Hindawi
Journal of Mathematics
Volume 2021, Article ID 6657594, 5 pages
https://doi.org/10.1155/2021/6657594

Research Article

A Polynomial Algorithm for Weighted Toughness of Interval Graphs

Ming Shi and Zongtian Wei

School of Science, Xi'an University of Architecture and Technology, Xi'an, Shaanxi 710055, China

Correspondence should be addressed to Ming Shi; m.shi@xauat.edu.cn

Received 25 November 2020; Revised 13 January 2021; Accepted 18 January 2021; Published 29 January 2021

Academic Editor: Kaleem R. Kazmi

The concept of toughness, introduced by Chvátal, has been widely used as an important invulnerability parameter. This parameter is generalized to weighted graphs, and the concept of weighted toughness is proposed. A polynomial algorithm for computing the weighted toughness of interval graphs is given.

1. Introduction

The concept of invulnerability was proposed in the early research on the connectivity of communication networks, which reflect the ability of a network to resist deliberate damage from the outside [1]. As a widely used invulnerability parameter, toughness was introduced by Chvátal in 1973.

Definition 1 (see [2]). Let G be a noncomplete graph. The toughness of G is defined as

$$t(G) = \min\left\{\frac{|X|}{\omega(G-X)}: X \subset V(G), \quad \omega(G-X) \geq 2\right\}. \tag{1}$$

If there exists $X^* \subseteq V(G)$ such that $t(G) = |X^*|/\omega(G-X^*)$, then X^* is called a t-tough set, denoted as t-set. In particular, for the complete graph K_n, define $t(K_n) = \infty$.

In reality, the vertices of a graph have different roles. Usually, vertex-weighted graphs are used to represent such network models, that is, each vertex is associated with a real number to distinguish such a difference. To measure the invulnerability of networks, we introduce the concept of weighted toughness of graphs.

Definition 2. Let G be a noncomplete vertex-weighted graph. The weighted toughness of G is defined as

$$t^w(G) = \min\left\{\frac{w(X)}{\omega(G-X)}: X \subset V(G), \quad \omega(G-X) \geq 2\right\}, \tag{2}$$

where $w: V \longrightarrow R^+$ is a nonnegative weight function and $w(X) = \sum_{v \in X} w(v)$ is the weight of the vertex cut X. For a vertex-weighted complete graph K_n, define $t^w(K_n) = \infty$.

By the definition, the greater the weighted toughness, the stronger the invulnerability of the vertex-weighted graph.

Definition 3. Let G be a vertex-weighted noncomplete graph. If there exists $X^* \subseteq V(G)$ such that $t^w(G) = w(X^*)/\omega(G-X^*)$, then X^* is called an achieving cut of G.

When the weights of all vertices are equal, weighted toughness is equivalent to toughness. The problem of toughness computation is NP-hard [3]. However, there exists a polynomial algorithm for toughness computation of interval graphs [4]. The definition of the interval graph is given below.

Definition 4 (see [5]). A graph G is called an interval graph, if $\forall v \in V(G)$ corresponds to a closed interval $I_v = [a_v, b_v]$, and $uv \in E(G)$ if and only if $I_u \cap I_v \neq \varnothing$. We call $\{I_v\}_{v \in V}$ the interval representation of G.

Due to the special structure and properties, interval graphs are widely used in theoretical research and engineering practice. For example, the archeology chronological order of unearthed items [6], the scheduling problems in computer science [7], and the determination of biology model of DNA sequences [8]. Many scholars have studied the algorithm and structure of interval graphs. Kratsch et al. gave a polynomial time algorithm for computing scattering numbers and toughness of interval graphs [4]. Broersma et al. gave a linear time algorithm for computing scattering numbers of interval graphs [9]. Li et al. gave a polynomial algorithm for computing weighted scattering number of interval graphs [10].

In this paper, we consider the algorithm of computing weighted toughness of interval graphs. The following work is arranged in two parts. In Section 2, some elementary definitions and notations, as well as some preliminary results are given. An algorithm for computing the weighted toughness of interval graphs is given based on the investigation of properties of achieving cuts and local minimum cuts. In Section 3, we summarize our work by the complexity analysis of the algorithm.

This paper only considers the finite simple undirected graphs. For the terminology and notations not defined here, we refer the reader to [11].

