Busca de Caminhos entre Usuários de Redes Sociais em Tempo Real

Wladston Viana, Mirella M. Moro*

[1]Departamento de Ciência da Computação, UFMG, Belo Horizonte / MG
{wladston, mirella}@dcc.ufmg.br

Abstract. *The average distance between users in a social network is small, given the six degree theory. Nonetheless, most online social networks do not offer any way for discovering paths between its users. Tradicional algorithms can be used to find static paths in offline copies of their graphs. However, in the online scenario, the ideal is to find such paths using live data, which is a hard task given the access constraints imposed by the social networks. In this work, we introduce a novel, real time path finding algorithm for social networks, called CUTE. It employs a heuristic that considers the geographical distance between users. In our experimental evaluation over Twitter, CUTE finds short paths for all types of users, while expanding less than 40 nodes.*

Resumo. *A distância média entre nós em uma rede social é pequena, considerando a teoria dos seis graus de separação. No entanto, as redes sociais online não oferecem formas de descobrir caminhos entre seus usuários. Algoritmos tradicionais são aplicáveis para cópias offline de seus grafos. Contudo, na Web, o ideal é encontrar caminhos utilizando dados online, o que é uma tarefa difícil considerando as limitações de acesso impostas pelas redes sociais. Neste trabalho, introduzimos um algoritmo para encontrar caminhos em tempo real, chamado CUTE. Ele utiliza uma heurística que considera a distância geográfica entre os usuários. Na nossa avaliação experimental com o Twitter, o CUTE encontra caminhos curtos entre usuários, expandindo menos de 40 nós.*

1. Introdução

O primeiro experimento que suporta a teoria de que as redes sociais têm a estrutura do tipo *small world* foi realizada por Travers and Milgram (1969). Os autores mostraram que grafos de redes sociais possuem uma pequena distância média entre os seus nós e um alto grau de agrupamento. Especificamente, eles descobriram em experimentos que os habitantes dos Estados Unidos estavam separados em média por 5,2 graus. Com o advento das redes sociais online, mais estudos foram realizados confirmando essa hipótese, trazendo a distância média para 4,67 graus no Twitter[1] e 3,74 graus no Facebook [Ugander et al. 2011]. Apesar disso, não existem ferramentas para encontrar caminhos entre usuários genéricos dessas redes sociais em tempo real.

Encontrar um caminho entre dois nós genéricos de um grafo é um problema clássico. Diversos algoritmos foram desenvolvidos para resolver esse problema, dentre eles o famoso algoritmo de Dijkstra [Dijkstra 1959]. Existem também algoritmos

*Trabalho parcialmente financiado por CNPq e FAPEMIG.

[1]Six Degrees of Separation, Twitter Style: http://www.sysomos.com/insidetwitter/sixdegrees

heurísticos, como o Best-First-Search [Pearl 1984] e o A* [Hart et al. 1968]. Esses algoritmos, no entanto, realizam uma busca extensa no grafo. Para utilizá-los, é necessário ler uma parte considerável do grafo, caso os nós para os quais se deseja encontrar um caminho estejam bem separados. No entanto, as redes sociais de maior adoção global, tais como o Twitter e o Facebook, não disponibilizam acesso amplo e irrestrito aos seus grafos. Tal fato inviabiliza a utilização de algoritmos e técnicas tradicionais para encontrar caminhos em tempo real nessas redes.

Especificamente, o Twitter é uma rede social de destaque, pois possui mais de 140 milhões[2] de usuários espalhados pelo mundo, entre eles muitas pessoas famosas, como o empresário Mark Zuckerberg, o senador Cristovam Buarque, o cantor Nando Reis, entre outros[3]. Uma característica importante é a disponibilização do acesso ao seu grafo através de uma API. Porém, as chamadas à API do Twitter possuem uma latência considerável, e há ainda uma restrição em relação ao número de requisições que podem ser realizadas, de modo que expadir um nó é uma operação muito cara. Em tempo hábil, é possível expandir apenas algumas dezenas de nós, o que inviabiliza os algoritmos de busca de caminhos conhecidos. Considerando um grafo em que cada nó tem 100 arestas saintes, para explorar 4 níveis de profundidade seria necessário expandir $100^3 = 10^6$ nós utilizando o algoritmo de Dijkstra.

