Em Busca de Redes Esparsas Robustas e Eficientes

Romeu Lustosa Pires[1,2], **Carlos Bravo Serrado**[1], **Daniel Sadoc Menasché**[1]

[1]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ), RJ – Brazil

[2]Anlix, Rio de Janeiro – RJ – Brazil

Abstract. *Motivated by the importance of building sparse, efficient and robust networks, we consider sparse networks with small sum of distances across nodes (Wiener index) and that are robust (biconnected). In particular, we propose a novel efficient algorithm to compute the Wiener index for those sparse networks. The algorithm allows us to exhaustively compute the Wiener index for a class of graphs with up to 20 vertices, verifying interesting properties of those graphs.*

Resumo. *Motivados pela importância de construir redes eficientes e robustas, apresentamos um algoritmo eficiente para calcular a soma das distâncias em redes esparsas. Tal índice, também chamado de índice de Wiener, está relacionado com a eficiência da rede. Aplicando o algoritmo, discutimos algumas propriedades das redes robustas (biconexas) e eficientes (com pequena soma das distâncias).*

1. Introdução

Robustez e eficiência são duas das características desejáveis em sistemas computacionais. Em redes de computadores, por exemplo, robustez refere-se ao fato de as redes serem tolerantes a falhas, e eficiência refere-se ao fato de os caminhos entre os nós serem curtos. A busca por topologias de redes robustas e eficientes motivou inúmeros trabalhos na interseção entre redes de computadores, teoria dos grafos e inteligência artificial.

Embora robustez e eficiência sejam ambas desejáveis, pode haver um compromisso entre as duas. Este compromisso se torna mais evidente à medida que o grafo se torna mais esparso. Considere, por exemplo, que possuímos um orçamento limitado, que nos permite construir, para um grafo de $|V|$ vértices, até $|E| = |V| + 1$ conexões diretas (arestas). Nesse caso, é possível alcançar eficiência máxima, por exemplo, conectando todos os vértices por meio de um nó central (semelhante a uma forma de estrela), envolvendo $n - 1$ arestas, conferindo distâncias ≤ 2. Em outras palavras, a distância entre nós é $O(1)$. Entretanto, esta topologia derivada da estrela é muito frágil, tendo em vista que a remoção do nó central desconecta os nós restantes. Alternativamente, pode-se ter uma solução muito robusta, contendo um ciclo envolvendo todos os nós, na qual é impossível desconectar quaisquer dois nós com a remoção de um terceiro, mas a distância entre os nós será $O(|V|)$. A topologia derivada a partir do ciclo é mais robusta, mas muito menos eficiente em comparação com a topologia derivada a partir da estrela.

Contribuição. Dados os critérios acima, visamos responder a seguinte pergunta: restringindo-se a redes robustas, ou seja, biconexas, quais as topologias mais eficientes, ou seja, que minimizam o índice de Wiener? *Neste trabalho, damos um passo para responder tal pergunta, desenvolvendo um algoritmo eficiente para calcular a soma das distâncias em uma rede esparsa.*

2. Trabalhos relacionados

Dada a importância do índice de Wiener, existem vários trabalhos com fórmulas fechadas para o índice de Wiener para certas classes de grafos [Graovac and Pisanski 1991].[1] O cálculo eficiente do índice de Wiener também foi considerado em [Klavzar et al. 1995, Chepoi and Klavžar 1997, Yeh and Gutman 1994]. Entretanto, não conhecemos nenhum resultado visando o cálculo do índice de Wiener para grafos esparsos.

É sabido que a soma das distâncias em uma árvore pode ser calculada em tempo linear [Mohar and Pisanski 1988], mas que o problema de determinar a soma das distâncias em grafos gerais, mesmo no caso em que o número de arestas cresce $\tilde{O}(n)$, não pode ser resolvido em tempo sub-quadrático, a não ser que a *Strong Exponential Time Hypothesis (SETH)* seja falsa [Roditty and Vassilevska Williams 2013, Impagliazzo et al. 2001]. Neste trabalho, propomos um algoritmo que, no pior caso, tem complexidade cúbica no número de vértices, mas que para uma vasta gama de redes esparsas mostra-se eficiente.