2. An Algorithm for Computing Weighted Toughness of Interval Graphs

In this section, we firstly investigate the properties of achieving cuts and local minimum cuts of interval graphs. The number of vertex cut of a graph increases with its order exponentially. Fortunately, it is not necessary to consider all vertex cuts when computing the weighted toughness of interval graphs.

Definition 5. (see [12]). Let G be an interval graph and X be a vertex cut. If for every proper subset $Y \subset X$ and $\omega(G-Y) < \omega(G-X)$, then X is called a strong cut of G.

Theorem 1. *Let G be an interval graph with nonnegative weights' function w: $V \longrightarrow R^+$. Then, any achieving cut of G is a strong cut.*

Proof. Let X be an achieving cut of G but not a strong cut. By Definition 5, there must exist a vertex $u \in X$ such that $\omega(G-X) \le \omega(G-(X-\{u\}))$. Since $w(X-\{u\}) < w(X)$,

$$\frac{w(X-\{u\})}{\omega(G-(X-\{u\}))} < \frac{w(X)}{\omega(G-X)} = t^w(G). \tag{3}$$

This contradicts to that X is an achieving cut of G. The proof is completed. □

Definition 6 (see [12]). Let G be an interval graph. Consider a point x such that $\min_i b_i < x < \max_i a_i$. If the end point immediately to the left of x is right end point and the end point immediately to the right of x is a left end point, then $C(x)$ is called a minimal local cut of G.

Figure 1 is an example of interval graph and its interval representation. The minimal local cuts of G are $C(4.5) = \{I_2, I_3\}$, $C(6.25) = \{I_3, I_4\}$, $C(8.75) = \{I_4, I_6\}$, $C(12.5) = \{I_4, I_6\}$, and $C(16.5) = \{I_9, I_{10}\}$.

Lemma 1 (see [12]). *Any strong cut of an interval graph can be expressed as a union of minimal local cuts.*

By Theorem 1 and Lemma 1, we know that any achieving cut can be expressed as a union of minimal local cuts. Therefore, the weighted toughness of an interval graph can be computing by the unions of minimal local cut. In 2006, Ray et al. presented an algorithm for computing minimal local cuts of an interval graph and proved that the time complexity is $O(n^2)$ [12].

Definition 7. Let $C(\alpha_1), C(\alpha_2), \ldots, C(\alpha_k)$ be all the minimal local cuts of an interval graph G, where $\alpha_1 < \alpha_2 < \cdots < \alpha_k$. Then, $C(\alpha_i)$ and $C(\alpha_{i+1})$ are called adjacent, $i = 1, 2, \ldots, k-1$.

Theorem 2. *Let $C(\alpha_1), C(\alpha_2), \ldots, C(\alpha_k)$ be all the minimal local cuts of a weighted interval graph G. If $C(\alpha_i)$ and $C(\alpha_j)$ are disjoint, then*

$$\min\left\{\frac{w(C(\alpha_i))}{\omega(G-C(\alpha_i))}, \frac{w(C(\alpha_j))}{\omega(G-C(\alpha_j))}\right\} \le \frac{w(C(\alpha_i)\cup C(\alpha_j))}{\omega(G-(C(\alpha_i)\cup C(\alpha_j)))}. \tag{4}$$

Proof. Since $C(\alpha_i)\cap C(\alpha_j) = \varnothing$, $w(C(\alpha_i)\cup C(\alpha_j)) = w(C(\alpha_i)) + w(C(\alpha_j))$. On the contrary, $\omega(G-C(\alpha_i)\cup C(\alpha_j)) \le \omega(G-C(\alpha_i)) + \omega(G-C(\alpha_j))$. Without loss of generality, suppose that $w(C(\alpha_i))/\omega(G-C(\alpha_i)) \ge w(C(\alpha_j))/\omega(G-C(\alpha_j))$. Then,

$$w(C(\alpha_i))\omega(G-C(\alpha_j)) - w(C(\alpha_j))\omega(G-C(\alpha_i)) \ge 0. \tag{5}$$

Therefore,

$$\begin{aligned}
&\frac{w(C(\alpha_i)\cup C(\alpha_j))}{\omega(G-(C(\alpha_i)\cup C(\alpha_j)))} - \frac{w(C(\alpha_j))}{\omega(G-C(\alpha_j))} \\
&\ge \frac{w(C(\alpha_i)) + w(C(\alpha_j))}{\omega(G-C(\alpha_i)) + \omega(G-C(\alpha_j))} - \frac{w(C(\alpha_j))}{\omega(G-C(\alpha_j))} \\
&= \frac{w(C(\alpha_i))\omega(G-C(\alpha_j)) - w(C(\alpha_j))\omega(G-C(\alpha_i))}{\omega(G-C(\alpha_j))\left[\omega(G-C(\alpha_i)) + \omega(G-C(\alpha_j))\right]} \\
&\ge 0.
\end{aligned} \tag{6}$$

The proof is completed. □

Remark 1. The conclusion of Theorem 2 is true for the cases where the number of minimal local cuts is greater than 2.

