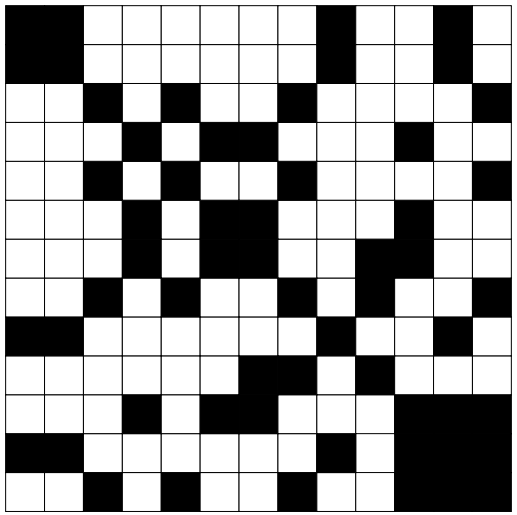
Le codage arithmétique

Théorème

Le codage arithmétique *permet d'encoder asymptotiquement une suite de n bits de densité d en n h(d) bits.*

On rappel que $d = \#(1)/n$ et que h représente l'entropie de Shannon de la suite de bits, c'est-à-dire que $h(x) = -x\log(x) - (1-x)\log(1-x)$.



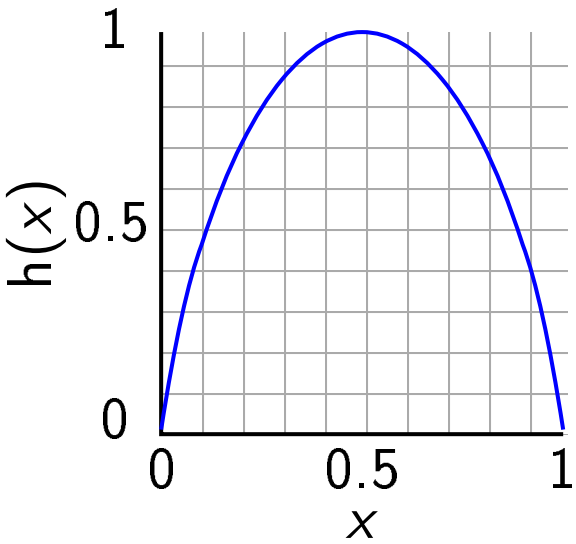


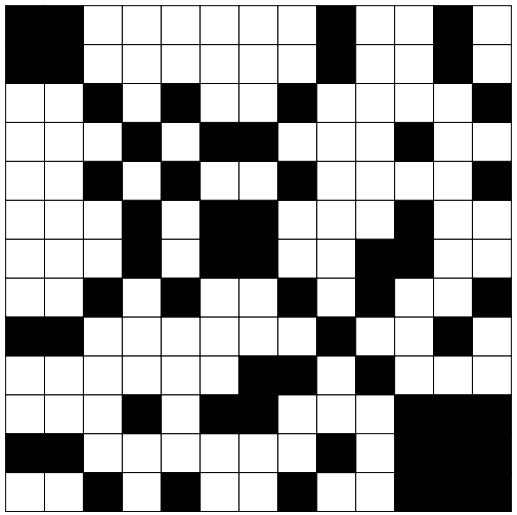
Le codage arithmétique

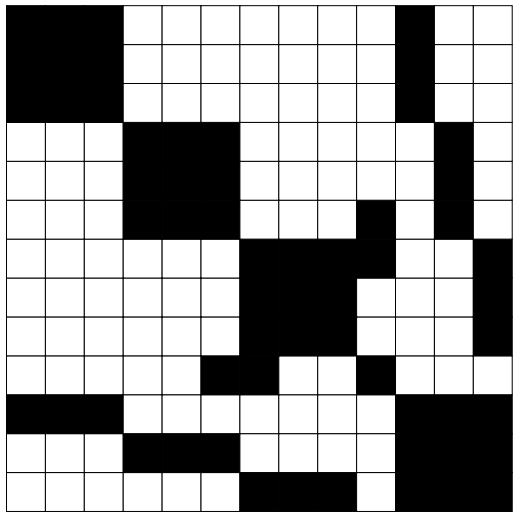
Théorème

Le codage arithmétique *permet d'encoder asymptotiquement une suite de n bits de densité d en n h(d) bits.*

On rappel que $d = \#(1)/n$ et que h représente l'entropie de Shannon de la suite de bits, c'est-à-dire que $h(x) = -x\log(x) - (1-x)\log(1-x)$.







