Análise de Algoritmos de Clusterização para Experimentos Randomizados em Redes Sociais de Larga Escala*

Bruno Dematttos Nogueira, Francisco Galuppo Azevedo, Fabricio Murai, Ana Paula Couto da Silva

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brazil

{bruno.dematttos, franciscogaluppo, murai, ana.coutosilva}@dcc.ufmg.br

Abstract. *Large companies conduct A/B tests to estimate the effect of changes in their websites. In these tests, users are randomly redirected to one of two versions of the site. However, in social networks, users that access different versions can influence each other if they are linked, making estimation more difficult. To minimize this interference, graph partitioning algorithms were proposed to find clusters of well-connected users (e.g. ϵ-net and FENNEL). All users within a cluster are redirected to the same version. In this work, we propose a parallel variant of ϵ-net and a new algorithm dubbed NoMAS, inspired on FENNEL. We present a theoretical analysis of the proposed algorithms' scalability complemented by empirical results on the estimation accuracy.*

Resumo. *Grandes empresas realizam testes A/B para estimar o efeito de mudanças nos seus websites. Nestes testes, usuários são redirecionados aleatoriamente para uma de duas versões do site. Porém, em redes sociais, usuários que acessam diferentes versões podem influenciar uns aos outros se estiverem relacionados, dificultando a estimação. Para minimizar esta interferência, foram propostos algoritmos para particionar a rede em clusters de usuários bem conectados (ϵ-net e FENNEL). Todos os usuários dentro de um cluster são redirecionados para uma mesma versão. Neste trabalho, propomos uma versão paralela do ϵ-net e um novo algoritmo chamado NoMAS, inspirado no FENNEL. Apresentamos uma análise teórica da escalabilidade dos algoritmos complementada por resultados empíricos sobre a acurácia da estimação.*

1. Introdução

Conduzir pesquisas de experiência dos usuários é peça-chave na tomada de decisões para grandes companhias que oferecem serviços na Internet (como Amazon, e-Bay, Facebook, Google e LinkedIn) [Kohavi et al. 2013]. Desde simples mudanças no *layout* das páginas até alterações substanciais em algoritmos de ranqueamento de conteúdo devem ser avaliadas de forma objetiva para determinar se estas são de fato melhores que o *status quo*. Algumas métricas relevantes são: tempo de carregamento das páginas, nível de engajamento dos usuários, número médio de cliques por usuário, lucro obtido através de anúncios etc.

*The authors' work has been partially funded by the EUBra-BIGSEA project by the European Commission under the Cooperation Programme (MCTI/RNP 3rd Coordinated Call), Horizon 2020 grant agreement 690116, CAPES, CNPq and FAPEMIG.

A avaliação do efeito de mudanças em um serviço oferecido pela Internet precisa ser feita através de experimentos controlados. Em particular, é importante considerar o mesmo período de medição ao comparar estatísticas obtidas para os dois tratamentos, pois podem existir fatores não-controlados relacionados ao tempo (ex.: o site aparece em manchete de jornal entre experimentos). Além disso, as populações expostas a cada uma das versões precisam ser equivalentes. Por exemplo, uma mudança bem recebida por um grupo de usuários que trabalham com TI pode não ser positiva para o restante.

Uma técnica clássica para realizar experimentos controlados são os testes A/B, que consistem em experimentos randomizados com duas variantes de um tratamento [Kohavi et al. 2009]. Em testes clínicos, parte dos indivíduos são selecionados aleatoriamente para receber um placebo, enquanto o restante recebe o medicamento cujo efeito deseja-se avaliar. No contexto da Internet, usuários que visitam um website são aleatoriamente redirecionados à versão atual ou à versão sendo testada.

Os testes A/B têm como premissa a *Stable Unit Treatment Value Assumption* (SUTVA), que assume que o comportamento de um indivíduo irá depender apenas do tratamento que lhe foi atribuído, ou seja, não há interferência entre os indivíduos. Enquanto esta suposição é válida para testes clínicos, experimentos em redes sociais normalmente a violam (efeito conhecido como *spillover*) [Xu et al. 2015]. Considere, por exemplo, um experimento para avaliar a criação de um chat para troca de mensagens entre usuários. Mesmo que as diferenças entre os grupos de controle e de tratamento não sejam significativas quando apenas 20% dos usuários tem acesso à funcionalidade, não podemos afirmar que essa conclusão continuaria válida caso 100% dos usuários pudessem usar o chat.

Recentemente, muitas abordagens foram propostas para estimar o efeito médio do tratamento (ATE) quando a SUTVA não é válida [Middleton and Aronow 2011, Backstrom and Kleinberg 2011, Katzir et al. 2012, Ugander et al. 2013, Gui et al. 2015]. A maioria delas utilizam algoritmos de clusterização para encontrar pequenos conjuntos de usuários bem conectados entre si. Nesse caso, a randomização é feita a nível de cluster: todos os usuários do mesmo cluster recebem o mesmo tratamento. O objetivo é emular dentro dos clusters os cenários em que toda a população está no controle ou no tratamento.

Algoritmos de clusterização para experimentos randomizados em redes sociais devem atender dois requisitos principais. Primeiramente, devem ser escaláveis. Uma forma de alcançar este objetivo é através da paralelização, utilizando, por exemplo, frameworks baseados em Map-Reduce como Hadoop ou Spark. Apesar disso, não existe na literatura uma descrição paralela do ϵ-net, um dos algoritmos que compõem o estado-da-arte nesta área [Ugander et al. 2013]. Como segundo requisito, devem resultar em pequenos erros na estimação do ATE. Para controlar o erro causado por diferenças nos tamanhos dos clusters, algumas abordagens, tais como a proposta em [Saveski et al. 2017], usam algoritmos que resultam em clusters de tamanhos semelhantes [Nishimura and Ugander 2013]. Esta restrição pode acabar aumentando o número de arestas entre clusters. É crucial entender em que cenários é vantajoso o uso de tais algoritmos.

Este trabalho apresenta três contribuições principais. A primeira é a proposta de uma versão paralela do algoritmo de clusterização ϵ-net. Como segunda contribuição, propomos o NoMAS, um algoritmo que lê o grafo que representa a rede como um *stream* de dados e, analisando cada nó uma única vez, determina um particionamento em clusters de

tamanho aproximadamente igual. Finalmente, avaliamos os algoritmos propostos quanto à escalabilidade e impacto na erros de estimação do ATE. O estudo de escalabilidade consiste na derivação de resultados analíticos complementados por resultados empíricos. O impacto dos algoritmos na qualidade das estimativas é feito através de simulações de modelos de comportamento de usuário sobre diferentes redes[1].

