UM ALGORITMO DE BUSCA POR VÉRTICES COM CARACTERÍSTICAS ESPECÍFICAS EM REDES

Pedro Vitor Pedroza Freitas

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Daniel Ratton Figueiredo

Rio de Janeiro
Julho de 2017

UM ALGORITMO DE BUSCA POR VÉRTICES COM CARACTERÍSTICAS ESPECÍFICAS EM REDES

Pedro Vitor Pedroza Freitas

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Daniel Ratton Figueiredo, D.Sc.

Prof. José Ferreira de Rezende, D.Sc.

Prof. Antonio Augusto de Aragao Rocha, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2017

Freitas, Pedro Vitor Pedroza

Um Algoritmo de Busca por Vértices com Características Específicas em Redes/Pedro Vitor Pedroza Freitas. – Rio de Janeiro: UFRJ/COPPE, 2017.

XII, 44 p.: il.; 29, 7cm.

Orientador: Daniel Ratton Figueiredo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 43 – 44.

1. Busca. 2. Redes Complexas. 3. Modelo Matemático. I. Figueiredo, Daniel Ratton. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A minha mãe, Helena.

Agradecimentos

Aos meus pais, Helena e Pedro, por sempre estarem ao meu lado nos momentos felizes e tristes, sempre me dando o incentivo, o consolo, o carinho, o puxão de orelha, a tranquilidade, a força para que em pudesse superar todas as barreiras que aparecessem. Essa conquista não poderia ser dedicada a alguém que não fossem eles.

A minha irmã, Larissa, por me servir de inspiração durante toda minha trajetória acadêmica. Exemplo de determinação, resiliência e, sobretudo, companherismo. Me sinto grato por todas as palavras de apoio nas minhas apresentações e pelas revisões de texto.

A minha tia, Elisabete, por todo o suporte dado. É de fundamental importância saber que temos mais um porto seguro além da casa dos nossos pais.

Ao meu orientador, prof. Daniel Figueiredo, seu apoio foi responsável por fazer essa minha caminhada durante o mestrado ser muito mais tranquila: desde a ajuda para resolver questões burocráticas até os momentos em que apareceu rapidamente com ideias para que esse trabalho pudesse ser concretizado.

Ao amigo Giulio Iacobelli por todo o seu conhecimento compartilhado comigo desde o início desse trabalho até sua conlusão.

Ao David Brilhante, Wladimir Cabral, Ronald Chiesse, Guilherme Iecker, Jose de Paula, Anderson Freitas, Matheus Correia, Jefferson Simões, Guilherme e Rapahel, apenas alguns dos amigos que fiz durante esse meu tempo na UFRJ, por todos os almoços, cafés, conversas e caronas.

Ao meus amigos de fora da UFRJ, que me proporcionaram os momentos de descontração necessários para que eu pudesse desenvolver a minha pesquisa o mais leve possível.

A todos os membros da banca por dela participarem e pelos comentários que contribuiram para esse trabalho.

À CAPES por ter em parte financiado a minha pesquisa.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM ALGORITMO DE BUSCA POR VÉRTICES COM CARACTERÍSTICAS ESPECÍFICAS EM REDES

Pedro Vitor Pedroza Freitas

Julho/2017

Orientador: Daniel Ratton Figueiredo

Programa: Engenharia de Sistemas e Computação

Neste trabalho, utilizamos o conceito de homofilia intrínseco a muitas redes para propor um modelo matemático que determina a probabilidade de um vértice não explorado possuir uma determinada característica em uma rede desconhecida *a priori*. O modelo utiliza as características dos vértices vizinhos 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 quatro redes reais buscando diferentes características por meio de variações do nosso modelo e comparando com outros algoritmos de busca informados e desinformados. Os resultados empíricos indicam que nenhum método considerado é consistentemente mais eficiente que os demais.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A SEARCH ALGORITHM FOR VERTICES WITH SPECIFIC FEATURES OVER NETWORKS

Pedro Vitor Pedroza Freitas

July/2017

Advisor: Daniel Ratton Figueiredo

Department: Systems Engineering and Computer Science

In this work, we use the homophily of the 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 four networks looking for different features using variations of the model and comparing with others informed and uninformed search algorithms. The empirical results show that none of considered methods is consistently more efficient than the others.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Redes são usadas para representar objetos - vértices - e o relacionamento - arestas - entre eles. Por sua vez, objetos podem possuir características que os personalizam. É comum nos perfis em redes sociais informarem características como nacionalidade, profissão e idade. Já em uma rede de informação, idioma, tema e ano da publicação de um documento podem caracterizá-lo. Porém, por razões como falta de espaço para armazenamento, políticas de segurança e privacidade, raramente as características dos vértices, assim como pesos das arestas e topologia da rede, estão disponíveis para serem acessadas de forma imediata. O que só ocorria se pudessemos baixar todos os dados de um único ponto central. Como não é o caso, precisamos de um processo de mineração para coletar esses dados da rede inicialmente desconhecida. Por exemplo, páginas da *web* podem ser coletadas navegando pelos *hiperlinks*, perfis de usuários do *Twitter* podem ser coletados através das redes de seguidores, uma rede de colaboração pode ser descoberta iterativamente a partir de um ou mais autores. Atualmente, o método mais empregado para esse tipo de coleta é o de *crawling*, onde são consultados os vizinhos de um vértice já anteriormente consultado, sucessivamente, a fim de descobrir parcela da (ou mesmo toda) informação da rede.

Neste contexto, surge o seguinte problema: como coletar eficientemente vértices de uma rede que possuam uma determinada característica? [1] Para ilustrar o problema, considere a rede da Figura 1.1. Nela, vértices rotulados por T e N possuem ou não determinada característica, respectivamente. Imagine agora que a rede é inicialmente desconhecida, mas que pode ser descoberta iterativamente a partir de um vértice inicial dado. Consequentemente, não é possível dar saltos na parte desconhecida da rede. Imagine ainda que estejamos interessados somente em vértices com a característica, e que a nossa capacidade de coleta de vértices seja limitada. Logo, o que queremos é um processo eficiente de mineração de vértices que maximize o número de vértices T encontrados ao explorar a rede.

O problema de minerar uma rede em busca de vértices com determinadas características encontra aplicações em diversas áreas, como (i) campanhas de *mar-*

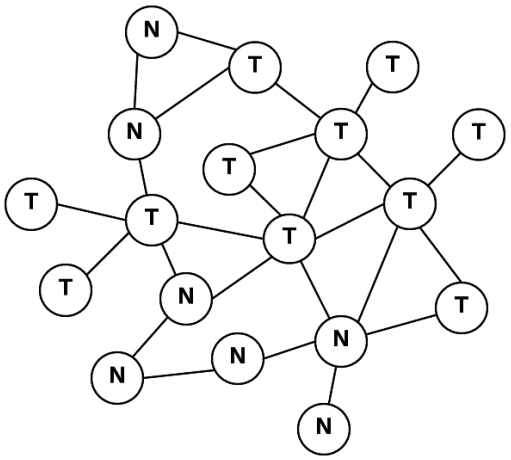

Figura 1.1: Rede com vértices que possuem (T) ou não (N) determinada característica.

keting personalizadas, (ii) sistemas de recomendação, (iii) investigação de fraudes mobiliárias e (iv) identificação de alunos envolvidos em desonestidades acadêmicas [2]. Dada a enormidade das redes que temos hoje em dia, resolver este problema de forma eficiente é fundamental. Considere que você trabalhe na área de *marketing* de uma empresa de *surfwear* e que sua tarefa atual seja divulgar seus produtos para moradores da cidade de Armação dos Búzios. Como meio, utiliza a rede do *Facebook* e inicia sua busca por moradores de Búzios a partir de um perfil que lhe é dado. Provavelmente, você irá verificar se o perfil que lhe foi dado é de um morador, depois passará para um amigo desse perfil, fará a mesma verificação, e assim sucessivamente. Só que Búzios possui menos de 10^5 habitantes [3] e a rede do *Facebook* tem mais de 10^9 pessoas [4]. Então, como identificar eficientemente essas pessoas?

Rede reais exibem frequentemente o princípio da *homofilia*, que diz que similaridade tende a cria conexão [5]. Isso se manifesta quando, para certas características, vértices similares se relacionam mais do que vértices diferentes em uma rede. Por exemplo, vértices com graus parecidos tendem a se conhecerem em uma rede (*degree-homophily*) [6], é mais provável que pessoas de mesma nacionalidade sejam amigas no *Facebook*, páginas na *web* tendem a ter *hyperlinks* para páginas do mesmo idioma, *blogs* de política tendem a referenciar *blogs* de mesma posição política. Dito isso, uma heurística para guiar a busca informada pode explorar essa questão da homofilia das redes. Naturalmente, considerando que a chance de um vértice possuir uma característica é proporcional ao número de vizinhos com a característica.

Neste trabalho, propomos um modelo matemático baseado em homofilia. O modelo determina a probabilidade de um vértice possuir uma característica em função de seus vizinhos conhecidos e de parâmetros globais da rede. Apresentamos três variações do modelo que são utilizadas para guiar uma busca informada pela rede.

Testamos também a dependência do nosso modelo em relação aos parâmetros, substituindo esses parâmetros por valores arbitrários e analisando a sua performance. Uma outra abordagem foi fazer a busca aprender esses parâmetros iterativamente. Para todos os casos, experimentamos o modelo em quatro redes reais para diferentes características, e comparamos com duas buscas desinformadas e um algoritmo informado recentemente proposto para este problema. Duas dessas redes foram coletadas das redes sociais *Facebook* e *Google Plus* [7]. A outra rede representa *hyperlinks* entre *blogs* políticos durante uma eleição presidencial americana [8]. E a última é uma rede de colaboração extraída do *dataset* do *DBLP* [9], cujas características dos vértices são as conferências em quais os autores publicaram. Nossos resultados finais mostram que nenhum algoritmo testado é consistentemente superior aos outros, sendo o desempenho relativo fortemente dependente do caso: densidade de positivos, frações de arestas, tamanho da rede, entre outras propriedades. O que indica a dificuldade de uma solução generalizada.

1.1 Organização do Texto

O restante deste trabalho encontra-se dividido da seguinte forma: no Capítulo 2 apresentamos alguns trabalhos relacionados; no Capítulo 3 apresentamos o modelo matemático proposto e o algoritmo desenvolvido; no Capítulo 4 discorremos a respeito dos algoritmos escolhidos para comparação com o nosso modelo; no Capítulo 5 apresentamos as rede reais utilizadas e a avaliação comparativa dos diferentes algoritmos; e finalmente, no Capítulo 6 apresentamos as conclusões.

Capítulo 2

Trabalhos Relacionados

As buscas propostas na literatura podem ser divididas em dois grandes grupos, em relação ao seu conhecimento a respeito de informações da rede: Busca Desinformadas e Buscas Informadas.

2.1 Buscas Desinformadas

Buscas desinformadas recebem esse nome pois não utilizam nenhuma informação sobre as características ou estrutura da rede para guiar o processo de busca. Logo, espera-se que seus algoritmos não consigam demonstrar bom desempenho na busca por vértices com uma característica específica, no sentido de explorarem muitos vértices que não possuem essa característica. Mas, quando isso não é um problema, pode-se utilizar os algoritmos de busca clássicos pertencentes a esse grupo, como Busca em Largura (BFS) [10], Busca em Profundidade (DFS) [11] e Passeios Aleatórios (RW) [12] para realizar uma busca exaustiva pela rede. São algoritmos amplamente difundidos e estudados na literatura, de fácil implementação, e podem ser implementados em qualquer rede.

2.2 Buscas Informadas

Do outro lado, temos os algoritmos de busca *informados* que utilizam algum conhecimento específico da rede como parte do processo de busca [13]. Esse conhecimento é geralmente transformado em uma prioridade por uma função heurística que é então utilizada para guiar a busca. Esta abordagem pode ser bem mais eficiente para o problema de busca em redes, exatamente como foi apresentado no Capítulo 1 (com descoberta gradual da rede e buscando por vértices com características específicas), uma vez que prioriza a exploração de vértices que potencialmente possuem a característica. Por conta disso, o algoritmo proposto neste trabalho se enquadra

justamente nesse grupo.

A seguir apresentamos alguns trabalhos que propõem (ou exploram) buscas informadas, tanto para solucionar problemas com redes desconhecidas a *priori* quanto para problemas com redes com a topologia conhecida a *priori*.

