Review : A Random Sampling O(n) Force-calculation Algorithm for Graph Layouts

H. Durand

Note de synthèse

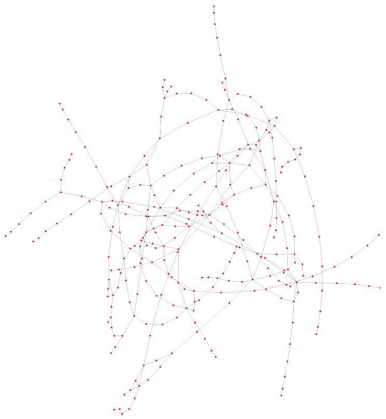

Fig. 1. Graphe du métro de Paris avec algorithme RVS

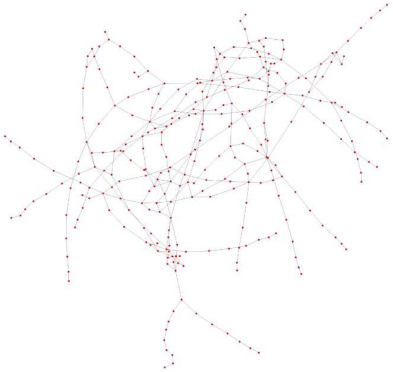

Fig. 2. Graphe du métro de Paris avec algorithme BH

Cette note de synthèse a pour objectif de décrire les travaux présentés par R. Gove dans l'article ***A Random Sampling O(n) Force-calculation Algorithm for Graph Layouts*** *[1]. Il y est d'abord décrit les motivations et le contexte dans lequel se place l'article, puis l'apport de l'auteur suivi d'un rapide état de l'art du tracé de graphes par les méthodes dîtes force-directed, d'une description concise des idées de l'algorithme proposé et des résultats obtenus en terminant par une courte critique de l'article et de l'approche de l'auteur.*

1 Motivations

L'utilisation de graphes comme structure de données est très courante dans plusieurs domaines scientifiques, que ce soit l'analyse de réseaux sociaux et de réseaux de télécommunication, la cartographie la linguistique ou la biologie. Cela s'explique en partie par le fait qu'un graphe $G = \{V, E\}$ présente une structure suffisamment simple (un ensemble de sommets V et un ensemble d'arêtes E reliant certains des sommets entre eux) pour être une abstraction intéressante de nombreux concepts. Cet objet mathématique intervient dès lors que l'on cherche à étudier des relations entre différentes entités (celles-ci pouvant prendre n'importe quelle forme). La représentation de ces graphes est un problème étudié depuis plusieurs siècles mais le développement de l'informatique dans la seconde moitié du $20^{ème}$ siècle, permettant de traiter une quantité de données plus importantes, a entraîné un essor de la recherche de méthodes de visualisation des graphes de grande dimension (qui sont en général *non planaires*). Même s'il n'existe pas de mesure objective unique de la qualité visuelle d'un graphe, certaines métriques sont tout de même intéressantes à observer, comme le nombre d'arêtes s'entrecoupant (il a été montré [2] qu'il existe une corrélation entre le nombre d'erreur d'interprétation d'un graphe et la proportion d'arêtes s'entrecoupant) ou l'angle moyen des intersections entre arêtes.

Les méthodes dîtes *force-directed*, dont il est question dans cet article, font parties des méthodes prédominantes de ce domaine. Elle sont introduites en 1963 par W. T. Tutte [3] et possèdent une analogie intéressante avec la mécanique. L'idée étant de définir une notion d'énergie interne du graphe et de positionner ses sommets de telle sorte qu'ils minimisent cette énergie. L'algorithme de Tutte prend, par exemple, pour notion d'énergie la somme des longueurs des arêtes au carré et on a ainsi $E(G) = \sum_{(u,v)\in E} d^2(u,v)$ qui atteint un minimum local en $x_n(u) = \frac{1}{|U|}\sum_{v\in V tq(u,v)\in E} x_n(v), \forall n \in \{1,...,p\}$ où $d(.,.)$ est la distance euclidienne et p la dimension de la représentation du graphe (en général 2 ou 3). En gardant l'analogie avec la mécanique et en s'appuyant plus particulièrement sur la loi de Hooke (qui modélise le comportement des solides élastiques soumis à une contrainte), une nouvelle approche est introduite en 1984 par P. Eads [4] qui modélise les interactions entre les sommets du graphe par deux forces. Une force d'attraction entre les sommets reliés par une arête, et une force de répulsion, entre les sommets n'étant pas reliés entre eux. On a alors

$F_{res}(u,v) = k_{uv}(d(u,v) - l_0)\vec{i_{uv}}$[1] (la force ressort qui attire les sommets reliés par une arête commune s'ils sont à une distance supérieure à l_0 et les repousse sinon) et $F_{rep}(u,v) = \frac{r_{uv}}{d^2(u,v)}\vec{i_{uv}}$[2] (la force de répulsion qui éloigne toute paire de sommets non reliés par une arête). Le problème de minimisation et en général non-convexe et on utilise une heuristique de type descente de gradient pour trouver un minimum local.