2. Trabalhos Relacionados

O estudo da resposta a um tratamento na presença de interações sociais é formalizado em [Manski 2013]. O autor considera a existência de pequenos grupos de indivíduos conectados, cujas respostas são fortemente dependentes. Neste caso, a randomização completa (i.e., a nível individual), não tem o poder estatístico para identificar o efeito do tratamento. Uma alternativa é particionar a população usando técnicas de detecção de comunidades. Este é um problema bem estudado, para o qual uma miríade de técnicas já foram propostas [Fortunato 2010]. Recentemente, os autores de [Eckles et al. 2014] mostraram que técnicas baseadas na otimização de uma métrica global, como a modularidade de Newman [Newman 2006], não são capazes de distinguir comunidades pequenas. Por esta razão, os autores argumentam que elas não são adequadas para testes A/B em redes.

Em [Backstrom and Kleinberg 2011] os autores assumem que um indivíduo é considerado "exposto" ao tratamento se ele e pelo menos τ de seus vizinhos recebem o tratamento. Para aumentar o tamanho da amostra efetiva dada uma porcentagem fixa da população a receber o tratamento, o método proposto realiza, a partir de um conjunto inicial de nodos, passeios aleatórios que tendem a fechar triângulos. Os nodos iniciais e visitados formam o grupo de tratamento. Infelizmente, este método não é escalável. Um abordagem similar é proposta em [Katzir et al. 2012], cuja diferença é permitir que o tratamento seja atribuído probabilisticamente. Neste trabalho, assim como outros em que o tratamento de um nodo é uma variável aleatória [Aronow and Samii 2012], a estimação é baseada nos estimadores Horvitz-Thompson e/ou Hajek.

Em [Gui et al. 2015], os autores propõem um modelo linear para a resposta de um nodo em função do tratamento dele e de seus vizinhos, permitindo o uso de todas as amostras. Para reduzir o impacto de supor incorretamente tal relação linear, os autores usam randomização a nível de cluster. Em particular, eles propõem um algoritmo que resulta em clusters de mesmo tamanho. A ideia do algoritmo é analisar cada par de vértices e trocar a etiqueta deles se houver uma redução no número de arestas entre partições. Portanto, devem ser analisados $|\mathcal{V}|^2$ pares e, para cada um deles, deve se descobrir quantos vizinhos há em cada partição para quantificar o ganho (ou perda), o que inviabiliza a aplicação desse algoritmo em redes com muitos nodos, como as descritas na introdução.

O algoritmo ϵ-net proposto em [Ugander et al. 2013] encontra um particionamento da rede onde cada nodo está a menos que ϵ saltos do centro de sua partição. Além disso, os centros das partições são nodos que distam pelo menos ϵ saltos entre si. Todos os nodos do grafo são inicializados como não-marcados. Um nodo v é escolhido uniformemente dentre os não-marcados para ser o centro de uma partição. Todos os nodos a uma distância menor que ϵ de v são marcados. Os dois últimos passos são repetidos até que todos os nodos sejam marcados. Ao final do processo, cada nodo u é associado à partição cujo centro está mais próximo de u (empates são resolvidos arbitrariamente). Apesar da

[1] As implementações estão disponíveis em `github.com/settifoglio/Redes`

simplicidade, a descrição do ϵ-net é inerentemente sequencial, inviabilizando seu uso em redes de larga escala.

Em [Nishimura and Ugander 2013], os autores propõem o FENNEL, um algoritmo para particionar um grafo lido como um *streaming* de dados. Todas as partições são inicializadas como conjuntos vazios. Ao analisar um nodo v, o método calcula uma pontuação associada a atribuir v a cada uma das partições e escolhe aquela que tem a maior pontuação. Essa pontuação é calculada descontando, do número de vizinhos do nodo que estão na partição, um termo que cresce de acordo com o tamanho da comunidade. Este desconto ou penalidade tem como objetivo fazer com que as comunidades tenham tamanhos semelhantes. Mais precisamente, o nodo v é colocado na partição $\arg\max_{i\in\{1,2,\ldots,\kappa\}} |P_i^t \cap N(u)| - \alpha\frac{\gamma}{2}(|P_i^t|)^{\gamma-1}$, onde P_i^t é o subconjunto de nodos no cluster i no passo t, $N(u)$ é o conjunto dos vizinhos do vértice v e α e γ são parâmetros dados como entrada. O único caso estudado é aquele em que $\gamma = 2$ (i.e., termo de penalidade é reduzido a $\alpha|P_i^t|$). Este algoritmo é utilizado no método proposto em [Saveski et al. 2017] para testar a hipótese de que a SUTVA é válida. Embora outro algoritmo de particionamento pudesse ter sido utilizado, o FENNEL é um dos mais escaláveis.

3. Algoritmos de clusterização propostos

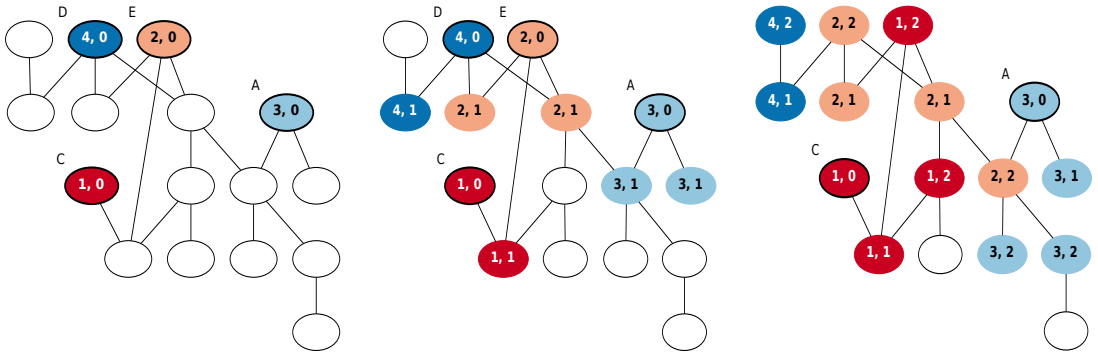
A seguir descrevemos os dois algoritmos de clusterização propostos neste trabalho. Estes algoritmos visam o particionamento de uma rede para a execução de experimentos randomizados, não sendo, portanto, algoritmos gerais para detecção de comunidades.

3.1. Algoritmo ϵ-net paralelo

Nesta seção propomos uma implementação paralela escalável do ϵ-net baseada no paradigma Map-Reduce, que é o bloco fundamental de frameworks distribuídos usados em redes sociais online de escala global tais como o Facebook e o LinkedIn. O Giraph e o Spark-GraphX são os exemplos mais proeminentes dos frameworks para executar operações em grafos. Esses frameworks mapeiam nodos, arestas, além de dados associados aos nodos e arestas para hosts de um cluster de computação. Além dos *joins* e filtros típicos das aplicações baseadas em Map-Reduce, esses frameworks permitem a execução de operações baseadas em *message passing* síncrono através das arestas do grafo.

