Desenvolvimento de um Algoritmo de Busca por Vértices Específicos em Redes

Pedro Freitas[1], Victor Cardoso[2], Giulio Iacobelli[1], Daniel Figueiredo[1]

[1] Programa de Engenharia de Sistemas e Computação/COPPE
Universidade Federal do Rio de Janeiro (UFRJ) - Rio de Janeiro, RJ

[2]Engenharia de Computação e Informação/Poli
Universidade Federal do Rio de Janeiro (UFRJ) - Rio de Janeiro, RJ

{pfreitas, giulio, daniel}@land.ufrj.br, vcardoso@poli.ufrj.br

Abstract. *Many real networks are not immediately available but can be collected through a mining process. However, in some scenarios we are only interested in the discovery of vertices that have certain features. The others vertices are irrelevant. For this problem, classic algorithms (ex. breadth-first search) are inefficient. In this paper we use the homophily - inherent in many real networks - to propose a mathematical model that determines the probability of an unexplored vertex to have the feature. This model uses the features of the neighbor vertices already explored and global parameters of the network. The probabilities assigned by the model are used to guide an informed search, which at each step makes a greedy choice. We evaluated the algorithm in three real networks comparing different variations of the model and others search algorithms. The results show that none of considered methods is consistently more efficient than the others.*

Resumo. *Muitas redes reais não estão disponíveis de forma imediata mas podem ser coletadas através de um processo de mineração. Entretanto, em alguns cenários estamos interessados em descobrir apenas vértices que possuem determinadas características, sendo irrelevantes os demais vértices. Para esse problema, algoritmos de busca clássicos (ex. busca em largura) se mostram ineficientes. Neste trabalho utilizamos a homofilia - inerente em muitas redes reais - para propor um modelo matemático que determina a probabilidade de um vértice não explorado possuir a característica. O modelo utiliza as características dos vértices vizinhos já explorados e parâmetros globais da rede. As probabilidades atribuídas pelo modelo guiam um algoritmo de busca informada, que a cada passo faz uma escolha gulosa. Avaliamos o algoritmo em três redes reais, comparando diferentes variações do modelo e outros algoritmos de busca. Os resultados indicam que nenhum dos métodos testados é consistentemente mais eficiente que os demais.*

1. Introdução

O relacionamento entre pares de objetos pode ser representado por redes, onde os objetos correspondem aos vértices e a existência do relacionamento entre dois objetos às arestas da rede. Em muitos contextos, objetos possuem características que os personalizam. Por exemplo, a nacionalidade, profissão ou idade das pessoas em redes socias, e o idioma,

tema ou ano de publicação de documentos em uma rede de informação. Redes muitas vezes não estão disponíveis de forma imediata, e precisam ser obtidas através de um processo de mineração. Por exemplo, a *web* pode ser coletada navegando pelos *hiperlinks* de suas páginas, e similarmente, usuários do *Twitter* podem ser coletados através da rede de seguidores.

Neste contexto, surge o seguinte problema: como coletar eficientemente vértices de uma rede que possuam determinada característica? Para ilustrar o problema, considere a rede da Figura 1 onde vértices rotulados por T e N possuem ou não determinada característica, respectivamente. Imagine que a rede é desconhecida mas pode ser descoberta iterativamente a partir de um vértice inicial. Idealmente, estamos interessados apenas nos vértices que possuem a característica, não sendo necessário descobrir os demais. Desta forma, um processo eficiente de mineração dos vértices deve maximinar o número de vértices T ao explorar a rede.

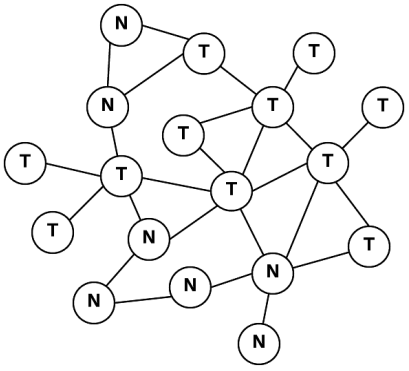

Figura 1. Rede com vértices que possuem (*T*) ou não (*N*) determinada característica.

O problema de minerar uma rede em busca de vértices com determinadas características encontra aplicações em muitos cenários, tais como campanhas de *marketing* personalizadas, sistemas de recomendação, detecção de fraudes de segurança, ou mesmo na identificação de estudantes que fumam para uma campanha antitabagismo eficaz [Murai et al. 2017]. Resolver este problema de forma eficiente é fundamental: considere o problema de encontrar pessoas no *Facebook* que declaram morar em Búzios. Búzios tem menos de 10^5 habitantes e a rede do *Facebook* tem mais de 10^9 pessoas, então como identificar estas pessoas de forma eficiente?

Algoritmos de busca em redes clássicos, tais como Busca em Largura (BFS), Busca em Profundidade (DFS), e Passeios Aleatórios (RW), são frequentemente empregados para descobrir redes reais. Entretanto, estes algoritmos são chamados de *desinformados* pois não utilizam nenhuma informação dos vértices para guiar o processo de busca. Intuitivamente, tais algoritmos podem não demonstrar bom desempenho na busca por vértices com determinadas características, no sentido de explorarem muitos vértices que não possuem a característica desejada.

