Busca de Caminhos em Redes Sociais Geolocalizadas em Tempo Real

Wladston Viana, Mirella M. Moro, Fabrício Benevenuto*

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

Abstract. *This paper tackles the problem of finding a path between two users in a social network. Classical solutions involve searching for paths in an entire snapshot of the social graph, which is impractical for large scale social networks (such as Twitter and Facebook) because they do not provide broad and unrestricted access to their social graph. Here, we propose a novel real time path finding algorithm for social networks, called CUTE, and an online tool that uses it, called FriendRouter. In our experimental evaluation over Twitter, the algorithm finds short paths for a number of users by expanding the friend list of no more than 40 users.*

Resumo. *Este artigo foca no problema de encontrar um caminho entre dois usuários de uma rede social. Soluções clássicas realizam a busca usando a cópia estática do grafo inteiro da rede; o que é impraticável para redes sociais grandes (como Twitter e Facebook) uma vez que elas não fornecem acesso amplo e irrestrito aos seus dados. Nesse trabalho, propomos um algoritmo para buscar caminhos em redes sociais em tempo real, chamado CUTE, e uma ferramenta online que o utiliza, chamada FriendRouter. Na nossa avaliação experimental utilizando o Twitter, o algoritmo encontra caminhos curtos para diversos pares de usuários através da expansão de menos de 40 listas de amigos.*

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 Djikstra [1959]. Existem também algoritmos heurísticos, como o Best-First-Search [Pearl 1984] e o A* [Hart et al. 1968]. Embora esses algoritmos sejam muito úteis em diversos cenários, eles são impraticáveis para busca em redes

*Trabalho parcialmente financiado por CNPq e FAPEMIG.

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

sociais online de larga escala (como o Twitter e o Facebook), uma vez que não é disponibilizado acesso amplo e irrestrito aos grafos dessas redes.

Todavia, é possível acessar os dados através de uma API (*Application Programming Interface*), geralmente com uma alta latência (uma vez que os dados estão vindo através da Internet), e uma restrição para o número de requisições à API que podem ser realizadas por hora. Obter a lista de amigos/seguidores (ao procurar por caminhos) se torna portanto a operação mais cara. Em tempo razoável, é possível expandir algumas dezenas de listas de amigos/seguidores, inviabilizando o uso de algoritmos conhecidos para busca em grafos. Por exemplo, considerando uma rede social na qual 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.
- 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.
- A implementação do algoritmo CUTE em uma ferramenta online, chamada FriendRouter (Seção 6), para a rede social Twitter.

2. Trabalhos Relacionados

Os trabalhos existentes para encontrar caminhos em grafos de redes sociais são em sua maioria baseados em *landmarking*, como por exemplo [Gubichev et al. 2010] e [Potamias et al. 2009]. Essas técnicas requerem que seja feito um pré-processamento intermediário no grafo, em parte ou em toda a sua extensão. Tal pré-processamento inviabiliza sua aplicação na maior parte das redes sociais online, pois é impossível ter acesso direto aos grafos das mesmas (a única maneira de realizar tal tarefa seria através do administrador da rede social).

Outra soluções propõem heurísticas para acelerar a busca, algumas através de agregação de informações adicionais ao grafo [Potamias et al. 2009]. Particularmente, [Thomsen et al. 2012] foca em encontrar caminhos em sistemas que utilizam geolocalização. O método utiliza estatísticas a partir de listas de registro de acessos e um mecanismo de cache complexo para melhorar o desempenho. Contudo, esses métodos

também dependem de uma imagem completa da rede, e portanto não são aplicáveis para o propósito desse trabalho.

Finalmente, a proximidade geográfica entre indivíduos socialmente ligados foi utilizada para explicar diversos fenômenos na sociedade, como a proliferação de um tipo específico de indústria em uma certa região e o índice de empregabilidade de indivíduos [Topa 2001]. No cenário da Internet, existem evidências quantitativas de maior proximidade geográfica entre usuários socialmente ligados em diferentes redes, como o Orkut [Benevenuto et al 2012], Twitter [Rodrigues et al 2011], Youtube [Brodersen et al. 2012] e Foursquare [Noulas et al 2011]. Esses trabalhos não somente inspiraram nosso método, mas também dão suporte à contribuição principal, que é a utilização de uma heurística baseada em geolocalização para diminuir o espaço de busca.

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. A busca por um caminho entre dois usuários de uma rede social pode ser descrita conforme a Definição 1.

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.*

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

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.

