Visualização de grandes redes em ambiente web

Felipe Buzatti Nascimento[1], **Raquel C. de Melo-Minardi**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo-Horizonte – MG – Brazil

{buzatti,raquelcm}@dcc.ufmg.br

Abstract. *O presente trabalho apresenta uma metodologia de visualização de grandes redes em um ambiente web através do uso de diferentes algoritmos de layout e agrupamento hierárquico, que são combinados e avaliados por diversas métricas de agrupamento. A partir da melhor combinação de layout-agrupamento avaliada é gerada uma visualização de grafos no plano utilizando múltiplos níveis de agrupamento, permitindo buscas por atributos, exploração das informações nos grupos e vértices em diversos níveis simultaneamente.*

1. Introdução

O presente trabalho procura desenvolver uma visualização de grafos não-direcionados em um ambiente web para tratar o problema de visualização de grandes redes. O objetivo é representar a gama de informações contidas em redes com milhares de vértices e milhões de arestas sem descartar informações cruciais ao mesmo tempo em que se utiliza uma representação que seja acessível ao usuário da visualização.

Uma vez que o desenho de toda a rede no plano seria intratável para o usuário e além da capacidade do ambiente *web*, utilizam-se algoritmos de agrupamento hierárquico (ver Seção 2.1) para determinar diferentes níveis de representação da rede, que são apresentados em maior detalhe apenas quando usuário julga necessário.

Diferentes algoritmos de *layout* e agrupamento foram combinados (ver Seção 2). A escolha da melhor combinação layout-agrupamento para cada rede é feita utilizando métricas internas de avaliação dos agrupamentos (mais detalhes na Seção 3).

A visualização, que foi implementada no ambiente D3[Heer et al. 2011], um framework javascript, utiliza as informações de posição fornecidas pelo algoritmo de layout, bem como a estrutura hierárquica do algoritmo de agrupamento para apresentar ao usuário uma visão inicialmente ampla da rede, ao mesmo tempo em que o permite explorar os agrupamentos que lhe são mais interessantes.

As principais contribuições do trabalho são: a combinação de métricas internas de avaliação dos agrupamentos utilizando tanto as informações de posição do layout do grafo quanto sua topologia para avaliar a qualidade dos layouts e agrupamentos; uma caracterização dos grupos na visualização; uma busca por qualquer atributo na rede; uma visualização de grandes redes que pode ser acessada em navegadores web.

2. Algoritmos de *layout* e agrupamento

Combinamos diferentes algoritmos de agrupamento e layout para criar diferentes representações da rede, que serão posteriormente comparadas entre si, como mostra a Seção 3. Exploramos quatro algoritmos de *layout* de grafos: *Linlog*[Noack 2007],

ForceAtlas2[Jacomy et al. 2014] (em sua modelagem clássica e utilizando a opção *linlog*), *Yifan-Hu[Hu 2005]* e *Fruchterman-Reingold*[Fruchterman and Reingold 1991]. Exceto pelo algoritmo LinLog, disponibilizado[1] pelo autor, os demais algoritmos de layout foram executados segundo a implementação disponível na ferramenta Gephi[Bastian et al. 2009].

Foram utilizados três algoritmos de agrupamento: *Kmeans*[Arthur and Vassilvitskii 2007], utilizando apenas a informação da posição de cada vértice fornecida pelo algoritmo de layout, *Ward*[Ward Jr 1963], utilizando a posição dada pelo algoritmo de layout e a conectividade fornecida pela topologia do grafo e *RNSC[King et al. 2004]* (Restricted Neighborhood Search Clustering Algorithm), utilizando apenas a informação topológica do grafo.

As implementações dos algoritmos *Ward* e *Kmeans* foram obtidas através da biblioteca *SKLearn[Pedregosa et al. 2011]* na linguagem Python. O algoritmo RNSC foi executado a partir de uma implementação fornecida[2] pelos autores. Além desses foi desenvolvido um novo algoritmo de agrupamento, com duas variações, uma em que são consideradas primariamente as posições definidas pelo algoritmo de layout e outra em que considera-se a topologia do grafo em primeiro lugar, e posteriormente a posição dos vértices para julgar a construção dos agrupamentos.