Selective Harvesting

O problema de busca em redes, com descoberta gradual da rede e buscando por vértices com características específicas, foi formulado e tratado recentemente em [2], recebendo o nome de *Selective Harvesting*. Neste trabalho, métodos desenvolvidos para problemas similares de busca em redes (*Maximum Observed Degree* [14], *Active Search* [15], *Active Sampling* [16] e *Social Network UCB1* [17]) foram usados diretamente ou adaptados para o problema em questão. Com uma abordagem alternativa (*data-driven*), também foram usados modelos estatíticos comumente aplicados em Aprendizado de Máquina, como EWLS (*Exponentially Weighted Least Squares*) e SVR (*Support Vector Regression*). A seguir, para uma leitura mais fácil do texto, nos referiremos aos métodos existentes e aos modelos estatísticos genericamente como **classificadores**. As simulações, em [2], mostraram que além de não existir um classificador predominantemente melhor, o melhor em uma rede pode ser o pior em outra. Então, considerou-se escolher um conjuto de classificadores dentre os que tiveram as melhores performances, e propor uma política para alternar o classificador empregado ao longo do processo de busca. Esperava-se assim mitigar um efeito característico de classificadores aplicados no modo *standalone*[1], chamado *tunnel vision*: sem qualquer recurso para amostragem independente, o impulso de explorar somente os vértices com determinada característica força os classificadores a coletar dados de treinamento enviesados, comprometendo significantemente o seu resultado final. Realmente, para as redes testadas, os resultados revelaram o aumento de vértices alvos explorados ao ser empregada a política de permutação de classificadores proposta no trabalho, chamada D^3TS. Sendo ela desenvolvida a partir da política *Dynamic Thompson Sampling* de aplicação em problemas *Multi-Armed Bandit*[2].

Active Search

Outros trabalhos na literatura, que se aproximam do problema aqui explorado, foram concebidos voltados para o problema de *Active Search*. Apesar do objetivo de

[1] Modo em qual um único classificador é empregado do início ao fim da busca.

[2] *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.

Active Search [15, 18–20] também ser o de maximizar a quantidade de vértices explorados pertencentes a uma determinada classe, assume-se que a rede é inteiramente conhecida e que por consequência qualquer vértice pode ser explorado a qualquer instante do processo de busca. Ou seja, os vértices candidatos a exploração não são apenas os vizinhos dos vértices já explorados. Entretanto, alguns métodos propostos para *Active Search* podem ser adaptados para o problema aqui explorado. Por exemplo, o método para *Active Search* foi adaptado em [2] para considerar apenas a rede conhecida até o presente momento da busca ao tentar estimar os rótulos desconhecidos através de sua função *smooth* sobre as arestas da rede.

Active Surveying

Um outro problema de calssificação binária assim com *active search* é o *active surveying*. Enquanto no primeiro o objetivo é descobrir ativamente a maior quantidade possível de vértices pertencentes a uma classe, no *active surveying*, o objetivo é consultar ativamente vértices para conseguir prever a proporção de uma determinada classe na rede. Estimar a intensão de voto em uma campanha presidencial seria um problema no mundo real. Este problema foi introduzido e tratado em [19]. O método proposto em [19] utiliza a teoria Bayesiana de decisão, um abordagem estatítica para classificação de padrões [21].

Active Sampling

Active Sampling é um problema de domínio específico de busca em que os rótulos e as arestas das instâncias (vértices) são adquiridas através de um processo iterativo para que se possa prever os rótulos de uma amostra de instâncias em uma rede. Quebrando assim, um pressuposto muito comum a um classificador de instâncias em um domíno relacional, o de que as relações entre instâncias são facilmente observáveis. Em domínios em que as relações não podem se tornar públicas esssa abordagem se encaixa perfeitamente. P.e., dado que foi pego um aluno em um caso de desonetidade acadêmica, seria antiético ter acesso a *e-mails* entre todos os estudantes na universidade para aumentar a investigação, uma vez que não se têm evidências para envolver esses outros alunos. O nosso problema generaliza *active sampling*, que considera as características dos vértices não observáveis. Para mais informações recomendamos a leitura de [16]. Um exemplo de método proposto para esse problema é o PNB [16]. PNB estima um valor de recompensa y_v usando uma média ponderada das recompensas dos vértices observados a dois saltos de distância de v, onde os pesos são os números de vizinhos comuns com v.

Maximum Observed Degree

Outro método da literatura adaptável ao problema em questão é o *Maximum Observed Degree* [14]. MOD é um algoritmo de busca míope que propõe maximizar o número de vértices descobertos. Ele é uma simplificação do MEUD (*Maximum Expected Uncovered Degree*), ambos propostos no mesmo trabalho. Ao contrário do MEUD que requer a distribuição de grau da rede, o MOD, assim como BFS, DFS e RW, não requer essa distribuição nem qualquer outra informação prévia da rede. Para redes que seguem uma lei de potência com coeficiente de distribuição de grau igual a um ou dois, MOD aproxima MEUD. A cada passo do MOD é explorado aquele vértice com maior grau observado. Intuitivamente, o MOD pode ser adaptado simplesmente priorizando os vértices a serem explorados de acordo com o número de vizinhos já explorados que possuem a característica buscada. De fato, esta adaptação foi também sugerida em [2] e será explorada neste trabalho.

Capítulo 3

O Algoritmo

3.1 Solução Genérica

A estrutura de qualquer rede em que estivermos interessado em realizar uma 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. Em um determinado instante da busca, um vértice do grafo só pode pertencer a apenas um dos três conjuntos: explorados (O), descobertos (D), e desconhecidos ($V \setminus (O \cup D)$). Na Figura 3.1, eles são ilustrados, respectivamente, pelas cores preta, cinza e branca.

Partindo da premissa de que a rede é inicialmente desconhecida, o algoritmo genérico só é capaz de saber quais vértices pertencem ao conjunto de explorados e quais vértices pertencem ao conjunto de descobertos. Dessa forma, para explorar a rede, basta que o algoritmo siga os seguintes passos:

1. Um vértice inicial v é dado (O = $\emptyset$ e D = $\{v\}$)
2. Escolhe o próximo vértice dentro do conjunto de descobertos e o explora:
   - Descobre se o vértice tem a característica ou não
   - Inclui todos os seus vizinhos não explorados no conjunto de descobertos
   - Inclui o vértice no conjunto de explorados
3. Volta ao passo 2

A iteração do algoritmo termina quando se atinge um número máximo de vértices que podem ser explorados. A esse limitante daremos o nome de **orçamento**.

O desempenho do algoritmo está ligado diretamente ao quão boa é a heurística empregada para escolher o próximo vértice a se explorar. A seguir, abordaremos justamente as heurísticas proposta neste trabalho que são fundamentadas em um modelo probabilístico que atribui aos vértices valores que representam a chance

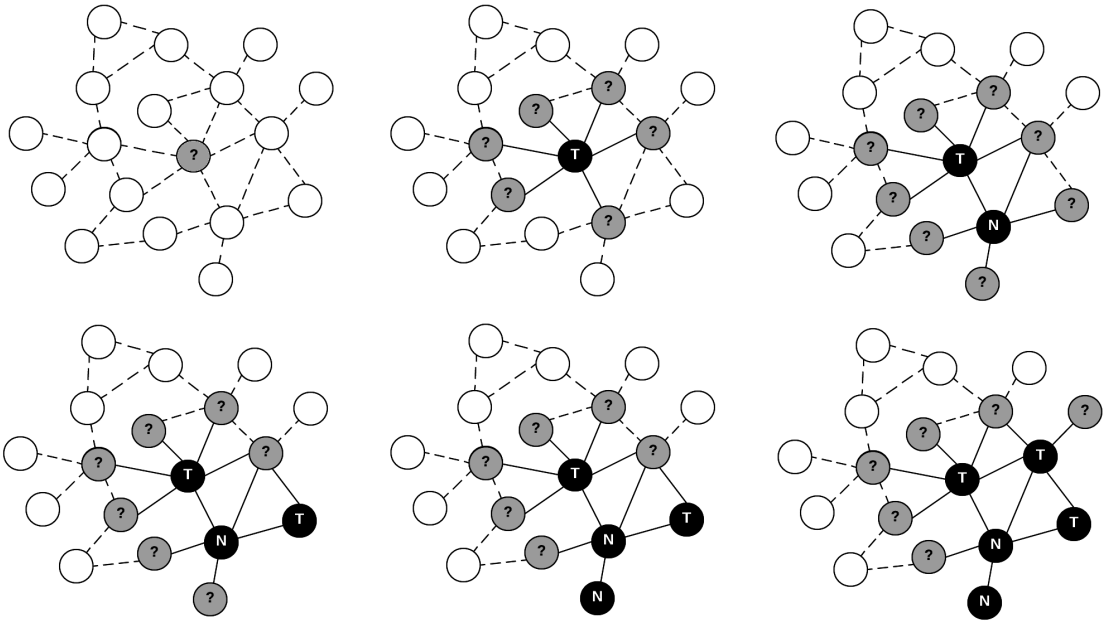

Figura 3.1: *Snapshots* da busca realizada em um grafo até t = 5 passos. Vértices e arestas sólidas representa o subgrafo conhecido. Os vértices pretos representam os vértices explorados. Vértices descobertos são marcados por '?'.

deles terem determinada característica. Resumindo o nosso algoritmo a uma busca gulosa por aquele vértice com maior valor atribuído pela heurística escolhida em um determinado instante da busca.

3.2 Modelo Probabilístico

A princípio, para o nosso modelo precisamos de informações a respeito da distribuição global dos tipos de arestas da rede. Dada uma aresta qualquer do grafo que já tenha os seus dois vértices adjacentes explorados, ela pode ser de um dos três tipos: aresta entre vértices com a caracterísitica (TT), aresta entre um vértice com e outro sem a caracterísitica (TN) ou aresta entre vértices sem a característica (NN). Chamaremos de P_T a fração de arestas do tipo TT, de P_D a fração de arestas do tipo TN, e de P_N a fração de arestas do tipo NN. Idealmente, esses valores devem ter sido, em algum momento, calculados sobre toda a rede. Pois eles servirão de base para o cálculo dos parâmetros do nosso modelo como veremos a seguir. Então podemos imaginar que um oráculo é responsável por nos informar esses valores, que essencialmente é uma informação muito rasa para afirmaramos que connhecemos a rede. Mas eles também podem ser estimados por um humano dada a sua experiência acumulada ao trabalhar com redes parecidas, por exemplo. Além de serem constantes, esses valores devem somar 1 obrigatoriamente. O mais importante é que essas frações nos passam implicitamente a intensidade da homofilia

na rede especificamente para a característica em questão.

A outra informação da qual o nosso modelo se utiliza é o número de 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. No fim, o que queremos modelar é a 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'*. Para tal, assumimos que a influência de cada um dos vizinhos é idêntica e ocorre de forma independente. Aqui quando falamos de influência, é no sentido do quanto uma pessoa que tem certa característica influência um amigo a também ter essa característica.

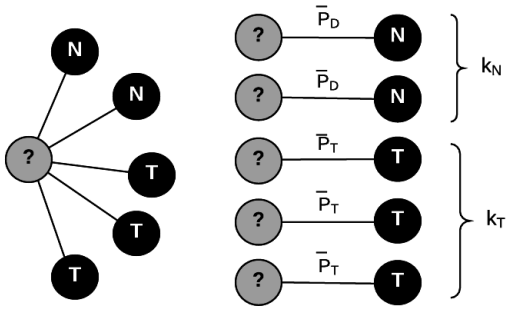

Figura 3.2: Ilustração do modelo probabilístico. Nesse exemplo, as probabilidades estão dispostas de forma que o seu produtório nos diz a probabilidade do evento *'ter a característica'* ocorrer em todas as arestas.

A aproximação que nosso modelo faz está em replicar o vértice em que estamos interessados em calcular a chance de ter a característica $k_T + k_N$ vezes, e então tratarmos cada aresta incidente no cenário real com uma aresta independente conforme pode ser visto na Figura 3.2. Portanto, o evento A pode ser desmembrado em um evento para cada aresta. A probabilidade condicional, $\mathbb{P}(A|k_T, k_N)$, então, pode ser 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 $T \in [0, ..., k_T + k_N]$.

