Supplemental Material for "MEGA: Machine Learning-Enhanced Graph Analytics for Infodemic Risk Management"

Ching Nam Hang, Pei-Duo Yu, Siya Chen, Chee Wei Tan and Guanrong Chen

Abstract—This document contains the supporting materials of the manuscript titled "MEGA: Machine Learning-Enhanced Graph Analytics for Infodemic Risk Management". In Appendix A, we provide the proofs of Lemma $1-4$ and Theorem $1-2$ of the main paper. We present the pseudocodes for the triangle motif counting algorithm and distance center computing algorithm of the feature engineering step of MEGA in Appendix B. In Appendix C, we analyze the time complexity of the proposed triangle motif counting algorithm of our MEGA framework. In Appendix D, we provide an illustrative example to demonstrate how the feature engineering step of MEGA computes the distance center for single message source detection. In Appendix E, we provide more detailed experiments to assess the performance of the feature engineering step of MEGA for triangle motif counting and distance center computation. In Appendix F, we provide extensive experimental results and in-depth analysis of finding the most prominent vertices of the graph via small-world analysis.

Index Terms—Infodemic, AutoML, feature engineering, fact-checking, graph neural network, network centrality.

APPENDIX A
PROOFS

A.1 Proof of Lemma 1

Proof. According to the *degree sum formula*, the sum of the degrees of all vertices in a finite graph is twice the number of edges in that graph. Since the degree of every vertex in G' must be at least $\theta+1$, the sum of the degrees of all vertices in G' must be greater than or equal to $\theta+1$ times the number of vertices in G'. That is,

$$(\theta+1)\cdot|V(G')| \leq \sum_{v\in V(G')} deg(v) = 2|E(G')|.$$

Since G' is a pruned graph of G, the number of edges in G' must be less than or equal to that in G. Hence, we have

$$|V(G')| \leq \frac{2|E(G')|}{\theta+1} \leq \frac{2|E(G)|}{\theta+1}.$$

□

A.2 Proof of Theorem 1

Proof. We denote a BA network with parameters m_0 and m as G and the initial complete network of m_0 vertices in G as G_{m_0}, where $m_0 > m$. Then, we have

$$\forall v \in G \setminus G_{m_0}, deg(v) \leq m.$$

If we set $\theta^* = m$, it is obvious that the size of the pruned graph must be equal to m_0 which is the size of G_{m_0}. If $m = m_0 - 1$, then we can completely decompose G. □

A.3 Proof of Lemma 2

Proof. Let $G(N,p)$ be an ER random network. Then, the distribution of the vertex degrees is Poisson as the graph size N goes to infinity. Thus, we can use the Chernoff bound [1] to compute the upper bound of the size of the pruned graph. Let v be any vertex in $G(N,p)$ and $\theta^* = kNp$ be the optimal threshold for pruning, where k is a positive integer, then we have

$$\begin{aligned}\textbf{prob}(deg(v)\geq\theta^*) &\leq \frac{(eNp)^{\theta^*}e^{-Np}}{(\theta^*)^{\theta^*}}\\ &= \Big(\frac{Np}{\theta^*}\Big)^{\theta^*}\cdot e^{\theta^*-Np}\\ &= \Big(\frac{1}{k}\Big)^{kNp}\cdot e^{(k-1)Np}\\ &< e^{-Np}\cdot\Big(\frac{e}{k}\Big)^{kNp}.\end{aligned}$$

Therefore, we can have a simple upper bound for the size of the pruned graph, N', as follows:

$$N' < e^{-Np}\cdot\Big(\frac{e}{k}\Big)^{kNp}\cdot N.$$

Note that for an ER random network $G(N,p)$, the value Np is the average degree of the graph, which is a constant. Hence, the size of the pruned graph decreases exponentially as k increases. It is worth noting that the key property used to prove Lemma 2 is that the probability distribution of a vertex has degree k satisfies the Poisson distribution when the size of the graph is large enough. □