A utilização de memória e o volume de dados transmitidos são restrições fundamentais nesses frameworks. Por essa razão, no algoritmo proposto, apenas um par (`etiqueta`, `distancia`) será armazenado por nodo, e as mensagens transmitidas tem essa mesma estrutura. Ao final de cada passo do *message passing*, uma operação de **reduce** é usada para sumarizar as mensagens recebidas. Este sumário poderá ser usado para atualizar a informação contida no nodo.

A chave para paralelizar o ϵ-net é constatar que, uma vez que os **centros dos clusters** sejam determinados, existe uma operação de *message passing* em tempo constante que associa os nodos do grafo aos seus respectivos clusters. Portanto, iremos focar em paralelizar a busca por centros dos clusters. Em particular, é crucial verificar de maneira eficiente quais nodos de uma dada lista de nodos são clusters válidos, i.e., consistentes com o algoritmo sequencial. Por exemplo, na Figura 1(a), consideramos a lista formada pelos nodos C, E, A e D. No algoritmo sequencial com $\epsilon = 3$, a escolha de C como centro de partição inviabiliza E como possível centro. Note que não é preciso determinar de uma



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$

Figura 1. Exemplo de chamada à PROPAGAETIQUETAS com `field=etiqueta` e $\epsilon = 3$. As sementes são C, E, A e D. No interior de cada nodo, x, y representa respectivamente a etiqueta sendo propagada e a distância em relação ao seed. No passo $t = 2$, observamos que: C é um centro válido, E foi invalidado por C, A é um centro válido e D deve ser testado novamente na próxima iteração, pois foi invalidado pela semente inválida E.

única vez uma lista de centros cujos discos irão cobrir o grafo: podemos encontrar mais centros repetidamente até que o grafo esteja completamente coberto.

Denominamos **sementes** os nodos que compõem uma lista $\mathbf{s}$, a partir da qual deseja-se extrair um suconjunto $\mathcal{C}$ de centros de clusters válidos. Para isso, estabelecemos uma ordem de precedência entre as sementes usando as posições $I(s)$ em que as sementes $s \in \mathbf{s}$ aparecem (posição anterior indica maior precedência). Desta forma, podemos propagar pelo grafo etiquetas que correspondem aos índices $I(s)$, dentro de um raio $\epsilon - 1$. Neste caso, o sumário das mensagens recebidas por um nodo é a mensagem cuja etiqueta é mínima. O conteúdo do sumário irá substituir os dados associados ao nodo se a etiqueta presente no sumário for menor. Na Figura 1(b), observamos que, após o primeiro passo da propagação, os vizinhos comuns a D e E mantiveram apenas a informação sobre E.

Ao final do processo de propagação, iremos analisar as sementes em ordem de precedência. As sementes válidas são aquelas que mantiveram a etiqueta inicial; as inválidas, possuem uma etiqueta que corresponde a uma semente válida. É possível ainda que uma semente X tenha sido invalidada por uma semente Y que foi, por sua vez, invalidada por uma semente Z. No exemplo da Figura 1(c), $X = D$, $Y = E$, $Z = C$. Ao invés de concluir que X é válida (dado que Y não é), optamos por interromper a análise da lista de sementes, pois é possível que existam outras sementes que teriam sido invalidadas por X, caso Y não estivesse em $\mathbf{s}$.

O Algoritmo 1 descreve o algoritmo ϵ-net paralelo sobre um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Na linha 1, uma lista $\mathbf{v}$ é criada embaralhando-se os índices dos nodos em $\mathcal{V}$. Em seguida, inicializa-se a variável $\mathcal{D}$, que irá mapear os nodos a pares cluster-distância. Os clusters serão identificados pelos índices dos seus centros. O laço das linhas 4-18 é repetido até que todos os nodos tenham sido mapeados. Na linha 5, a função OBTEMSEMENTES varre a lista $\mathbf{v}$ e retona a lista $\mathbf{s}$ contendo os k primeiros nodos ainda não-mapeados (i.e., $v : \mathcal{D}(v) = (\texttt{NULL}, \epsilon)$). Em seguida, inicializa-se a variável I, que mapeia os nodos de $\mathbf{s}$ aos índices $1, \ldots, |\mathbf{s}|$. Esses índices são usados para criar $\mathcal{G}'$, uma cópia de $\mathcal{G}$, usada

Algoritmo 1: ϵ-NET PARALELO encontra um particionamento do grafo

Input: grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, distância ϵ, grau de paralelismo k
Output: mapa $\mathcal{D}$, onde $\mathcal{D}(v) =$ (etiqueta do cluster ao qual v pertence, distância ao centro do cluster) para todo $v \in \mathcal{V}$

1 $\mathbf{v} \leftarrow$ EMBARALHAR$(\mathcal{V})$
2 $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E}, \mathcal{D})$ onde $\mathcal{D}(v) = (\texttt{NULL}, \epsilon)$ para todo $v \in \mathcal{V}$
3 $n \leftarrow 0$ `// número de clusters encontrados`
4 **while** *existe* $v \in \mathcal{V}$ *tal que* $\mathcal{D}(v) = (\mathit{NULL}, \epsilon)$ **do**
5 $\quad \mathbf{s} \leftarrow$ OBTEMSEMENTES$(\mathbf{v}, \mathcal{D}, k)$
6 $\quad$ Inicialize $I(s_i) = n + i$ para $\mathbf{s} = (s_1, \ldots, s_{|S|})$
7 $\quad \mathcal{G}' \leftarrow (\mathcal{V}, \mathcal{E}, \mathcal{D}')$ onde $\mathcal{D}'(v) = \begin{cases} (I(v), 0.0) & \text{se } v \in \mathbf{s} \\ (\texttt{NULL}, \epsilon) & \text{caso contrário} \end{cases}$
8 $\quad \mathcal{D}'' \leftarrow$ PROPAGAETIQUETAS$(\mathcal{G}', \epsilon, \mathit{etiqueta})$ `// passo paralelizado`
9 $\quad \mathcal{C} \leftarrow \emptyset$ `// inicializa lista de centros`
10 $\quad$ **for** $s \in \mathbf{s}$ **do**
11 $\quad\quad$ **if** $\mathcal{D}'(s) = \mathcal{D}''(s)$ **then**
12 $\quad\quad\quad \mathcal{C} \leftarrow \mathcal{C} + \{s\}$
13 $\quad\quad$ **else if** $\mathbf{s}[\mathcal{D}''(s).first] \notin \mathcal{C}$ **then**
14 $\quad\quad\quad$ **break** `// foi invalidada por outra inválida`
15 $\quad n \leftarrow n + |\mathcal{C}|$
16 $\quad \mathcal{G}'' \leftarrow (\mathcal{V}, \mathcal{E}, \mathcal{D}'')$ onde $\mathcal{D}''(v) = \begin{cases} (v, 0.0) & \text{se } v \in \mathcal{C} \\ (\texttt{NULL}, \epsilon) & \text{caso contrário} \end{cases}$
17 $\quad \mathcal{D}''' \leftarrow$ PROPAGAETIQUETAS$(\mathcal{G}'', \epsilon, \mathit{distancia})$ `// passo paralelizado`
18 $\quad \mathcal{D}(v) = \begin{cases} \mathcal{D}'''(v) & \text{se } \mathcal{D}'''(v).second < \mathcal{D}(v).second \\ \mathcal{D}(v) & \text{caso contrário} \end{cases}$
19 **return** $\mathcal{D}$