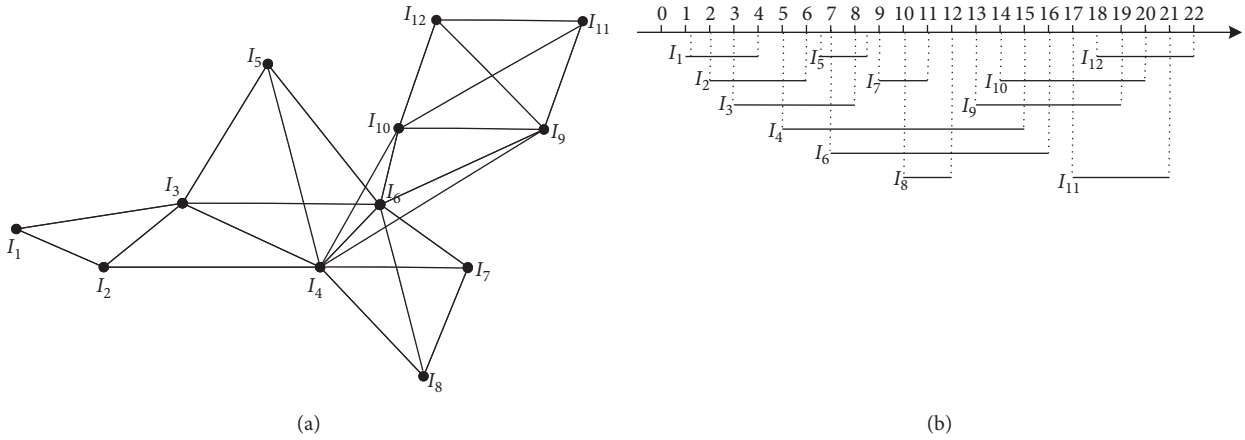

Figure 1: An interval graph G and its interval representation.

Theorem 3. *Let $C(\alpha_i), C(\alpha_{i+1}), C(\alpha_{i+2})$ be three sequential adjacent minimal local cuts of an interval graph G. If $C(\alpha_i) \cap C(\alpha_{i+1}) = \varnothing$, then $C(\alpha_i) \cap C(\alpha_{i+2}) = \varnothing$.*

Proof. By contradiction, if $C(\alpha_i) \cap C(\alpha_{i+2}) \neq \varnothing$, then there exists a vertex $v \in C(\alpha_i) \cap C(\alpha_{i+2})$. Let $u \in C(\alpha_{i+1})$, $z \in C(\alpha_{i+2}) - C(\alpha_i)$, and the interval representation of v, u, z be $[a, b]$, $[a_{i+1}, b_{i+1}]$, and $[a_{i+2}, b_{i+2}]$, respectively.

By the definition of interval representation, $a < a_{i+1} < a_{i+2}$. Since $v \in C(\alpha_{i+2})$, $b_{i+1} < b$, and $[a_{i+1}, b_{i+1}] \subset [a, b]$. Thus, $v \in C(\alpha_{i+1})$. This contradicts that $C(\alpha_i) \cap C(\alpha_{i+1}) = \varnothing$. The proof is completed.

Theorem 3 shows that, for an interval graph, nonadjacent minimal local are disjoint. The algorithm for finding the union of minimal local cuts with nonempty intersection is given below. □

Theorem 4. *The time complexity of Algorithm 1 is $O(n^3)$*

Proof. Step 3 needs $|C(\alpha_i)||C(\alpha_{i+1})|$ comparisons to determine whether $C(\alpha_i) \cap C(\alpha_{i+1}) = \varnothing$. Step 5 finds union of two minimal local cuts with nonempty intersection, which need $|C(\alpha_i)||C(\alpha_{i+1})|$ comparisons. Similarly, Step 7 needs $|X_{r-1}||S_j|$ comparisons. It is easy to know that Step 2 to Step 8 need $k-1$ circulations, and the total computations does not exceed $(k-1)[2(|C(\alpha_i)||(\alpha_{i+1})|) + |X_{r-1}||S_j| + 2]$. Since an interval graph of order n has at most $n-2$ minimal local cuts and each minimal local cut contains at most $n-2$ vertices, the complexity of Algorithm 2 is $O(n^3)$. The proof is completed.