Nesse trabalho apresentamos um algoritmo para encontrar caminhos curtos entre usuários de redes sociais geolocalizadas de acesso restrito. As contribuições desse trabalho podem ser resumidas da seguinte maneira.

- O Algoritmo CUTE (Seção 3), um algoritmo de busca bidirecional baseado no A* que utiliza uma heurística não admissível (ou seja, não se garante encontrar o caminho mínimo) para encontrar um caminho curto entre dois usuários de uma rede social. Tal algoritmo apresenta duas novidades científicas: ele minimiza o número de acessos feitos ao grafo e permite que a busca seja feita em tempo real.
- Uma heurística (Seção 4) a qual permite que caminhos curtos, próximos à separação média entre os usuários, sejam encontrados através do algoritmo. Em resumo, essa heurística atribui uma penalidade a cada nó, baseado: no grau de entrada e saída de cada nó, e na distância geográfica até o nó-alvo.
- Uma análise experimental (Seção 5) que utiliza dados reais de uma rede social (Twitter) mostra que o CUTE encontra caminhos entre quaisquer dois usuários da rede (incluindo usuários genéricos, usuários famosos e usuários geograficamente muito afastados). Esses experimentos enfatizam as vantagens do CUTE, com o qual a busca é realizada de maneira otimizada, expandindo tipicamente menos de 40 nós, um ganho de desempenho de cerca de 10^5 vezes se comparado com o algoritmo ótimo de Dijkstra.

2. Trabalhos Relacionados

Os trabalhos existentes para encontrar caminhos em grafos de redes sociais são em sua maioria baseados em *landmarking* [Cohen et al. 2002, Gubichev et al. 2010, Potamias et al. 2009, Zhao et al. 2010]. Essas técnicas requerem que seja feito um pré-processamento intermediário no grafo, em parte ou em toda a sua extenção. Tal pré-processamento inviabiliza sua aplicação na maior parte das redes sociais online, pois é

[2]Twitter turns six (março 2012): http://blog.twitter.com/2012/03/twitter-turns-six.html

[3]Celebrity Twitter Users: http://wefollow.com/twitter/celebrity

impossível ter acesso direto aos grafos das mesmas (a única maneira de realizar tal tarefa seria sendo o "dono" da rede).

Dentre as diversas redes sociais, provavelmente a mais utilizada em pesquisas científicas é o Twitter. Um dos principais motivos é o fácil acesso a seus dados através de sua API. O assunto dessas pesquisas é muito variado, incluindo estudos sobre: a influência dos usuários [Bigonha et al. 2011], a topografia do grafo do Twitter [Takhteyev et al. 2012], mecanismos de coordenação sem organização central durante catástrofes naturais [Sarcevic et al. 2012], e até como o humor do Twitter influencia a bolsa de valores [Bollen et al. 2011].

Nosso trabalho avança o estado-da-arte pois não existe atualmente nenhum estudo sobre a busca de caminhos entre dois nós genéricos de um grafo com acesso restrito aos dados (ou seja, sem acesso aos dados dos usuários e da topologia da rede). Além disso, o algoritmo proposto realiza uma expansão mínima de nós para encontrar um caminho curto, necessitando apenas de um pré-processamento simples o suficiente para ser executado em tempo real juntamente com a busca.

Diferentemente dos trabalhos relacionados, para reduzir consideravelmente a complexidade da busca, o algoritmo proposto utiliza dados geográficos e explora a característica de agrupamento geolocalizado das redes sociais. Finalmente, para averiguar o funcionamento e o desempenho do algoritmo proposto, a avaliação experimental realizada segue a tendência apresentada e utiliza dados reais do Twitter.

3. CUTE - Caminhos entre Usuários em Tempo rEal

Nesse trabalho, apresentamos o CUTE – um algoritmo para encontrar **C**aminhos entre **U**suários em **T**empo r**E**al. O CUTE é um algoritmo de busca que encontra um caminho curto entre dois usuários (vértices) genéricos de uma rede social de acesso restrito. O acesso restrito é definido pela própria rede social e varia de uma rede para outra. É comum, por exemplo, existir um limite absoluto de número de acessos por hora. Além disso, existe a latência de acesso intrínseca à maioria das redes online (ou seja, não é de interesse comercial suportar o acesso imediato e ilimitado).

Para contrapor o acesso restrito das redes sociais online, o CUTE foi projetado para minimizar o acesso aos dados da rede social. Desse modo, o algoritmo utiliza busca bidirecional e emprega uma heurística que considera primariamente a distância geográfica entre os usuários.