Em contraposição, algoritmos de busca *informados* utilizam conhecimento específico dos vértices como parte do processo de busca [Russel and Norvig 2003]. Esse conhecimento é geralmente transformado em uma prioridade por uma função heurística

que é então utilizada para guiar a busca, explorando primeiro os vértices de maior prioridade. Esta ideia poder ser aplicada no problema em questão, pois as características dos vértices explorados podem guiar o processo de busca. Intuitivamente, esta abordagem pode ser bem mais eficiente, uma vez que prioriza a exploração de vértices que potencialmente possuem a característica.

Redes reais frequentemente exibem forte *homofilia*, que é a tendência de vértices a se relacionarem com vértices que possuem caracterísitcas similares. Por exemplo, amigos no *Facebook* tendem a ter a mesma nacionalidade, e páginas *web* tendem a apontar para páginas do mesmo idioma. Desta forma, uma heurística para a busca informada pode explorar a inerente homofilia das redes e guiar o processo de busca. Intuitivamente, a chance de um vértice possuir uma característica aumenta com o número de vizinhos que possuem a característica.

Neste trabalho propomos um modelo matemático probabilístico baseado em homofilia. O modelo determina a probabilidade de um vértice possuir a característica em função de seus vizinhos já conhecidos e de parâmetros da rede. Apresentamos três variações do modelo que são utilizadas para guiar uma busca informada pela rede. Experimentamos o modelo em três redes reais diferentes, e em diferentes características, comparando os resultados com buscas desinformadas, e também com um algoritmo informado recentemente proposto para este problema. Nossos resultados mostram que nenhum algoritmo testado é consistentemente superior aos outros, sendo o desempenho relativo fortemente dependente do caso. Isto indica a dificuldade de uma solução generalizada para este problema.

O restante desse artigo encontra-se organizado da seguinte forma: na Seção 2 apresentamos alguns trabalhos relacionados; na Seção 3 apresentamos o modelo matemático proposto e o algoritmo desenvolvido; na Seção 4 apresentamos as redes reais e a avaliação comparativa dos diferentes algoritmos; e finalmente, na Seção 5 são feitas as considerações finais.

2. Trabalhos Relacionados

O problema de busca em redes, exatamente como foi apresentado na Seção 1 (com descoberta gradual da rede e utilizando informação dos vértices), foi formulado e tratado recentemente em [Murai et al. 2017]. Empregando e adaptando métodos desenvolvidos para outros problemas ([Avrachenkov et al. 2014], [Wang et al. 2013], [Pfeiffer et al. 2012] e [Bnaya et al. 2013]), além de classificadores clássicos de Aprendizado de Máquina, os autores chegaram a conclusão de que é possível obter uma eficiência muito maior variando o classificador (ou método) utilizado em cada passo da busca. Para variar o método a cada passo da busca, é proposta um novo algoritmo chamado D^3TS, que baseia-se na política *Dynamic Thompson Sampling* empregada em problemas *Multi-Armed Bandit*[1].

Outros trabalhos na literatura que se aproximam do problema em questão foram desenvolvidos para *Active Search*. O objetivo da *Active Search* [Wang et al. 2013]

[1]*Multi-Armed Bandit* é um problema em que um apostador em uma fileira de caça níqueis tem que se decidir pelas máquinas em que jogar, quantas vezes jogar em cada máquina e em que ordem jogar. Quando jogada, cada máquina oferece um prêmio segundo uma distribuição de probabilidade específica da máquina. O objetivo é maximizar a soma de prêmios recebida pelo apostador.

[Ma et al. 2015] [Garnett et al. 2012] [Xie et al. 2016] também é descobrir a maior quantidade possível de vértices pertecentes a uma determinada classe, porém a rede é conhecida neste caso. Isso permite que qualquer vértice da rede possa ser explorado a qualquer instante do algoritmo. Entretanto, alguns métodos propostos para *Active Search* podem ser adaptados para o problema em questão. Por exemplo, o método para *Active Search* proposto em [Wang et al. 2013] foi adaptado em [Murai et al. 2017] para considerar apenas a rede conhecida até o momento presente da busca.

Um outro método da literatura que pode ser adaptado para o problema em questão é o MOD (*Maximum Observed Degree*) [Avrachenkov et al. 2014]. MOD é um algoritmo de busca míope que propõe maximizar o número de vértices descobertos. Através de um processo iterativo, a cada passo o vértice com o maior grau observado é explorado. Intuitivamente, o MOD pode ser adaptado para o problema em questão simplesmente priorizando os vértices a serem explorados de acordo com o número de vizinhos já explorados que possuem a característica. De fato, esta adaptação foi também sugerida em [Murai et al. 2017], e será explorada neste trabalho.

3. Algoritmo

Partindo da premissa de que a estrutura da rede é inicialmente desconhecida, o algoritmo deve seguir os seguintes passos para descobrir a maior quantidade possível de vértices com a característica desejada (veja a Figura 2):

1. Escolhe-se um vértice para iniciar a busca e o explora (i.e., pergunta se ele possui a característica);
2. Ao explorar, além de ficar sabendo se o vértice tem a característica (vértices T) ou não (vértices N), o algoritmo também passa a conhecer todos os vizinhos do vértice explorado. Esses são os vértices descobertos porém ainda não explorados (vértices '?');
3. Dentro do conjunto de vértices descobertos e não explorados, o algoritmo deve escolher o próximo vértice a explorar segundo uma heurística. Esse vértice assumirá o papel do vértice inicial no passo 1;

A iteração do algoritmo só termina quando se atinge um número máximo de perguntas a serem feitas, que chamaremos de *orçamento*.