A relação do índice de Wiener com outras métricas de grafos foi extensamente considerada na literatura [Dankelmann et al. 2009]. Neste trabalho, focamos exclusivamente no índice de Wiener, apresentando um algoritmo que é particularmente eficiente para grafos esparsos.

3. O índice de Wiener de grafos muito esparsos

Grafos esparsos ocorrem em cenários nos quais o custo de criação de arestas é alto – por exemplo, em redes elétricas ou redes rodoviárias. A soma das distâncias entre os pares de vértices de um grafo é também chamada de índice de Wiener. É possível verificar que para grafos biconexos, por exemplo, dados $|E|$ e $|V|$ no regime em que $|E| \geq 2|V| - 5$ existem grafos que ao mesmo tempo minimizam a soma das distâncias e são biconexos [Bollobás 2004]. Entretanto, para $|E| < 2|V| - 5$ isso não é mais verdade. No regime $|E| < 2|V| - 5$ precisamos escolher entre reorganizar as arestas para minimizar a soma das distâncias ou preservar a biconectividade.

O meio mais usual de se calcular o índice de Wiener de um grafo genérico é a partir da matriz de caminhos mínimos, podendo ser obtida a partir do algoritmo de Floyd-Warshall, de complexidade $O(|V|^3)$. Desde pelo menos 1988 existe um esforço para encontrar maneiras de se calcular o índice de Wiener sem a necessidade da matriz de caminho mínimo, que é um método considerado de força-bruta [Paul Manuel, Indra Rajasingh, Bharati Rajan, R. Sundara Rajan 2013].

Neste trabalho buscamos um algoritmo para calcular o índice de Wiener, especialmente focados na classe de grafos na qual estamos interessados: a de grafos biconexos com $|E| < 2|V| - 5$. Embora o algoritmo seja geral, ele é mais eficiente no regime esparso. A principal vantagem de se trabalhar no regime esparso consiste na ocorrência de muitas sequências de vértices de grau 2, os quais chamamos de pontes.[2] É possível utilizar essas premissas para sermos mais eficiente no cálculo do índice de Wiener. Em resumo, o algoritmo apresentado tem complexidade $O((|V| - |V_1| - |V_2|)^3)$, sendo $|V_1|$ e $|V_2|$ o número de vértices de grau 1 e grau 2, respectivamente.

[1] `https://cstheory.stackexchange.com/questions/17424/complexity-of-computing-the-average-distance-of-a-graph`

[2] A terminologia é sutilmente diferente daquela usualmente adotada em teoria dos grafos.

3.1. O índice de Wiener por partes

Calcularemos o índice de Wiener por partes, dividindo os vértices em duas categorias: (1) *vértices internos* e (2) *vértices centrais*. Chamamos de *vértices internos* todos os vértices de grau 2 ou menos, e de *vértices centrais* todo o restante, ou seja, de grau 3 ou mais. O índice de Wiener pode ser calculado pelo somatório...

1. ... das distâncias entre os vértices centrais.
2. ... das distâncias entre os vértices internos com os vértices centrais.
3. ... das distâncias entre os vértices internos pertencentes à mesma ponte.
4. ... das distâncias entre os vértices internos pertencentes a pontes diferentes.

Iremos denotar por $W_{cc}(G), W_{pc}(G), W_p(G)$ e $W_{pp}(G)$ as somas das distâncias correspondentes aos quatros elementos acima. Sendo $W(G)$ o índice de Wiener do grafo,

$$W(G) = W_{cc}(G) + W_{pc}(G) + W_p(G) + W_{pp}(G).$$

A seguir, descrevemos como calcular cada um dos termos da soma acima.

3.2. Algoritmos

Nesta seção apresentamos algoritmos eficientes para encontrar menores caminhos e calcular o índice de Wiener em grafos esparsos. É conveniente que a estrutura de dados utilizada para armazenar o grafo de entrada G seja uma lista/vetor dinâmico de adjacências, de modo que seja possível devolver o grau de determinado vértice em tempo constante.