A.4 Proof of Lemma 3

Proof. Let $G = (V(G), E(G))$ be a planar graph. It is known that the average degree of G is strictly less than 6. It implies that there must be at least one vertex with degree less than or equal to 5 in G. After removing such vertices from G, there is always at least one new vertex to prune since any subset of a planar graph is also a planar graph. Ultimately, G can be completely decomposed. □

A.5 Proof of Theorem 2

Proof. Let $v \in K_j$, where j is an integer such that $0 \leq j \leq maxC$. Let $\tilde{v}$ denote the set of pruned vertices such that v is the ancestor of these pruned vertices (e.g., $v_5 \in \tilde{v_4}$ in Fig. 14). First, we consider the distance from vertex u to all the vertices in $\tilde{v}$ including v itself. Without loss of generality, we assume that the common ancestor w of u and v is in level h. Then, we have

$$d(u,v) = |h-i| + |h-j| \geq |i-j|,$$

where the equality holds if $u = w$ or $v = w$. Then, for other vertices in $\tilde{v}$, we have

$$\begin{aligned}\sum_{x\in\tilde{v}} d(u,x) &= \sum_{x\in\tilde{v}} [d(u,v) + d(v,x)] \\ &= T(v)\cdot d(u,v) + D(v) \\ &\geq T(v)\cdot |i-j| + D(v).\end{aligned}$$

Hence, we have

$$\begin{aligned}S(u,G) &\geq \sum_{0\leq j\leq maxC} \left\{ \sum_{v\in K_j} [T(v)\cdot|i-j| + D(v)] \right\} \\ &= \sum_{0\leq j\leq maxC} \left[|i-j| \cdot \sum_{v\in K_j} T(v) + \sum_{v\in K_j} D(v) \right] \\ &= \sum_{0\leq j\leq maxC} [|i-j|\cdot T(K_j) + D(K_j)].\end{aligned}$$

□

A.6 Proof of Theorem 3

Proof. First, we show that G' must contain $C_{dist}(G)$ whenever $N' \geq N/2$. We prove this part by contradiction. Assume that $N' \geq N/2$ and the distance center $v_c \in G\backslash G'$.

To prove this theorem, we define an edge (u,v) as a bridge if the removal of (u,v) disconnects G. Assume (u,v) is a bridge. Let C_u^v and C_v^u denote the connected component containing u and v respectively after removing (u,v).

Lemma 5. If edge (u,v) is a bridge in G, then we have

$$S(u,G) = S(v,G) + |C_v^u| - |C_u^v|. \tag{1}$$

Proof. Let u and v be two vertices of G such that (u,v) is a bridge in G. Then, for any vertex $v_i \in C_v^u$ and $u_j \in C_u^v$, we have

$$dist(v_i,u_j) = dist(v_i,v) + 1 + dist(u_j,u).$$

Hence, the distance centrality of u in G, $S(u,G)$, can be rewritten in terms of the distance centrality of u and v in C_u^v and C_v^u respectively. We have

$$\begin{aligned}S(u,G) &= \sum_{w\in G} dist(u,w) \\ &= \sum_{w\in C_u^v} dist(u,w) + 1 + \sum_{w\in C_v^u} dist(u,w) \\ &= S(u,C_u^v) + 1 + \sum_{w\in C_v^u} [dist(v,w)+1] \\ &= S(u,C_u^v) + 1 + S(v,C_v^u) + |C_v^u|.\end{aligned}$$

By following the same approach, we have

$$S(v,G) = S(u,C_u^v) + 1 + S(v,C_v^u) + |C_u^v|.$$

Combining two results above, we have

$$S(u,G) - S(v,G) = |C_v^u| - |C_u^v|.$$

□

Lemma 6. Let G' be the pruned graph and $v \in G\backslash G'$. Then, v is either a degree-1 vertex or an endpoint of a bridge.