A estrutura de qualquer rede que estivermos interessados em realizar a busca pode ser representada por um grafo não-direcionado $G = (V, E)$, onde V é o conjunto de vértices da rede e E é o conjunto de arestas [Bondy and Murty 1976] [Netto 1996]. Desta forma, em um determinado instante da busca, um vértice do grafo só pode pertencer a apenas um dos três conjuntos: vértices explorados (O), vértices descobertos mas não explorados (D), e vértices não descobertos ($V \setminus (O \cup D)$). Na Figura 2, eles são representados, respectivamente, pelas cores preta, cinza e branca.

A heurística empregada em nosso algoritmo precisa de informações a *priori* da distribuição global dos tipos de aresta da rede. Dada uma aresta qualquer do grafo que já tenha os seus dois vértices adjacentes explorados, ela pode ser de um dos 3 tipos: aresta entre vértices com a característica (TT), aresta entre um vértice com e outro sem a característica (TN) ou aresta entre vértices sem a característica (NN). Chamaremos de P_T a fração de arestas do tipo TT, P_D a fração de arestas do tipo TN, e de P_N a fração

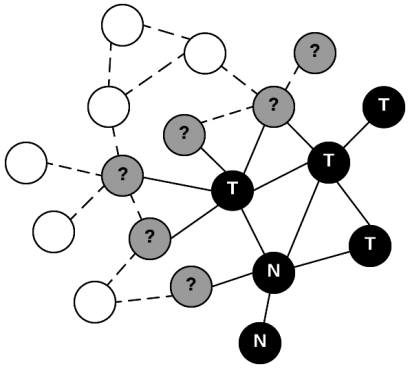

Figura 2. *Snapshot* de busca em um grafo após t = 6 passos. Vértices e arestas sólidas representa o subgrafo conhecido. Os vértices pretos representam os vértices explorados. Vértices descobertos mas não explorados são marcados por '?'.

de arestas do tipo NN sobre o grafo inteiramente explorado. Além de serem constantes, esses valores devem somar 1 obrigatoriamente. Essas frações indicam a intensidade da homofilia na rede, especificamente para a característica em questão.

Outra informação da qual a heurística se utiliza é o número de vértices vizinhos a um vértice $v \in D$ que possuem a característica e o número dos que não possuem. Chamaremos esses valores de k_T e k_N, respectivamente. Diferentemente de P_T, P_D e P_N, esses valores são calculados ao decorrer do processo de busca e estão sujeitos a atualizações a cada passo. A função heurística que usamos se baseia na probabilidade de um vértice ter a característica dado que o mesmo possui k_T vizinhos com a característica e k_N vizinhos sem a característica, ou seja $\mathbb{P}(A|k_T, k_N)$, onde A é o evento 'ter a característica'. Além disso, assumimos que a influência de cada um dos vizinhos é idêntica e ocorre de forma independente.

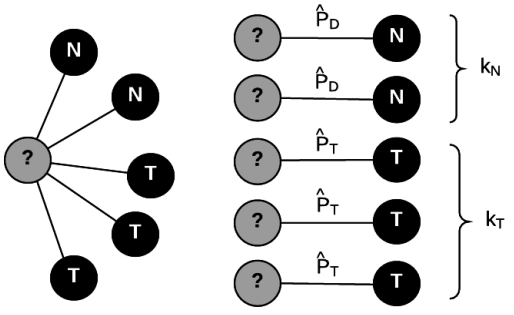

Figura 3. Ilustração do modelo probabilístico. Nesse exemplo, as probabilidades estão dispostas de forma que é calculada a probabilidade do evento 'ter a característica' ocorrer em todas as arestas.

Como estamos tratando a influência dos vizinhos como eventos independentes, o evento A, 'ter a característica', pode ser desmembrado em um evento para cada aresta (veja a Figura 3). A probabilidade condicional, $\mathbb{P}(A|k_T, k_N)$, então, pode ser melhor

representada por $\mathbb{P}(T = k|k_T, k_N)$, onde T é uma variável aleatória que indica o número de realizações do evento A. E $T \in [0, ..., k_T + k_N]$.

As frações de arestas P_T, P_D e P_N não é bem o que a nossa herística irá usar. Para a heurística só interessa arestas em que já é conhecido o rótulo (T ou N) de um dos vértices (Figura 3). Portanto, usaremos as probabilidades condicionais chamadas, $\hat{P}_T$ e $\hat{P}_D$. $\hat{P}_T$ se trata da probabilidade de ocorrer o evento A em um vértice da aresta dado que na outra ponta da aresta temos um vértice T. $\hat{P}_D$ é a probabilidade de ocorrer o evento A em um vértice da aresta dado que na outra ponta da aresta temos um vértice N. As fórmulas dessas probabilidades condicionais são descritas a seguir:

$$\hat{P}_T = P_T/(1 - P_N) \tag{1}$$

$$\hat{P}_D = P_D/(1 - P_T) \tag{2}$$

Para $k = 0$, não temos nenhuma realização do evento A, i.e., em todas as arestas deve ocorre N para o vértice analisado.