2.1. Definição do número de agrupamentos e agrupamento em multinível

O presente trabalho procura determinar o número de agrupamentos através de métricas de validação internas que indiquem a qualidade dos diferentes agrupamentos (também conhecidas como *stopping rules*[Milligan and Cooper 1985]). Foram utilizadas métricas que levam em consideração tanto a posição dos vértices no plano quanto as arestas entre os vértices. A posição dos vértices foi obtida a partir do algoritmo de *layout.*

Nesse trabalho foram utilizadas as seguintes métricas que consideram a posição: *Davies-Bouldin*[Davies and Bouldin 1979], *Dunn*[Dunn† 1974],*Calinski-Harabasz[Cali´nski and Harabasz 1974]* e *GS**[Boutin and Hascoet 2004]. As métricas utilizadas que analisam arestas são as seguintes: *MinMaxCut*[Ding et al. 2001], *Coverage*[Brandes et al. 2003], *MQ*[Boutin and Hascoet 2004]* e *modularity*[Newman and Girvan 2004].

Para cada layout são gerados agrupamentos utilizando o algoritmo *kmeans* com número de grupos variando entre 1 e k, sendo $k < N$, em que N é o número de vértices no grafo. Dessa forma é montado um ranking do número de grupos para cada layout. Esse ranking é utilizado para determinar os grupos de forma que seu número não seja maior que o limite L de elementos que podem ser visualizados eficientemente e o número de elementos dentro de cada grupo também não exceda esse mesmo limite L. Utilizamos um método *bottom-up* que executa os algoritmos de agrupamento da Seção 2. Esse algoritmo percorre o *ranking* do número de grupos, e toda vez que um agrupamento consegue cumprir o limite para o número máximo L de elementos em cada grupo, mas falha no limite máximo de grupos, esse agrupamento é utilizado como entrada para o algoritmo de agrupamento no ciclo seguinte até que o número de grupos no nível mais superior também não exceder L.

[1] disponível em `https://code.google.com/p/linloglayout/`

[2] disponível em `http://www.cs.toronto.edu/~juris/data/rnsc/rnsc.zip`

3. Comparação dos algoritmos de layout e agrupamento

Todo algoritmo de layout é utilizado como entrada para todos os algoritmos de agrupamento. Para definir um só resultado utilizamos as métricas mencionadas na Seção 2.1. Ordenamos para cada métrica o valor medido em cada instância. Então cada instância recebe uma pontuação de acordo com sua posição no ranking, de forma que, seja sua posiçao p_i na métrica i, sua pontuação é $R = \sum_i (n - p_i)$, em que n é o número de instâncias testadas.

Executamos esse ranking para todas as métricas e somamos em cada instância os valores de R obtidos para cada métrica. A melhor combinação nessa análise será utilizadas para representar a rede. A Figura 1 apresenta os resultados dessa combinação em um conjunto de instâncias.

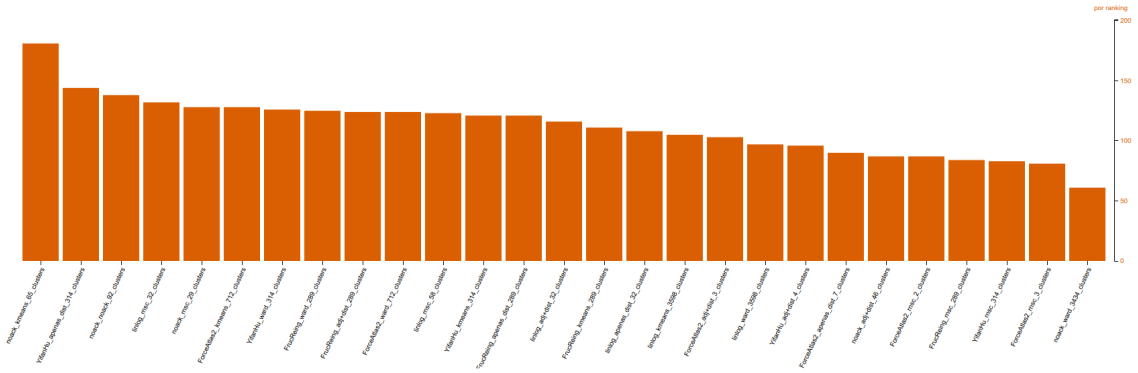

Figura 1. Comparativo do resumo das métricas para combinações de algoritmos de layout e agrupamento.

4. Visualização

A visualização, exemplificada na Figura 2, apresenta os vértices e os agrupamentos através de círculos, com seus centros definidos segundo as coordenadas calculadas pelo algoritmo de layout (e pela média aritmética das coordenadas dos vértices que compõem o grupo). A cor da circunferência diferencia vértices, que possuem uma circunferência branca, de agrupamentos, com uma circunferência vermelha.