como entrada para a primeira chamada ao Procedimento PROPAGAETIQUETAS. Este procedimento retorna $\mathcal{D}''$, o mapeamento obtido ao se propagar a informação em $\mathcal{D}'$ por uma distância $\epsilon - 1$, repassando-se apenas as mensagens cuja `etiqueta` (índice do cluster) é mínima. O laço das linhas 10-14 calcula $\mathcal{C}$, o subconjunto de sementes de $\mathbf{s}$ que serão os centros dos novos clusters. Caso a etiqueta associada à semente s em $\mathcal{D}''$ corresponda a uma semente que não está em $\mathcal{C}$ (i.e., uma semente inválida), o laço é interrompido na linha 14. Na linha 16, um novo grafo é criado para a propagação dos índices dos novos clusters, que se dá na linha 17. Desta vez, são propagadas as etiquetas dos centros encontrados até o momento, repassando-se apenas as mensagens cuja `distancia` (até o cluster) é mínima. Por fim, atualiza-se o mapeamento de nodos aos clusters na linha 18.

O Procedimento propagaEtiquetas é baseado no paradigma *message passing* síncrono. As linhas 1 e 2 indicam que apenas as sementes ou centros irão trocar mensagens no passo $t = 0$. As linhas 3-8 descrevem como as mensagens recebidas serão processadas, gerando, potencialmente, novas mensagens a serem tratadas no passo seguinte. O processo termina quando nenhuma mensagem é trocada.

3.2. NoMAS: novo algoritmo de clusterização baseado em streaming

O NoMAS (**No**rmalized Intersection **M**inimization **A**lgorithm for Graph **S**treams) é o segundo algoritmo de clusterização proposto neste trabalho. Sua principal vantagem é

Procedimento propagaEtiquetas

Input: grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D})$, distância ϵ, campo $field \in$ {etiqueta,distancia} usado para sumarizar mensagens
Output: mapa $\mathcal{D}$

if $\mathcal{D}(v).second = 0.0$ **then**
  Envie $m = (\mathcal{D}(v).first, 1.0)$ a todos os vizinhos em $\mathcal{G}$

Seja $\mathcal{M}(v)$ o conjunto de mensagens recebidas por v no passo anterior
if $field = etiqueta$ **then** $m^\star \leftarrow \arg\min_{m \in \mathcal{M}(v)}\{m.first\}$
else $m^\star \leftarrow \arg\min_{m \in \mathcal{M}(v)}\{m.second\}$
if $(field = etiqueta$ *and* $\mathcal{D}(v).first > m^\star.first)$ *or* $(field = distancia$ *and* $\mathcal{D}(v).second > m^\star.second)$ **then**
  $\mathcal{D}(v) \leftarrow m^\star$
  Envie $m = (\mathcal{D}(v).first, \mathcal{D}(v).second + 1)$ a todos os vizinhos em $\mathcal{G}$
Quando nenhuma mensagem for trocada, **return** $\mathcal{D}$

permitir controlar o compromisso entre obter partições de tamanhos semelhantes e a qualidade dessas partições, medida segundo a fração de arestas que ligam nodos em partições diferentes. Este algoritmo é baseado no modelo de *streaming* de grafos e pode ser facilmente paralelizado, características que o tornam altamente escalável.

Diferente do ϵ-net, o número de partições κ obtidas pelo NoMAS é fixo, fornecido como entrada. O modelo de streaming assume que a ordem em que os vértices são analisados é dada por um caminhamento em largura (BFS) em um grafo direcionado[2], assim como foi feito por [Stanton and Kliot 2012]. Neste modelo, a comunidade a que um nodo pertence deve ser determinada no momento em que a BFS o explora. Apesar do algoritmo ser sequencial, é possível fazer com que ele seja executado em um cluster computacional aplicando-se a mesma ideia usada em [Nishimura and Ugander 2013] para paralelizar os algoritmos FENNEL e LDG.

Assim como o FENNEL, o algoritmo NoMAS determina a qual partição um nodo será associado a partir de uma regra simples. Seja $P_i^t = \{P_1^t, \cdots, P_\kappa^t\}$ um particionamento do grafo no passo t, onde P_i^t é o conjunto de vértices do cluster i, $N(u)$ é o conjunto de vizinhos do vértice u e α um parâmetro fornecido com entrada. O vértice u será membro da comunidade cujo índice maximiza a pontuação, i.e.

$$\arg\max_{i \in \{1,2,\ldots,\kappa\}} \frac{\max(1, |P_i^t \cap N(u)|)}{|P_i^t|^\alpha} \tag{1}$$

Em caso de empate entre comunidades, o nodo será colocado naquela de menor tamanho.

O Algoritmo 2 descreve o NoMAS aplicado a um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Na linha 1, o vetor $\mathcal{C}$ de tamanho $|\mathcal{V}|$ é inicializado. Nas linhas 2-4, k nodos uniformemente aleatórios são selecionados sem repetição e mapeados às partições 1 a k. No laço das linhas 5-12, é feita uma busca em largura pelo grafo onde cada nodo é avaliado. Na linha 6, é inicializado o vetor $\mathbf{Coeficiente}$ de tamanho k, que irá armazenar a pontuação de cada comunidade em relação ao vértice u. Nas linhas 7 e 8, incrementa-se a pontuação da comunidade a que cada vizinho de u está associado. A pontuação final de cada comunidade

[2]Caso o grafo seja não-direcionado, cada aresta original $(u, v) \in \mathcal{E}$ deve ser transformada em duas arestas direcionadas, (u, v) e (v, u).