$$\mathbb{P}(T = 0|k_T, k_N) = (1 - \hat{P}_T)^{k_T}(1 - \hat{P}_D)^{k_N} \tag{3a}$$

Para $k = 1$, temos que deve ocorrer exatamente uma realização do evento A, seja em alguma das arestas em que o vértice conhecido é T ou em alguma das arestas em que o vértice conhecido é N.

$$\begin{aligned}\mathbb{P}(T = 1|k_T, k_N) = \binom{k_T}{1}\hat{P}_T(1 - \hat{P}_T)^{k_T-1}(1 - \hat{P}_D)^{k_N} + \\ + \binom{k_N}{1}\hat{P}_D(1 - \hat{P}_D)^{k_N-1}(1 - \hat{P}_T)^{k_T}\end{aligned} \tag{3b}$$

Para $k = 2$, temos que devem ocorrer duas realizações do evento A, e existem três formas possívies disso acontecer. A primeira é que as duas ocorram em duas quaisquer arestas em que o vértice conhecido é T, a segunda é que as duas ocorram em duas quaisquer arestas em que o vértice conhecido é N, e a terceira, e última, forma é que uma ocorra em alguma das arestas em que o vértice conhecido é T e a outra em alguma das arestas em que o vértice conhecido é N.

$$\begin{aligned}\mathbb{P}(T = 2|k_T, k_N) = \binom{k_T}{2}\hat{P}_T^2(1 - \hat{P}_T)^{k_T-2}(1 - \hat{P}_D)^{k_N} + \\ + \binom{k_N}{2}\hat{P}_D^2(1 - \hat{P}_D)^{k_N-2}(1 - \hat{P}_T)^{k_T} + \\ + \binom{k_T}{1}\binom{k_N}{1}\hat{P}_T(1 - \hat{P}_T)^{k_T-1}\hat{P}_D(1 - \hat{P}_D)^{k_N-1}\end{aligned} \tag{3c}$$

Agora fica fácil notar que as probabilidades calculadas seguem duas funções binomiais, uma com parâmetros $\hat{P}_T$ e k_T e outra com parâmetros $\hat{P}_D$ e k_N. Sendo assim,

para um valor k qualquer temos que:

$$\mathbb{P}(T = k|k_T, k_N) = \sum_{i=max(0,k-k_N)}^{min(k_T,k)} \binom{k_T}{i} \hat{P}_T^i (1 - \hat{P}_T)^{k_T - i} \binom{k_N}{k-i} \hat{P}_D^{k-i} (1 - \hat{P}_D)^{k_N - (k-i)} \tag{4}$$

Desse nosso modelo (Equação 4) surgem as três heurísticas utilizadas no trabalho. Sendo a mais rígida aquela que para calcular a probabilidade de um vértice ter a característica considera que o evento A tem que ocorrer nas $k_T + k_N$ arestas (Equação 5); uma menos rígida, e mais intuitiva, que calcula a mesma probabilidade mas considerando o evento A ocorrer na maioria das arestas (Equação 6); e uma menos rígida ainda que baseia o seu cálculo do vértice ter a característica na ocorrência do evento A em pelo menos uma das arestas (Equação 7).

$$H_1 = \mathbb{P}(T = k_T + k_N|k_T, k_N) = \hat{P}_T^{k_T} \hat{P}_D^{k_N} \tag{5}$$

$$H_2 = \sum_{k=\left\lceil \frac{k_T + k_N}{2} \right\rceil}^{k_T + k_N} \mathbb{P}(T = k) \tag{6}$$

$$H_3 = 1 - \mathbb{P}(T = 0) = 1 - (1 - \hat{P}_T)^{k_T} (1 - \hat{P}_D)^{k_N} \tag{7}$$

Começando com um vértice de entrada, o algoritmo o explora, inclue seus vizinhos no conjunto de descobertos (D) e calcula a função heurística (H_1, H_2 ou H_3) para cada um deles. Dentre todos os vértices pertecentes a D o vértice com o maior valor para a função utilizada (o mais provável de ter a característica) é escolhido como o próximo a ser explorado. Ele então é retirado do conjunto D e incluído no conjunto de vértices explorados (O). Esse processo se repete até se atingir o *orçamento* e o algoritmo retorna o conjunto O. Quanto mais vértices com a característica retornados melhor é a heurística.

Para cada vizinho de um vértice explorado o algoritmo paga um custo $\mathcal{O}(1)$ para atualizar os valores k_T ou k_N mais o custo $\mathcal{O}(H_i)$ para calcular H_i, com $i = 1, 2, 3$. Então, para cada vértice explorado temos um custo $\mathcal{O}(g \cdot H_i)$, onde g é o grau do vértice. Após essas atualizações, é preciso encontrar o vértice com o maior valor para a função heurística no conjunto D. Essa última operação custa $\mathcal{O}(n)$, onde n é o total de vértices no grafo. Logo, para apenas um vértice explorado a complexidade é $\mathcal{O}(g \cdot H_i + n)$. Considerando que o número de vértices explorados é no máximo igual ao *orçamento*, a complexidade do algoritmo é $\mathcal{O}(\textit{orçamento} \cdot (g_{max} \cdot H_i + n))$, onde g_{max} é o maior grau no grafo.