L'avantage de ces méthodes est qu'elles sont conceptuellement simples et faciles à mettre en oeuvre et cela pourrait expliquer, selon l'auteur de l'article, leur prépondérance dans la communauté scientifique. Elles sont par contre relativement coûteuses en temps de calcul et en espace mémoire, et ceux principalement du fait du calcul des forces de répulsion qui, contrairement aux forces d'attraction qui sont uniquement calculées sur les voisins de chaque sommet (soit en $O(|V|)$), doivent être calculées sur l'ensemble du graphe et l'approche force brute à alors une complexité en temps quadratique (en $O(|V|^2)$). Cela peut donc être problématique dès lors que l'on considère un graphe de grande dimension.

2 Contribution

L'auteur de l'article [1] propose dans un premier temps une nouvelle approche de type *force-directed* en temps linéaire pour le tracé de graphe. Les résultats obtenus semblent très intéressants car l'algorithme proposé *Random Vertex Sampling* (*RVS*) est jusqu'à 6.1 fois plus rapide que la méthode BH (qui est l'algorithme de référence pour cet article). Les performances mesurées (en terme de qualité visuelle) par l'algorithme RVS sont similaires à celles obtenues par l'algorithme BH. En revanche l'auteur remarque que les graphes réalisés par l'algorithme RVS sont moins symétriques que ceux obtenus par l'algorithme BH et il propose donc un algorithme combinant RVS et BH qui améliore la symétrie des graphes tout en gardant une complexité en temps de calcul linéaire.

3 État de l'art

L'alogorithme proposé par P. Eads dans [4] pour le calcul des forces de répulsion est un algorithme naïf dont la complexité en temps de calcul est quadratique en le nombre de sommets ce qui le rend en pratique peu utilisable pour des graphes de grande dimension. Différentes heuristiques ont été proposées pour accélérer ce calcul.

Fructerman et Reingold propose en 1991 [5] de diviser l'espace par une grille en réduisant ainsi le temps de calcul moyen mais le calcul des forces de répulsion reste, dans le pire des cas, quadratique.

Plusieurs algorithmes en temps de calcul de l'ordre de $O(|V|log(|V|))$ ont été proposés entre 1980 et 1990. Particulièrement l'algorithme de Barnes-Hut [6] est une méthode de référence dans la communauté scientifique car, selon R. Gove, les méthodes concurrentes (Fast Multipole Method [7] et Well-Separated Pair Decomposition [8]) sont considérées comme plus difficiles à implémenter. Le calcul de la force de répulsion par l'algorithme BH est basé sur la création d'une structure d'arbre hiérarchique.

4 Méthodologie

L'auteur, avec l'algorithme *Random Vertex Sampling*, propose d'effectuer l'étape de mise à jour des positions des sommets dûe à la force de répulsion uniquement sur un sous-ensemble U de l'ensemble V des sommets du graphe à chaque itération. On calcule les forces de répulsion de chacun des sommets $u \in U$ en les comparant avec un autre sous-ensemble de sommet $R \subset V$, différent pour chaque sommet de U. Plus particulièrement, si on choisit U et V de telle sorte à avoir $|U| = |V|^{\phi}$ et $|R| = |U|^{1-\phi}$ avec $0 < \phi < 1$, on peut calculer les forces de répulsion en $O(|U|.|R|) = O(|V|)$. Le sous-ensemble R est choisit aléatoirement par un algorithme semblable à l'algorithme de mélange de Fisher-Yates. Le sous-ensemble U est quand à lui choisit par fenêtre glissante sur U. Ces deux méthodes permettent de sélectionner les sous-ensembles R et U en temps linéaire.

Au départ les sommets sont disposés de manière à former un disque (ceux se trouvant au début du tableau des sommets étant les plus au centre). Puis, on applique l'algorithme RVS sur $|V|^{\phi}$ sommets du tableau en commençant par les premiers et en glissant vers les derniers à chaque itération.

L'auteur de l'article estime de manière empirique une valeur $\phi = 3/4$, bien que la méthode employée pour cela ne soit pas détaillée. Il précise ainsi que l'algorithme semble plus performant si les sommets sont mis à jour plus souvent mais de manière moins précise (valeur de ϕ plus grande) que si les sommets sont mis à jour moins souvent de manière plus précise donc avec un plus grand ensemble R (valeur de ϕ plus petite). La loi des grands nombres semble jouer un rôle ici même s'il n'est pas fait mention de théorèmes asymptotiques dans l'article.

Finalement, en observant que les résultats produits par l'algorithme Random Vertex Sampling sont moins symétriques que ceux produits par l'algorithme Barnes-Hut, l'auteur propose de combiner l'algorithme RVS et l'algorithme BH. Pour cela, il suffit de faire tourner l'algorithme RVS sur un nombre suffisant d'itérations pour obtenir une bonne position initiale que l'on pourra alors ajuster en lançant quelques itérations de l'algorithme BH. L'expérimentation semble montrer qu'une dizaine d'itérations seraient suffisantes pour améliorer la symétrie du rendu.