Algoritmo 2: NoMas encontra um particionamento do grafo

Input: grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, distância ϵ, número de partições k
Output: mapa $\mathcal{C}$, onde $\mathcal{C}(v)$ indica o cluster ao qual v pertence, $\forall v \in \mathcal{V}$

1 $\mathcal{C} \leftarrow [-1, \ldots, -1]$ `// inicializa partição de cada nó`
2 $\mathbf{v} \leftarrow$ EMBARALHAR$(\mathcal{V})$
3 **for i** $\in 1, \ldots, k$ **do**
4 $\quad \mathcal{C}[\mathbf{v}[\mathbf{i}]] \leftarrow \mathbf{i}$ `// associa nó aleatório a cada partição`
5 **for** $u \in$ BFS$(\mathcal{G}, \mathbf{v}[1])$ **do**
6 $\quad$ **Coeficiente** $\leftarrow [0, 0, \ldots, 0]$ `// pontuação de cada partição`
7 $\quad$ **for** $s \in N(\mathbf{u})$ **do**
8 $\quad\quad$ **Coeficiente**$[\mathcal{C}[s]] \leftarrow$ **Coeficiente**$[\mathcal{C}[s]] + 1$
9 $\quad$ **for i** $\in 1, \ldots, \kappa$ **do**
10 $\quad\quad$ **Coeficiente**$[\mathbf{i}] \leftarrow \max(1,$ **Coeficiente**$[\mathbf{i}])$
11 $\quad\quad$ **Coeficiente**$[\mathbf{i}] \leftarrow$ **Coeficiente**$[\mathbf{i}]/$TAMANHOPARTIÇÃO$(\mathbf{i})^{\alpha}$
12 $\quad \mathcal{C}[u] \leftarrow \arg\max_{\mathbf{i} \in 1, \ldots, \kappa}$ **Coeficiente**$[\mathbf{i}]$ `// Equação (1)`
13 **return** $\mathcal{C}$

é calculada nas linhas 9-11. Finalmente, a partição do nodo u é escolhida na linha 12.

O motivo pelo qual o algoritmo funciona é simples e similar ao que explica o êxito do LDG e do FENNEL. Por um lado, as comunidades que possuem mais vizinhos com o vértice sendo analisado terão um numerador maior. Por outro, comunidades que já possuem muitos nodos são penalizadas, o que favorece o aparecimento de comunidades com tamanho similar. O parâmetro α determina se a divisão será feita priorizando comunidades de tamanho similares ou poucas arestas cortadas.

4. Avaliação dos algoritmos de clusterização

Nesta seção iremos avaliar os algoritmos ϵ-net paralelo e NoMAS em relação à escalabilidade e a acurácia na estimação do ATE usando três datasets de redes reais.

4.1. Datasets

Utilizamos três redes sociais (nodos representam indivíduos) disponíveis na coleção de datasets SNAP, de Stanford[3]. A Tabela 1 mostra as estatísticas básicas delas:

- **Bitcoin OTC:** Rede de pessoas que fizeram transações utilizando o Bitcoin. Uma aresta direcionada (i, j) indica que i confia em j (*who-trusts-whom*).
- **Enron Email:** Rede formada a partir de cerca 500 mil emails trocados dentro da companhia Enron. Nodos são endereços de email e arestas não-direcionadas (i, j) indicam que pelo menos um email foi enviado de i para j ou de j para i.
- **Wiki-Vote:** Rede que representa a votação entre colaboradores da Wikipedia para a escolha de administradores das páginas. Nodos na rede representam colaboradores e uma aresta direcionada (i, j) indica que o usuário i votou no usuário j.

4.2. Análise de escalabilidade

Nesta seção iremos avaliar a escalabilidade dos algoritmos de clusterização propostos segundo os requisitos de memória, custo de transmissão de mensagens e tempo de execução. Para o ϵ-net, estudamos o caso em que $\epsilon = 3$, pois este é o valor utilizado na literatura.[4]

[3] http://snap.stanford.edu/

[4] Ademais, $\epsilon = 2$ considera apenas vizinhos imediatos e $\epsilon > 3$ tende a gerar clusters de tamanho $O(n)$.

Dataset	Total de Nodos	Total de Arestas
Bitcoin OTC	5.881	35.592
Enron Email	36.692	183.831
Wiki-vote	7.115	103.689

Tabela 1. Total de nodos e arestas das redes em cada dataset.

Algoritmo ϵ-net paralelo com $\epsilon = 3$

Custo de memória. O ϵ-net paralelo é projetado para frameworks distribuídos onde os vértices são mapeados para hosts de um cluster $\mathcal{H}$. Cada host $h \in \mathcal{H}$ é responsável por um subconjunto $\mathcal{V}_h \subset \mathcal{V}$ e pelo conjunto de arestas $\mathcal{E}_h$ adjacentes a vértices em $\mathcal{V}_h$. Para a execução do algoritmo, é necessário armazenar um par (`etiqueta`,`distancia`) para cada vértice. Portanto, o requisito mais estrito de memória é dado pelo número de arestas armazenadas em um host. Havendo memória suficiente para carregar o grafo no cluster $\mathcal{H}$, haverá memória suficiente para executar o ϵ-net paralelo.

Custo de transmissão. Derivar resultados analíticos para o custo de transmissão no caso de grafos gerais é muito complicado. Por essa razão, faremos a análise apenas para o caso de grafos regulares, que será complementada com resultados empíricos para os datasets utilizados neste artigo. Suponha um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ escolhido uniformemente dentre todos os grafos d-regulares com $|\mathcal{V}|$ vértices. Na primeira chamada à PROPAGAETIQUETAS, são enviadas mensagens de cada semente $s \in \mathbf{s}$ para seus vizinhos. Seja $\mathcal{N}_i$ a variável aleatória que denota a cardinalidade do conjunto de vizinhos dos nodos em $\mathbf{s}$ quando $|\mathbf{s}| = i$. Em particular, sabemos que $\mathcal{N}_1 = d$ e que $\mathcal{N}_i \leq di$ com probabilidade 1. No caso geral, o valor esperado dessa v.a. é dado pela relação de recorrência homogênea

$$\mathrm{E}[\mathcal{N}_i] = \mathrm{E}[\mathcal{N}_{i-1}] + d - d\frac{\mathrm{E}[\mathcal{N}_{i-1}]}{|\mathcal{V}|}. \tag{2}$$

Note que estamos apenas acrescentando d e descontando os vizinhos da nova semente que são adjacentes aos nodos em $\mathcal{N}_{i-1}$. A solução da recorrência em (2) é dada por

$$\mathrm{E}[\mathcal{N}_i] = |\mathcal{V}| \left(1 - \left(1 - \frac{d}{|\mathcal{V}|}\right)^i\right), \tag{3}$$

Quando $d \ll |\mathcal{V}|$ e $i \ll |\mathcal{V}|$, podemos usar a aproximação de primeira ordem $(1 - d/|\mathcal{V}|)^i \approx 1 - di/|\mathcal{V}|$ e assim obter $\mathrm{E}[\mathcal{N}_i] \approx di$. Essencialmente, esse resultado diz que quando d e $|\mathbf{s}|$ são pequenos, analisar o caso médio é equivalente a analisar o pior caso. Na segunda chamada à PROPAGAETIQUETAS, cada um dos $\mathcal{N}_{|\mathbf{s}|}$ nodos irá enviar mensagens para os seus d vizinhos. As mensagens trocadas entre nodos armazenados no mesmo host não consomem recursos da rede de comunicação. Portanto, o custo de transmissão é dado pelo número de mensagens trocadas entre hosts. A probabilidade de dois nodos estarem no mesmo host do cluster $\mathcal{H}$ é $1/|\mathcal{H}|$. Logo, o custo de transmissão por iteração é aproximadamente $d^2|\mathbf{s}|(|\mathcal{H}| - 1)/|\mathcal{H}|$.