O procedimento proposto se divide em três algoritmos. O **Algoritmo de Contração** calcula estruturas básicas, decompondo o grafo em "pontes" (caminhos onde todo os vértices internos tem grau 2). O **Algoritmo de Distâncias entre Dois Vértices** calcula em tempo constante a distância entre dois vértices usando dados providos pelo algoritmo anterior. Finalmente, o **Algoritmo de Distâncias usando Fórmulas Fechadas** usa fórmulas fechadas para W_{pp}, W_p e W_{pc}, para calcular o índice de Wiener do grafo G, usando os dois algoritmos anteriores.

Em particular, o último algoritmo faz uso da matriz de caminhos mínimos de G_{aux}, grafo auxiliar com os vértices de grau maior ou igual à 3 de G, gerada pelo **Algoritmo de Contração**. A matriz pode ser calculada usando um algoritmo clássico para distâncias entre todos os pares de vértices de um grafo, como Floyd-Warshall.

Algoritmo de Contração. O laço principal varre todos os vértices e se o grau do vértice for maior ou igual a três, ou seja, se for um nó central, o nó é incluído em conjunto e será tratado posteriormente. Finalmente, se o nó for interno, procede-se com o tratamento de sua respectiva ponte. Para cada ponte tratada, inclui-se uma respectiva aresta no grafo de saída.

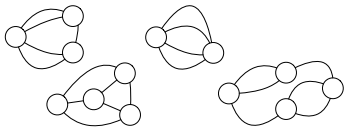


Figura 1. Multigrafos biconexos, $|E| = |V| + 2$

Exemplo do Algoritmo de Contração. Considere grafos biconexos extremamente esparsos, $|E| = |V| + 2$. Nesse caso, pode-se verificar que existem apenas 4 tipos de grafos biconexos, obtidos pela inserção de vértices nos multigrafos indicados na Figura 1. Assim, dado um grafo de entrada, a saída do **Algoritmo de Contração** será um dos 4 multigrafos apresentadas na Figura 1, bem como o número de vértices a ser alocado em cada aresta do multigrafo.

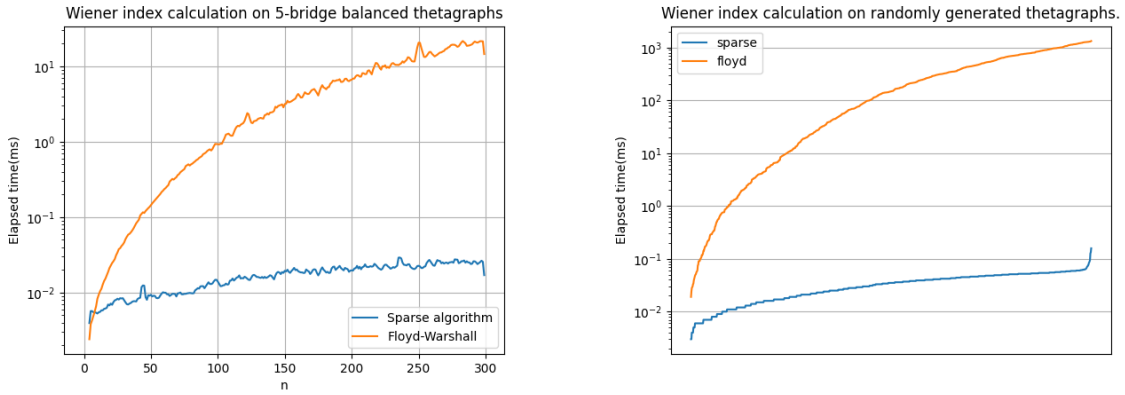
3.3. Comparativo de desempenho

A seguir, apresentamos alguns resultados que comparam o desempenho do cálculo do índice de Wiener pelo usual algoritmo de Floyd-Warshall contra o algoritmo proposto neste trabalho. Os experimentos foram executados sequencialmente, ou seja, sem uso de processamento paralelo, todos em um processador *Intel Core i5-3330*.

Como era de se esperar, o tempo de execução nos grafos biconexos esparsos, com apenas dois vértices com grau maior que dois, se mostrou quase constante independentemente do número de vértices, mas apenas do número de pontes. As Figuras 2(a) e 2(b) mostram o comparativo do tempo necessário para a execução dos dois algoritmos em grafos biconexos esparsos, com apenas dois vértices com grau maior que dois.