Proof. Assume that v is not an endpoint of a bridge. Then, v must be contained in a cycle of G. The degree of v must be larger than or equal to 2 during pruning, which implies that $v \in G'$ and contradicts the assumption that $v \in G\backslash G'$. □

By Lemma 6, we conclude that v_c must be an endpoint of a bridge in G. We consider two distinct cases based on the distance between v_c and G' in the following, which is defined as usual by $dist(v_c,G') = \min_{w\in G'}\{dist(v_c,w)\}$.

Case 1: Assume that $dist(v_c,G') = 1$. Then, there is only one vertex $u \in G'$ such that $dist(v_c,u) = 1$ and (u,v_c) is a bridge by Lemma 6. Moreover, we have $S(v_c,G) = S(u,G) + |C_u^{v_c}| - |C_{v_c}^u|$ by Lemma 5. Since $|C_u^{v_c}| \geq N' \geq N/2$, we have $S(v_c,G) \geq S(u,G)$, which contradicts the assumption that v_c is the distance center.

Case 2: Assume that $dist(v_c,G') \geq 2$. From Case 1, we can deduce that for any two vertices u and v in $G\backslash G'$ and (u,v) is a bridge with $dist(u,G') < dist(v,G')$. Then, $S(u,G) < S(v,G)$. This implies that, for all v with $dist(v,G') \geq 2$, there is a neighbor u of v such that $S(u,G) < S(v,G)$, which is a contradiction.

Therefore, we conclude that if $N' \geq N/2$, then G' must contain the distance center.

Now, let us consider the case that $C_{dist}(G) \notin G'$. Note that for each pruned neighbor u of $v' \in G'$, we know that (v',u) is a bridge of G. Hence, we can compute $S(u,G)$ immediately by using Lemma 5. In particular, since v' has the minimum distance centrality on G', there is a unique pruned neighbor of v', say u', such that $S(u',G) < S(v',G)$. Since $C_{u'}^{v'}$ is a tree, we can apply the result of Theorem 3 in [2] to complete the rest of the proof. □

APPENDIX B
PSEUDOCODES

B.1 Triangle Motif Counting

The triangle motif counting algorithm (i.e., the computing in the feature engineering step) is implemented by Algorithm 1, where l is the number of disconnected components after graph pruning, $maxC$ is the total number of clusters in G_i, and MEGA($G_{i,j}$) is the recursive function of the feature engineering step for triangle motif counting in $G_{i,j}$.

Algorithm 1: Triangle Motif Counting

Input : Graph G', vertex sets $K_{i,j}$, number $l, maxC$
Output: Triangle count $\lambda(v)$ of each vertex $v \in V(G)$
for $i = 1, 2, \ldots, l$ **do**
  for $a, b \in N_{G_i}(v_r^i)$ **do**
    if $(a, b) \in E(G)$ **then**
      $\lambda(v_r^i) \leftarrow \lambda(v_r^i) + 1$
      $\lambda(a) \leftarrow \lambda(a) + 1$
      $\lambda(b) \leftarrow \lambda(b) + 1$
    end
  end
  remove vertex v_r^i from G_i
  for $j = 1, 2, \ldots, maxC$ **do**
    for $v \in K_{i,j}$ **do**
      $N^{\uparrow}_{G'}(v) \leftarrow N_{G'}(v) \bigcap K_{i,j-1}$
      $N^{\downarrow}_{G'}(v) \leftarrow N_{G'}(v) \bigcap K_{i,j+1}$
      for $a, b \in N^{\uparrow}_{G'}(v)$ *or* $a, b \in N^{\downarrow}_{G'}(v)$ **do**
        if $(a, b) \in E(G)$ **then**
          $\lambda(v) \leftarrow \lambda(v) + 1$
          $\lambda(a) \leftarrow \lambda(a) + 1$
          $\lambda(b) \leftarrow \lambda(b) + 1$
        end
      end
      $G_{i,j} \leftarrow (V(K_{i,j}), E(K_{i,j}))$
      MEGA($G_{i,j}$)
    end
  end
end
return λ

B.2 Distance Center Computation

Algorithm 2 describes the feature engineering step of MEGA for distance center computation in the input graph G. Based on Theorem 3, if $N' < N/2$, where N' is the cardinality of $V(G')$ and N is the cardinality of $V(G)$, we can use Corollary 1 to backtrack the removed distance center in G.

