UM ALGORITMO EFICIENTE PARA CONTAR TRIÂNGULOS EM GRAFOS

Gustavo Schmid de Jesus - Marcus Ritt (orientador)

INSTITUTO DE INFORMÁTICA UFRGS

1 Motivação

Contar o número de triângulos em grafos é importante pois utilizamos este valor para obter estatísticas de grafos. Abaixo um grafo e seus triângulos:

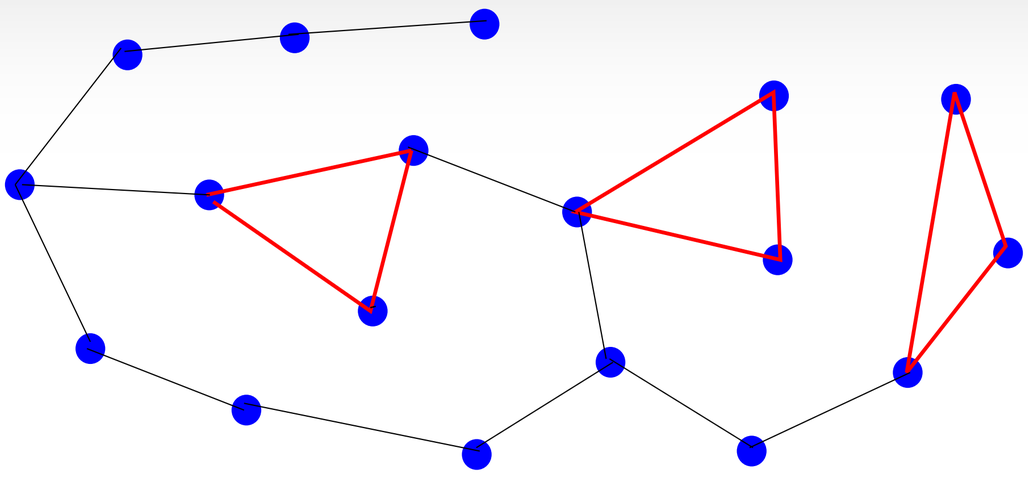

Figura 1: vértices em azul, arestas em cinza e triângulos em vermelho.

2 Aplicações

Exemplos de aplicações com triângulos:

- Calcular coeficiente de clustering;
- Calcular coeficiente de transitividade;
- Analisar grafos de redes sociais;
- Detectar subgrafos com alta relação;

3 O Algoritmo

- O algoritmo mais rápido para contar triângulos utiliza multiplicação de matrizes;

- Para grafos grandes isto se torna inviável pois a complexidade de memória é $\theta(n^2)$, sendo n o número de vértices;

- A solução é utilizar algoritmos que utilizem somente a memória mínima $\theta(m)$, sendo m o número de arestas;

- O melhor algoritmo conhecido foi proposto por Latapy (2008) e possui complexidade de tempo $O(m\sqrt{m})$;

4 O algoritmo proposto

A proposta é implementar e avaliar um algoritmo que possua:

-Complexidade de tempo $O(md)$;

-Complexidade de memória $\theta(m)$;

- O **d** em $O(md)$ representa a degeneração do grafo definido por Lick e White (1970);

- Podemos ordenar os vértices por degeneração em $O(m)$ para que cada vértice dependa de no máximo d vértices (Batagelj e Zaversnik, 2001) ;

perguntas ou sugestões: gsjesus@inf.ufrgs.br

5 Resultados

Os testes foram feitos com grafos propostos por Latapy e alguns dos maiores grafos do repositório SNAP da Universidade de Stanford.

Tabela 1: Resultados de tempo e número médio de comparações(por aresta). *Core* (algoritmo proposto)

				Tempo (s)		#Comparacões	
Instância	Degeneração	Vértices	Arestas	*Core*	*latapy*	*Core*	*latapy*
ip	274	467.273	1.744.214	0,77	0,75	97	90
actor-2002	365	383.640	15.038.083	14,10	13,50	138	132
p2p	854	6.235.399	159.870.973	383	355	375	327
web	588	39.459.925	783.027.125	264	322	39	30
amazon0601	10	403.394	3.387.388	0,32	0,40	5,10	4,90
roadNet-CA	3	1.965.206	5.533.214	0,25	0,34	0,39	0,44
web-BerkStan	201	685.230	7.600.595	1,11	1,08	14,16	11,38
web-Google	44	875.713	5.105.039	0,58	0,72	5,45	4,54
WikiTalk	131	2.394.385	5.021.410	1,83	1,68	35,63	30,93
as-skitter	111	1.696.415	11.095.298	3,67	3,29	27,63	19,32
cit-Patents	64	3.774.768	16.518.948	4,30	4,50	6,10	5,80
soc-LiveJornaul	372	4.847.571	68.993.773	27,13	28,48	40	36