Para o cálculo de cada uma das heurísticas H_1 e H_3 temos um custo $\mathcal{O}(1)$. Logo, para essas heurísticas, a complexidade do algoritmo é $\mathcal{O}(\textit{orçamento} \cdot (g_{max} + n)) = \mathcal{O}(\textit{orçamento} \cdot n)$.

Para H_2 temos um custo $\mathcal{O}(g_{max}^2)$. Isso ocorre pois na equação de H_2 o somatório interno (Equação 4) tem $\min(k_T, k) - \max(0, k - k_N) + 1$ parcelas. Se $k < k_T$ e $k > k_N$, teremos exatamente $k_N + 1$ parcelas. Se $k < k_T$ e $k < k_N$, teremos exatamente $k + 1$

parcelas. Se $k > k_T$ e $k > k_N$, teremos exatamente $k_T - (k - k_N) + 1$ parcelas. Para todos os casos, teremos até $k_T + 1$ parcelas, que é da ordem do maior grau, $\mathcal{O}(g_{max})$. No somatório externo (Equação 6), no pior caso, teremos metade do maior grau parcelas, que também é $\mathcal{O}(g_{max})$. O que nos dá, $\mathcal{O}(g_{max}^2)$. Logo, a complexidade do algoritmo para H_2 é $\mathcal{O}(\textit{orçamento} \cdot (g_{max}^3 + n))$.

O algoritmo que citamos anteriormente na Seção 2, MOD, é um algoritmo guloso que busca atingir a maior cobertura possível da rede através das conexões dos vértices com um baixo número de vértices explorados. A cada passo do algoritmo, o próximo vértice a ser explorado será aquele que apresentar o maior grau observado. Para os resultados na Seção 4, o MOD foi adapatado para que ao invés de considerarmos o maior grau observado com critério de escolha, considerarmos o maior número de vizinhos com a característica. Nas nossas definições apresentadas anteriormente isso significa o maior valor de k_T. Nos referenciamos a essa variação do MOD como MOD*.

4. Avaliação

Para a avaliação do desempenho do nosso algoritmo selecionamos três redes reais extraídas de *datasets* disponíveis na *web*. Sobre essas redes, basicamente, calculamos a quantidade de vértices explorados positivos (que possuem a característica) por *orçamento*. Em adição ao nosso algoritmo com as três heurísticas (H_1, H_2 e H_3), rodamos as buscas desinformadas, Busca em Largura (BFS) e Busca em Profundidade (DFS), e a busca informada MOD*. É importante ressaltar que essa três últimas buscas (BFS, DFS e MOD*) não precisam de parâmetros, enquanto as heurísitcas apresentadas nesse trabalho precisam de dois parâmetros, $\hat{P}_T$ e $\hat{P}_D$, derivados de PT, PD e PN. Nessa avaliação utilizamos a linguagem de programação *Python* com o auxílio do módulo *graph-tool*[2]. O código-fonte encontra-se disponível em `https://github.com/freitaspedro/SearchOverGraphs/tree/wperfor`.

4.1. Datasets

Os dois primeiros *datasets* adotados neste trabalho foram utilizados para um estudo sobre detecção de círculos sociais [Leskovec and Krevl 2014]. Enquanto um contém parte da rede do ***Facebook***, o outro contém parte da rede do ***Google Plus***. Ambos são formados por um conjunto de *egonets*[3]. Para cada *egonet*, existe uma lista de características em cada vértice, onde os valores 1 e 0 simbolizam a presença ou não da característica. Essa lista, dependendo da *egonet*, pode apresentar até 200 características. Em cada *egonet*, algumas características foram escolhidas de forma que representassem diferentes proporções de valores positivos, frações de tipos de arestas (P_T, P_D e P_N), e $\hat{P}_T$ e $\hat{P}_D$. Nosso interesse é entender o quanto essas disparidades impactam nos resultados. Foram testadas as *egonets* com maior quantidade de vértices tanto para *Facebook* quanto para *Google Plus*.

O último *dataset* utilizado, ***PolBlogs***, é parte de um estudo a respeito do comportamento de *blogs* políticos durante a eleição presidencial norte-americana de 2004 [Adamic and Glance 2005]. O *dataset* é composto por *hiperlinks* entre os *blogs*,

[2]*Graph-tool* é um módulo *Python* eficiente para manipulação e análise estatística de grafos (`https://graph-tool.skewed.de/`).

[3]*Egonets* são redes centradas em um indivíduo, com suas arestas para seus vizinhos, e as possíveis arestas entre os vizinhos.

onde um *hiperlink* se trata da referência de uma página a outra. Esse *dataset* traz como característica a inclinação política dentro do contexto daquele país. Nos vértices com valor 0, temos os *blogs* com inclinação à esquerda (ou liberais). Nos vértices com valor 1, temos os *blogs* com inclinação à direita (ou conservadores).

4.2. Resultados

Para a avaliação dos algoritmos foi considerada a média amostral de 20 rodadas (com inicialização aleatória e uniforme) da quantidade de vértices explorados com a característica escolhida. O *orçamento* é aumentado gradualmente até que chegue em torno da metade do total de vértices da rede. Os vértices escolhidos aleatoriamente em cada rodada são mantidos para todos os *orçamentos* e todos os algoritmos em cada rede, e devem pertencer a maior componente conexa caso a rede não seja conexa. Para todos os experimentos, as quantidades totais de vértices explorados atingiu os *orçamentos* disponíveis.