Esta seção segue com a visão geral do algoritmo (seção 3.1), sua descrição completa (seção 3.2) e um exemplo de sua execução (seção 3.3). Finalmente, para facilitar o entendimento do algoritmo CUTE, a heurística proposta é apresentada na próxima seção.

3.1. Visão Geral

A busca por um caminho entre dois usuários é definida da seguinte maneira.

Definição 1. *Dado um grafo $G = \{V, E\}$, onde os vértices (ou nós) $v \in V$ representam os usuários de uma rede social, e as arestas (ou arcos) $e \in E$ representam as conexões (relacionamentos) entre cada par de usuários $\langle u, v \rangle$, para $u \neq v$. Para quaisquer vértices u e $v \in V$, denomidados vértices origem e destino, o algoritmo CUTE é definido como uma função δ dada pela Equação 1.*

$$\delta(u,v) = \begin{cases} u \xrightarrow{\text{p}} v & \text{se existe um caminho } p \text{ de } u \text{ a } v \\ \\ \emptyset & \text{caso contrário} \end{cases} \quad (1)$$

Para retornar o caminho p de u a v, o CUTE é derivado do A* bidirecional, utilizando uma heurística não admissível. Nesse caso, para a busca partindo de u até v, é utilizada uma função heurística $h(n_x)$ que determina em qual ordem o algoritmo visitará os vértices. Para a busca inversa (de v a u), são visitados apenas vértices geograficamente próximos ao nó-alvo v.

3.2. O Algoritmo CUTE

Conforme explicado anteriormente, o algoritmo CUTE realiza uma busca bidirecional com heurísticas complementares para cada direção da busca. Primeiro, é realizada a busca a partir do nó-alvo. Em seguida é realizada a busca a partir do nó-origem até que se encontre um vértice já visitado pela busca anterior.

O Algoritmo 1 apresenta esse processo de busca. Inicialmente, são iniciadas as estruturas de dados. A lista *closedset* contém os vértices já visitados e é iniciada como uma lista vazia (linha 1). A pilha *openset* contém vértices ordenados pelos próximos candidatos para exploração. Inicialmente o *openset* contém apenas o nó-origem (linha 2). A lista *goalset* contém vértices conectados ao nó-alvo, construído na linha 2 através da função GeraGoalSet, a ser detalhada mais adiante.

Algoritmo 1 CUTE - main()

Entrada: nó-origem v_o, nó-destino v_d
Saída: Uma lista de vértices C interligando v_o até v_d

```
 1: C ← ∅ // closedset: lista de vértices já visitados
 2: O ← {v_o} // openset: pilha de vértices a serem visitados
 3: G ← GeraGoalSet(v_d) // goalset: lista de vértices conectados ao nó-destino
 4: while not O.empty() do
 5:    current ← O.pop()
 6:    C.add(current)
 7:    for all node ∈ current.getNeighbours() do
 8:       if node ∉ C then
 9:          if node ∈ G then
10:             return rebuildPath(node)
11:          end if
12:          O.push(node)
13:       end if
14:    end for
15: end while
```

Com as estruturas de dados inicializadas, executa-se o laço principal do CUTE. Em cada iteração, retira-se o elemento do *openset* com o menor valor da nova heurística (Seção 4) na linha 5, e insere-se o mesmo no *closedset* (linha 6). Através das diferentes iterações, os vértices adjacentes que não pertencem ao *closedset* são adicionados ao *openset* (linha 12). A função *rebuildPath* é responsável por reconstruir o caminho do vértice

Algoritmo 2 CUTE - GeraGoalSet()

Entrada: Um vértice v_x pertencente ao grafo
Saída: Uma lista de vértices conectados ao vértice de entrada

1: $G \leftarrow \emptyset$
2: $H \leftarrow \{v_x\}$
3: **while** $G.length() < 1000$ **do**
4: $\quad current \leftarrow H.pop()$
5: $\quad G.add(current)$
6: $\quad$ **for all** $node \in current.getNeighbours()$ **do**
7: $\quad\quad$ **if** $distancia(node, v_x) < 1$ **and** $node \notin G$ **and** $node \notin H$ **then**
8: $\quad\quad\quad H.push(node)$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end while**
12: **return** $H + G$