By the discussion above, to computing the weighted toughness of an interval graph, it is sufficient to consider the union of minimally local cuts with nonempty intersection, as well as the number of connected components of the remaining subgraphs. □

Theorem 5. *The time complexity of Algorithm 2 is $O(n^3)$.*

Proof. The total computations of Step 2 is n. In Step 3, obtaining graph $G = G - S_0$ requires $2|V||S_0|$ operations. Step 4 requires $1 + |V - X|\log^{|V-X|}$ comparisons at most. Step 6 requires $|S_{i-1}||M_j|$ comparisons. Step 7 requires $|M_j|$ comparisons to determine whether $M_j = \varnothing$. Similarly, $|V - S_i|$ comparisons are needed to determine whether $V - S_i = \varnothing$. If $M_j = \varnothing$ and $V - S_i = \varnothing$, then the total computations for Step 4 to Step 7 does not exceed $\omega(|V - X|\log^{|V-X|} + 2 + |S_{i-1}||M_j| + |M_j| + |V - S_i| + 3)$. If $M_j \neq \varnothing$, finding vertex $\{y\}$ with the largest subscript in M_j, $|M_j|\log^{|M_j|}$ comparisons are needed. If $\{y\}$ has an 'unvisited' neighbor $\{z\}$ in $V - X$, finding the neighbor of $\{y\}$ in $V - X$ needs at most $n-1$ comparisons. The total circulations of Step 9 to Step 13 are $|V - X| - 1$, so the total computations of Step 8 to Step 13 is $(|V - X| - 1)(|M_j|\log^{|M_j|} + n - 1)$. If $\{y\}$ does not have 'unvisited' neighbors, the computations for Step 7 to Step 8 is $|M_j|$. Therefore, the total computations for Step 7 to Step 13 is $(|V - X| - 1)(|M_j|\log^{|M_j|} + n - 1) + |M_j|$. Since $|M_j| \leq n - |X|$, $|S_{i-1}| \leq n$, $|S_i| \leq n$, $\omega \leq n$, $|X| \leq n - 2$, the time complexity of algorithm 2 is $O(n^3)$. The proof is completed.

Now, we give an algorithm for computing the weighted toughness and achieving cuts of interval graphs. □

Example 1. Use Algorithm 3 to compute the weighted toughness of G_1 and G_2 in Figure 2.

Consider cuts $X_1 = \{I_2, I_3\}$, $X_2 = \{I_3, I_4\}$, $X_3 = \{I_4, I_6\}$, $X_4 = \{I_9, I_{10}\}$, $X_5 = \{I_2, I_3, I_4\}$, and $X_6 = \{I_3, I_4, I_6\}$. Let $\lambda_i = w(X_i)/\omega(G - X_i)$, $i \in \{1, 2, \ldots, 6\}$.

For G_1, $\lambda_1 = 7.00$, $\lambda_2 = 7.50$, $\lambda_3 = 5,33$, $\lambda_4 = 7.00$, $\lambda_5 = 12.00$, and $\lambda_6 = 5.25$. Therefore, $t^w(G_1) = \min_{i\in\{1,2,\ldots,6\}}\{\lambda_i\} = \lambda_6 = 5.25$, and the achieving cut is $X_2 \cup X_3$. Similarly, $t^w(G_2) = 0.20$, and the achieving cut is X_3.

This example shows that the value of weighted toughness is not only related to the structure and weights of the graph but also related to the way of weight association.

Remark 2. The weight of G_1 and G_2 are randomly generated.

3. Conclusion Remarks

At last, we consider the complexity of Algorithm 3. Let $A(G) = (a_{ij})_{n\times n}$ be the adjacency matrix of an interval graph G of order n. In Algorithm 3, computing $w(X)$ and

Input: An interval graph G; minimal local cuts $C(\alpha_1)$, $C(\alpha_2), \ldots, C(\alpha_k)$.
Output: Union of minimal local cuts with nonempty intersection.