Algorithm 2: Distance Center Computation

Input : Graph G, graph G', lower bounds S_{LR}
Output: distance center of G, $C_{dist}(G)$
$Q_s \leftarrow$EnQueue(v_r, $S(v_r, G')$)
foreach v *in* $V(G')\backslash\{v_r\}$ **do**
  $Q_s \leftarrow$EnQueue(v, $S_{LR}(v, G')$)
end
while *continue* = *true* **do**
  (v, $S_{LR}(v, G')$)$\leftarrow$ DeQueue(Q_s)
  if $S(v, G') = S_{LR}(v, G')$ **then**
    $C_{dist}(G') \leftarrow v$
    continue$\leftarrow$false
  else
    $Q_s \leftarrow$EnQueue(v, $S(v, G')$)
  end
end
if $N' \geq N/2$ **then**
  $C_{dist}(G) \leftarrow C_{dist}(G')$ (see Theorem 3)
end
return $C_{dist}(G)$

APPENDIX C
TIME COMPLEXITY OF TRIANGLE MOTIF COUNTING

The time complexity of the graph pruning step depends on the number of pairs of vertices that we need to verify (whether an edge exists between them). Let $P(G)$ be the set of pruned vertices of G with threshold θ, i.e., $P(G) = \{v \in V(G) \mid v \notin V(G')\}$. Then, the time complexity of the pruning step is bounded above by $\binom{\theta}{2} \cdot |P(G)|$. Assume that $\theta \ll N$. Then, the time complexity of the pruning step is bounded above by $O(N)$ since $|P(G)| \leq N$.

In hierarchical clustering, we select a vertex v_r^i as an optimal root of the BFS for G_i to obtain the distances from v_r^i to any other vertex in G_i. Hence, the time complexity of the hierarchical clustering step is $O(|V(G')| + |E(G')|) = O(|E(G')|)$ since we only consider graphs with $|V(G')| \leq |E(G')|$ in the triangle motif counting problem.

In the computing step, each vertex v has three types of neighbors: the neighbors in the upper cluster $N^{\uparrow}_{G'}(v)$, the neighbors in the lower cluster $N^{\downarrow}_{G'}(v)$, and the neighbors in the same cluster $N^{\rightarrow}_{G'}(v)$. Therefore, the number of pairs of vertices that we need to check for each vertex v is at most

$$\binom{|N^{\uparrow}_{G'}(v)|}{2} + \binom{|N^{\downarrow}_{G'}(v)|}{2} + \binom{|N^{\rightarrow}_{G'}(v)|}{2} \leq \binom{deg_{\max}}{2},$$

where $deg_{\max}$ is the maximum degree of G'. Thus, the time complexity of the computing step is $O(N' \cdot \binom{deg_{\max}}{2})$.

We thus conclude that the total time complexity of the feature engineering step of MEGA for solving the triangle motif counting problem is $O(|E(G')| + N' \cdot \binom{deg_{\max}}{2})$.

APPENDIX D
ILLUSTRATIVE EXAMPLE OF DISTANCE CENTER COMPUTATION

We use an example to illustrate the difference of finding the message source estimator between the feature engineering step of MEGA and the BFS heuristic algorithm in [3]. Using the same example as in [3], we have a spread graph as shown in Fig. 14. There are 5 vertices received the message (v_1 to v_5), and we want to find the distance-based message source estimator of this spread graph using MEGA.