5 Résultats

L'analyse des résultats est présentée selon deux aspects : le temps de calcul de l'algorithme et la qualité du rendu. Trois algorithmes sont comparés, l'algorithme de référence Barnes-Hut, l'algorithme Random Vertex Sampling et une combinaison des algorithmes BH et RVS comme décrite précédemment. Chacun des algorithmes est lancé 20

[1] k_{uv} : raideur du ressort, i_{uv} : vecteur unitaire dirigé de u vers v

[2] r_{uv} est une constante de répulsion

fois pour une durée de 300 itérations pour chacun des 109 graphes, de plus ou moins grande dimension.

L'évaluation du temps de calcul montre que RVS permettrait d'être en moyenne 2.9 fois plus rapide que BH et la combinaison de RVS et BH d'être en moyenne 2.8 fois plus rapide que BH seul. Sur des graphes de très grande dimension (générés artificiellement tels que $|E| = 2|V|$) l'accélération de l'algorithme RVS et de la combinaison RVS BH semble non constante (plus les graphes sont grands plus l'accélération du temps de calcul est grande) ce qui est prometteur pour l'application de l'algorithme sur des graphes de très grande dimension. Cela est d'ailleurs cohérent avec les complexités en temps des algorithmes RVS (complexité linéaire) et BH (complexité log-linéaire).

La mesure des performances en terme de qualité du rendu est analysé selon cinq métriques permettant d'avoir une vision globale de la qualité visuelle des graphes. Il ne semble pas y avoir de différences significatives, concernant ces mesures, entre les résultats obtenus par les trois algorithmes étudiés. Aussi il semble qu'ils permettent tous de renvoyer un rendu de qualité similaire bien que comme il a été précédemment expliqué, RVS donne des tracés moins symétriques que BH (ce qui peut être compensé en appliquant quelques itérations de l'algorithme BH à l'algorithme RVS).

6 Analyse critique et perspectives

L'article est dans l'ensemble très clair tant dans les explications de l'algorithme proposé que dans la mise en contexte. Les résultats qu'il propose sont intéressants et ceux particulièrement pour le tracé de graphes de grande dimension qui demande des longs temps de calcul avec les méthodes de référence actuelles (Barnes-Hut en particulier). Il permet ainsi d'obtenir des résultats dont les qualités visuelles sont similaires à celles des algorithmes de référence mais avec un temps de calcul jusqu'à 6.1 fois plus rapide.

Par ailleurs l'auteur est dans une démarche de facilitation de la reproductibilité des résultats en mettant à disposition le code source de son algorithme, les résultats de ses expérimentations ainsi qu'une interface graphique conviviale permettant d'expérimenter les modèles RVS, BH et RVS & BH simplement. Son travail semble d'ailleurs s'appuyer sur l'observation fait dans [9] : les méthodes de visualisation s'appuyant sur des algorithmes ou des objets mathématiques complexes ne semblent pas prendre la place qui leur serait dûe au vu de leurs performances. La méthode proposée est donc conceptuellement simple tout en affichant des performances intéressantes.

Il semble enfin qu'il pourrait être intéressant de formaliser la méthode en montrant, si c'est le cas, que l'algorithme converge bien vers un minimum local et de borner sa vitesse de convergence. Il serait également pertinent de détailler la méthode utilisée pour le choix de la valeur de ϕ et de voir si sa valeur optimale dépend du graphe étudié.

References

[1] Gove, R., 2019. "A random sampling o(n) force-calculation algorithm for graph layouts". *Computer Graphics Forum,* **38**, 06, pp. 739–751.

[2] Purchase, H., Cohen, R., and James, M., 1997. "An experimental study of the basis for graph drawing algorithms. acm journal of experimental algorithmics, 2(4), 4 es". *Journal of Experimental Algorithmics (JEA),* **2**, 01, p. 4.

[3] Tutte, W. T., 1963. "How to Draw a Graph". *Proceedings of the London Mathematical Society,* **s3-13**(1), 01, pp. 743–767.

[4] Eades, P., 1984. "A heuristic for graph drawing".

[5] Fruchterman, T. M. J., and Reingold, E., 1991. "Graph drawing by force-directed placement". *Software: Practice and Experience,* **21**.

[6] Barnes, J., and Hut, P., 1986. "A hierarchical O(N log N) force-calculation algorithm". *,* **324**(6096), Dec., pp. 446–449.

[7] Grengard, L., and Rokhlin, V., 2006. *The Rapid Evaluation of Potential Fields in Particle Systems in Three Dimensions*, Vol. 1360. 11, pp. 121–141.

[8] Callahan, P., and Kosaraju, S., 1995. "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields.". *J. ACM,* **42**, 01, pp. 67–90.

[9] Naps, T., Cooper, S., Koldehofe, B., Leska, C., Rößling, G., Dann, W., Korhonen, A., Malmi, L., Rantakokko, J., Ross, R., Anderson, J., Fleischer, R., Kuittinen, M., and Mcnally, M., 2003. "Evaluating the educational impact of visualization (report of the working group on evaluating the educational impact of visualization)". *ACM SIGCSE Bulletin,* **35**, 12, pp. 124–136.