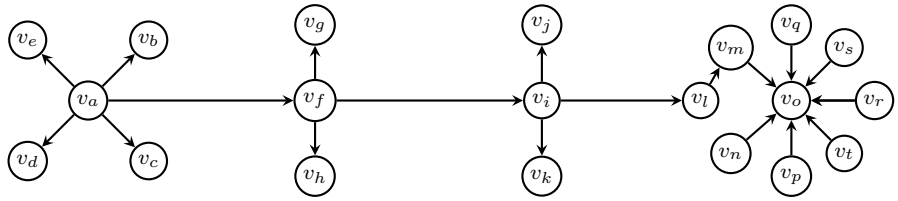


Figura 1. Exemplo de grafo direcionado representando uma pequena rede social.

recebido como seu parâmetro até o nó-origem e o nó-destino. A busca então termina quando o *openset* e o *goalset* possuírem algum elemento em comum (linha 10). É esperada uma rápida convergência da busca. No pior caso, o algoritmo continuará até explorar todo o componente conectado, se não encontrar um caminho válido.

O Algoritmo 2 descreve o processo para gerar o *goalset*, que é uma lista de vértices sabidamente conectados ao nó-alvo. São inseridos até 1000 vértices no *goalset*, a partir do nó-alvo (linha 3). Nessa busca somente são expandidos vértices que estão geograficamente localizados a menos de 1 Km do alvo (linha 7). Uma expansão é representada pela operação *getNeighbours* do Algoritmo 2. Foi escolhido o valor de 1 Km para escolher vértices localizados no mesmo agrupamento (*cluster*). O *goalset* é construido desse modo pois a busca subsequente (a partir do nó-origem) será digirida para esse agrupamento (*cluster*). Se tal busca posterior alcançar qualquer vértice pertencente ao *goalset*, é possível fechar um caminho do nó-origem até o nó-alvo.

3.3. Exemplo

A Figura 1 apresenta um grafo para exemplificar um processo de busca do vértice v_a ao vértice v_o. Nesse exemplo, não são calculados valores heurísticos para os vértices, pois são definidos a partir das equações apresentadas na Seção 4.

Primeiramente, é realizada a busca reversa partindo de v_o, adicionando os vértices $v_m, v_n, v_p, v_q, v_r, v_s, v_t$ e v_l, vértices geograficamente próximos de v_o, para o *goalset*. O *goalset* resultante para esse exemplo é apresentado na Figura 2. Em seguida, é iniciado o

v_o	v_m	v_q	v_r	v_s	v_t	v_n	v_p	v_l

Figura 2. *goalset* após a execução do Algoritmo 2 para o vértice v_o.

v_a	v_f	v_i	v_l	v_m	v_o	*caminho calculado*

v_a	v_f	v_i	*closedset*

v_b	v_c	v_d	v_e	v_g	v_h	v_j	v_k	*openset*

Figura 3. *openset* e *closedset* ao final da execução do exemplo.

loop principal. Move-se v_a do *openset* para o *closedset* (linha 7 e 8 do algoritmo 1). Os vértices conectados v_b, v_c, v_d, v_e e v_f são inseridos no *openset* (linha 14 do algoritmo 1). Em seguida, é movido o vértice com o menor valor da função heurística (esse caso v_q) do *openset* para o *closedset*. Os vértices v_g, v_h e v_i são adicionados ao *openset*. Em outra interação, move-se v_i do *openset* para o *closedset*, e adiciona-se v_j e v_k para o *openset*. Ao se deparar com o v_l, a busca termina, pois v_l está presente no *goalset* e já é possível fechar o caminho entre v_a e v_o. O conteúdo das estruturas de dados nesse ponto, juntamente com o caminho calculado, são apresentados na Figura 3.

4. Heurística para Diminuir Espaço de Busca

4.1. Requisitos

A principal novidade do CUTE, que permite fazer a busca em tempo real, é utilizar uma heurística baseada na geolocalização dos usuários para reduzir o espaço de busca. Em outras palavras, a heurística procura conduzir a busca até o grupo no qual nó alvo está melhor conectado. Para que tal heurística possa ser usada, é necessário que se conheça a localização dos usuários da rede social. É necessário também que o grafo da rede forme grupos (*clusters*) geolocalizados, pois essa é a propriedade explorada pela heurística.