Para complementar os resultados teóricos obtidos para o caso de grafos d-regulares, calculamos o número de mensagens trocadas em cada iteração a partir de 1000 execuções do algoritmo sobre os três datasets de redes, quando $|\mathbf{s}| \in \{2, 4, 8\}$. A Figura 2 mostra os boxplots do número de mensagens trocadas nas iterações 1, 2, 5, 10, 20

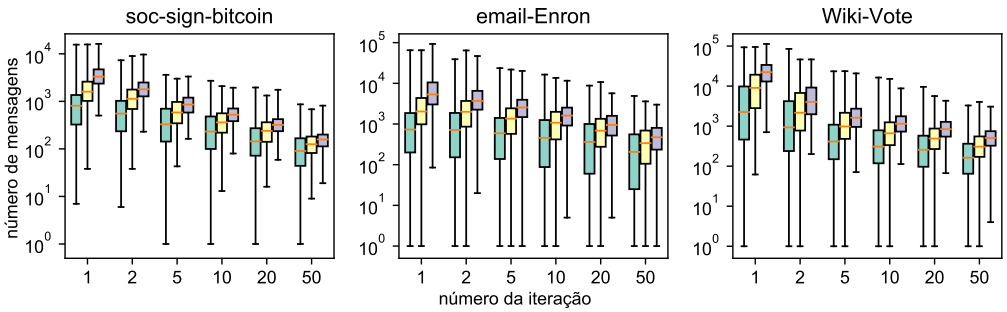


Figura 2. Boxplots do custo de transmissão (número de mensagens) do ϵ-net paralelo para certas iterações, com $|\mathbf{s}|$ = 2 (verde), 4 (amarelo) e 8 (rosa). A relação entre o número de mensagens e $|\mathbf{s}|$ é aproximadamente linear.

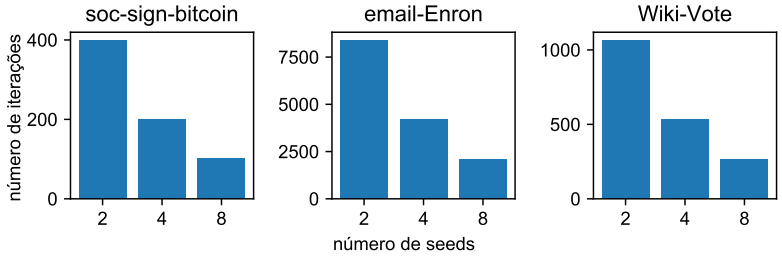


Figura 3. Numéro médio de iterações do ϵ-net paralelo até a parada, variando-se o número de seeds por iteração $|\mathbf{s}| \in \{2, 4, 8\}$.

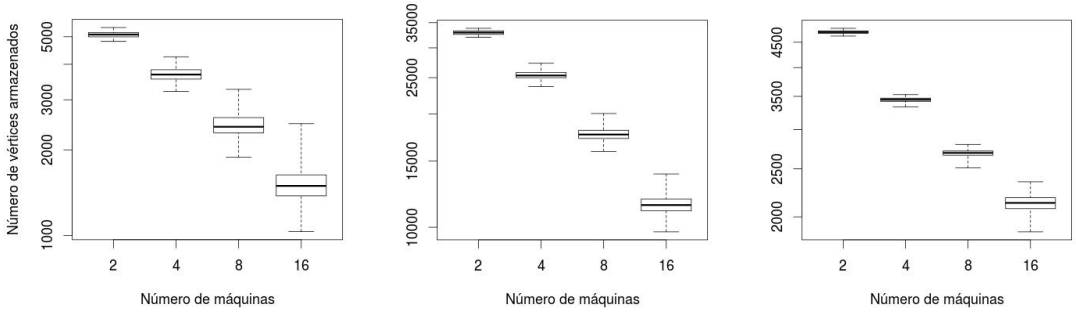
e 50. Cada boxplot indica, de cima para baixo, máximo, 3º quartil, mediana, 1º quartil e mínimo. Verificamos que a mediana varia de maneira aproximadamente linear com $|\mathbf{s}|$. Além disso, o número de mensagens transmitidas diminui com o número de iterações, pois os vértices de maior grau tendem a ser associados às partições primeiro.

Tempo de execução. O número de iterações realizadas até que o algoritmo termine é aproximadamente a razão entre o total de sementes utilizadas e $|\mathbf{s}|$. Contudo, não é trivial calcular esse total, mesmo no caso de um grafo d-regular. Assim, incluímos resultados empíricos que relacionam o número de iterações com a escolha de $|\mathbf{s}|$. A Figura 3 mostra o número médio de iterações em cada dataset baseado em 1000 execuções. Como o desvio padrão nesses casos foi muito pequeno, não incluímos intevalos de confiança. O número médio de iterações decresce linearmente com o número de seeds.

Algoritmo NoMAS

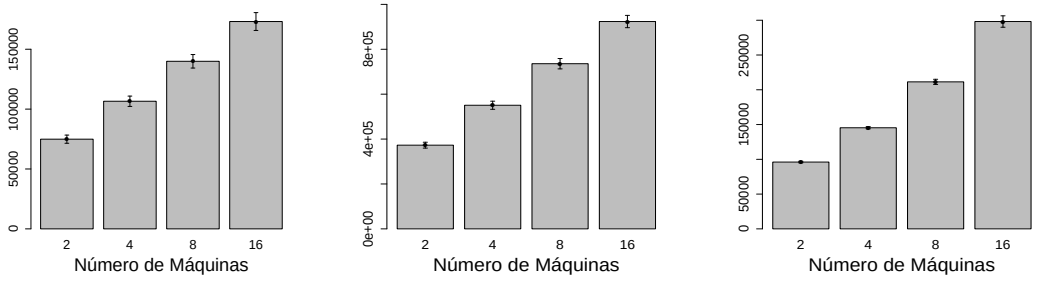
Custo de memória. Na versão serial do NoMAS o custo de memória consiste no espaço para armazenar o grafo, o tamanho atual de cada partição e, para cada vértice: sua etiqueta e um vetor de tamanho κ contendo o número de vizinhos em cada partição. Em um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a complexidade assintótica de espaço é $O(\max\{|\mathcal{V}| + |\mathcal{E}|,\ \kappa|\mathcal{V}|\})$. Quando o grau médio $\bar{d} > \kappa$, o custo dominante é o custo de se armazenar o grafo. Logo, se a máquina puder comportá-lo, é possível executar o NoMAS.