A Tabela 1 apresenta algumas propriedadas das redes usadas em nossa avaliação.

Dataset	**Vértices**	**Arestas**	**Grau Médio**	**Diâmetro**	**Clusterização Global**
Facebook	747	60050	160,776	7	0,697
Google Plus	4872	416992	171,179	6	0,248
PolBlogs	1490	19090	25,624	8	0,251

Tabela 1. Propriedades das redes[4]

4.2.1. Facebook

Para a *egonet* do *Facebook* consideramos duas características, que chamaremos de A e B. Se tratam de características anônimas e os seus significados não afetariam o objetivo da nossa avaliação.

A distribuição do tipo das arestas (P_T, P_D e P_N), e as probabilidades $\hat{P}_T$ e $\hat{P}_D$, para cada uma dessas 2 características podem ser vistas na Tabela 2. Na Tabela 2, ainda temos a quantidade de positivos, que é o mesmo que a quantidade de vértices com rótulo T dada uma característica.

Caracterísitca	P_T	P_D	P_N	$\hat{P}_T$	$\hat{P}_D$	**Positivos**
A	0,047	0,231	0,722	0,061	0,832	63 (8%)
B	0,133	0,413	0,454	0,226	0,757	222 (29%)

Tabela 2. Distribuição do tipo de arestas - *Facebook*

Na Figura 4 temos o desempenho dos algoritmos de busca para a característica A. Como pode ser visto, a BFS tem um desempenho ruim, porém ainda consegue se sair melhor que a H_1. Isso ocorre pois na equação usada em H_1 (Equação 5) conforme os valores de k_T e k_N crescem ela se aproxima de 0 rapidamente. Até chegar em um momento

[4]Os diâmetros das *egonets* não são de tamanho igual a 2 pois as arestas incidentes no vértice ego foram desconsideradas a fim de dificultar o processo de busca.

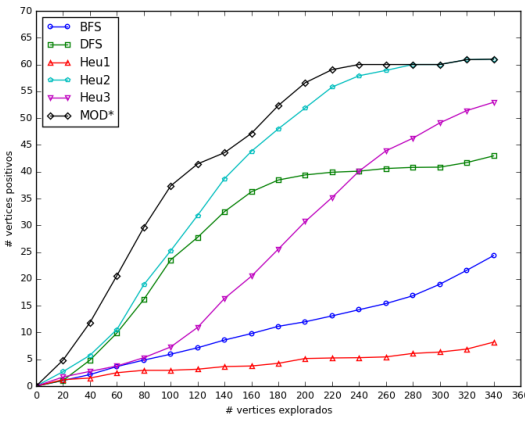

Figura 4. Desempenho dos algoritmos de busca para a característica A - *Facebook*

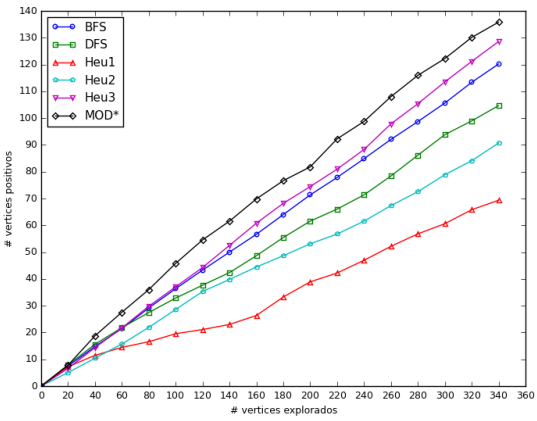

Figura 5. Desempenho dos algoritmos de busca para a característica B - *Facebook*

em que k_T e k_N são grandes o suficiente para transformar a H_1 em quase uma busca aleatória, e como a procentagem de positivos é apenas 8% isso justifica seu desempenho ruim. Para a H_3 ocorre o contrário, conforme os valores de k_T e k_N crescem, sua equação (Equação 7) se aproxima de 1. Contudo, isso não impacta tanto o desempenho da H_3, que a partir 100 vértices explorados tem um crescimento significativo. Como já se esperava a H_2 tem um bom desempenho, perdendo, surpreendentemente, apenas para o MOD*, que possui uma heurística muito simples. Essas duas últimas buscas acabam por convergir a partir de 280 vértices explorados.

A busca pela característica B resulta em um gráfico com os desempenhos das buscas se resumindo basicamente a retas com diferentes inclinações, conforme pode ser visto na Figura 5. Novamente temos a MOD* com o melhor desempenho, porém ela não consegue se distanciar tanto assim das outras buscas. A H_3 e BFS vêm em seguida, ganhando por exemplo da H_2 que para essa característica tem um resultado ruim. Esse baixo desempenho pode ser justificado, em parte, por conta de P_D assumir um alto valor (aprox. 0,413), o que atrapalha na capacidade da heurística em prever relações que influenciem na presença da característica em um vértice.

4.2.2. Google Plus

A rede do *Google Plus* também é uma *egonet* assim como no caso do *Facebook* e por isso elas são muito parecidas estuturalmente. Entretanto, ela tem oito vezes mais vértices que a do *Facebook*. Esse aumento significativo da rede é importante para observarmos o desempenho dos algoritmos de busca em redes maiores.