Considerando as grandes redes sociais online, tais requisitos são satisfeitos através da geolocalização da string de localidade informada pelo usuário no seu perfil ou em suas postagens. Mesmo que alguns usuários deixem em branco ou informem erroneamente a sua localização, é possível assumir que a informação contida no seu perfil e nas suas postagens é suficiente para determinar a localização aproximada da maioria dos usuários [Takhteyev et al. 2012]. Quanto à existência dos grupos geolocalizados, pelo menos para a rede do Twitter (utilizada na avaliação experimental), o estudo publicado em [Takhteyev et al. 2012] mostra que seu grafo contém grupos densos de usuários que habitam em uma mesma cidade.

4.2. Função Heurística

Considerando que as redes sociais globais possuem milhões de vértices e que se deseja minimizar o número de acessos ao grafo (lembrando que é necesário minimizar o acesso em função da latência de acesso e do número restrito de acessos possíveis por hora): é necessário utilizar uma heurística para *reduzir o espaço de busca*. A heurística proposta

é apresentada na Equação 2. No processo de busca, é escolhido o vértice com o menor valor de $f(x)$, dentre os vértices disponíveis.

$$f(x) = g(x) + h(x) \tag{2}$$

onde a função $g(x)$ é o número de passos (*hops*) entre o vértice x e o nó-origem, e a função $h(x)$ é definida pela Equação 3.

$$h(x) = h_d(x) + h_{out}(x) + h_{in}(x) \tag{3}$$

onde $h_d(x)$ é o componente relacionado à distância geográfica (seção 4.3), $h_{out}(x)$ é o componente relacionado ao grau de saída do vértice (seção 4.4) e $h_{in}(x)$ é o componente relacionado ao grau de entrada do vértice (seção 4.5).

4.3. Distância Geográfica

Para reduzir o espaço de busca (minimizando o acesso ao grafo), é utilizada uma função heurística baseada na distância geográfica, uma vez que as redes sociais formam clusters geolocalizados [Takhteyev et al. 2012]. Note que, uma busca realizada entre usuários geograficamente próximos tem uma complexidade reduzida, se comparada à busca no grafo completo [Travers and Milgram 1969].

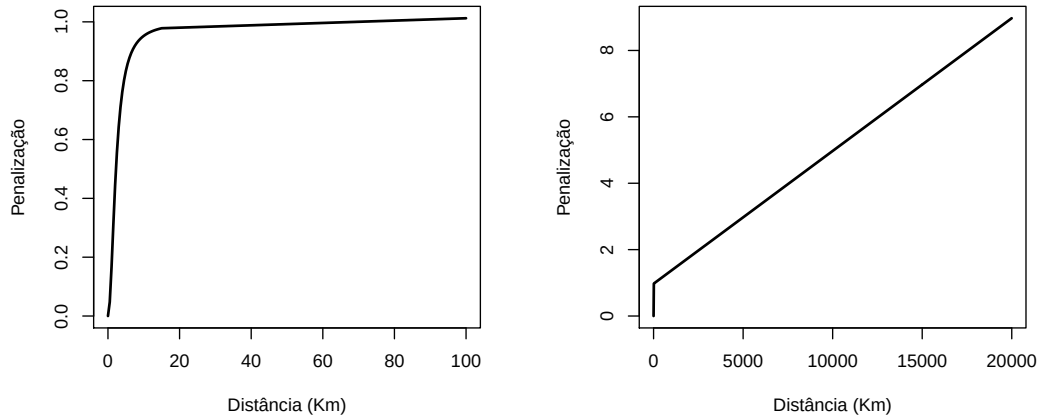
Para atingir tal objetivo, a função h_d aplica uma grande penalidade para os vértices a uma distância geográfica maior, e uma penalidade mínima para vértices a uma distância geográfica mais próxima. Especificamente, a penalidade pode chegar a 9 para os pontos mais distantes, e fica entre 0 e 1 para distâncias menores de 350 Km. A função foi montada deste modo, pois foi percebido durante a avaliação experimental que usuários com uma distância de 15 Km estão no mesmo agrupamento, e portanto devem receber uma penalização mínima. Tais parâmetros foram formulados na Equação 4.

$$h_d(x) = \begin{cases} \dfrac{x^2}{x^2+5} & \text{se } x \leq 15 \\ \\ \dfrac{45}{46} + 0.0004(x-15) & \text{se } x > 15 \end{cases} \tag{4}$$

Os parâmetros das Equações 4, 5 e 6 foram obtidos através de uma série de pré-processamento, onde foram testados várias curvas definidas por parâmetros de valores diferentes. As equações presentes neste trabalho possuem os melhores parâmetros encontrados que maximizaram a convergência da busca. A Figura 4 ilustra a Equação 4 em dois intervalos: 4(a) com o comportamento da função para valores pequenos, e 4(b) com o comportamento da função para todo o seu domínio, considerando a distância máxima entre dois pontos na Terra.