Notre algorithme

Notre algorithme de compression par partitionnement structurel se divise essentiellement en 3 grandes étapes :

1. Recherche structurelle : Trouver des structures* dans le graphe.
2. Ordonnancement : Trouver un ordre des sommets qui soit compatible avec les structures trouvées à l'étape 1.
3. Compression : Compresser la matrice d'adjacence associé à l'ordre trouvé à l'étape 2.

*Remarque : les structures peuvent éventuellement se chevaucher.

Recherche structures

Pour trouver des structures, on fait appel à d'autres algorithmes dédiés à la recherche de structures.

- BigClam [Yang and Leskovec, 2013]
- KCBC [Liu et al., 2015]
- Metis [Karypis et al., 1999]
- Spectral [Hespanha, 2004]
- HyCoM [Araujo et al., 2014]
- Louvain [Blondel et al., 2008]
- SlashBurn [Kang and Faloutsos, 2011]

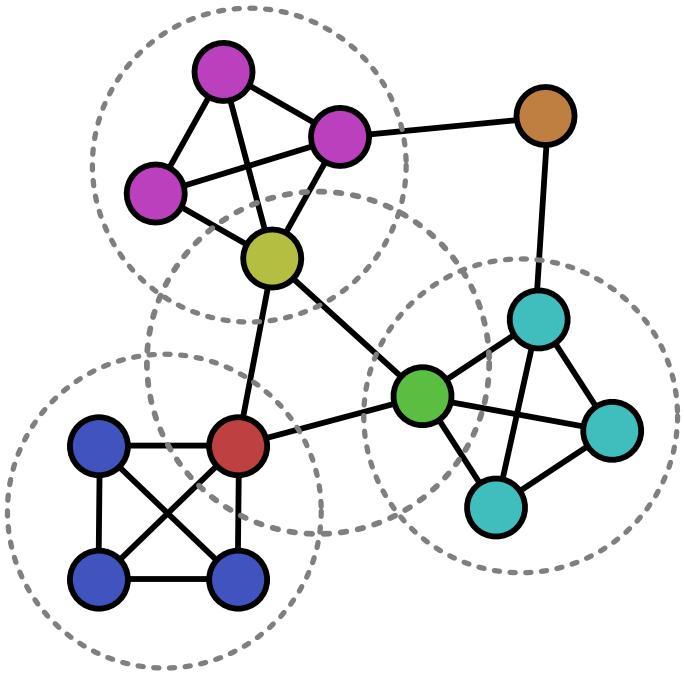
Sélection des structures

Algorithme : Heuristique gloutonne

Données : un graphe G, la liste des structures trouvées T
Résultat : une sélection de structures S incluses dans T
initialiser l'ensemble des structures sélectionnées à $S = \emptyset$
début
  pour chaque *structure s dans T* **faire**
    estimer le taux de compression si on ajoute s à S
  si *il existe une structure qui améliore la compression* **alors**
    ajouter la meilleure structure à l'ensemble S
  retourner S