Já na versão paralela, em um *setup* com $|\mathcal{H}|$ hosts, cada um armazena: (i) a lista de vértices pelos quais ela é responsável, (ii) o conjunto de vizinhos de cada um deles e



(a) Bitcoin (b) Enron (c) Wiki-Vote

Figura 4. Custo de memória por máquina do NoMAS (soma do número de nodos com o tamanho da lista de vizinhos), variando-se o número de máquinas.



(a) Bitcoin (b) Enron (c) Wiki-Vote

Figura 5. Número total de mensagens trocadas para cada dataset variando-se o número de máquinas no cluster.

(iii) a partição a qual cada um dos vizinhos pertence. Dessa forma, cada máquina guarda uma lista de tamanho $|\mathcal{V}|/|\mathcal{H}|$, que representa os nodos que são gerenciados por ela, além dos vizinhos de cada um desses vértices e suas respectivas partições, que tem tamanho aproximado de $\bar{d} \cdot |\mathcal{V}|/|\mathcal{H}|$.

Custo de transmissão. Assim como no caso do ϵ-net, faremos uma análise para um caso simples e depois apresentaremos resultados empíricos usando grafos reais. Novamente, assuma um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ escolhido uniformemente dentre todos os grafos d-regulares com $|\mathcal{V}|$ vértices. Cada um dos hosts do cluster $\mathcal{H}$ será responsável por $|\mathcal{V}|/|\mathcal{H}|$ nodos. No pior caso, em que os vizinhos desses vértices são distintos, serão armazenados os dados de $d \cdot |\mathcal{V}|/|\mathcal{H}|$ vértices, como dito na seção anterior. A cada *stream* do algoritmo, a partição a que pertencem os vizinhos dos nós armazenados em cada máquina deve ser atualizada. Para cada uma, a etiqueta dos seus $|\mathcal{V}|/|\mathcal{H}|$ nodos terá que ser transmitida para até d outros computadores do *cluster*, enquanto ela receberá a informação dos $d \cdot |\mathcal{V}|/|\mathcal{H}|$ vértices vizinhos. Dessa maneira, o custo de transmissão total será $|\mathcal{V}| \cdot d$, no pior caso.

Tempo de execução. A análise do tempo de execução deve levar em conta o caminhamento feito no grafo para visitar todos os vértices, que é $O(|\mathcal{V}| + |\mathcal{E}|)$ e que, para cada nodo, deve ser analisada a pontuação de cada comunidade, o que custa $O(\kappa)$, onde κ é o número de comunidades. Logo, a complexidade total de tempo é dada por $O(\kappa(|\mathcal{V}|+|\mathcal{E}|))$.

Datasets	Corte			Carga máxima normalizada ρ		
	ϵ-net	FENNEL	NoMAS	ϵ-net	FENNEL	NoMAS
Bitcoin OTC	0.745	0.554	0.580	106.0	1.002	1.020
Enron Email	0.884	0.563	0.400	197.8	1.000	1.000
Wiki-Vote	0.882	0.813	0.798	464.6	1.001	1.071

Tabela 2. Avaliação da qualidade do particionamento em termos de fração de arestas entre comunidades (Corte) e o tamanho normalizado da maior comunidade (ρ). Para ambas as métricas, quanto menor, melhor.

4.3. Avaliação da qualidade do particionamento

Nesta seção avaliamos a qualidade do particionamento encontrado pelos algoritmos ϵ-net paralelo (com $\epsilon = 3$) e NoMAS (com $\kappa = 10$ partições e $\alpha = 2$). Comparamos os resultados com aqueles do FENNEL. O ϵ-net original não foi incluído na comparação, pois sua saída é idêntica a da versão paralela proposta neste artigo. As métricas valiadas são as mesmas utilizadas em [Tsourakakis et al. 2014] para a avaliação do FENNEL:

- **Corte:** fração das arestas do grafo que conecta nodos em diferentes partições.
- **Carga máxima normalizada** ρ: razão entre o tamanho da maior partição e $|\mathcal{V}|/\kappa$. Quando $\rho = 1$, todas as partições tem o mesmo tamanho.

A Tabela 2 mostra as médias obtidas para cada métrica para os algoritmos em cada um dos três datasets baseados em 100 execuções. Observa-se que o ϵ-net resultou em valores maiores de **Corte**. Isso acontece porque o número de partições encontradas é uma variável aleatória, tipicamente muito maior que $\kappa = 10$: o número médio de partições encontradas foi 1326,3 para o Bitcoin, 5505,3 para o Enron Email e 4131,6 para o Wiki-Vote. Também conforme esperado, o ϵ-net apresenta variabilidade muito maior no tamanho das partições (i.e., maior ρ) porque não leva esse critério em consideração.

Comparando o NoMAS com o FENNEL, observamos um dataset onde o NoMAS é superior (Enron Email), um dataset onde o FENNEL é superior (Bitcoin OTC), e um dataset onde o NoMAS tem **Corte** menor, mas ao custo de uma variabilidade ρ maior. Em termos quantitativos, o NoMAS chegou a reduzir **Corte** em 29.0% e aumentou-o em 4.7% no pior caso. Já a carga máxima normalizada ρ foi até 7% maior para o NoMAS.

4.4. Acurácia na estimação do ATE

Nesta seção, comparamos os algoritmos propostos com o FENNEL em relação à acurácia na estimação do *average treatment effect* (ATE). Para isto, simulamos a resposta (ou comportamento) Y_i de cada indivíduo i na rede a partir do tratamento $Z_i \in \{0, 1\}$ atribuído por randomização a nível de cluster (i.e., todos os nodos j no mesmo cluster tem mesmo Z_j). O modelo utilizado para geração de respostas é o modelo linear

$$Y_i = \beta_0 + \beta_1 Z_i + \beta_2 \sigma_i + \omega_i, \tag{4}$$

onde $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$ são parâmetros do modelo, σ_i é a fração de vizinhos j do nodo i com $Z_j = 1$, e $\omega_i \sim \text{Normal}(0, 1)$.