A descrição completa do algoritmo utilizado foi publicada em [Viana and Moro 2012]. Em resumo, são utilizadas três estruturas de dados: uma lista (*closedset*) para os vértices já visitados; uma pilha (*openset*) para os vértices

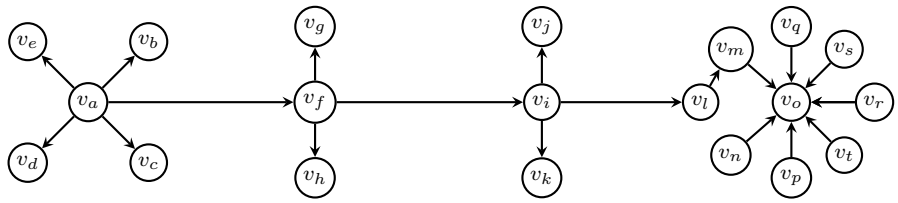

Figura 1. Exemplo de grafo direcionado representando uma pequena rede social.

v_o	v_m	v_q	v_r	v_s	v_t	v_n	v_p	v_l

Figura 2. *goalset* para o vértice v_o.

ordenados pelos próximos candidatos para exploração (iniciada com o nó de origem); e uma lista (*goalset*) para os vértices conectados ao nó-alvo. Após inicializar as estruturas de dados, executa-se o laço principal do CUTE: em cada iteração, retira-se o elemento da *openset* com o menor valor da nova heurística e armazena-o na *closedset*; os vértices adjacentes que não pertencem à *closedset* são adicionados à *openset*; então o algoritmo reconstrói o caminho do vértice recebido até o nó-origem e o nó-destino; finalmente, a busca termina quando *openset* e *goalset* possuem um elemento comum. 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.

É importante notar que *goalset* é uma lista de vértices sabidamente conectados ao nó-alvo. São inseridos até 1000 vértices na *goalset*, a partir do nó-alvo. Nessa busca somente são expandidos vértices que estão geograficamente localizados a menos de 1 Km do alvo (proximidade geográfica). Foi escolhido o valor de 1 Km para escolher vértices localizados no mesmo agrupamento (*cluster*). A *goalset* é construída desse modo pois a busca subsequente (a partir do nó-origem) será dirigida para esse agrupamento (*cluster*). Se tal busca posterior alcançar qualquer vértice pertencente à *goalset*, é possível fechar um caminho do nó-origem até o nó-alvo.

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 *loop* principal. Move-se v_a do *openset* para o *closedset*. Os vértices conectados v_b, v_c, v_d, v_e e v_f são inseridos no *openset*. 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.

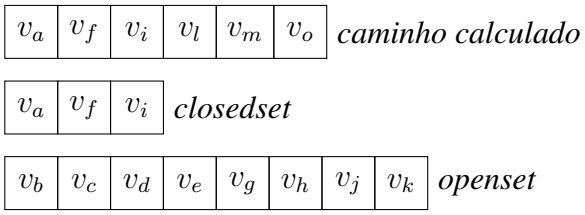

Figura 3. *openset* e *closedset* ao final da execução do exemplo.

4. Heurística para Diminuir Espaço de Busca

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 o 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.

4.1. 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 dada por $f(x) = g(x) + h(x)$, 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 por $h(x) = h_d(x) + h_{out}(x) + h_{in}(x)$, onde $h_d(x)$ é o componente relacionado à distância geográfica (seção 4.2), $h_{out}(x)$ é o componente relacionado ao grau de saída do vértice (seção 4.3) e $h_{in}(x)$ é o componente relacionado ao grau de entrada do vértice (seção 4.4). No processo de busca, é escolhido o vértice com o menor valor de $f(x)$, dentre os vértices disponíveis.

4.2. 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 são formulados na Equação 2.

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

Os parâmetros das Equações 2, 3 e 4 foram obtidos através de uma série de pré-processamentos, nos quais foram testadas várias curvas definidas por parâmetros de valores diferentes. As equações presentes neste trabalho possuem os melhores parâmetros encontrados que maximizaram a convergência da busca. O trabalho publicado em [Viana and Moro 2012] apresenta a visualização dessas três equações de acordo com os parâmetros definidos.

4.3. 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. 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. 2012]. A função é expressa pela Equação 3.

$$h_{out}(x) = \begin{cases} -0.025 * x + 1 & \text{se } x \leq 40 \\ 0 & \text{se } 40 < x \leq 300 \\ \exp\left(\dfrac{x-300}{500}\right) - 1 & \text{se } x > 300 \end{cases} \tag{3}$$

4.4. 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 4.

$$h_{in}(x) = \exp(-0.007 * x) \tag{4}$$

5. Avaliação Experimental

A avaliação experimental do algoritmo CUTE utiliza a rede social de microblogs Twitter. Essa rede foi escolhida pois sua política de acesso permite aos 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).

A Figura 4 apresenta alguns resultados obtidos utilizando o CUTE na busca de caminhos entre os autores deste artigo e personalidades brasileiras e internacionais. Para melhor análise, 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 4. Conjunto de rotas obtidas para a avaliação experimental.

Tabela 1. Número de vértices explorados para as rotas da Figura 4.

Caminho	Vértices explorados		
	partindo do alvo	partindo da origem	Total
A	15	8	23
B	3	18	21
C	4	10	14
D	3	6	9
E	3	6	9
F	3	18	21
G	2	10	12

Figura 5. Página inicial

Figura 6. Interface principal

6. Descrição da Ferramenta

Foi construída uma ferramenda Web, chamada FriendRouter, empregando o algoritmo CUTE. A ferramenta requer que o usuário se autentique com a sua conta do Twitter, conforme ilustrado na Figura 5. A interface principal do sistema é bem simples, como mostrado na Figura 6: requer o identificador do destino da busca (dado por padrão) e o identificador do usuário origem, e dá a opção de publicar o resultado da busca através da conta do usuário autenticado.