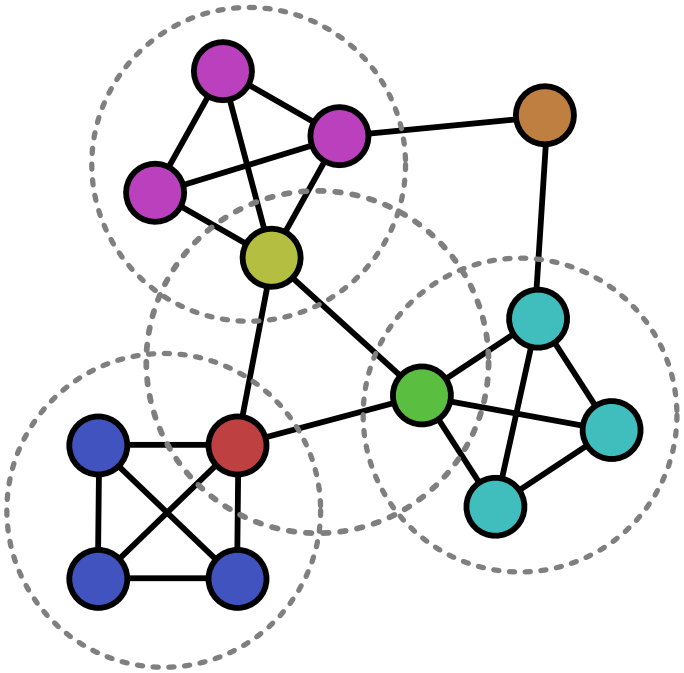
Partitionnement structurel

On partitionne les sommets du graphe suivant les structures sélectionnées, c'est-à-dire que deux sommets sont dans la même partie si et seulement si ils appartiennent exactement aux mêmes structures.



Ordonner les sommets

On ordonne les sommets du graphe suivant la partition que l'on vient de créer, c'est-à-dire que les sommets d'une même partie doivent être consécutifs.