The input graph has five vertices (v_1 to v_5) that received the message, and our goal is to find the distance-based message source estimator using the MEGA framework. In the first step of graph pruning with $\theta = 1$, we remove v_5 and update its parent v_4 such that $T(v_4) = 2$ and $D(v_4) = 1$, resulting in the pruned graph G'. We then use every vertex in the pruned graph as the root to perform BFSs and determine the worst-case performance. Starting with v_1 as the root, we obtain the following results:

$$\begin{aligned} S(v_1, G') &= 0 + 1 + 3 + 2 = 6 \\ S_{LR}(v_2, G') &= 1 + 0 + 1 + 5 = 7 \\ S_{LR}(v_3, G') &= 2 + 1 + 0 + 3 = 6 \\ S_{LR}(v_4, G') &= 1 + 2 + 1 + 1 = 5. \end{aligned}$$

Since $S_{LR}(v_4, G')$ is the smallest, we need to check if $S(v_4, G') = S_{LR}(v_4, G')$ by using v_4 as the root for the second BFS tree traversal. As we have $S(v_4, G') = S_{LR}(v_4, G')$ and the lower bound of every vertex remains unchanged,

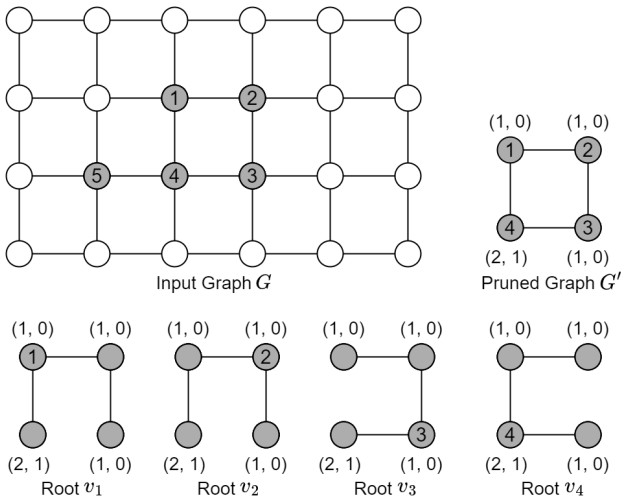


Fig. 14. Example illustrating how MEGA finds the distance-based message source estimator of a spread graph. There are totally 5 vertices received the message (in gray color). After graph pruning, we have a pruned graph G'. We then select v_1 to v_4 as the root in hierarchical clustering respectively and see how many BFSs we have to build for each root.

we find that v_4 is the distance-based message source estimator of G'. Since $N' > N/2$, based on Theorem 3, we conclude that v_4 is the message source of G.

Similarly, for the roots v_2 and v_3, we need to perform two BFS tree traversals as well. However, for the root v_4, we have

$$\begin{aligned} S(v_4, G') &= 1+2+1+1 = 5 \\ S_{LR}(v_1, G') &= 0+1+2+3 = 6 \\ S_{LR}(v_2, G') &= 1+0+1+5 = 7 \\ S_{LR}(v_3, G') &= 2+1+0+3 = 6. \end{aligned}$$

Since $S(v_4, G')$ is already the smallest, we conclude that v_4 is the message source of G by using only one BFS tree. As such, in the worst case, we only need to perform BFS twice for finding the distance-based message source estimator of the spread graph. Meanwhile, v_4 is also identified as the rumor center using the BFS heuristic algorithm in [3] but it requires to construct BFS trees starting from every vertex in the spread graph. Thus, in this example, the number of BFSs performed and the number of edges visited by MEGA are both less than that of the algorithm in [3].

APPENDIX E
EXPERIMENTAL PERFORMANCE EVALUATION

In this section, we provide more detailed experiments to assess the performance of the feature engineering step of MEGA for triangle motif counting and distance center computation on (simple undirected) social networks.

E.1 Triangle Motif Counting on Social Networks

We evaluate the performance of the feature engineering step of MEGA on the triangle motif counting problem using the social networks provided by Stanford Large Network Dataset Collection (SNAP) [4].