Os gráficos apresentados nas Figuras 3(a) e 3(b) mostram o desempenho dos algoritmos sendo executados em grafos aleatórios. O modo de geração de cada grafo seguiu um algoritmo simples: inicializamos um grafo com um dado número de vértices e, enquanto este grafo não for conexo, criamos uma aresta entre dois vértices arbitrários. Nenhum dos grafos avaliados por essa abordagem é biconexo, e a razão do número de arestas e número de vértices se manteve em aproximadamente 3.

O histograma da Figura 3 mostra o tempo de execução em 300 grafos diferentes, todos com 200 vértices. Nele, fica explícito que, embora o algoritmo que apresentamos possa ser mais útil em determinadas classes de grafos, em alguns casos pode haver um tempo extra necessário. Estamos atualmente avaliando sob quais condições o algoritmo proposto é mais vantajoso, e conjecturamos que a vantagem do algoritmo decresça com relação ao número de arestas, e cresça com o número de vértices de grau 1 ou 2.

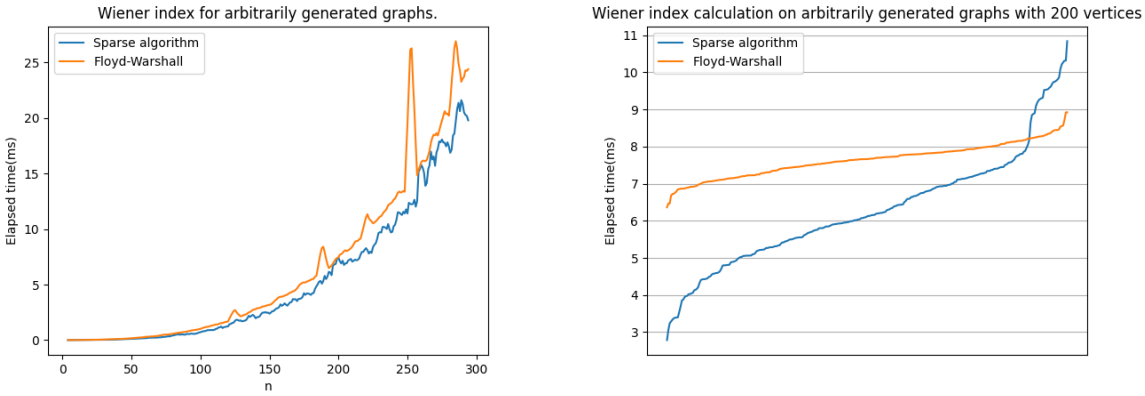


(a) (b)

Figura 2. (a) Tempo de execução em função do número de vértices em grafos biconexos com apenas 2 vértices com grau > 2 e com 5 pontes; (b) Tempo de execução em grafos biconexos com 2 vértices com grau > 2, gerados aleatoriamente (número de vértices distribuído uniformemente entre 20 e 1000)

4. Resultados numéricos para grafos biconexos esparsos

Entre todas as topologias biconexas possíveis com um dado número de vértices e arestas, qual possui o menor índice de Wiener? Realizamos alguns experimentos de força-bruta em grafos relevantes (biconexos). Para tal, usamos o



(a) (b)

Figura 3. (a) Histograma do tempo de execução em função do número de vértices em grafos arbitrariamente gerados; (b) Histograma do tempo de execução em diferentes grafos arbitrários com 200 vértices.

geng [Brendan McKay, Adolfo Piperno 2022]. O tempo de execução para um dado grupo cresce em função não só do número de vértices mas também do número de arestas. Devido à natureza combinatorial do problema, podemos perceber na Tabela 1 que mesmo para um valor de n tão pequeno como 18 pode ter o tempo de exploração em ordens de grandeza de frações de segundos ($|E| = |V| + 1$) até anos ($|E| = 2|V| - 6$).