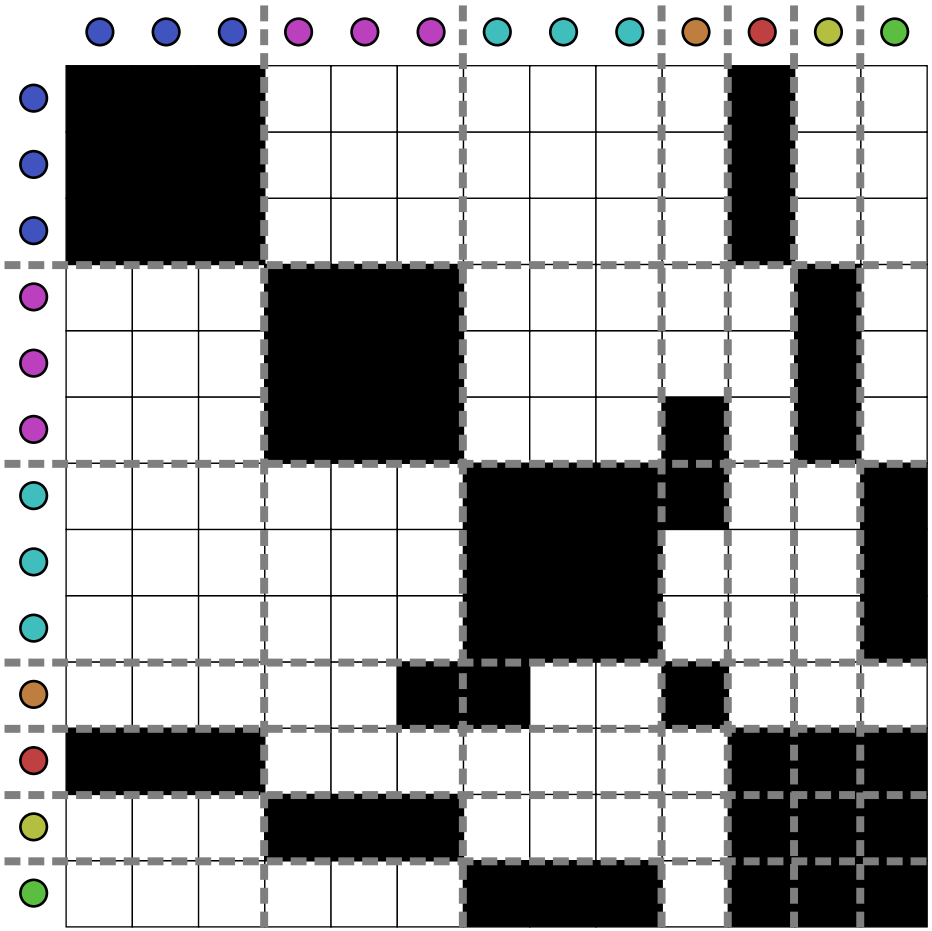
Découpage en blocs

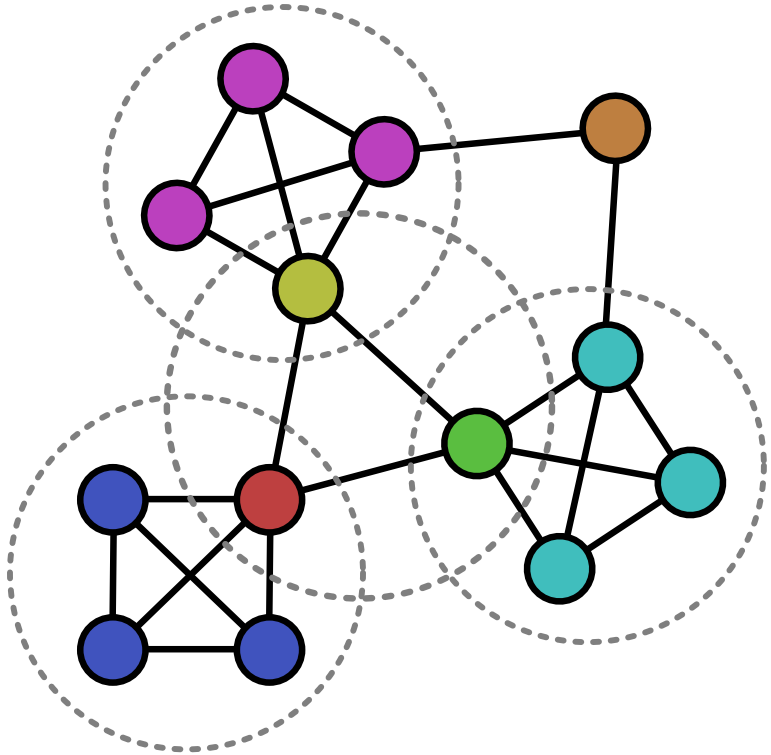
On considère la matrice d'adjacence associé à l'ordonnancement que l'on vient de trouver. On sait que l'ordonnancement choisi rend consécutif des sommets qui appartiennent à une même partie, et donc a une même structure. Cela permet de découper la matrice d'adjacence en *blocs*.

Définition (Bloc)