4.4. Grau de Saída

Além de considerar a distância geográfica, O CUTE utiliza também o grau de saída para minimizar ainda mais o espaço de busca. Para tanto, são penalizados os usuários que tenham poucas arestas saintes, pois deseja-se escolher vértices com maior conectividade.



(a) distâncias entre 0 e 100 Km (b) distâncias entre 0 e 20000 Km

Figura 4. Modelo de penalização para distância geográfica entre usuários.

Tal escolha é justificada pelo fato de que vértices mais conectados tendem a chegar no destino mais rapidamente. Desse modo, aplica-se uma penalidade de até 1 para usuários que não possuírem um número de arestas saintes entre 40 e 300.

Aplica-se também uma penalidade para usuários que possuam muitas arestas saintes, pois presume-se que na vida real as pessoas têm um número pequeno de amizades, uma vez que é impraticável acompanhar a atualização de muitos usuários. Além disso, trabalhos recentes mostram que usuários que têm muitas arestas saintes possuem grande probabilidade de serem classificados como *spammers* ou robôs [Bigonha et al. 2011]. A função é expressa pela Equação 5 e ilustrada na Figura 5.

$$h_{out}(x) = \begin{cases} -0.025 * x + 1 & \text{se } x \leq 40 \\ 0 & \text{se } 40 < x \leq 300 \\ \exp\left(\frac{x - 300}{500}\right) - 1 & \text{se } x > 300 \end{cases} \tag{5}$$

4.5. Grau de Entrada

Igualmente, são penalizados usuários com poucas arestas entrantes, pois deseja-se escolher vértices com maior conectividade. Aplica-se uma penalidade variando de 0 a 1 para vértices com menos de 700 arestas entrantes. A função é expressa pela Equação 6 e ilustrada pela Figura 6.

$$h_{in}(x) = \exp(-0.007 * x) \tag{6}$$

5. Avaliação Experimental

Para realizar a avaliação experimental do algoritmo CUTE, foi escolhida a rede social de microblogs Twitter. Essa rede social se destaca, pois sua política de acesso permite aos

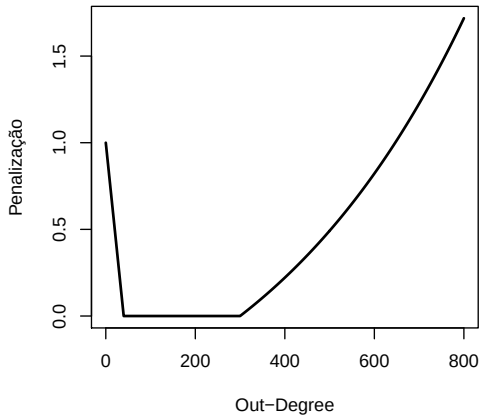


Figura 5. Modelo de penalização para arestas saintes do vértice.

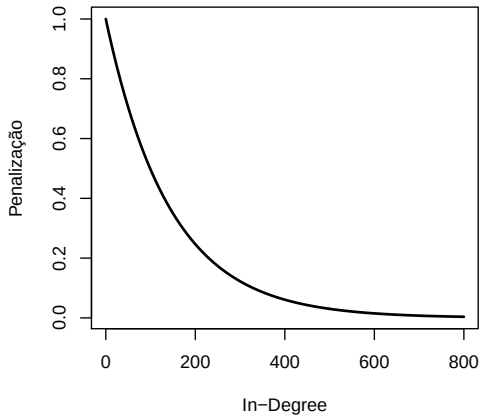


Figura 6. Modelo de penalização para arestas entrantes do vértice

seus usuários acessar a maioria dos dados da rede. Essa análise é difícil (senão impossível) de ser executada em outras redes sociais (tais como Facebook) que exigem autorização dos usuários para ter acesso às suas informações. Por outro lado, a API do Twitter é simples de usar e é baseada em REST (*Representational State Transfer*), outro fator vantajoso em relação às demais redes sociais (como o Orkut).