Não são especificamente as frações de arestas P_T, P_D e P_N que utilizamos no nosso modelo, e sim, probabilidades condicionais derivadas dessas frações. Para o algoritmo só interessa arestas em que já é conhecido o rótulo de um dos vértices (Figura 3.2). Portanto, usaremos as probabilidades condicionais chamadas, $\overline{P}_T$ e $\overline{P}_D$. $\overline{P}_T$ se trata da probabilidade de ocorrer o evento A em um vértice da aresta dado que adjacente a ele temos um vértice T. $\overline{P}_D$ é a probabilidade de ocorrer o evento A em um vértice da aresta dado que adjacente a ele temos um vértice N. As fórmulas dessas probabilidades são descritas a seguir:

$$\overline{P}_T = P_T/(1 - P_N) \tag{3.1}$$

$$\overline{P}_D = P_D/(1 - P_T) \tag{3.2}$$

Uma vez que já sabemos os parâmetros do nosso modelo resta apresentarmos o modelo em si. A seguir demostramos o passo-a-passo do raciocínio lógico para construção do nosso modelo probabilístico:

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

$$\mathbb{P}(T = 0|k_T, k_N) = (1 - \overline{P}_T)^{k_T}(1 - \overline{P}_D)^{k_N} \tag{3.3a}$$

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.

$$\begin{aligned}\mathbb{P}(T = 1|k_T, k_N) = \binom{k_T}{1}\overline{P}_T(1 - \overline{P}_T)^{k_T-1}(1 - \overline{P}_D)^{k_N} + \\ + \binom{k_N}{1}\overline{P}_D(1 - \overline{P}_D)^{k_N-1}(1 - \overline{P}_T)^{k_T}\end{aligned} \tag{3.3b}$$

Para $k = 2$, temos que devem ocorrer duas realizações do evento A. 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.

$$\begin{aligned}\mathbb{P}(T = 2|k_T, k_N) = \binom{k_T}{2}\overline{P}_T^2(1 - \overline{P}_T)^{k_T-2}(1 - \overline{P}_D)^{k_N} + \\ + \binom{k_N}{2}\overline{P}_D^2(1 - \overline{P}_D)^{k_N-2}(1 - \overline{P}_T)^{k_T} + \\ + \binom{k_T}{1}\binom{k_N}{1}\overline{P}_T(1 - \overline{P}_T)^{k_T-1}\overline{P}_D(1 - \overline{P}_D)^{k_N-1}\end{aligned} \tag{3.3c}$$

Observando as equações acima (Equações 3.3a, 3.3b e 3.3c), fica fácil notar que as probabilidades calculadas seguem duas distribuições binomiais, uma com parâmetros $\overline{P}_T$ e k_T e outra com parâmetros $\overline{P}_D$ e k_N. Sendo assim, para um valor k qualquer

temos que:

$$\mathbb{P}(T = k|k_T, k_N) = \sum_{i=max(0,k-k_N)}^{min(k_T,k)} \binom{k_T}{i} \overline{P}_T^{\,i} (1 - \overline{P}_T)^{k_T - i} \binom{k_N}{k-i} \overline{P}_D^{\,k-i} (1 - \overline{P}_D)^{k_N - (k-i)} \tag{3.4}$$

Desse nosso modelo (Equação 3.4) surgem as três heurísticas utilizadas no trabalho. Que encontram-se em ordem decrescente de rigidez. A primeira 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 3.5); a segunda (mais intuitiva) calcula a mesma probabilidade mas considerando o evento A ocorrer na maioria das arestas (Equação 3.6); e a terceira baseia o seu cálculo do vértice ter a característica na ocorrência do evento A em pelo menos uma das arestas (Equação 3.7).

$$H_1 = \mathbb{P}(T = k_T + k_N|k_T, k_N) = \overline{P}_T{}^{k_T}\ \overline{P}_D{}^{k_N} \tag{3.5}$$

$$H_2 = \sum_{k=\left\lceil \frac{k_T+k_N}{2} \right\rceil}^{k_T+k_N} \mathbb{P}(T = k) \tag{3.6}$$

$$H_3 = 1 - \mathbb{P}(T = 0) = 1 - (1 - \overline{P}_T)^{k_T} (1 - \overline{P}_D)^{k_N} \tag{3.7}$$

Aplicando qualquer uma das heurísticas (H_i, $i = 1, 2, 3$) separadamente no algoritmo genérico anteriormente apresentado na Seção 3, o algoritmo se comporta da seguinte forma:

1. Um vértice inicial v é dado (O = $\emptyset$ e D = $\{v\}$)
2. Atualiza k_T e k_N e (re)calcula H_i para todos os vértices no conjunto de descobertos. Escolhe o vértice com maior H_i para explorar:
   - Descobre se o vértice tem a característica ou não
   - Inclui todos os seus vizinhos não explorados no conjunto de descobertos
   - Inclui o vértice no conjunto de explorados
3. Volta ao passo 2

Esse processo se repete até se atingir o orçamento e o algoritmo retorna o conjunto de explorados. Quanto mais vértices com a característica retornados melhor é a heurística.

3.2.1 Estimando $\overline{P}_T$ e $\overline{P}_D$

Ao invés de passarmos os valores das probabilidades condicionais, $\overline{P}_T$ e $\overline{P}_D$, como entrada do nosso algoritmo, uma outra abordagem é aprender esses valores a medida que o algoritmo vai descobrindo a rede. Para isso, o algoritmo deve atualizar os valores de seus dois estimadores (um para $\overline{P}_T$ e um para $\overline{P}_D$) a cada passo com as informações descobertas até aquele momento. Podemos estimar essas probabilidades da seguinte forma:

$$\overline{P}_T(t) = P_T(t)/(1 - P_N(t)) \tag{3.8}$$

$$\overline{P}_D(t) = P_D(t)/(1 - P_T(t)) \tag{3.9}$$

, onde $P_T(t)$, $P_D(t)$ e $P_N(t)$ são respectivamente as frações de arestas TT, TN e NN no instante t.

Porém, sabe-se que:

$$P_T(t) = \frac{n_T(t)}{n_T(t) + n_D(t) + n_N(t)} \tag{3.10}$$

$$P_D(t) = \frac{n_D(t)}{n_T(t) + n_D(t) + n_N(t)} \tag{3.11}$$

$$P_N(t) = \frac{n_N(t)}{n_T(t) + n_D(t) + n_N(t)} \tag{3.12}$$

, onde $n_T(t)$, $n_D(t)$ e $n_N(t)$ são respectivamente os números de arestas TT, TN e NN no instante t.

Substituindo 3.10, 3.11 e 3.12 em 3.8 e 3.9, chegamos as fórmulas finais de nossos estimadores:

$$\overline{P}_T(t) = n_T(t)/(n_T(t) + n_D(t)) \tag{3.13}$$

$$\overline{P}_D(t) = n_D(t)/(n_D(t) + n_N(t)) \tag{3.14}$$

Ainda é preciso determinar um valor inicial para n_T, n_D e n_N. Em nossa avaliação testamos $n_T(0) = n_D(0) = n_N(0) = 1$.

3.3 Complexidade

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. 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 a um orçamento b, a complexidade do algoritmo é $\mathcal{O}(b \cdot (g_{max} \cdot H_i + n))$, onde g_{max} é o maior grau no grafo.

3.3.1 Complexidade de H_1 e H_3

Tanto para o cálculo de H_1 quanto para o cálculo de H_3 temos um custo $\mathcal{O}(1)$. Logo, para essas heurísticas, a complexidade do algoritmo é $\mathcal{O}(b \cdot (g_{max} + n)) = \mathcal{O}(b \cdot n)$.

3.3.2 Complexidade de H_2

Para H_2 temos um custo $\mathcal{O}(g_{max}^2)$. Isso ocorre pois na equação de H_2 o somatório interno (Equação 3.4) 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 3.6), 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 $\mathcal{O}(b \cdot (g_{max}^3 + n))$.

3.4 Exemplo

Consideremos o grafo da Figura 1.1. As frações dos tipos de aresta nesse grafo valem $P_T = \frac{12}{25} = 0,48$, $P_D = \frac{8}{25} = 0,32$ e $P_N = \frac{5}{25} = 0,2$. Usando as Equações 3.1 e 3.2, temos que $\overline{P}_T = \frac{0,48}{(1-0,2)} = 0,6$ e $\overline{P}_D = \frac{0,32}{(1-0,48)} \approx 0,61$.

Assumimos que após 5 passos da busca, a configuração do grafo é igual a representada na Figura 3.3.

Se considerarmos a nossa outra abordagem em que $\overline{P}_T$ e $\overline{P}_D$ são calculados dinamicamente, e que $n_T(0) = n_D(0) = n_N(0) = 1$, então temos que $n_T(4) = 1 + 2 = 3$, $n_D(4) = 1 + 3 = 4$ e $n_N(4) = 1 + 1 = 2$. E a partir da substituição desses valores em nossos estimadores (Equações 3.13 e 3.14) é possível chegar a $\overline{P}_T(4) = \frac{3}{3+4} \approx 0,43$ e $\overline{P}_D(4) = \frac{4}{2+4} \approx 0,67$.

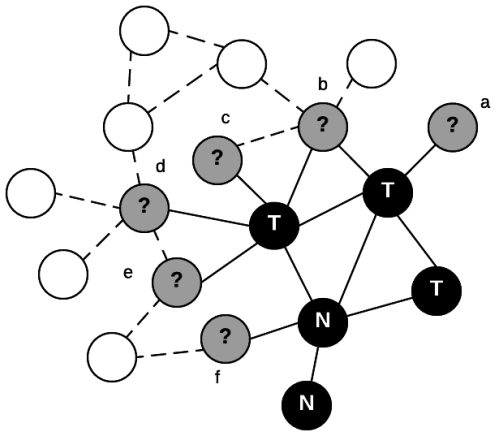

Figura 3.3: *Snapshot* de busca em um grafo após t = 5 passos. Os próximos vértices que podem ser explorados estão indicados pelas letras a-f.

3.4.1 H_1

Aplicando a heurística H_1 (Equação 3.5) a cada um dos vértices de a a f na Figura 3.3, temos que:

$$H_1(a) = H_1(c) = H_1(d) = H_1(e) = 0,6^1 \cdot 0,61^0 = 0,6$$

$$H_1(b) = 0,6^2 \cdot 0,61^0 = 0,36$$

$$H_1(f) = 0,6^0 \cdot 0,61^1 = 0,61$$

O vértice f obteve o maior valor para essa heurística. Portanto, ele deve ser escolhido como o próximo vértice a se explorar. Observe que o vértice em questão possui apenas um vizinho explorado e que esse vizinho não possui a característica. Assim o cálculo de seu H_1 se resume a probabilidade de um vértice ter a caracterítica dado que o seu único vizinho não tem, ou seja, $\overline{P}_D$.

Usando os valores estimados para $\overline{P}_T$ e $\overline{P}_D$, temos que:

$$H_1(a) = H_1(c) = H_1(d) = H_1(e) = 0,43^1 \cdot 0,67^0 = 0,43$$

$$H_1(b) = 0,43^2 \cdot 0,67^0 \approx 0,18$$

$$H_1(f) = 0,43^0 \cdot 0,67^1 = 0,67$$

O vértice com o maior valor permanece sendo o f. É importante notar que se $\overline{P}_T$ fosse maior que $\overline{P}_D$ o vértice escolhido seria qualquer um dentre a, c, d e e. O mesmo vale para $\overline{P}_T$ e $\overline{P}_D$ estimados.

3.4.2 H_2

Aplicando a heurística H_2 (Equação 3.6) a cada um dos vértices de a a f na Figura 3.3, temos que:

$$H_2(a) = H_2(c) = H_2(d) = H_2(e) = \mathbb{P}(T = 1|k_T = 1, k_N = 0) = \overline{P}_T = 0,6$$

$$H_2(b) = \mathbb{P}(T = 1|k_T = 2, k_N = 0)+\mathbb{P}(T = 2|k_T = 2, k_N = 0) = 2\overline{P}_T(1-\overline{P}_T)+\overline{P}_T{}^2 = 0,84$$

$$H_2(f) = \mathbb{P}(T = 1|k_T = 0, k_N = 1) = \overline{P}_D = 0,61$$

Para essa heurística, o vértice b obteve o maior valor. Logo, ele será o escolhido para exploração. Esse vértice é o único com mais de um vizinho explorado, além disso os seus dois vizinhos explorados tem a característica. Assim, o H_2 desse vértice é a probabilidade do vértice ter a característica por pelo menos uma das duas arestas. Exatamente o que faz H_3.

Usando os valores estimados para $\overline{P}_T$ e $\overline{P}_D$, temos que:

$$H_2(a) = H_2(c) = H_2(d) = H_2(e) = \overline{P}_T(4) = 0,43$$

$$H_2(b) = \mathbb{P}(T = 1|k_T = 2, k_N = 0)+\mathbb{P}(T = 2|k_T = 2, k_N = 0) = 2\overline{P}_T(4)(1-\overline{P}_T(4))+\overline{P}_T(4)^2 \approx 0.675$$

$$H_2(f) = \mathbb{P}(T = 1|k_T = 0, k_N = 1) = \overline{P}_D(4) = 0,67$$

O vértice b novamente apresenta o maior valor para a heurística, só que com uma diferença bem pequena para o vértice f.

3.4.3 H_3

Aplicando a heurística H_3 (Equação 3.7) a cada um dos vértices de a a f na Figura 3.3, temos que:

$$H_3(a) = H_3(c) = H_3(d) = H_3(e) = 1 - (1 - 0,6)^1(1 - 0,61)^0 = 0,6$$

$$H_3(b) = 1 - (1 - 0,6)^2(1 - 0,61)^0 = 0,84$$

$$H_3(f) = 1 - (1 - 0,6)^0(1 - 0,61)^1 = 0,61$$

Por conta do vértice b ter obtido o maior valor para essa heurística, ele será o próximo explorado.