Ao processar a busca, FriendRouter mostra resultados intermediários. Como ilustrado na Figura 7, a interface mostra como o caminho entre os usuários é encontrado, desenhando uma árvore com os usuários explorados durante o processo. No final, o resultado da busca é inserido em um painel de buscas já efetuadas para prévia consulta, como é visto na Figura 8.

Para explorar rodas adicionais, ou rotas especiais, Friendrouter conta com uma função de filtro. Quando a opção de filtro é utilizada, o algoritmo ignora os usuários filtrados, construindo um goalset que não contém os usuários filtrados e posteriormente não os escolhendo durante o processo de busca. Essa funcionalidade pode ser utilizada para descartar usuários spam ou institucionais, por exemplo.

7. 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,

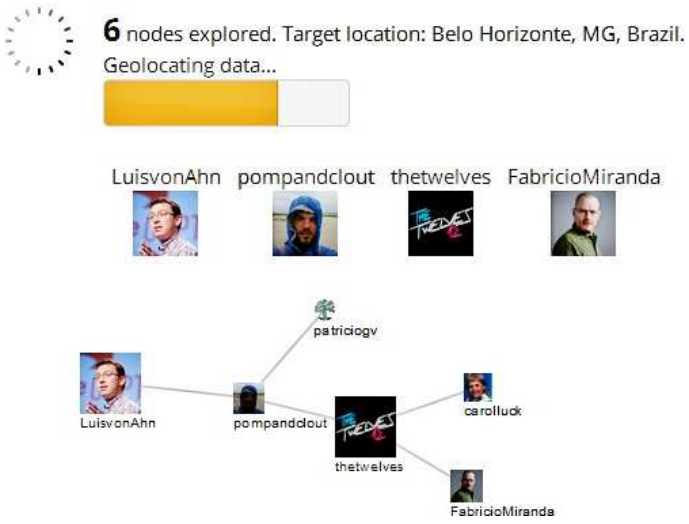

Figura 7. Durante o processo de busca: número de nós explorados, a localização do usuário-destino (obtida através de seu perfil), a barra de progresso, o caminho atual e um grafo interativo mostrando estados intermediários.

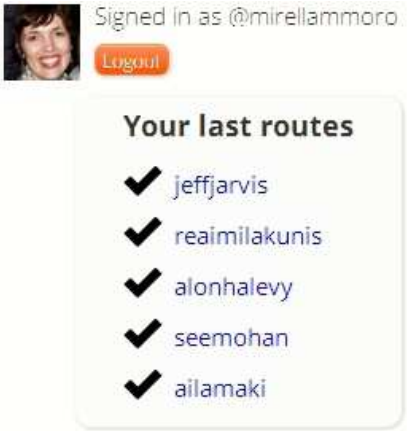

Figura 8. Buscas anteriores são armazenadas em cache e disponibilizadas

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 explorar outras informações que possam ajudar a direcionar o processo de busca ainda mais. Finalmente, este trabalho foi publicado no Workshop Brasnam [Viana and Moro 2012] e aceito para publicação na competição de trabalhos de alunos de graduação do ACM SIGMOD [Viana and Moro 2013].

Referências

Benevenuto et al, F. (2012). Characterizing user navigation and interactions in online social networks. *Information Sciences*, 195(15):1–24.

Bigonha, C., Cardoso, T. N. C., Moro, M. M., Gonçalves, M. A., and Almeida, V. A. F. (2012). Sentiment-based influence detection on twitter. *J. Braz. Comput. Soc.*, 18(3):169–183.

Brodersen, A., Scellato, S., and Wattenhofer, M. (2012). YouTube around the world: geographic popularity of videos. In *WWW*, pages 241–250.

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 *CIKM*, pages 499–508, Toronto, Canada.

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.

Noulas et al, A. (2011). An empirical study of geographic user activity patterns in foursquare. In *ICWSM*.

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 *CIKM*, pages 867–876, Hong Kong, China.

Rodrigues et al, T. (2011). On word-of-mouth based discovery of the web. In *IMC*, pages 381–393.

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

Thomsen, J. R., Yiu, M. L., and Jensen, C. S. (2012). Effective caching of shortest paths for location-based services. In *SIGMOD*, pages 313–324.

Topa, G. (2001). Social interactions, local spillovers and unemployment. *Review of Economic Studies*, 68(2):261–95.

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.

Viana, W. and Moro, M. M. (2012). Busca de caminhos entre usuários de redes sociais em tempo real. In *BRASNAM*, Curitiba, Brazil.

Viana, W. and Moro, M. M. (2013). Friendrouter - real-time path finder in social networks. In *SIGMOD Undergrad Research Competition*, New York City, USA.