São apresentadas rotas entre alguns usuários escolhidos do Twitter. Escolhemos quatro membros do comitê de programa: Fabrício Benevenutto (*@fbnevenutto*), Luciana Nedel (*@lunedel*), Geraldo Xexeo (*@xexeo*) e Jonice Oliveira (*@joniceoliveira*). Para tornar os experimentos mais reais e mostrar buscas possíveis de acontecer, escolhemos usuários famosos e residentes em lugares diferentes. Especimente, escolhemos um usuário famoso — o jornalista Willian Bonner (*@realwbonner*), um cientista residente nos Estados Unidos (*@cookp*) e um usuário com muitos seguidores residente na Rússia (*@tsyplukhin*).

A Figura 7 apresenta alguns resultados obtidos utilizando o CUTE. Para analisar

o desempenho do CUTE em relação a essas buscas, o número de vértices expandidos para cada busca é apresentado na Tabela 1. Nesse caso, o tempo para realizar tais buscas é irrelevante, pois depende do tempo de resposta da API do Twitter e do serviço de Geolocalização, ambos extremamente variáveis. Então, a fim de curiosidade, considerando a cache local completamente vazia, essas buscas demoraram em média 5 minutos para completar.

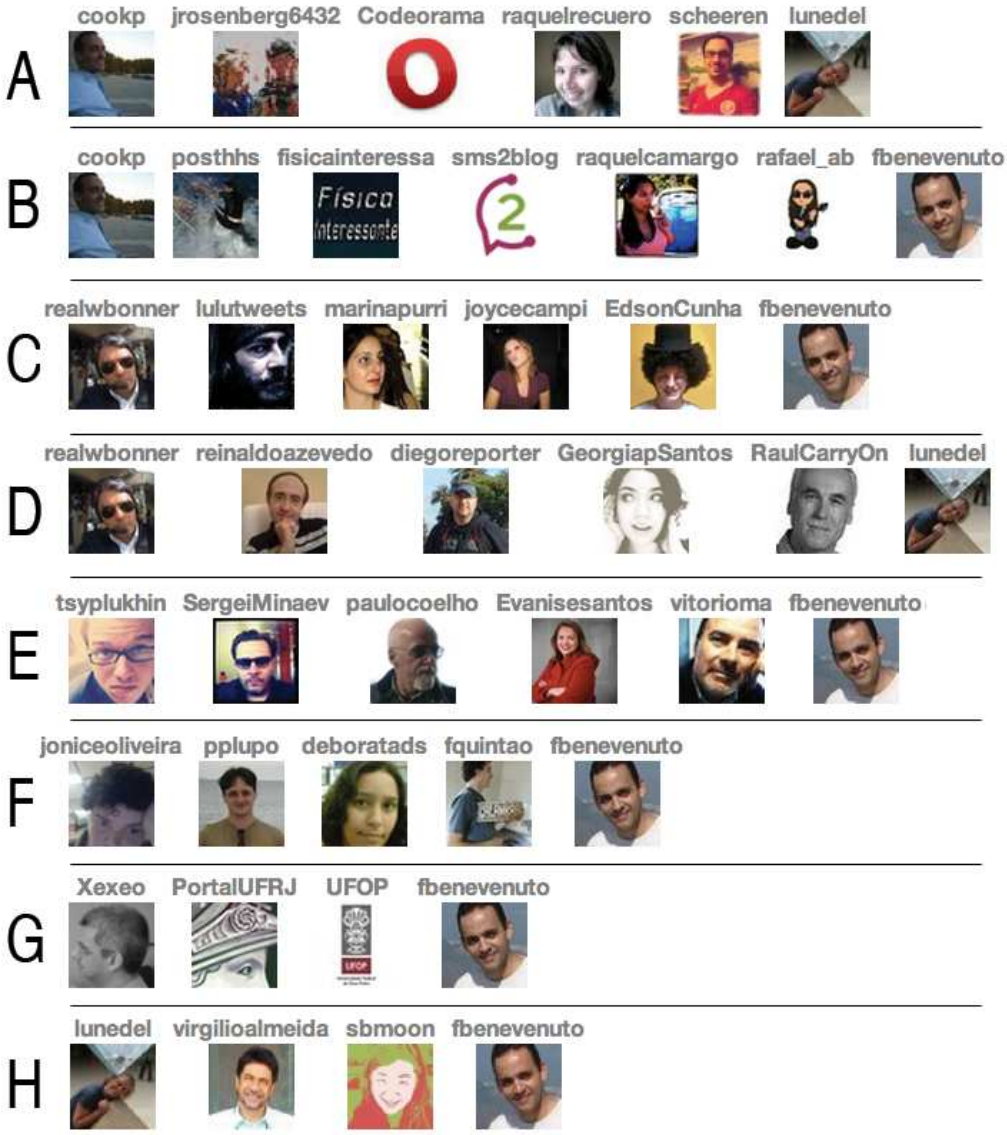


Figura 7. Conjunto de rotas obtidas para a avaliação experimental.

6. Conclusão

Este artigo apresentou um algoritmo para encontrar caminhos em tempo real para grafos do tipo *small world* grandes e de acesso restrito, minimizando o acesso ao grafo. Foi utilizada uma heurística que considera primariamente a distância geográfica entre os vértices,

Tabela 1. Número de vértices explorados para as rotas da Figura 7.

Caminho	Vértices explorados		
	partindo do alvo	partindo da origem	Total
A	8	10	18
B	17	12	29
C	18	12	30
D	4	10	14
E	12	12	24
F	11	12	23
G	2	12	14
H	2	12	14

a fim de explorar o agrupamento do grafo. Essa heurística é responsável por diminuir consideravelmente o espaço de busca de centenas de milhares de vértices para apenas algumas dezenas.

Na avaliação experimental, foram encontrados caminhos com menos de 6 graus de separação, realizando uma leitura de menos de 40 vértices, para um grafo com mais de 100 milhões de vértices, o que permite que a busca seja realizada em tempo real. Algoritmos tradicionais encontrados na literatura, em contraposição, expandem milhares de vértices, inviabilizando uma pesquisa em tempo real, uma vez que expandir um vértice é uma operação muito cara.