Usando os valores estimados para $\overline{P}_T$ e $\overline{P}_D$, temos que:

$$H_3(a) = H_3(c) = H_3(d) = H_3(e) = 1 - (1 - 0,43)^1(1 - 0,67)^0 = 0,43$$

$$H_3(b) = 1 - (1 - 0,43)^2(1 - 0,67)^0 \approx 0,675$$

$$H_3(f) = 1 - (1 - 0,43)^0(1 - 0,67)^1 = 0,67$$

Nesse caso, o vértice escolhido continua sendo o b.

Para esse exemplo simples, todos os valores de H_3 foram iguais aos valores de H_2. Isso ocorreu pois todos os vértices de a a f possuem até dois vizinhos explorados. Assim, H_2 que considera o evento *'ter a característica'* ocorrer na maioria das arestas, acaba sendo equivalente a H_3, que considera o evento ocorrer em pelo menos uma das arestas. Para outros casos em que quantidade de vizinhos explorados for maior, naturalmente as heurísticas irão divergir.

Capítulo 4

Métodos Existentes

Diversos problemas envolvendo grafos podem ser resolvidos através de um processo de busca. Em parte deles é preciso visitar todos os vértices de um grafo, às vezes basta visitar um subconjunto de vértices. Fazer um percurso exaustivo pelos vértices pode ser a solução mais fácil mas não a mais inteligente. A maioria dos trabalhos na literatura que propõem uma solução para o encontro de vértices pertencentes a uma classe consideram que a topologia da rede é conhecida *a priori*. O que não é uma premissa do problema tratado neste trabalho. Para poder comparar o desempenho das nossas heurísticas escolhemos dois algoritmos de busca desinformados largamente utilizados, a **Busca em Largura** e a **Busca em Profundidade**, e também um algoritmo informado recentemente proposto para o escopo do nosso problema que surgiu da adaptação do ***Maximum Observed Degree*** [14].

4.1 Busca em Largura

Busca em largura (BFS) [10] é um método de busca desinformada em grafos. Sua busca é realizada com o auxílio de uma estrutura de dados chamada fila que com o seu mecanismo FIFO (*First In First Out*) garante que nenhum vértice seja explorado mais de uma vez e que os primeiros vértices a entrarem na fila sejam os primeiros a sairem. A BFS explora todos os vértices de um certo nível do grafo antes de passar a explorar os vértices do nível imediatamente abaixo. Por exemplo, se a busca partir de um vértice u, espera-se que todos os vizinhos de u sejam explorados antes de prosseguir para outros níveis. Esse processo se repete sistematicamente até a BFS explorar todos os vértices do grafo ou achar o vértice buscado. Na nossa implementação adaptamos a busca em largura para parar quando se atinge o orçamento.

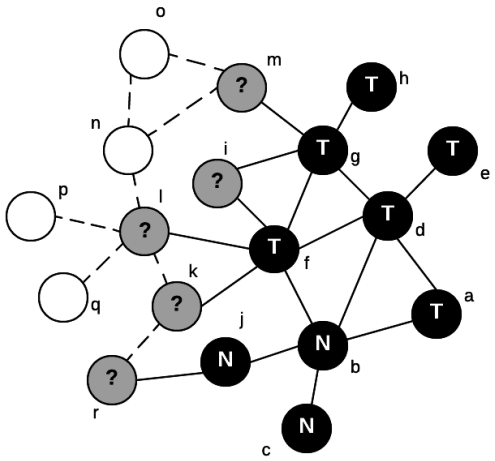

Figura 4.1: *Snapshot* da BFS em um grafo após t = 9 passos.

Exemplo

Consideremos que a BFS é iniciada no vértice a de um grafo com a mesma topologia representado na Figura 4.1 mas com todos os vértices inicialmente desconhecidos. E que temos um orçamento igual a 9, i.e., metade do total de vértices no grafo. Se a ordem de escolha das arestas para inserção dos vértices na fila se der da direita para esquerda, a sequência de vértices explorados e a dinâmica da fila ocorrem de forma que seus estados finais são:

`explorados = {a, d, b, e, g, f, j, c, h}; fila = {m, i, f, i, l, k, r}`

Para esse exemplo, a Figura 4.1 nos revela que a BFS conseguiu explorados 7 vértices com a característica dos 9 permitidos pelo orçamento, sendo eles os vértices a, d, e, g, f, j e h.

Complexidade

Para cada vizinho de um vértice explorado temos um custo $\mathcal{O}(1)$ para enfileirá-lo. Então, para cada vértice explorado temos um custo $\mathcal{O}(g)$, onde g é o grau do vértice. Considerando que o número de vértices explorados é no máximo igual a um orçamento b, a complexidade final da BFS é $\mathcal{O}(b \cdot g_{max})$, onde g_{max} é o maior grau na rede.

4.2 Busca em Profundidade

Busca em profundidade (DFS) é um algoritmo de busca desinformada que percorre e explora toda extensão do primeiro filho do vértice inicial u da árvore de busca até se atinigir um vértice folha (i.e, sem filhos) ou obtiver sucesso na busca. Caso falhe, a busca retrocede (*backtrack*) e recomeça no último vértice a ser explorado que ainda tiver filhos não explorados. Para esse *backtrack*, o algoritmo da DFS conta com o auxílio de uma pilha, uma estrutura de dados do tipo LIFO (*Last In First Out*). Também adaptamos a busca em profundidade para parar quando se atinge o orçamento.

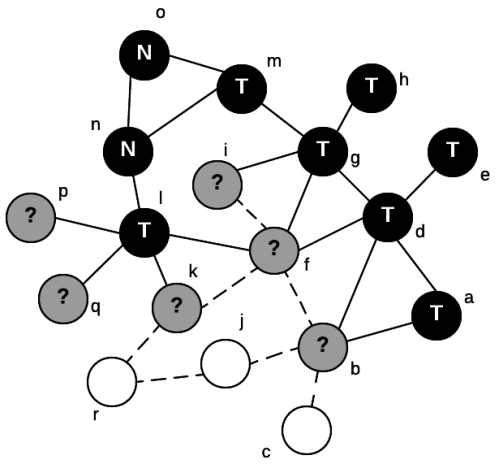

Figura 4.2: *Snapshot* da DFS em um grafo após t = 9 passos.

Exemplo

Consideremos que a DFS é iniciada no vértice a de um grafo com a mesma topologia do utilizado na BFS (Figura 4.2) e com os vértices incialmente desconhecidos. E que nosso orçamento é igual a 9. Dada que a ordem de escolha das arestas para inserção dos vértices na pilha é da esquerda para direita, a sequência de vértices explorados e a dinâmica da pilha ocorrem de forma que seus estados finais são:

`explorados = {a, d, e, g, h, m, o, n, l}; pilha = {b, b, f, f, i, n, k, q, p, f}`

Assim como a BFS, a DFS, conforme pode ser visto na Figura 4.2, conseguiu explorados 7 vértices com a característica dos 9 permitidos pelo orçamento. Mas não os mesmos, e sim os vértices a, d, e, g, h, m e l.

Complexidade

Para cada vizinho de um vértice explorado temos um custo $\mathcal{O}(1)$ para empilhá-lo. Então, para cada vértice explorado temos um custo $\mathcal{O}(g)$, onde g é o grau do vértice. Considerando que o número de vértices explorados é no máximo igual a um orçamento b, a complexidade final da DFS é $\mathcal{O}(b \cdot g_{max})$, o g_{max} é o maior grau na rede.

4.3 *Maximum Observed Degree**

O *Maximum Observed Degree* (MOD) é um algoritmo míope que busca maximizar a cobertura da rede a medida que explora os vértices de um grafo. Para tal, opta sempre por explorar aquele vértice que tiver o maior grau observado, i.e. o maior número de vizinhos explorados. Para a nossa avaliação, adaptamos a heurística do MOD para que a cada passo fosse explorado o vértice com o maior número de vizinhos com a característica. O que significa o vértice como o maior k_T, conforme as definições apresentadas no Capítulo 3. Todo o restante do algoritmo segue exatamente a lógica introduzida na Seção 3.1. Nos referenciamos a essa variação do MOD como MOD*.

Exemplo

Considere que temos um grafo com o estado da busca conforme apresentado na Figura 3.3. Aplicando a heurística do MOD* a cada um dos vértices de a a f, temos que:

$$k_T(a) = k_T(c) = k_T(d) = k_T(e) = 1$$

$$k_T(b) = 2$$

$$k_T(f) = 0$$

O vértice b, por ter obtido o maior valor para essa heurística, deve ser o próximo a ser explorado. Uma boa escolha pois b possui a característica (vide Figura 1.1).

Complexidade

Essa busca segue o mesmo algoritmo genérico das heurísticas propostas neste trabalho. Além disso, assim como H_1 e H_3 o cálculo de k_T também é $\mathcal{O}(1)$. Isso nos

leva a conclusão de que a complexidade do MOD* também é $\mathcal{O}(b \cdot (g_{max} + n))$, onde b é o orçamento e g_{max} é o maior grau no grafo.

Capítulo 5

Avaliação

Para a avaliação de desempenho do nosso algoritmo selecionamos quatro redes reais extraídas de *datasets* disponíveis na *web*. Sobre essas redes aferimos o número 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), testamos as buscas desinformadas, BFS e DFS, e a busca informada MOD*. Como essas três últimas buscas não utilizam parâmetros de entrada, também testamos as versões das nossas heurísticas em que os parâmetros são aprendidos ao decorrer da busca a partir dos nosso estimadores com inicialização igual a 1, chamada dyH_i.

Ainda testamos uma outra abordagem em que atribuímos o valor arbitrário 0,5 a $\overline{P}_T$ e $\overline{P}_D$ ($symH_i$). Para o caso de H_1 e H_3, essas atribuições degeneram as heurísticas a ponto delas não levarem mais em consideração a diferença de vizinhos com ou sem a características em seus cálculos, e sim, apenas a quantidade total de vizinhos explorados do vértice que se deseja calcular a probabilidade. Mais especificamente, H_1 passa a escolher sempre o vértice descoberto com menor número de vizinhos explorados e H_3 passa a escolher sempre o vértice com maior número de vizinhos explorados. Já no caso de H_2, $\overline{P}_T = \overline{P}_D = 0{,}5$ significa não considerar a diferença entre vizinhos T e N, e nem o total de vizinhos explorados, i.e, vértices descobertos com uma quantidade grande de vizinhos tem a mesma chance de descobertos com uma quantidade pequena de vizinhos.

Nessa avaliação utilizamos a linguagem de programação *Python* com o auxílio do módulo para manipulação e análise estatística de grafos chamado *graph-tool*[22]. O código-fonte pode ser acessado através do *link*: `https://github.com/freitaspedro/SearchOverGraphs/disser`.

5.1 Datasets

Os dois primeiros *datasets* adotados neste trabalho surgiram de um estudo sobre detecção de círculos sociais [7]. 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*[1]. 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. Para cada *egonet*, diversas características com variados valores para $\overline{P}_T$ e $\overline{P}_D$ e quantidade de positivos foram testadas. Foram testadas as *egonets* disponíveis nos *datasets* com maior quantidade de vértices tanto para *Facebook* quanto para *Google Plus*. A avaliação foi realizada sobre a maior componente conexa de ambos, que representam mais de 99% dos vértices das *egonets*.

O outro *dataset* utilizado, ***PolBlogs***, é parte de um estudo a respeito do comportamento de *blogs* políticos durante a eleição presidencial norte-americana de 2004 [8]. O *dataset* é composto por *hiperlinks* entre os *blogs*, 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). Na nossa avaliação buscamos pelos *blogs* conservadores trabalhando apenas com a maior componente conexa da rede que representa cerca de 82% da rede.

O nosso último *dataset* utilizado, ***DBLP***, é reponsável por armazenar cerca de 4 milhões de publicações de quase 2 milhões de autores na área de Ciência da Computação. Na rede de colaboração que montamos a partir desse *dataset*, os vértices representam os autores, e a ocorrência de uma aresta simboliza a ocorrência de pelo menos uma publicação em conjunto. As características extraídas de cada autor e inserida nos vértices são as suas participações em conferências através de suas publicações.

Na montagem, foram considerados apenas autores com publicações do tipo `inproceedings` publicadas por meio de `conf`, em outras palavras, com artigos em anais de conferências. Além disso, houve um processo de filtragem para o descarte de autores representados pelas expressões `et al.` e `anonymous`. Possíveis inconsistências em relação a homônimos não foram tratadas por acreditarmos não ser tão relevante para o objetivo final deste trabalho. Dessa rede construída, amostramos aleatoriamente 200 mil vértices e extraímos a maior componente conexa - contendo mais de 60 mil vértices - para rodarmos nossos experimentos.