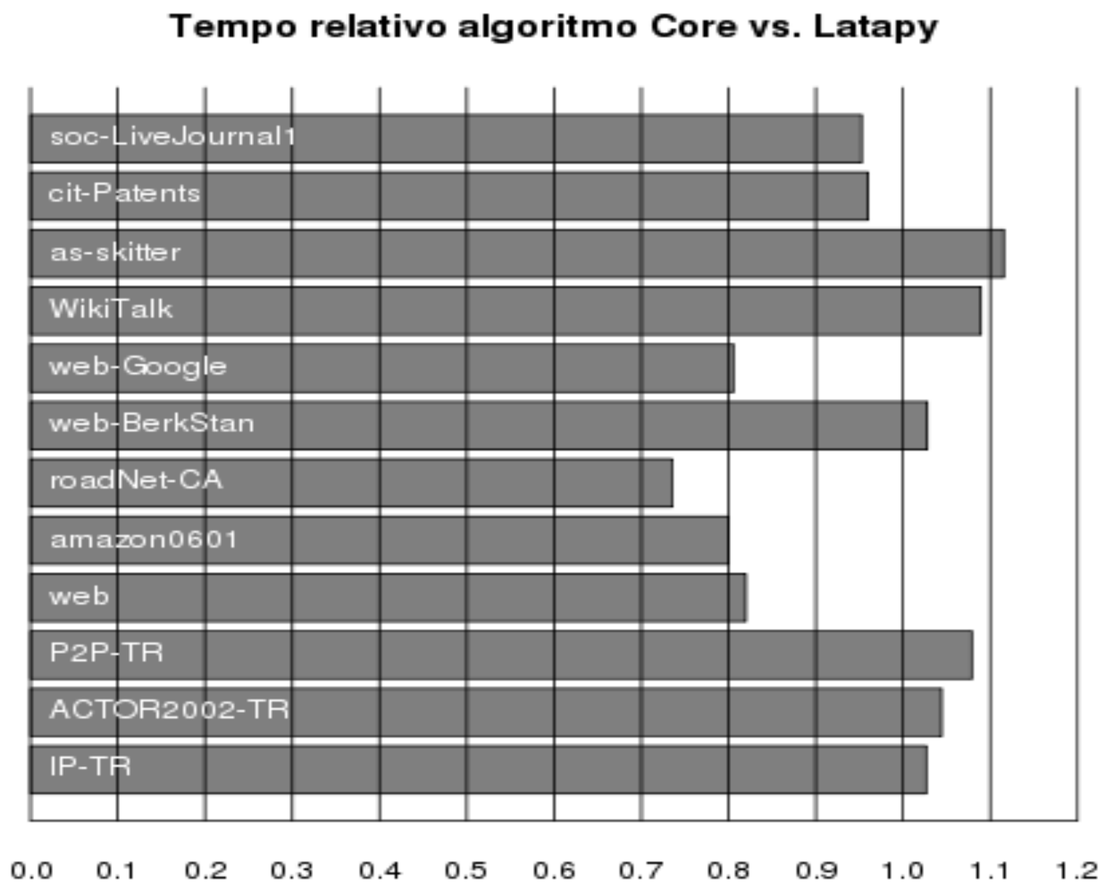

Figura 2: Relacão entre tempo (*Core* / *latapy*)

6 Conclusões

- Em 50% das instâncias o algoritmo *Core* foi superior e considerando todas instâncias, ele foi 4,5% mais rápido, em média;

- Grafos com degeneração alta em relação ao número de arestas tendem a demorar mais com o algoritmo *Core*;

- O algoritmo *Core* possui a taxa de comparações média inferior ao *latapy* apenas no grafo roadNet-CA;

7 Referências

Latapy, M (2008), ¨Main-memory triangle computations for very large (sparse (power-law)) graphs¨, presented at Theor. Comput. Sci., 2008, pp.458-473.

Lick, Don R. e White, Arthur T. (1970), "k-degenerate graphs", Canadian Journal of Mathematics 22: 1082–1096

Batagelj e Zaversnik (2001),¨An O(m) Algorithm for Cores Decomposition of Networks¨, CoRR, 2003