(1) **Step 1:** $j = 0; S_j = \varnothing; r = 0; X_r = \varnothing; i = 1$;
(2) **Step 2:** If $i \neq k$, turn to Step 3, otherwise, stop;
(3) **Step 3:** If $C(\alpha_i) \cap C(\alpha_{i+1}) \neq \varnothing$, turn to Step 4, otherwise, $j = 0$, output X_r, $i = i + 1$, turn to Step 2;
(4) **Step 4:** $j = j + 1$;
(5) **Step 5:** $S_j = C(\alpha_i) \cup C(\alpha_{i+1})$;
(6) **Step 6:** $r = r + 1$;
(7) **Step 7:** $X_r = X_{r-1} \cup S_j$;
(8) **Step 8:** $i = i + 1$.

Algorithm 1: Union of minimal local cuts with nonempty intersection.

Input: interval graph G, a union of minimally local cuts with nonempty intersection X.
Output: $\omega(G - X)$.

(1) **Step 1:** $\omega = 0, S_0 = \varnothing, j = 0, i = 1$.
(2) **Step 2:** set the vertices in X to 'visited', the vertices in $V - X$ to 'unvisited', $S_0 = X$.
(3) **Step 3:** $G = G - S_0$.
(4) **Step 4:** visit the vertices of $V - X$ in order of the vertex subscripts, find the 'unvisited' vertex u which has the minimum subscript, and set the vertex 'visited'.
(5) **Step 5:** $M_j = M_j \cup \{u\}$.
(6) **Step 6:** $S_i = S_{i-1} \cup M_j$.
(7) **Step 7:** if $M_j \neq \varnothing$, turn to Step 8; otherwise, if $V - S_i \neq \varnothing$, $\omega = \omega + 1$, $j = 0, i = i + 1$, return to Step 4 and if $V - S_i = \varnothing$, output $\omega + 1$.
(8) **Step 8:** find a vertex with the largest subscript in M_j, $\{y\}$; if $\{y\}$ has an 'unvisited' neighbor $\{z\}$, turn to Step 9; otherwise, $M_j = M_j - \{y\}$, return to Step 7.
(9) **Step 9:** set $\{z\}$ to 'visited'.
(10) **Step 10:** $j = j + 1$.
(11) **Step 11:** $M_j = M_{j-1} \cup \{z\}$.
(12) **Step 12:** $i = i + 1$.
(13) **Step 13:** $S_i = S_{i-1} \cup \{z\}$, return to Step 8.

Algorithm 2: Component number corresponding to the union of minimally local cuts with nonempty intersection.

Input: An interval weighted graph G, all minimal local cuts $C(\alpha_1), C(\alpha_2), \ldots, C(\alpha_k)$.
Output: $t^w(G)$ and achieving cuts.

(1) **Step 1:** use Algorithm 1 to find the union of minimal local cuts with nonempty intersection.
(2) **Step 2:** use Algorithm 2 to find the number of connected components.
(3) **Step 3:** for each cut X, compute $w(X)/\omega(G - X)$.
(4) **Step 4:** compute $\min w(X)/\omega(G - X)$; output $t^w(G)$ and achieving cuts.

Algorithm 3: Weighted toughness and achieving cuts of interval graphs.