Outras propriedades a respeito das redes finais testadas podem ser vistas na Tabela 5.1.

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

Dataset	**Vértices**	**Arestas**	**Grau Médio**	**Diâmetro**	**Clusterização Global**
Facebook	744	60046	161,413	7	0,697
Google Plus	4867	416988	171,353	6	0,248
PolBlogs	1222	19089	31,242	8	0,251
DBLP	63276	102416	3,237	29	0,145

Tabela 5.1: Propriedades das maiores componentes conexas das redes do *Facebook*, *Google Plus*, *PolBlogs* e *DBLP*.

5.2 Resultados

Nos nossos experimentos consideramos a média amostral e o desvio padrão da quantidade de vértices positivos explorados em 100 rodadas (50 para DBLP) com inicialização aleatória e uniforme. I.e., para cada rodada escolheu-se aleatoriamente um vértice dentre todos os vértices na rede com iguais chances. O orçamento foi aumentado gradualmente até que chegasse por volta da metade do total de vértices da rede testada. Em uma mesma rede, o mesmo conjunto de vértices iniciais escolhidos aleatoriamente foi mantido para todos os tipos de busca. Para todos os experimentos, as quantidades totais de vértices explorados atingiu os *orçamentos* disponíveis.

5.2.1 *Facebook*

Para a *egonet* do *Facebook*, exibimos a seguir os resultados obtidos para as características com *ids* '119', '91', '240' e '219'. Se tratam de características anônimas retiradas do *dataset* e os seus significados não são tão relevantes para o objetivo da nossa avaliação.

A distribuição dos tipos de arestas, as probabilidades $\overline{P}_T$ e $\overline{P}_D$ usadas como parâmetros do nosso modelo e quantidade total de positivos na rede para essas características podem ser vistas na Tabela 5.2.

Característica (*id*)	P_T	P_D	P_N	$\overline{P}_T$	$\overline{P}_D$	**Positivos**
119	0,047	0,231	0,722	0,060	0,832	63 (8%)
91	0,033	0,073	0,894	0,036	0,686	115 (15%)
240	0,045	0,311	0,644	0,064	0,874	139 (18%)
219	0,133	0,413	0,454	0,226	0,756	216 (29%)

Tabela 5.2: Distribuição dos tipos de arestas - *Facebook*.

Na Figura 5.1, temos os desempenhos dos algoritmos de busca para a característica '119'. Como podemos ver, H_1 tem desempenho pior até do que a BFS. Esse baixo desempenho de H_1 em grande parte é porque a sua equação (Equação

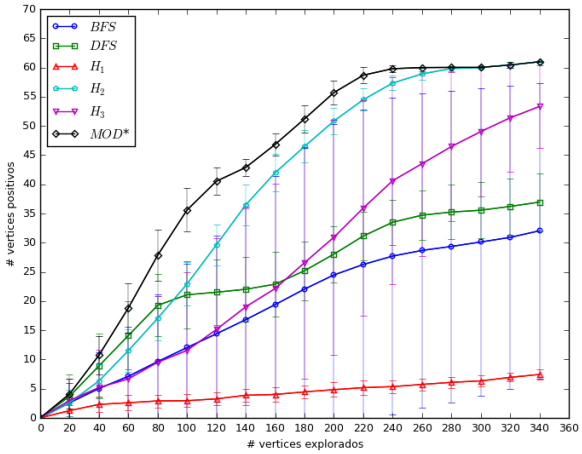

Figura 5.1: Desempenho dos algoritmos de busca para a característica '119' - *Facebook*

3.5) é muito sensível ao valores de k_T e k_N. Conforme esses valores crescem ela se aproxima rapidamente de 0, até chegar em um momento em que k_T e k_N são grandes o suficiente para transforma H_1 em um busca quase aleatória. E como a porcentagem de positivos para essa característica é apenas 8%, uma busca aleatória não poderia ter um resultado bom. Para H_3 ocorre justamente o oposto, conforme k_T e k_N crescem, sua equação (Equação 3.7) se aproxima de 1. Contudo, isso não impacta tanto o desempenho de H_3, que a partir de 160 vértices explorados tem um crescimento significativo, superando as buscas desinformadas. Como esperado a H_2 tem um bom desempenho, mas mesmo assim, em certos pontos, é surpreendentemente superado pelo MOD*, que possui uma heurística muito simplória. Contudo, essas duas buscas acabam por convergir a partir dos 280 vértices explorados, que representa cerca de 37% da rede.

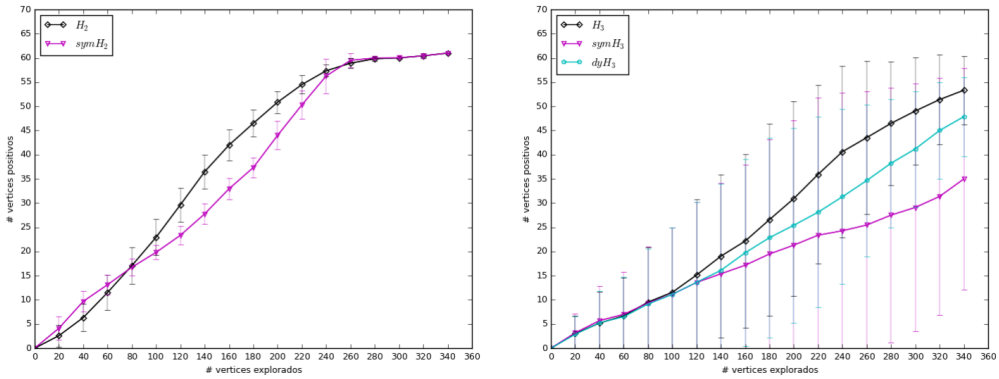

Figura 5.2: Desempenho de H_2 e H_3 e suas variações para a característica '119' - *Facebook*

Na Figura 5.2, temos o gráfico da esquerda com a mesma curva de H_2 apresentado no gráfico anterior e a sua variação em que utilizamos 0,5 como parâmetro do modelo, $symH_2$. Já o gráfico da direita replicamos a curva de H_3 e plotamos juntamente as suas variações $symH_3$ e dyH_3. Em ambos os gráficos as heurísticas com os parâmetros reais tiveram uma melhor peformance que as heurísticas com parâmetros arbitrários, atestando assim a importância de utilizarmos os parâmetros corretos no nosso modelo para esse cenário. O desempenho de dyH_3 ainda nos mostra que nossos estimadores fazem um bom trabalho, aproximando significativamente dyH_3 de H_3.

Para a característica '219', que possui uma porcentagem mais alta de positivos (29%), os desempenhos das buscas ficam muito mais próximos, conforme podemos ver no gráfico da Figura 5.3. Isso faz total sentido, pois quanto maior a densidade da característica na rede, menor a importância de uma heurística para guiar a busca.

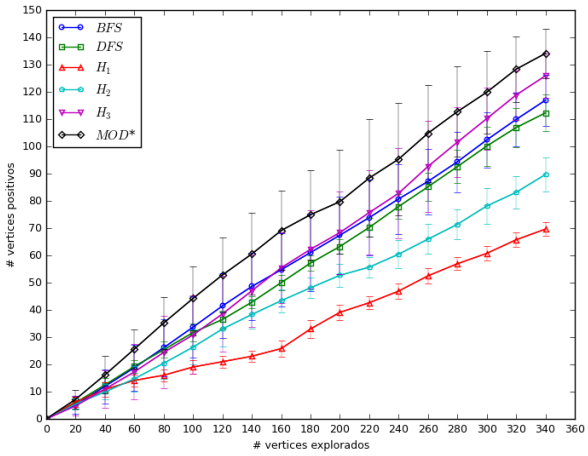

Figura 5.3: Desempenho dos algoritmos de busca para a característica '219' - *Facebook*

Na Figura 5.4, refizemos o mesmo processo da característica '119' para a característica '219', só que dessa vez dando destaque a H_1 e H_3. Novamente, os estimadores potencializaram as buscas com parâmetros dinâmicos para as duas heurísticas, atingindo assim o desempenho das heurísticas com parâmetros globais. Para $symH_1$ observamos uma leve superioridade, mas para $symH_3$ nada mudou.

Os resultados obtidos para as demais características são apresentados na Figura 5.5.

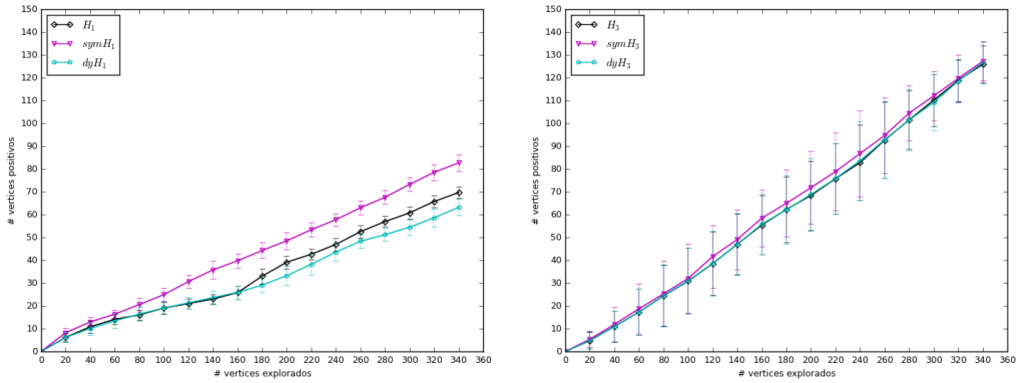

Figura 5.4: Desempenho de H_1 e H_3 e suas variações para a característica '219' - *Facebook*

5.2.2 *Google Plus*

A rede do *Google Plus* também é uma *egonet* assim como no caso do *Facebook* e por isso são muito parecidas estruturalmente. Porém, contendo quase 7 vezes mais vértices que a do *Facebook*, essa rede é importante para vermos como os algoritmos de busca escalam.

A distribuição dos tipos de arestas, as probabilidades $\overline{P}_T$ e $\overline{P}_D$ usadas como parâmetros do nosso modelo e quantidade total de positivos na rede para as características buscadas podem ser vistas na Tabela 5.3.

Característica (*id*)	P_T	P_D	P_N	$\overline{P}_T$	$\overline{P}_D$	**Positivos**
798	0,006	0,102	0,892	0,007	0,935	247 (5%)
862	0,007	0,127	0,866	0,007	0,949	367 (7%)
0	0,595	0,341	0,064	0,903	0,364	3522 (72%)

Tabela 5.3: Distribuição dos tipos de arestas - *Google Plus*.

No gráfico da Figura 5.6 temos os desempenhos dos algoritmos de busca para a característica '862' que ocorre em aproximadamente 7% dos vértices na rede do *Google Plus*. Assim como na característica '119' do *Facebook*, a melhor das nossas heurísticas é a H_2. As nossas outras duas heurísticas se misturam às buscas desinformadas, que de certa forma não ficam tão longe das com melhores desempenho. Isso mostra a dificuldade de nossas heurísticas encontrarem vértices com a característica em redes em que a fração de arestas do tipo TT é extremamente pequena (P_T = 0,006).