A distribuição do tipo das arestas, as probabilidades $\hat{P}_T$ e $\hat{P}_D$ e a porcentagem de positivos para cada uma das 2 características escolhidas podem ser vistas na Tabela 3.

No gráfico na Figura 6 temos o desempenho dos algoritmos de busca para a característica A que ocorre em aproximadamente 72% dos vértices da rede. Esse alto percentual faz com que as diferentes técnicas de busca tenham um resultado extremamente

Característica	P_T	P_D	P_N	$\hat{P}_T$	$\hat{P}_D$	Positivos
A	0,594	0,342	0,064	0,903	0,365	3528 (72%)
B	0,007	0,102	0,891	0,008	0,935	247 (5 %)

Tabela 3. Distribuição do tipo de arestas - *Google Plus*

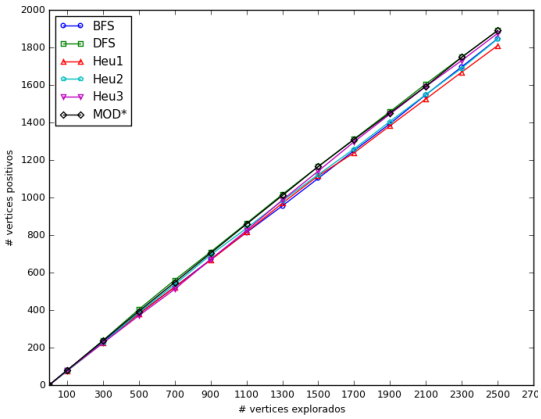

Figura 6. Desempenho dos algoritmos de busca para a característica A - *Google Plus*

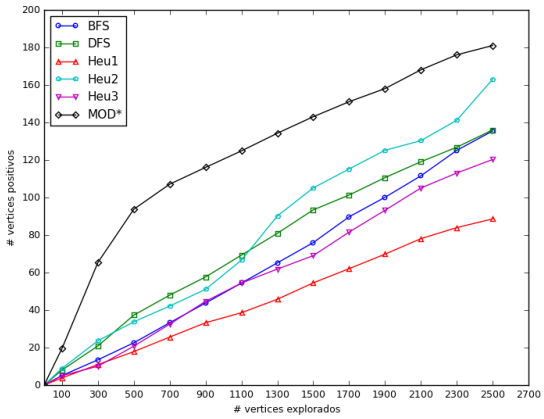

Figura 7. Desempenho dos algoritmos de busca para a característica B - *Google Plus*

parecido. Devido à abundância de vértices com a característica buscada, qualquer busca consegue encontrar esses vértices com certa facilidade.

Quando existe proporcionalmente poucos vértices explorados, os valores das funções heurísticas para os vértices são muito semelhantes ou até mesmo iguais, o que faz com que o próximo vértice a ser explorado seja escolhido de forma quase aleatória. Conforme se percorrem mais vértices, acumula-se informação a respeito do surgimento da característica, consequentemente os valores de heurísitca são mais afiados para que se tome a decisão correta. No caso da característica B (Figura 7), isso pode ser observado no comportamento da curva de desempenho da H_2, que no início tem um crescimento parecido com o das outras buscas, mas que a partir de 1100 vértices explorados (quase $1/4$ do número total de vértices) consegue atingir os melhores segundos resultados.

4.2.3. PolBlogs

Essa rede, apresentada em [Adamic and Glance 2005], possui apenas uma característica como dito anteriormente. Contudo, a apresentação do seu resultado traz uma importante confirmação a respeito do impacto da distribuição de arestas na eficiência do algoritmo proposto, e também é relevante por trazer um rede mais "rica"que as anteriores que eram centradas em indivíduos (*egonets*).

Na Tabela 4 podem ser vistas a distribuição do tipo das arestas, as probabilidades $\hat{P}_T$ e $\hat{P}_D$ e a quantidade positivos para essa rede.

Para a característica A nessa rede de *blogs*, que representa a inclinação política de

Característica	P_T	P_D	P_N	$\hat{P}_T$	$\hat{P}_D$	**Positivos**
A	0,471	0,089	0,440	0,517	0,158	732 (49%)

Tabela 4. Distribuição do tipo de arestas - *PolBlogs*

cada *blog*, as diversas técnicas de busca têm desempenhos parecidos até mais ou menos 150 vértices explorados (Figura 8). Depois desse ponto se destacam o bom desempenho da H_3 e o mau desempenho da H_2. O fato dessa rede ser formada por dois grandes *clusters* - um com os *blogs* de esquerda e outro com os *blogs* de direita - e a busca ser feita em cima dos *blogs* de direita, nos dão dicas do porquê desse comportamento. Relembrando, a H_3 para calcular a probabilidade de um vértice ter a característica considera que o evento 'ter a característica' tem que ocorrer em pelo menos uma das arestas. H_3 se encaixa bem nessa rede em que a correlação entre a presença da característica e a presença de uma aresta é alta - uma vez dentro do *cluster* dos *blogs* de direita é difícil sair. H_2 é mais rígida e considera a maioria das arestas, isso pode ser um problema quando o algoritmo percorre os vértices nos extremos dos *clusters*, por isso o mau desempenho. Isso também pode ter ocorrido no MOD* que não conseguiu um bom desempenho como nos outros experimentos, tendo inclusive desempenho inferior às buscas desinformadas.