O raio dos círculos nos vértices pode ser definido por qualquer atributo numérico presente em todos os vértices da rede bem como pelos valores de *betweenness* ou *coeficiente de clusterização* (ver Figura 2.d). No caso dos agrupamentos, o raio é definido pela média aritmética dos atributos e métricas mencionados para o raio dos vértices nos elementos que pertencem ao grupo.

As arestas entre vértices de grupos diferentes têm seus pesos somados e uma aresta com essa soma é criada entre os grupos. De forma análoga, as arestas entre vértices de um grupo e um vértice fora desse grupo são combinadas para que sejam visualizadas como uma só aresta quando esse grupo e vértice estiverem presentes na visualização. Para não tumultuar a visualização, apenas as arestas do vértice/grupo sob o cursor são visualizadas e nenhuma aresta é visualizada quando o cursor não está sobre um círculo.

Inicialmente, a visualização apresenta todos os grupos e vértices que não estão agrupados em nenhum outro grupo. O usuário pode expandir os grupos que mais lhe

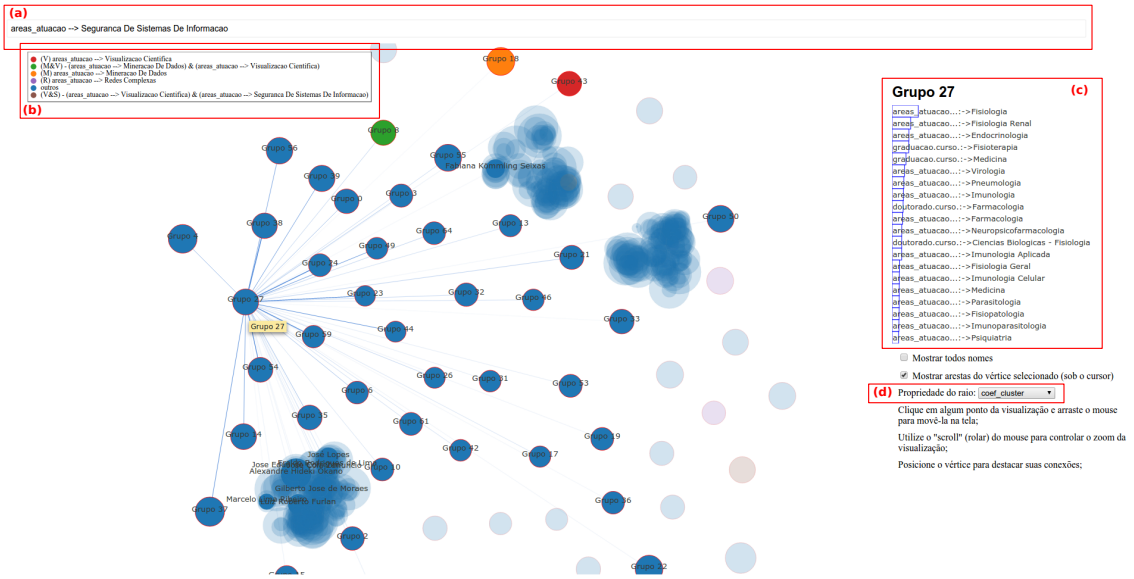

Figura 2. Visão de exemplo da ferramenta de visualização. (a) Caixa de pesquisa textual; (b) legenda com atributos pesquisados; (c) caracterização do grupo sob o cursor; (d) seletor da propriedade que define o raio dos elementos.

interessam clicando sobre esses grupos. De forma a impedir que a expansão de muitos grupos gere um número muito grande de elementos na visualização, resultando em um custo computacional muito alto, há um mecanismo que registra o número de elementos expandidos e retrai os grupos expandidos quando é necessário reduzir o número de elementos. Esse mecanismo utiliza uma fila *FIFO* e um limite para o número de elementos ($limiteAbertos$). Toda vez que um grupo é expandido, esse grupo é adicionado à fila. Quando a fila excede $limiteAbertos$ desenfileiram-se elementos, tomando o cuidado de garantir que um pai não será desenfileirado antes dos filhos.

4.1. Caracterização dos grupos

Para permitir que o usuário seja capaz de identificar um grupo, uma visualização é gerada para cada grupo sob o cursor. Essa visualização tem o objetivo de apresentar um resumo dos atributos que destacam esse grupo em comparação com o restante dos grupos. Para realizá-la processamos todos os atributos nos vértices de cada grupo. Calculamos para cada grupo e cada atributo a seguinte expressão:

$$TI\,(g,t) = \frac{|m \mid v \in m \iff v \in g \cap t \in v|}{|g|} \times \log\left(\frac{|G|}{|M \mid g' \in M \iff v \in g' \cap t \in v|}\right)$$

em que v é um vértice, t é um atributo, g é um grupo, $v \in g$ significa que v pertence diretamente ao grupo g ou indiretamente através de grupos que pertencem a g, $|g|$ é o número de vértices no grupo g e $|G|$ é o número de grupos. TI é uma expressão para a medida clássica da área de recuperação de informação conhecida como *TF-IDF*[Salton and Buckley 1988].

Assim, espera-se que o valor de TI seja alto para os termos que estão presentes em muitos vértices no grupo e não estão presentes, ou estão pouco presente nos demais grupos. Como alguns atributos tendem a ser exclusivos de alguns vértices por definição,

tais como nome, identificador, etc. TI simplesmente não é calculado para atributos que aparecem em apenas um vértice. De posse desse valor TI para cada par grupo-atributo podemos montar uma lista ordenada L para cada grupo dos atributos que o caracterizam segundo esse valor em ordem decrescente.

Essa visualização consiste em um gráfico de barras com os primeiros vinte atributos da lista L. Para cada barra é apresentado o nome do atributo a que ela se refere e seu comprimento é diretamente proporcional ao valor de TI. A Figura 2.c destaca essa visualização na ferramenta.

4.2. Pesquisa por atributos

Para permitir que o usuário pesquise e visualize vértices a partir de atributos textuais foi implementada uma pesquisa de texto que destaca na visualização e localiza os vértices que possuem o atributo pesquisado. Essa funcionalidade está destacada na Figura 2.a. Ao selecionar o atributo que deseja, os vértices que possuem esse atributo têm sua cor alterada. Se apenas um vértice corresponder à pesquisa a visualização também produz um efeito de *zoom*, centralizando e aproximando-se do vértice que corresponde à pesquisa.

Se mais de uma pesquisa é realizada os vértices que correspondem a uma e outra pesquisa ganham cores diferentes, bem como vértices ou grupos que correspondem a mais de uma pesquisa. Para identificar essas diferentes combinações é utilizada uma legenda listando as combinações presentes (ver Figura 2.b).

Há um limite pré-determinado de dez cores diferentes que permite o uso de cores contrastantes e evita que a visualização fique super-saturada. Quando ao realizar uma nova busca o número de pesquisas e combinações diferentes sendo visualizadas excede o limite remove-se a pesquisa mais antiga até que o limite seja respeitado.

5. Conclusão

O presente trabalho apresentou um método capaz de visualizar redes de dezenas de milhares de vértices em um ambiente web interativo capaz de se moldar à necessidade do usuário. As técnicas de busca por atributos, a caracterização dos agrupamentos e a expansão seletiva permitem uma exploração da rede em seus aspectos macro e micro, de forma fluida.

Foi realizado o estudo de diversos algoritmos de layout de grafos e agrupamentos, abordando tanto novos trabalhos quanto algoritmos clássicos. A combinação desses algoritmos gera uma variada gama de abordagens possíveis. A comparação dessas abordagens através de um bom número de métricas internas de validação torna a metodologia aqui apresentada bastante apta a tratar diversas redes diferentes. De fato, o uso dessa metodologia em diversas redes é um trabalho que deve ser apresentado futuramente.

Referências

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035.

Bastian, M., Heymann, S., and Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks.

Boutin, F. and Hascoet, M. (2004). Cluster validity indices for graph partitioning. In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on*, pages 376–381.

Brandes, U., Gaertler, M., and Wagner, D. (2003). Experiments on graph clustering algorithms. In *ESA*, pages 568–579.

Cali´nski, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):224–227.

Ding, C., He, X., Zha, H., Gu, M., and Simon, H. (2001). Spectral min-max cut for graph partitioning and data clustering. *Lawrence Berkeley National Lab. Tech. report*, 47848.

Dunn†, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95–104.

Fruchterman, T. M. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164.

Heer, M. B., Ogievetsky, V., and Jeffrey (2011). D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*.

Hu, Y. (2005). Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71.

Jacomy, M., Venturini, T., Heymann, S., and Bastian, M. (2014). Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PLoS ONE*, 9(6):e98679.

King, A. D., Przulj, N., and Jurisica, I. (2004). Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–20.

Milligan, G. W. and Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113.

Noack, A. (2007). Energy models for graph clustering. *J. Graph Algorithms Appl.*, 11(2):453–480.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.