No gráfico da esquerda, na Figura 5.7, traçamos um comparativo entre H_2 e a sua variação $symH_2$. Como pode ser observado, essa heurística com os parêmetros iguais a 0,5, resultou em um melhor desempenho nessa rede. Ou seja, uma busca com

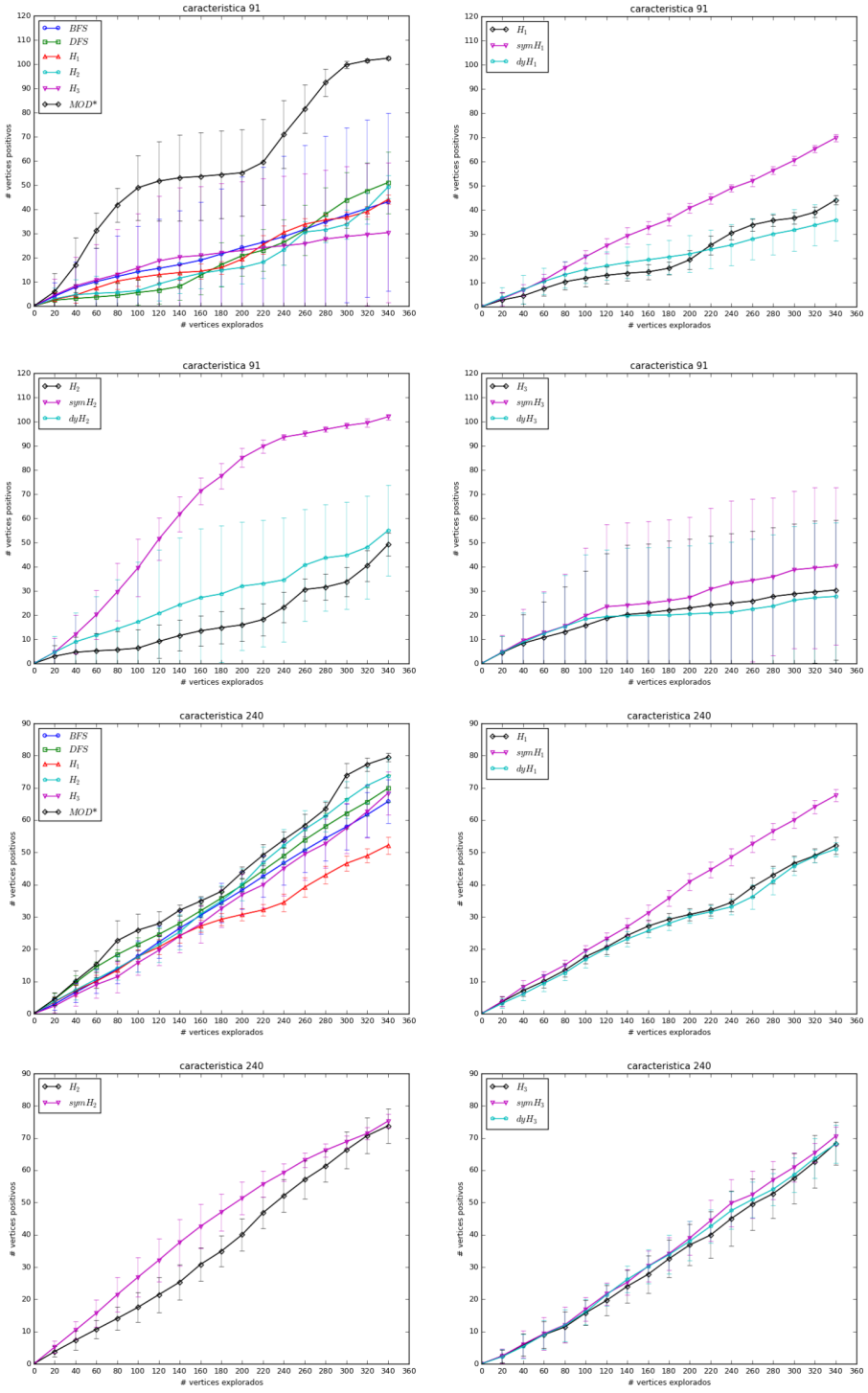

Figura 5.5: Resultados para as características '91' e '240' - *Facebook*

uma heurística que não difere entre vértices com mais ou menos vizinhos explorados tem um desempenho tão bom quanto o MOD* (Figura 5.6). Com isso, só podemos concluir que para essa rede as lógicas por trás das nossas heurísticas não conseguem extrair a essência necessária para melhor guiar a busca. Isso fica claro também ao

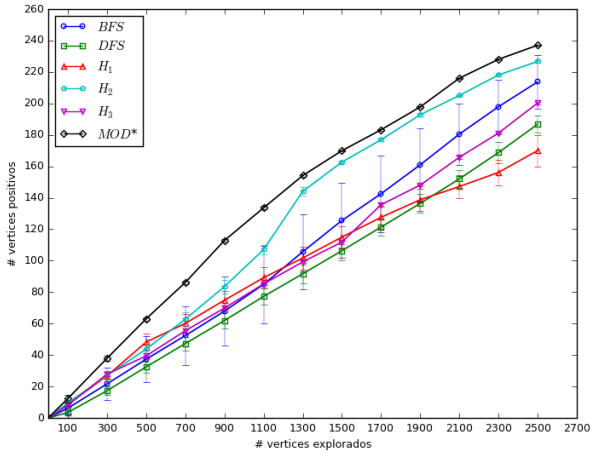

Figura 5.6: Desempenho dos algoritmos de busca para a característica '862' - *Google Plus*

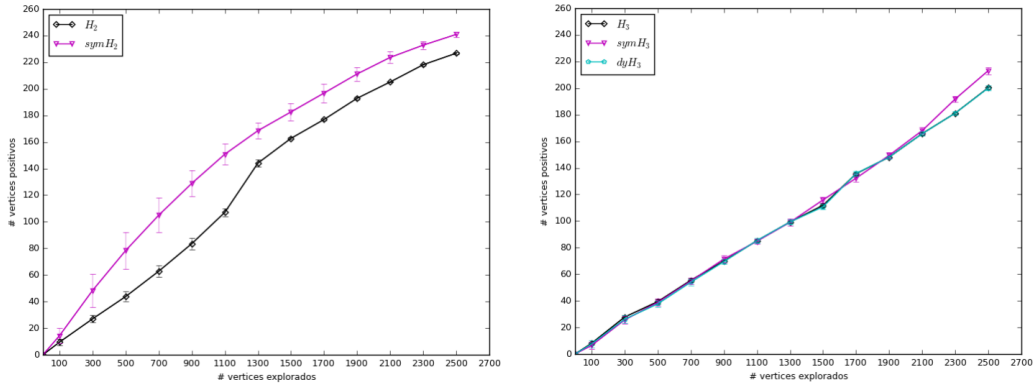

Figura 5.7: Desempenho de H_2 e H_3 e suas variações para a característica '862' - *Google Plus*

olharmos os desempenhos de H_3, $symH_3$ e dyH_3 praticamente iguais no gráfico a direita na Figura 5.7.

Propositalmente, pegamos uma outra característica com as propriedades (distribuição dos tipos de arestas, $\overline{P}_T$, $\overline{P}_D$ e positivos) bem próximas da característica anterior. O objetivo era mostrar a coerência do nosso modelo. O gráfico, na Figura 5.8, evidencia que os desempenhos relativos das nossas heurísticas se mantêm os mesmos do gráfico da característica '862'. Indicando assim que as nossas heurísticas ao se depararem com características parecidas (em suas propriedades) obtêm os mesmos desempenhos na busca.

Os gráficos com as variaçãos de H_2 e H_3, na Figura 5.9, reforçam a ideia de que nosso modelo é coerente. Inclusive, para as duas características é interessante observar que as $symH_2$ supera o desmepenho do MOD*.

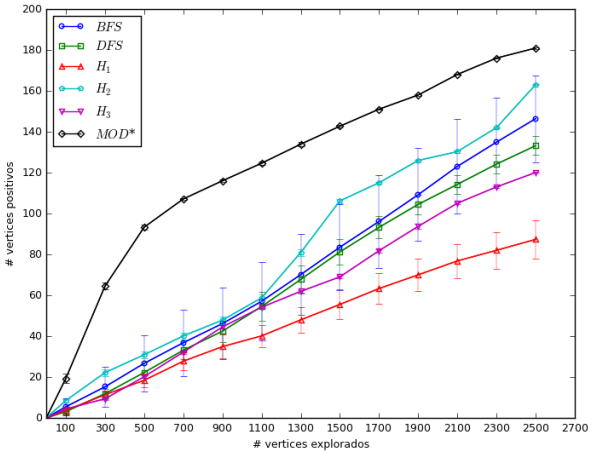

Figura 5.8: Desempenho dos algoritmos de busca para a característica '798' - *Google Plus*

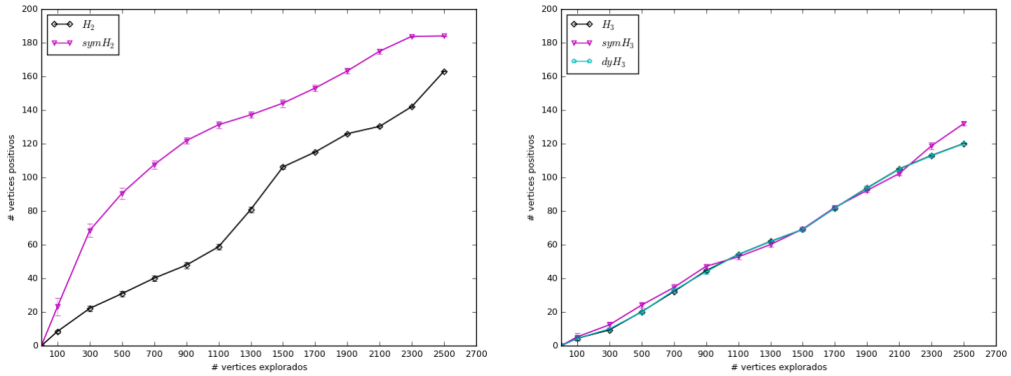

Figura 5.9: Desempenho de H_2 e H_3 e suas variações para a característica '798' - *Google Plus*

O fenômeno de buscas com desempenhos iguais para características abundantes fica mais evidente para a característica '0' (Figura 5.10)

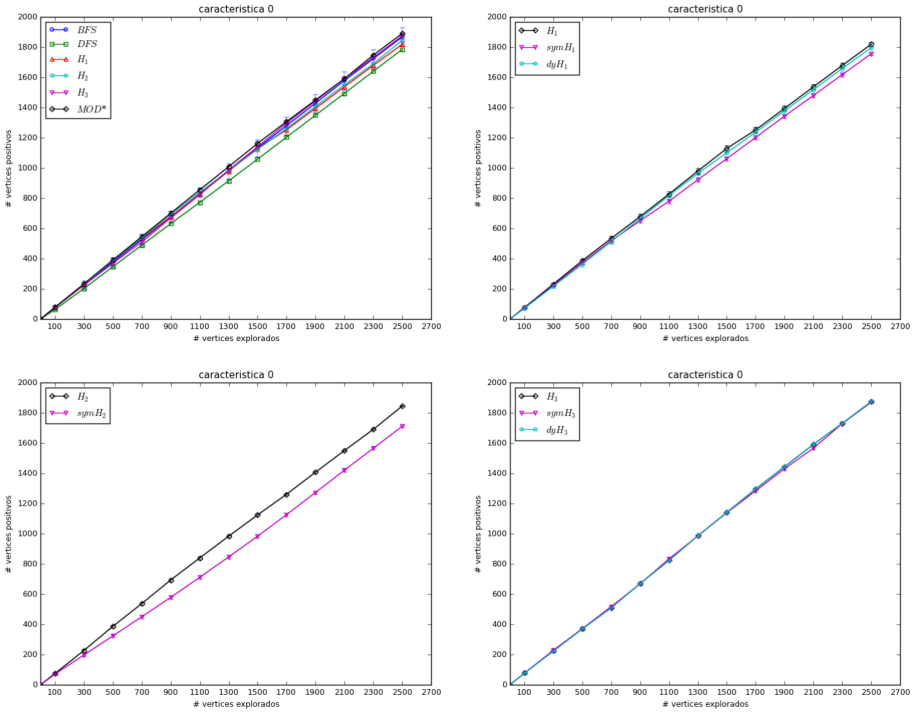

Figura 5.10: Resultados para a característica '0' - *Google Plus*

5.2.3 *PolBlogs*

Como dito anteriormente, *PolBlogs* apresenta apenas uma característica, representando a inclinação política de um *blog*. A busca foi realizada sobre os *blogs* conservadores, que são um pouco mais que a metade de todos os *blogs* na rede. Os resultados nessa rede são importantes pois confirmam o impacto da distribuição dos tipos de arestas em nosso modelo. Indicando principalmente que uma das nossas heurísticas consegue tomar decisões acertadas quando os valores de P_T e P_N são altos comparados a P_D. Outro motivo para apresentarmos os resultados dessa rede é que ela é mais "rica" que as anteriores, no sentido de que ela não é centrada em apenas um indivíduo.

A distribuição dos tipos de arestas, as probabilidades $\overline{P}_T$ e $\overline{P}_D$ usadas como parâmetros do nosso modelo e quantidade total de positivos na rede pela característica 'conservadores' podem ser vistas na Tabela 5.4.

Característica	P_T	P_D	P_N	$\overline{P}_T$	$\overline{P}_D$	**Positivos**
Conservadores	0,471	0,089	0,440	0,516	0,158	636 (52%)

Tabela 5.4: Distribuição dos tipos de arestas - *PolBlogs*.