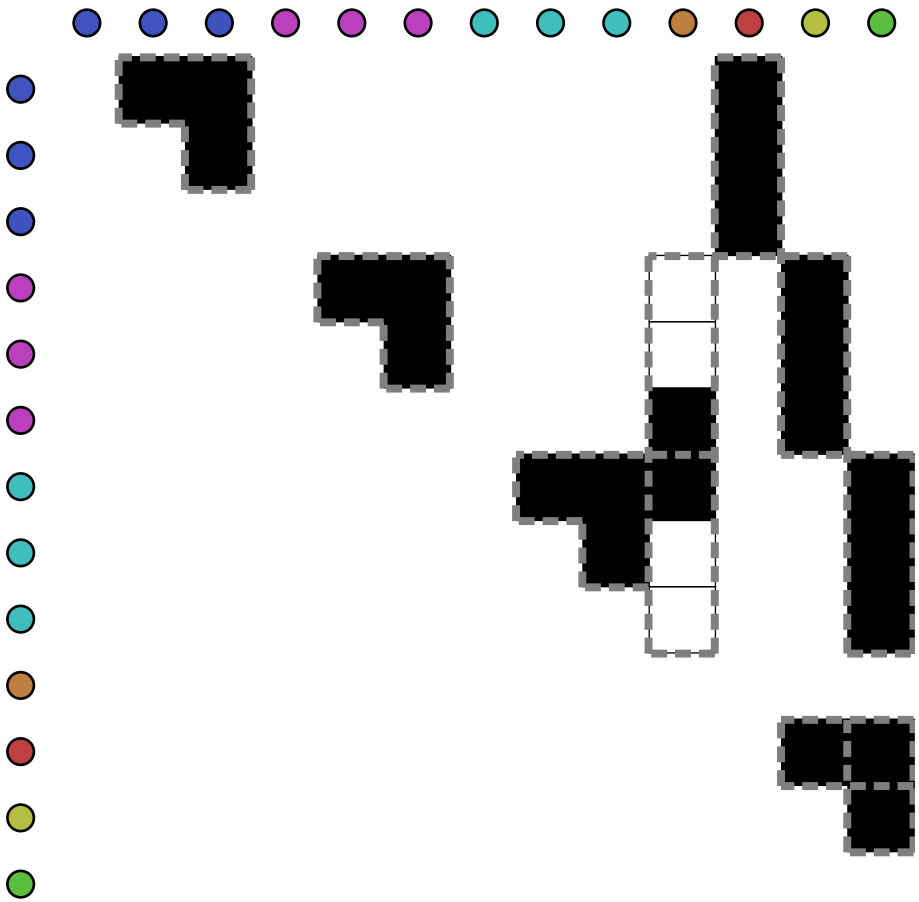
Un *bloc* associé à deux parties est la sous-matrice de la matrice d'adjacence pour laquelle on a gardé uniquement les lignes correspondant aux sommets de la première partie et les colonnes correspondant aux sommets de la deuxième.

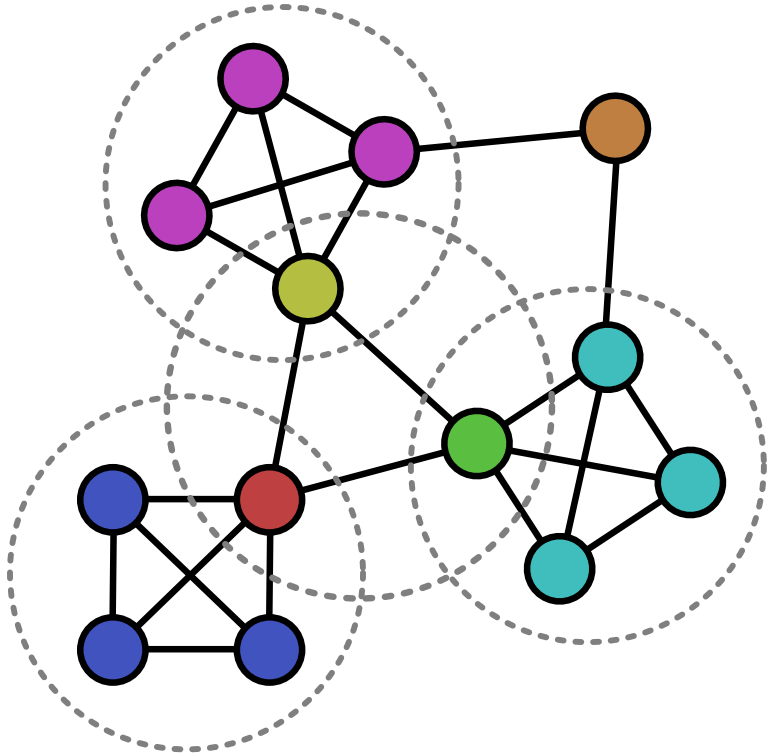
Exemple de découpage en blocs





Visualisation de la matrice compressée





Résumé des données à compresser

Un graphe compressé par notre algorithme est constitué de :

- La liste des tailles des différentes parties : $\mathcal{O}(\sum \log k_i)$, où k_i est le nombre de sommets dans la partie i.
- La matrice globale qui indique quels sont les blocs non-vides : $\mathcal{O}(p^2\ \mathrm{h}(d))$, où p est le nombre de parties, $\mathrm{h}(x) = -x\log(x) - (1-x)\log(1-x)$ représente l'entropie de cette matrice, et d est la densité de la matrice.
- La liste des blocs non-vides : $\mathcal{O}(\sum k_i\ k_j\ \mathrm{h}(d_{i,j}))$, où $d_{i,j}$ est la densité du bloc correspondant aux parties i et j.