Como trabalhos futuros, planeja-se realizar experimentos em um tipo diferente de rede social. Especificamente, pretende-se considerar uma rede social acadêmica, como a apresentada no portal CiênciaBrasil[4], e verificar os desempenho do algoritmo utilizando a heurística atual e outras próprias para redes acadêmicas.

Referências

Bigonha, C., Cardoso, T. N. C., Moro, M. M., Gonçalves, M. A., and Almeida, V. A. F. (2011). Sentiment-based influence detection on twitter. *J. Braz. Comput. Soc.*, online first.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1 – 8.

Cohen, E., Halperin, E., Kaplan, H., and Zwick, U. (2002). Reachability and distance queries via 2-hop labels. In *Procs. of ACM-SIAM Symposium on Discrete Algorithms*, pages 937–946, San Francisco, California.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Gubichev, A., Bedathur, S., Seufert, S., and Weikum, G. (2010). Fast and accurate estimation of shortest paths in large graphs. In *Procs. of ACM Int'l Conf. on Information and Knowledge Management*, pages 499–508, Toronto, Canada.

[4]CiênciaBrasil: http://www.pbct.inweb.org.br

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost path. *IEEE Transactions on Systems Science and Cybernetics*, 2(2):100–107.

Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Potamias, M., Bonchi, F., Castillo, C., and Gionis, A. (2009). Fast shortest path distance estimation in large networks. In *Procs. of ACM Conf. on Information and Knowledge Management*, pages 867–876, Hong Kong, China.

Sarcevic, A., Palen, L., White, J., Starbird, K., Bagdouri, M., and Anderson, K. (2012). "beacons of hope" in decentralized coordination: learning from on-the-ground medical twitterers during the 2010 haiti earthquake. In *Procs. of ACM Conference on Computer Supported Cooperative Work*, pages 47–56, Seattle, Washington, USA.

Takhteyev, Y., Gruzd, A., and Wellman, B. (2012). Geography of twitter networks. *Social Networks*, 34(1):73–81.

Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, 32(4):425–443.

Ugander, J., Karrer, B., Backstrom, L., and Marlow, C. (2011). The anatomy of the facebook social graph. *CoRR*, abs/1111.4503.

Zhao, X., Sala, A., Wilson, C., Zheng, H., and Zhao, B. Y. (2010). Orion: shortest path estimation for large social graphs. In *Procs. of Conference on Online Social Networks*, page 9, Boston, MA, USA.