Nessa rede, os diversos algoritmos buscando por *blogs* conservadores - que representam 52% da rede - conseguem manter um certo padrão a medida que o orçamento

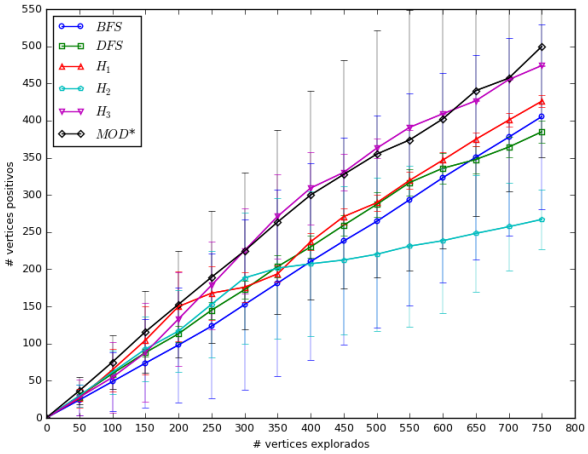

Figura 5.11: Desempenho dos algoritmos de busca - *PolBlogs*

é aumentado. Com exceção da H_2 que, após um início promissor, tem uma queda acentuada de rendimento na exploração de vértices positivos a partir dos 300 vértices explorados. Contudo, a nossa outra heurística, a H_3, consegue manter um aproveitamento em média de 70%. O fato dessa rede ser formada por dois grandes *clusters* - um com os *blogs* liberais e outro com os *blogs* conservadores - e a busca ser feita em cima dos *blogs* conservadores, 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* conservadores 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.

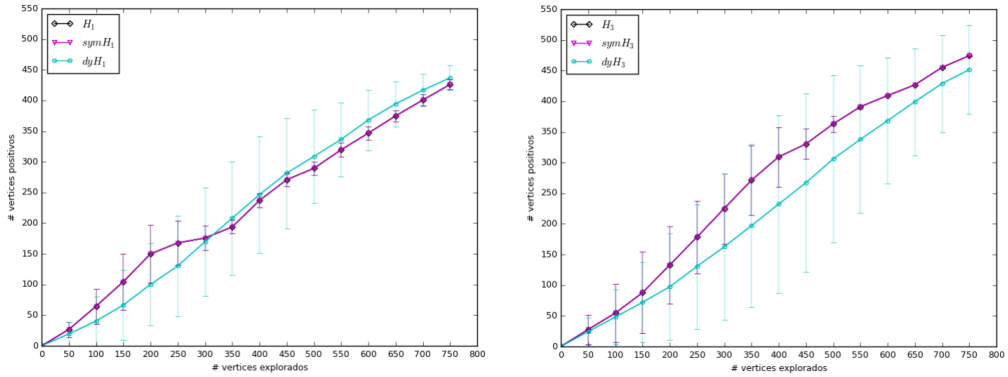

Figura 5.12: Desempenho de H_1 e H_3 e suas variações - *PolBlogs*

Nos gráficos da Figura 5.12, destacamos H_1 e suas variantes (dyH_1 e $symH_1$) e a direita H_3 e suas variantes (dyH_3 e $symH_3$). Neles é mostrado um empate entre as duas heurísticas e suas variações em que são utilizados $\overline{P}_T$ e $\overline{P}_D$ iguais a 0,5. Mas quando olhamos para as versões em que é aprendido esses parâmetros há uma inversão de desempenhos de uma heurística para a outra. De qualquer forma, em ambos os casos, ocorre uma boa aproximação com as heurísticas com os parâmetros globais corretos.

5.2.4 *DBLP*

A rede do *DBLP* sobre a qual performamos os algoritmos de busca é significativamente maior que as outras redes deste trabalho. Tanto em relação ao número de vértices quanto a variedade de características dos vértices presentes nessa rede. Uma consequência direta disso é a baixíssima densidade de positivos para a maioria das características na rede. Consideramos como característica a participação dos autores em alguns congresso, que pudessem apresentar grandes diferenças na quantidade de autores. Sendo a primeira característica responsável por sinalizar aqueles autores com publicações no "*Workshop on Graph-based Methods for Natural Language Processing (TextGraphs)*", a segunda representando aqueles que publicaram em "*IEEE International Conference on Peer-to-Peer Computing (P2P)*", a terceira no "*IEEE Wireless Communications and Networking Conference (WCNC)*", para as duas últimas consideramos uniões de diferentes conferências que tivessem alguma interseção (Tabela 5.5 e Tabela 5.6).

Característica	**Positivos**
IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)	3120 (4,9%)
IEEE International Conference on Image Processing (ICIP)	2433 (3,8%)
International Conference on Multimedia Computing and Systems (ICMCS)	1632 (2,5%)
Conference of the International Speech Communication Association (INTERSPEECH)	1530 (2,4%)
International Conference on Human-Computer Interaction (HCI)	1657 (2,6%)
International Conference on Human Factors in Computing Systems (CHI)	1764 (2,7%)
International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)	1062 (1,6%)
Grupo 1	9767 (15,4%)

Tabela 5.5: Conferências pertencentes ao *Grupo 1*.

A distribuição dos tipos de arestas, as probabilidades $\overline{P}_T$ e $\overline{P}_D$ usadas como parâmetros do nosso modelo e quantidade total de positivos na rede para as carac-

Característica	**Positivos**
IEEE International Symposium on Circuits and Systems (ISCAS)	2174 (3,4%)
IEEE International Conference on Robotics and Automation (ICRA)	2124 (3,3%)
IEEE/RJS International Conference on Intelligent RObots and Systems (IROS)	2132 (3,3%)
IEEE International Conference on Systems, Man and Cybernetics (SMC)	1528 (2,4%)
IEEE Vehicular Technology Conference (VTC)	1849 (2,9%)
European Signal Processing Conference (EUSIPCO)	1248 (1,9%)
Grupo 2	8757 (13,8%)

Tabela 5.6: Conferências pertencentes ao *Grupo 2*.

terísticas escolhidas podem ser vistas na Tabela 5.7.

Característica	P_T	P_D	P_N	$\overline{P}_T$	$\overline{P}_D$	**Positivos**
TextGraphs	$1,95 \times 10^{-5}$	0,001	0,999	$1,95 \times 10^{-5}$	0,976	18 (0,02%)
P2P	0,001	0,012	0,987	0,001	0,937	135 (0,2%)
WCNC	0,017	0,097	0,886	0,019	0,847	1483 (2%)
Grupo 2	0,113	0,267	0,620	0,153	0,703	8757 (13%)
Grupo 1	0,135	0,279	0,586	0,187	0,673	9767 (15%)

Tabela 5.7: Distribuição dos tipos de arestas - *DBLP*.

Como pode ser visto no gráfico da Figura 5.13, para a primeira característica, *TextGraphs*, com presença extremamente baixa a H_3 se destacou de todas as outras buscas. É um resultado interessante pois como $\overline{P}_T$ e $\overline{P}_D$ são também extremamente pequenos a equação dessa heurística (Equação 3.7) acaba não atribuindo valores tão diferentes assim aos vértices descobertos, de qualquer forma essas diferenças sutis são capazes de melhor guiar a busca. O que já não é verdade para as nossas outras heurísticas que têm desempenhos equivalentes as buscas desinformadas e ao MOD*. Esse último revela ter muitas dificuldades em encontrar vértices com uma característica rara na rede mesmo com uma quantidade muito grande de vértices explorados.

Na Figura 5.14 destacamos a H_3 e as duas variações a fim de tentar elucidar o porquê dela conseguir esse desempenho. Imediatamente, podemos ver que ela tem um comportamento muito parecido com a sua versão que não difere entre vértices com mais ou menos vizinhos na hora de atribuir uma probabilidade ($symH_3$). Isso revela uma natureza quase aleatória que também poderia se aplicar as nossas duas outras heurísticas, mas que por algum motivo não trivial, os desempenhos delas não são nem de longe tão bons quanto o da H_3.

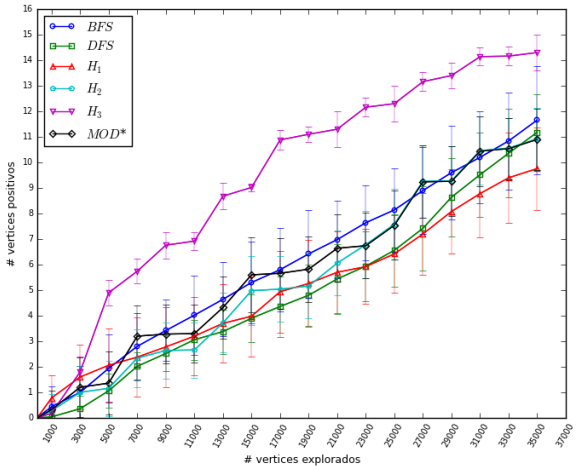

Figura 5.13: Desempenho dos algoritmos de busca - *TextGraphs/DBLP*

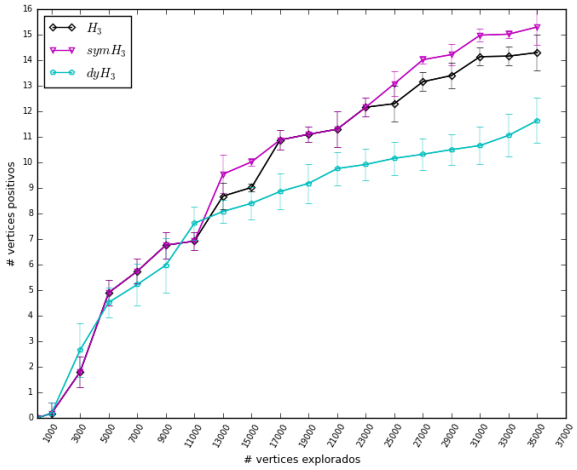

Figura 5.14: Desempenho de H_3 e suas variações - *TextGraphs/DBLP*

Com quase 100 vezes mais positivos que o *TextGraphs*, em *WCNC* (Figura 5.15) a H_3 já consegue obter um bom desempenho, ficando abaixo das buscas desinformadas assim como a nossa H_1. Por outro lado, a H_2 consegue aproveitar melhor a rede descoberta para encontrar os positivos, obtendo assim o segundo melhor desempenho dentre todos os algoritmos de busca.

Destacamos a melhor das nossas heurísticas nesse cenário no gráfico da Figura 5.16, comparamos com a $symH_2$ e o que podemos observar foi que os seus desempenhos são equivalentes.

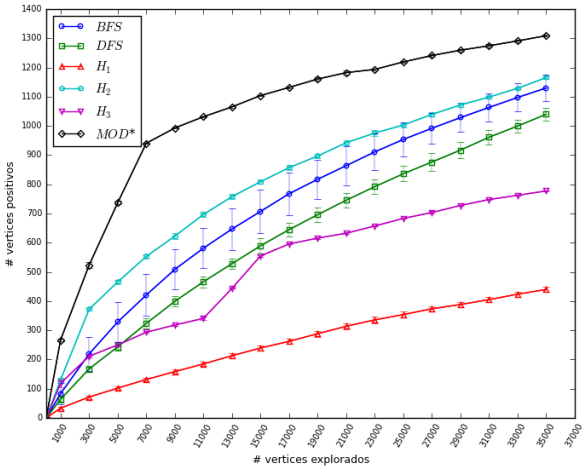

Figura 5.15: Desempenho dos algoritmos de busca - *WCNC/DBLP*

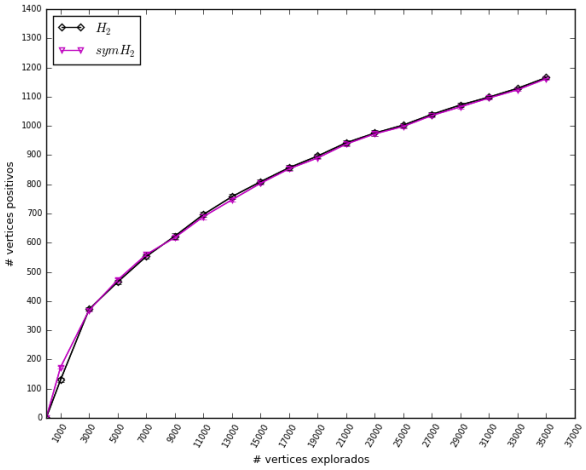

Figura 5.16: Desempenho de H_2 e suas variações - *WCNC/DBLP*

A Figura 5.17 reforça a hipótese já observada para a característica anterior, de que, para essa rede, as nossa heurísticas não tiram proveito suficiente da rede descoberta quando temos uma quantidade grande de positivos. O que nos leva a crer que a aproximação realizada pela modelagem matemática parece não ser boa, e por isso as nossas heurísticas falham.

H_1 quando comparada as suas variações (Figura 5.18), observa-se que ao empregarmos os parâmetros simétricos ($\overline{P}_T = \overline{P}_D = 0{,}5$) essa heurística tem uma melhora signicativa, se equiparando à H_2 (melhor das nossas heurísticas) e as buscas desinformadas.