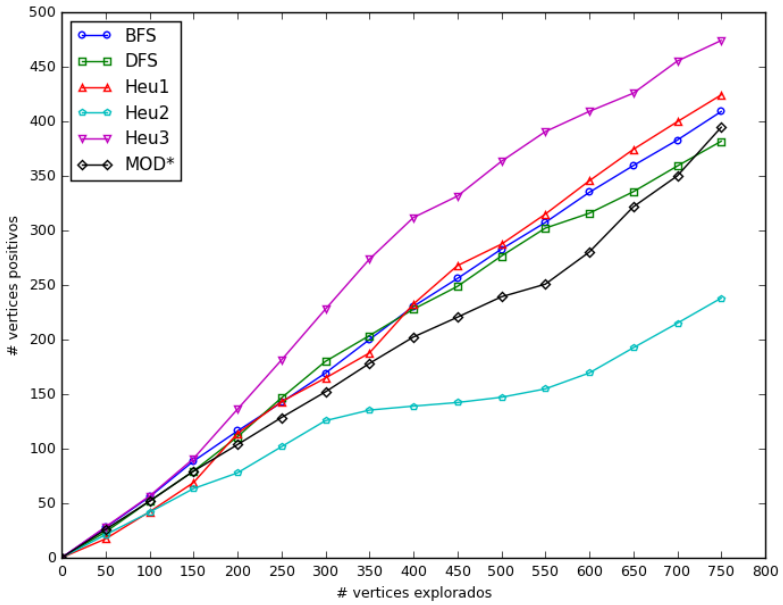

Figura 8. Desempenho dos algoritmos de busca - *PolBlogs*

5. Conclusão

Neste artigo apresentamos um modelo probabilístico que utiliza a homofilia da rede para determinar a probabilidade de um vértice possuir uma determinada característica em função das características do seus vizinhos conhecidos e da distribuição dos tipos de arestas da rede. Três variações do modelo foram utilizadas por um algoritmo de busca informado, que explora a rede de forma iterativa. O objetivo do algoritmo é encontrar

o maior número de vértices com uma determinada característica dentro de um número limitado de vértices que podem ser explorados.

A avaliação do algoritmo em três redes reais, utilizando diferentes características, e a comparação do desempenho com outros algoritmos, indicam o potencial e a fragilidade da metodologia proposta. Em particular, nenhuma abordagem testada se mostrou consistentemente superior, considerando todos os cenários avaliados. O desempenho do algoritmo MOD* foi melhor nas *egonets* de redes sociais, cuja heurística é simples e pode ser calculada rapidamente. Mas dependendo da caracterísitca, houve o empate entre diversos algoritmos. Por outro lado, uma das variações do modelo proposto (H_3) teve desempenho bem superior na rede de *blogs* onde os vértices possuem inclinação política, cenário no qual a heurística MOD* tem desempenho pior que buscas desinformadas.

Os resultados empíricos indicam que o problema de encontrar vértices com determinadas características em redes não tem fácil solução que possa ser generalizada, apresentando bom desempenho em todos os casos. De fato, esta observação também foi feita em um recente trabalho, que utiliza e mistura diversos métodos distintos em uma única proposta [Murai et al. 2017]. Deixamos como trabalho futuro o melhor entendimento do problema e das características da rede que possam ser exploradas para elucidar um algoritmo generalizado.

Referências

Adamic, L. A. and Glance, N. (2005). The political blogosphere and the 2004 US Election. In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*.

Avrachenkov, K., Basu, P., Neglia, G., and Ribeiro, B. (2014). Pay Few, Influence Most: Online Myopic Network Covering. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*.

Bnaya, Z., Puzis, R., Stern, R., and Felner, A. (2013). Bandit algorithms for social network queries. In *Social Computing (SocialCom), 2013 International Conference on*, pages 148–153. IEEE.

Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. Macmillan/Elsevier, 5 edition. ISBN 0-44-19451-7.

Garnett, R., Krishnamurthy, Y., Xiong, X., Schneider, J., and Mann, R. (2012). Bayesian optimal active search and surveying. In *International Conference on Machine Learning, ACM*, pages 1239–1246.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Ma, Y., Huang, T.-K., and Schneider, J. G. (2015). Active Search and Bandits on Graphs using Sigma-Optimality. In *Conference on Uncertainty Artificial Inteligence*, pages 542–551.

Murai, F., Renno, D., Ribeiro, B., Gisele, P., Towsley, D., and Gile, K. (2017). Selective Harvesting over Networks. *arXiv preprint arXiv:1703.05082*.

Netto, P. O. B. (1996). *Grafos: teoria, modelos, algoritmos*. Edgard Blücher.

Pfeiffer, J. J., Neville, J., and Bennett, P. (2012). Active sampling of networks. In *Workshop on Mining and Learning with Graphs*.

Russel, S. and Norvig, P. (2003). Artificial intelligence: A modern approach. *EUA: Prentice Hall*.

Wang, X., Garnett, R., and Schneider, J. (2013). Active Search on Graphs. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 731–738. ACM.

Xie, P., Zhu, J., and Xing, E. P. (2016). Diversity-promoting bayesian learning of latent variable models. In *International Conference on Machine Learning*.