Tabela 1. Tempo (segundos) usando geng para gerar exaustivamente topologias

m/n	16	17	18	19	20
17	0	-	-	-	-
18	0	0	-	-	-
19	12	12	0	-	-
20	95	93	0	0	-
21	608	483	18	8	0
22	4243	2571	308	66	71
23	30890	17510	3623	985	746
24	226782	136300	37951	10501	6121
25	1245289	1036436	393006	112077	65796
26	1595565	1193888	3938628	1609835	471579
27	-	*	26769599	*	5661408
28	-	*	131226063	*	51692392

Depois de gerar todos os grafos definidos, calculamos os índices de Wiener. Verificamos, no conjunto de grafos biconexos avaliados, a seguinte propriedade: *dados $|V|$ e $|E|$, é sempre possível encontrar uma rede biconexa ótima, ou seja, que minimiza o índice de Wiener, na qual no máximo 2 vértices têm grau maior que dois.*

5. Conclusão e trabalhos futuros

Apresentamos um algoritmo eficiente para o cálculo da soma das distâncias entre os vértices de uma rede. O algoritmo apresentado é útil para calcular a soma das distâncias entre vértices em redes esparsas, e se aproveita da classificação dos pares de vértices em quatro categorias: distâncias entre vértices centrais, distância entre vértices internos e

vértices centrais, distância entre vértices internos numa mesma ponte, e distância entre vértices internos em pontes diferentes. Vislumbramos que o algoritmo aqui apresentado seja útil para avaliar o desempenho de redes reais, bem como para propor e resolver conjecturas envolvendo o índice de Wiener [Deng et al. 2010].[3]

A ideia do algoritmo proposto de decompor o grafo em "pontes" (caminhos onde todo os vértices internos tem grau 2), que podem ser encaradas como contrações de partes do grafo em um único vértice, é empregrada em outros algoritmos em grafos, e.g., Blossom Algorithm. Pretendemos verificar outros algoritmos de distâncias em grafos esparsos que adotam ideias parecidas de contração, para comparar com o algoritmo aqui proposto.

Pretendemos também expandir a avaliação do algoritmo proposto, em função da fração de vértices com grau 1 e grau 2. Esse tipo de avaliação será realizada com o Configuration Model, onde o grafo aleatório é parametrizado com a sequência de graus dos vértices (em comparação ao Floyd-Warshall). Sobre a caracterização do valor ótimo de Wiener para algumas classes de grafos, pretendemos caracterizar esses grafos, e apresentar propriedades dos mesmos, como o número de motifs especiais, bem como determinar quais famílias de grafos tradicionais, como Debruijn, Hipercubo, CubeConnectedCycles etc respeitam as propriedades apresentadas neste trabalho.

Referências

Bollobás, B. (2004). *Extremal graph theory*. Courier Corporation.

Brendan McKay, Adolfo Piperno (2022). Nauty traces. Acessado em 7 abril 2022.

Chepoi, V. and Klavžar, S. (1997). The Wiener index and the Szeged index of benzenoid systems in linear time. *Journal of chemical info. computer science*, 37(4):752–755.

Dankelmann, P., Gutman, I., Mukwembi, S., and Swart, H. C. (2009). The edge-wiener index of a graph. *Discrete Mathematics*, 309(10):3452–3457.

Deng, H., Xiao, H., and Tang, F. (2010). On the extremal wiener polarity index of trees with a given diameter. *MATCH Commun. Math. Comput. Chem*, 63(1):257–264.

Graovac, A. and Pisanski, T. (1991). On the wiener index of a graph. *Journal of mathematical chemistry*, 8(1):53–62.

Impagliazzo, R., Paturi, R., and Zane, F. (2001). Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530.

Klavzar, S., Gutman, I., and Mohar, B. (1995). Labeling of benzenoid systems which reflects the vertex-distance relations. *Journal of chemical info. and comp. sci.*, 35(3):590–593.

Mohar, B. and Pisanski, T. (1988). How to compute the Wiener index of a graph. *Journal of mathematical chemistry*, 2(3):267–277.

Paul Manuel, Indra Rajasingh, Bharati Rajan, R. Sundara Rajan (2013). A new approach to compute wiener index. *Journal Computational Theoretical Nanoscience* `https://arxiv.org/abs/1307.6502`.

Roditty, L. and Vassilevska Williams, V. (2013). Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC*, pages 515–524.

Yeh, Y.-N. and Gutman, I. (1994). On the sum of all distances in composite graphs. *Discrete Mathematics*, 135(1-3):359–365.

[3]O algoritmo proposto neste trabalho, implementado em C, está disponível publicamente em: `https://github.com/rilpires/sparse_graphs_tools`