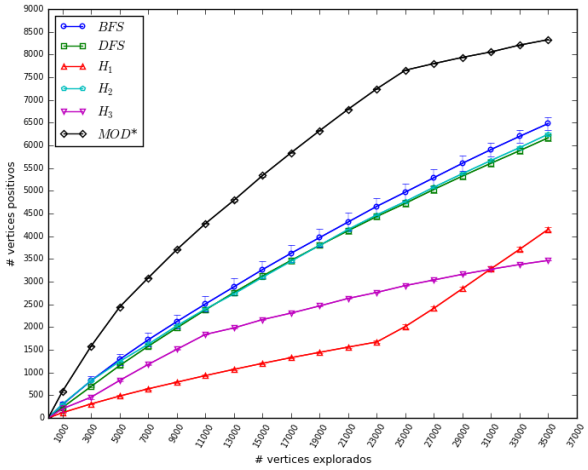

Figura 5.17: Desempenho dos algoritmos de busca - *Grupo 1/DBLP*

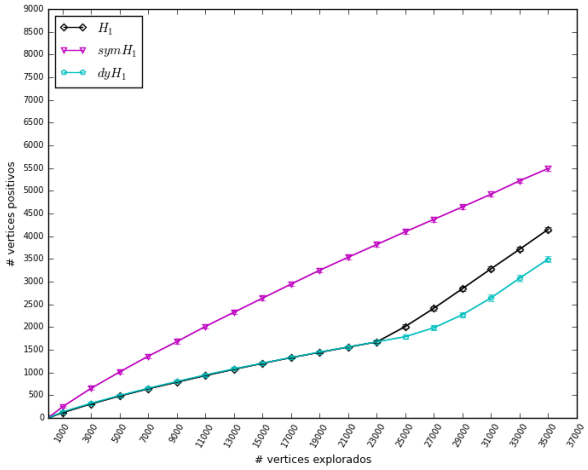

Figura 5.18: Desempenho de H_1 e suas variações - *Grupo 1/DBLP*

Os comportamentos das buscas para *P2P* e as conferências no *Grupo 2*, na Figura 5.19, são análogos ao que já vimos nas outras características.

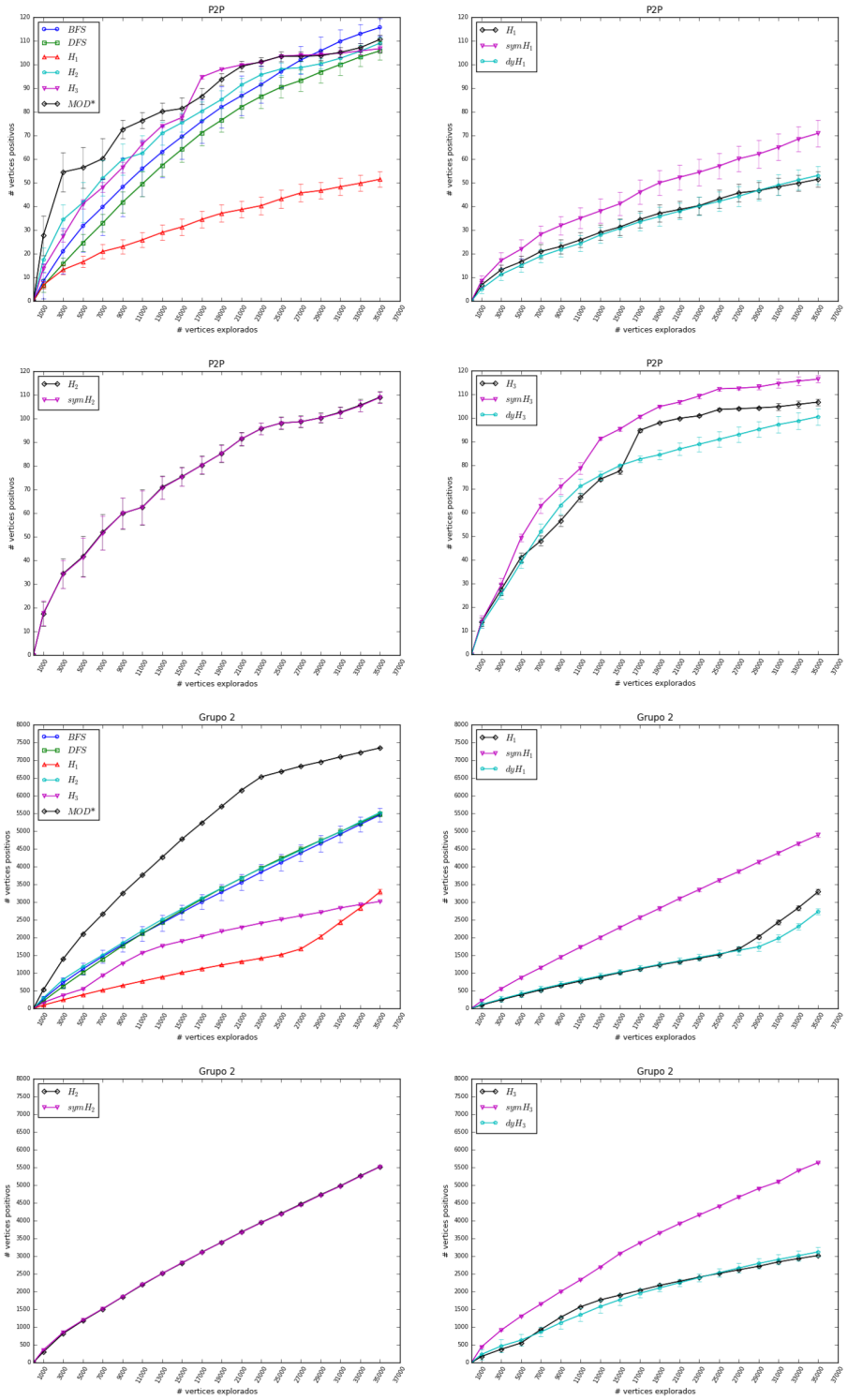

Figura 5.19: Resultados - *P2P* e *Grupo 2/DBLP*

5.2.5 *Resumo*

A seguir, na Tabela 5.8, apresentamos um ranqueamento dos desempenhos das buscas para cada característica de cada uma das redes experimentadas. A par do que vimos anteriormente, para classificar o desempenho de uma busca olhou-se não só para a quantidade de vértices positivos encontrados com o orçamento final, mas também o aproveitamento para todos os diferentes orçamentos. Para certas características, que se encontram assinaladas, não obtivemos os resultados para dyH_2, devido ao alto tempo de processamento.

Rede/Característica		**Positivos**	**1ª**	**2ª**	**3ª**
Facebook	119†	63 (8%)	$MOD*$	H_2	H_3
	91	115 (15%)	$symH_2$	$MOD*$	$symH_1$
	240†	139 (18%)	$symH_2$	$MOD*$	H_2
	219	216 (29%)	$MOD*$	H_3	BFS
Google Plus	798†	245 (5%)	$symH_2$	$MOD*$	H_2
	862†	367 (7%)	$symH_2$	$MOD*$	H_2
	0†	3522 (72%)	BFS	H_3	dyH_3
PolBlogs	Conservadores	636 (52%)	$MOD*$	H_3	$symH_3$
DBLP	TextGraphs†	18 (0,02%)	$symH_3$	H_3	dyH_3
	P2P†	135 (0,2%)	$symH_3$	$MOD*$	H_3
	WCNC†	1483 (2%)	$MOD*$	H_2	$symH_2$
	Grupo 2†	8757 (13%)	$MOD*$	H_2	DFS
	Grupo 1†	9767 (15%)	$MOD*$	BFS	H_2

Tabela 5.8: Ranqueamento das buscas nas redes do *Facebook*, *Google Plus*, *PolBlogs* e *DBLP*. † características sem resultados para dyH_2.

Capítulo 6

Conclusões

Neste trabalho 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. Apresentamos três variações do modelo que são utilizadas para guiar uma busca informada pela rede. Onde o objetivo do algoritmo de busca é encontrar o maior número de vértices com uma determinada característica dentro de um número limitado de vértices que possam ser explorados. Testamos a dependência do nosso modelo em relação aos parâmetros, substituíndo esses parâmetros por valores arbitrários e analisando a sua performance. Uma outra abordagem foi fazer a busca aprender esses parâmetros iterativamente.

A avaliação do algoritmo em quatro redes reais - buscando em cada uma delas características diferentes - 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*, cuja heurística é simples e pode ser calculada rapidamente, foi melhor nas *egonets* de redes sociais e na rede *DBLP* para características mais presentes, ocupando os primeiros e segundos lugares. Mas a nossa H_2, seja na sua versão original ou através de sua variação simétrica, também desempenhou um bom papel nas redes sociais. Já na rede do *DBLP*, para características muito escassas, uma das variações do modelo proposto ($symH_3$) teve desempenho bem superior as demais buscas, cenário no qual MOD* tem desempenho equivalente a das buscas desinformadas. Ocorreram casos ainda em que diversos algoritmos de busca empataram.

Os resultados empíricos mostraram que as nossas heurísticas apesar de ter uma fundamentação matemática podem ter abstraído informações importantes no passo em que realizam a aproximação do cenário real para um cenário com eventos independentes. Em alguns casos, as heurísticas acabaram tendo desempenho inferior às suas próprias versões com escolha quase aleatória.

Por fim, concluimos 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 mistura diversos métodos distintos em uma única proposta [2].

Deixamos como trabalho futuro o melhor entendimento de características da rede que possam ser exploradas para montar um modelo mais condizente com a realidade a fim de elucidar um algoritmo generalizado que possa maximizar os vértices positivos encontrados.

Referências Bibliográficas

[1] FREITAS, P., CARDOSO, V., IACOBELLI, G., et al. "Desenvolvimento de um Algoritmo de Busca por Vértices Específicos em Redes". In: *Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance)*, 2017.

[2] MURAI, F., RENNÓ, D., RIBEIRO, B., et al. "Selective Harvesting over Networks", *CoRR*, v. abs/1703.05082, 2017. Disponível em: <`http://arxiv.org/abs/1703.05082`>.

[3] "IBGE — Cidades — Rio de Janeiro — Armação dos Búzios". `http://cidades.ibge.gov.br/xtras/perfil.php?codmun=330023`. Último acesso em 26 de Maio de 2017.

[4] "Facebook: global daily active users 2017 — Statistic". `https://www.statista.com/statistics/346167/facebook-global-dau/`. Último acesso em 26 de Maio de 2017.

[5] MCPHERSON, M., SMITH-LOVIN, L., COOK, J. M. "Birds of a feather: Homophily in social networks", *Annual review of sociology*, v. 27, n. 1, pp. 415–444, 2001.

[6] BARABÁSI, A.-L. *Network science*. Cambridge university press, 2016.

[7] LESKOVEC, J., KREVL, A. "SNAP Datasets: Stanford Large Network Dataset Collection". `http://snap.stanford.edu/data`, jun. 2014.

[8] ADAMIC, L. A., GLANCE, N. "The political blogosphere and the 2004 US Election". In: *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

[9] LEY, M. "DBLP: some lessons learned", *Proceedings of the VLDB Endowment*, v. 2, n. 2, pp. 1493–1500, 2009.

[10] MOORE, E. F. *The shortest path through a maze*. New York: Bell Telephone System, 1959.

[11] TARJAN, R. "Depth-first search and linear graph algorithms", *SIAM journal on computing*, v. 1, n. 2, pp. 146–160, 1972.

[12] SPITZER, F. *Principles of random walk*, v. 34. Springer Science & Business Media, 2013.

[13] RUSSEL, S., NORVIG, P. "Artificial Intelligence: A Modern Approach", *EUA: Prentice Hall*, 2003.

[14] AVRACHENKOV, K., BASU, P., NEGLIA, G., et al. "Pay Few, Influence Most: Online Myopic Network Covering". In: *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, 2014.

[15] WANG, X., GARNETT, R., SCHNEIDER, J. "Active Search on Graphs". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 731–738. ACM, 2013.

[16] PFEIFFER, J. J., NEVILLE, J., BENNETT, P. "Active sampling of networks". In: *Workshop on Mining and Learning with Graphs*, 2012.

[17] BNAYA, Z., PUZIS, R., STERN, R., et al. "Bandit algorithms for social network queries". In: *Social Computing (SocialCom), 2013 International Conference on*, pp. 148–153. IEEE, 2013.

[18] MA, Y., HUANG, T.-K., SCHNEIDER, J. G. "Active Search and Bandits on Graphs using Sigma-Optimality." In: *Conference on Uncertainty Artificial Inteligence*, pp. 542–551, 2015.

[19] GARNETT, R., KRISHNAMURTHY, Y., XIONG, X., et al. "Bayesian optimal active search and surveying". In: *International Conference on Machine Learning, ACM*, pp. 1239–1246, 2012.

[20] XIE, P., ZHU, J., XING, E. P. "Diversity-promoting bayesian learning of latent variable models". In: *International Conference on Machine Learning*, 2016.

[21] BERGER, J. O. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[22] "graph-tool: Efficient network analysis". `https://graph-tool.skewed.de/`. Último acesso em 8 de Junho de 2017.