Usamos os cinco vetores de parâmetros $\boldsymbol{\beta}$ sugeridos em [Azevedo et al. 2018] para contemplar os casos mais representativos. Dado um vetor $\boldsymbol{\beta}$ e um algoritmo, existe

	Algoritmo	Vetor de parâmetros $\boldsymbol{\beta}$ e $\text{MSE}_{\text{Linear}}$				
		$\boldsymbol{\beta} = (0, 0, 1)$	$\boldsymbol{\beta} = (0, 1, 0, 5)$	$\boldsymbol{\beta} = (0, 1, 0)$	$\boldsymbol{\beta} = (0, 1, 1)$	$\boldsymbol{\beta} = (0, 1, 2)$
bitcoin	FENNEL	0,00189	0,00174	0,00192	0,00194	0,00214
	NoMAS	0,00212	0,00190	0,00200	0,00190	0,00188
	ϵ-net	0,00224	0,00214	0,00204	0,00208	0,00205
Enron	FENNEL	0,00032	0,00034	0,00036	0,00034	0,00033
	NoMAS	0,00030	0,00032	0,00032	0,00031	0,00031
	ϵ-net	0,00027	0,00025	0,00028	0,00027	0,00030
Wiki-Vote	FENNEL	0,00198	0,00181	0,00188	0,00187	0,00186
	NoMAS	0,00187	0,00189	0,00204	0,00208	0,00198
	ϵ-net	0,00201	0,00217	0,00204	0,00203	0,00209

Tabela 3. Avaliação dos algoritmos quanto ao erro de estimação do ATE, medido em termos do MSE. Modelo de resposta linear usado para geração dos dados.

aleatoriedade na geração das partições, no mapeamento de partições para os grupos de controle e tratamento (que resultam no vetor $\mathbf{Z}$), e na geração das respostas (que resultam no vetor $\mathbf{Y}$). Para cada ponto de aleatoriedade realizamos 10 execuções, resultando em 5 vetores $\boldsymbol{\beta} \times 3$ algoritmos $\times$ 10 vetores $\mathbf{Z} \times 10$ vetores $\mathbf{Y}$, totalizando 15.000 simulações.

Os resultados são avaliados em termos do erro médio quadrático (MSE), dado por

$$\text{MSE}(\text{algoritmo}, \boldsymbol{\beta}) = \frac{1}{100} \sum_{\mathbf{Z},\mathbf{Y}} (\widehat{\text{ATE}}(\mathbf{Z}, \mathbf{Y}) - \text{ATE}(\boldsymbol{\beta}))^2, \tag{5}$$

onde $\widehat{\text{ATE}}(\mathbf{Z}, \mathbf{Y})$ é o ATE estimado por um modelo de regressão linear para os vetores $\mathbf{Z}$ e $\mathbf{Y}$ produzidos pelo algoritmo e $\text{ATE}(\boldsymbol{\beta})$ é o valor real do ATE dado $\boldsymbol{\beta}$. Consulte [Azevedo et al. 2018] para mais informações sobre como calcular estas quantidades.

A Tabela 3 sumariza os resultados obtidos para cada dataset e vetor de parâmetros. Observamos que o NoMAS resulta em menor MSE que o FENNEL em 7 de 15 casos. Em particular, NoMAS supera o FENNEL no dataset Email Enron para todos valores de $\boldsymbol{\beta}$ testados. É interessante notar que o ϵ-net obtém os melhores resultados naquele dataset.

5. Conclusões

Neste trabalho propusemos e analisamos dois algoritmos escaláveis de particionamento de redes para a realização de experimentos randomizados. O primeiro é uma versão paralela do algoritmo ϵ-net. O segundo, NoMAS, é inspirado no FENNEL, mas usa uma função diferente para controlar o compromisso entre número de arestas conectando vértices em diferentes partições e balanceamento do tamanho das partições. Comparando os resultados obtidos para NoMAS e FENNEL em relação à qualidade do particionamento e à acurácia na estimação do ATE, concluímos que as duas formas de penalizar partições grandes geram resultados semelhantes para o modelo de resposta linear. Além disso, a qualidade das partições não exibiu uma correlação direta com a acurácia da estimação: embora o ϵ-net tenha levado aos piores resultados de particionamento, o MSE resultante em um dos datasets foi menor que dos outros algoritmos. É possível que diferenças mais acentuadas entre os algoritmos seriam observadas quando o modelo de estimação é diferente do modelo usado para gerar as respostas. Isto será investigado em trabalhos futuros.

Referências

Aronow, P. M. and Samii, C. (2012). Estimating average causal effects under general interference. In *Summer Meeting of the Society for Political Methodology*, pages 19–21.

Azevedo, F. G., Nogueira, B. D., Murai, F., and da Silva, A. P. C. (2018). Modelos de resposta para experimentos randomizados em redes sociais de larga escala. arXiv:1803.03497.

Backstrom, L. and Kleinberg, J. M. (2011). Network bucket testing. In *WWW*, pages 615–624.

Eckles, D., Karrer, B., and Ugander, J. (2014). Design and analysis of experiments in networks: Reducing bias from interference. *CoRR abs/1507.00803*, stat.ME.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports-Review Section of Physics Letters*, 486(3-5):75–174.

Gui, H., Xu, Y., Bhasin, A., and Han, J. (2015). Network A/B Testing: From Sampling to Estimation. *WWW*, pages 399–409.

Katzir, L., Liberty, E., and Somekh, O. (2012). Framework and algorithms for network bucket testing. In *WWW*, pages 1029–1036.

Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., and Pohlmann, N. (2013). Online controlled experiments at large scale. In *KDD*, pages 1168–1176.

Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. (2009). Controlled experiments on the web: survey and practical guide. *DMKD*, 18(1):140–181.

Manski, C. F. (2013). Identification of treatment response with social interactions. *Econometrics Journal*, 16(1):S1–S23.

Middleton, J. A. and Aronow, P. M. (2011). Unbiased Estimation of the Average Treatment Effect in Cluster-Randomized Experiments. *SSRN Electronic Journal*.

Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

Nishimura, J. and Ugander, J. (2013). Restreaming graph partitioning: simple versatile algorithms for advanced balancing. In *KDD*, pages 1106–1114.

Saveski, M., Pouget-Abadie, J., Saint-Jacques, G., Duan, W., Ghosh, S., Xu, Y., and Airoldi, E. M. (2017). Detecting Network Effects. In *KDD*, pages 1027–1035.

Stanton, I. and Kliot, G. (2012). Streaming graph partitioning for large distributed graphs. In *KDD*, pages 1222–1230. ACM.

Tsourakakis, C., Gkantsidis, C., Radunovic, B., and Vojnovic, M. (2014). Fennel: Streaming graph partitioning for massive scale graphs. In *WSDM*, pages 333–342. ACM.

Ugander, J., Karrer, B., Backstrom, L., and Kleinberg, J. (2013). Graph cluster randomization: Network exposure to multiple universes. In *KDD*, pages 329–337.

Xu, Y., Chen, N., Fernandez, A., Sinno, O., and Bhasin, A. (2015). From infrastructure to culture: A/B testing challenges in large scale social networks. In *KDD*, pages 2227–2236.