Complexité

Complexité des différentes parties de l'algorithme :

- Recherche des structures : $\mathcal{O}(n^3)$
- Sélection des structures : $\mathcal{O}(s^2\ (n+m))$
- Compression : $\mathcal{O}(n+m)$

(n est le nombre de sommets, m le nombre d'arêtes, s est le nombre de structures trouvées)

Évaluation : dataset

3 graphes ont été utilisés pour tester notre algorithme :

- `ia-infect` : Contacts humains à Dublin
- `choc` : Co-éditeurs de la page Wikipédia Chocolate
- `as-oregon` : Réseau de routage dans l'Oregon

Graphe	`ia-infect`	`choc`	`as-oregon`
Nombre de sommets	392	2 899	13 579
Nombre d'arêtes	2 765	5 467	37 448

Évaluation : résultat

Graphe	ia-infect	choc	as-oregon
Liste des tailles	214 bits	886 bits	1 771 bits
Matrice globale	529 bits	5 405 bits	35 621 bits
Blocs non-vides	9 577 bits	31 411 bits	221 898 bits
Compressé	10 320 bits	37 702 bits	259 290 bits
Baseline	17 560 bits	60 310 bits	475 911 bits
Taux de compression	59 %	63 %	54 %
Bits par arête	3,73	6,90	6,92
ConDeNSe	3,5	9	8,5
SlashBurn			7,72

Perspectives

Plusieurs perspectives :

- Faire une recherche en amont pour décider quels algorithmes lancer parmi les sept.
- Lors de la recherche structurelle, améliorer l'heuristique gloutonne, optimiser l'estimation du taux de compression.
- Faire des tests sur des graphes plus gros.
- Voir comment on peut exploiter le graphe compressé.

Merci

Merci !

Des questions ?

References I

Araujo, M., Günnemann, S., Mateos, G., and Faloutsos, C. (2014).
Beyond blocks : Hyperbolic community detection.
In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008).
Fast unfolding of communities in large networks.
Journal of statistical mechanics : theory and experiment, 2008(10) :P10008.

References II

Chatterjee, A., Levan, M., Lanham, C., Zerrudo, M., Nelson, M., and Radhakrishnan, S. (2016).
Exploiting topological structures for graph compression based on quadtrees.
In *2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 192–197. IEEE.

Fan, W., Li, J., Wang, X., and Wu, Y. (2012).
Query preserving graph compression.
In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 157–168. ACM.

References III


Hespanha, J. P. (2004).
An efficient matlab algorithm for graph partitioning.
Santa Barbara, CA, USA : University of California.


Kang, U. and Faloutsos, C. (2011).
Beyond 'caveman communities' : Hubs and spokes for graph compression and mining.
In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 300–309.


Karypis, G., Aggarwal, R., Kumar, V., and Shekhar, S. (1999).
Multilevel hypergraph partitioning : applications in vlsi domain.
IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 7(1) :69–79.

References IV


Liu, Y., Safavi, T., Shah, N., and Koutra, D. (2018).
Reducing large graphs to small supergraphs : a unified approach.
Social Network Analysis and Mining, 8 :1–18.


Liu, Y., Shah, N., and Koutra, D. (2015).
An empirical comparison of the summarization power of graph clustering methods.
arXiv preprint arXiv :1511.06820.


Maneth, S. and Peternek, F. (2018).
Grammar-based graph compression.
Information Systems, 76 :19–45.

References V


Yang, J. and Leskovec, J. (2013).
Overlapping community detection at scale : a nonnegative matrix factorization approach.
In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM.