Paralelismo em Técnicas de *Sampling* de Grafos e seus Impactos na Visualização de Grafos*

Gabriel G. Santos[1], **César A. F. De Rose**[1]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

gabriel.giordani@acad.pucrs.br, cesar.derose@pucrs.br

Resumo. *Técnicas de sampling de grafos têm como objetivo diminuir a quantidade de processamento necessário para a análise de grandes grafos. Este trabalho verifica a possibilidade de paralelismo para acelerar a produção de samples, assim como seu impacto em visualizações de grafos. Os resultados dos testes realizados até o momento indicam uma melhora na produção de samples apenas em uma abordagem de memória compartilhada.*

1. Introdução

Com o crescimento de volumes de dados, técnicas que aceleram ou simplificam o processamento destes dados tornaram-se essenciais. Técnicas de *sampling* de grafos atingem esses objetivos construindo um sub-grafo (*sample*) que consegue representar algumas características do grafo original, permitindo que análises sobre a *sample* sejam verdadeiras para o grafo original. Para construir a *sample* são selecionados vértices e arestas do grafo, com um critério previamente definido, até que se obtenha o tamanho definido.

Usualmente, técnicas de *sampling* são avaliadas pelas características topológicas que conseguem manter. No entanto, nos últimos anos, novos métodos de avaliação foram propostos, um deles sendo a capacidade de manter propriedades visuais dos grafos. Para que seja possível analisar estas propriedades, [Nguyen et al. 2017] criaram métricas de qualidade de visualização, que foram utilizadas neste trabalho. Outros trabalhos como [Wu et al. 2016] utilizam de testes com voluntários para determinar a fidelidade da visualização de uma *sample* em relação ao grafo original.

Como o objetivo de técnicas de *sampling* é facilitar o processamento de grafos, acelerar a criação dessas *samples* faz com que o processo de análise desses grafos seja realizado em tempos ainda menores. No entanto, tendo em vista que, atualmente, pesquisadores estão analisando estas técnicas de outras perspectivas, é importante que os métodos utilizados para acelerar a criação de *samples* garantam que mais propriedades, além das topológicas, sejam preservadas. Portanto, explorar versões paralelas que também mantenham as características visuais dos grafos se torna importante.

2. Técnicas de Sampling

Sampling de grafos, de acordo com [Leskovec and Faloutsos 2006], pode ser dividido em três categorias: *sampling* por vértices, *sampling* por arestas e *sampling* por caminho. Cada uma dessas categorias diz respeito a forma com que vértices e arestas são selecionadas para participarem da *sample*.

*Trabalho desenvolvido como Bolsista CAPES/BRASIL a partir da Chamada INCT - MCTI/CNPq/CAPES/FAPs nº 16/2014

Sampling por vértices seleciona vértices do grafo para participarem da *sample*. Após selecionado o número desejado de vértices são adicionadas a *sample* todas as arestas que conectam tais vértices. O algoritmo desta categoria que foi implementado se chama Random Node, que consiste em selecionar os vértices de forma aleatória.

Sampling por arestas seleciona arestas do grafo para participarem da *sample*. Após selecionado o número desejado de arestas são induzidos os vértices que se conectam a elas para participarem da *sample*. O algoritmo desta categoria que foi implementado se chama Random Edge, que consiste em selecionar as arestas de forma aleatória.

Sampling por caminho realiza um percurso pelo grafo e adiciona a *sample* todas os vértices e arestas que foram percorridos até que a *sample* tenha o tamanho desejado. O algoritmo desta categoria que foi implementado se chama Random Jump, que realiza um caminho sobre o grafo e, a cada vértice, possui uma chance de pular para um vértice aleatório e continuar caminhando.

3. Discussão

Para implementar os algoritmos foi utilizada a linguagem C++. A implementação da versão em memória compartilhada foi feita com OpenMP. A implementação da versão em memória distribuída foi feita com MPI.

Os testes realizados com as versões em memória compartilhada apresentaram um aumento de desempenho. No entanto, os testes realizados com as versões em memória distribuída não obtiveram melhoras.

Atualmente, as versões em memória distribuída não contam com uma separação do grafo, o que resulta em diversas colisões entre os processos, já que dois processos diferentes podem selecionar um mesmo nodo ou aresta para participar da *sample*. Portanto, devido ao baixo custo das operações dos algoritmos e a necessidade de tratar as colisões, a distribuição de trabalho não compensa o custo de envio de mensagens. No futuro será explorado se diferentes formas de separação do grafo irão proporcionar um ganho de desempenho.

As versões em OpenMP e sequenciais apresentaram um comportamento similar quando suas visualizações foram submetidas às métricas de visualização. Ambas apresentam uma queda na qualidade da visualização para *samples* maiores. Em ambas abordagens o algoritmo com melhores resultados foi o Random Jump.

Referências

Leskovec, J. and Faloutsos, C. (2006). Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636.

Nguyen, Q. H., Hong, S.-H., Eades, P., and Meidiana, A. (2017). Proxy graph: Visual quality metrics of big graph sampling. *IEEE transactions on visualization and computer graphics*, 23(6):1600–1611.

Wu, Y., Cao, N., Archambault, D., Shen, Q., Qu, H., and Cui, W. (2016). Evaluation of graph sampling: A visualization perspective. *IEEE transactions on visualization and computer graphics*, 23(1):401–410.