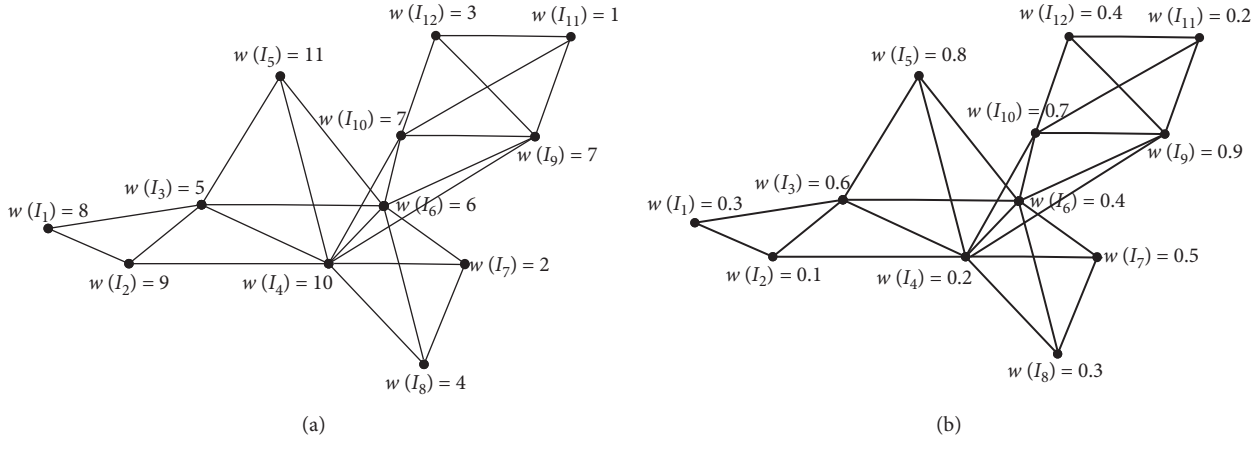

Figure 2: The weighted interval graphs (a) G_1 and (b) G_2.

$w(X)/\omega(G-X)$ in Step 3 requires $|X|-1$ additions and 1 division. Since at most $k(k+1)/2$ cuts be considered, the total operations of Step 3 is at most $k(k+1)/2|X|$. Step 4 requires $k(k+1)/2\log^{k(k+1)/2}$ comparisons, where $k \leq n-2$. Combining Theorem 4 and Theorem 5, the complexity of Algorithm 3 is $O(n^3)$. Therefore, it is a good algorithm.

In this paper, we introduce a new parameter, weighted toughness, which generalizes the concept of toughness to weighted graphs. This parameter can be used to measure the invulnerability of weighted graphs. A polynomial algorithm for computing the weighted toughness of interval graphs is given.

However, computing the weighted toughness for general graphs is NP-hard. Algorithms and complexity analysis of weighted toughness for graphs such as trees, chordal graphs, and permutation graphs may be interesting. We can also use other parameters to measure the invulnerability of weighted graphs.

Data Availability

No data were used to support this study

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (1661066), Science and Technological Fund of Shaanxi Province (2016JM1035), and Science and Technological Fund of Qinghai Province (2017-ZJ-701).

References

[1] Z. T. Wei, Y. Liu, W. Yang, and T. Wang, *In Network Invulnerability*, Xi'an Jiao Tong University Press, Xi'an, China, 2015.

[2] V. Chva, tal, Tough graphs and Hamiltonian circuits," *Discrete Math*, vol. 5, pp. 215–228, 1973.

[3] D. Bauer, S. L. Hakimi, and E. Schmeichel, "Recognizing tough graphs is NP-hard," *Discrete Applied Mathematics*, vol. 28, no. 3, pp. 191–195, 1990.

[4] D. Kratsch, T. Kloks, and H. Mller, "Computing the toughness and the scattering number for interval and other graphs," *IRISA Research Report France*, vol. 5, pp. 1–19, 1994.

[5] P. C. Gilmore and A. J. Hoffman, "A characterization of comparability graphs and of interval graphs," *Canadian Journal of Mathematics*, vol. 16, pp. 539–548, 1964.

[6] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, vol. 57, pp. 171–202, Academic Press, Cambridge, MA, USA, 2004.

[7] M. Habib, R. Mcconnell, C. Paul, and L. Viennot, "Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing," *Theoretical Computer Science*, vol. 234, no. 1-2, pp. 59–84, 2000.

[8] J. R. Jungck, G. Dick, and A. G. Dick, "Computer-assisted sequencing, interval graphs, and molecular evolution," *Biosystems*, vol. 15, no. 3, pp. 259–273, 1982.

[9] H. J. Broersma, J. Fiala, P. A. Golovach, T. Kaiser, D. Paulusma, and A. Proskurowski, "Linear-time algorithms for scattering number and Hamilton-connectivity of interval graphs," *Journal of Graph Theory*, vol. 79, pp. 259–273, 2015.

[10] F. Li, X. Zhang, and H. Broersma, "A polynomial algorithm for weighted scattering number in interval graphs," *Discrete Applied Mathematics*, vol. 264, pp. 118–124, 2019.

[11] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North-Holland, Oxford, UK, 1982.

[12] S. Ray, R. Kannan, D. Zhang, and H. Jiang, "The weighted integrity problem is polynomial for interval graphs," *Ars Combinatoria*, vol. 79, pp. 77–95, 2006.