E.1.1 Optimal Threshold Tuning

Fig. 15 shows how the threshold θ affects the computation time of the feature engineering step of MEGA for triangle motif counting on eight real-world networks provided by SNAP. We also compare the performance of MEGA using the degree center and a random vertex as the BFS root in hierarchical clustering since other centrality measures such as PageRank require global information from all other vertices that must increase the computational complexity. We see that MEGA with the degree center performs better than that with a random vertex. We also note that finding an optimal threshold for an arbitrary graph is not a straightforward task unless we further exploit the inherent structure of the graph.

Based on Lemma 1, we use a data-driven (statistical) model to approximate the optimal threshold θ^* for MEGA. We generate 6,000 synthetic networks in which the optimal thresholds θ^* are computed. Each network has 1,000 vertices, and the number of edges of each network is not fixed. Note that the number of vertices of the input graph is not a critical parameter for computing θ^*. In Fig. 16, we see that the gradient of the blue line, which is a linear regression model that fits the results, is relatively small that keeps θ^* always less than 100, even the graph size starts to increase. We also note that noisy data only appear when the graph size is comparatively small. Therefore, the optimal threshold θ^* computed by this model must satisfy the condition of $\theta^* \ll N$ and balance the trade-off for large-scale networks.

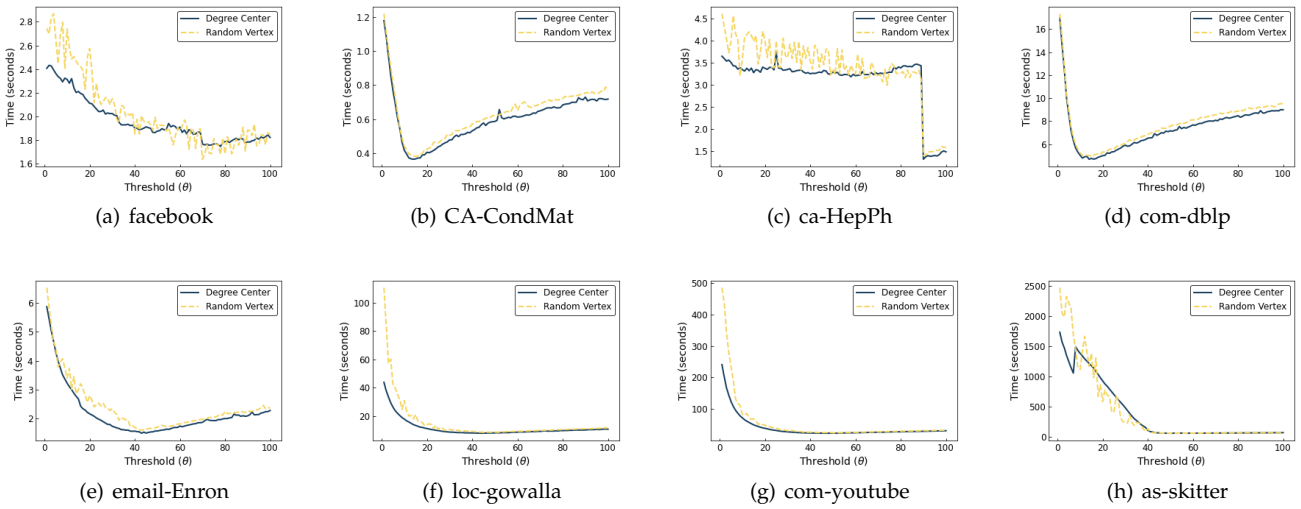
E.1.2 Evaluation on Social Networks

We compare MEGA with the algorithm in [5], which is an award-winning work of the MIT/Amazon/IEEE Graph Challenge. The algorithm in [5] assigns direction to each edge based on the degree of each vertex. It implies that, for each vertex v, there are $\binom{deg^+(v)}{2}$ pairs of vertices needed to be checked, where $deg^+(v)$ is the outdegree of v. Hence, its time complexity is $O(|E(G)|+N\cdot\binom{deg^+_{\max}}{2})$, where $|E(G)|$ is the time complexity of the direction assignment and $deg^+_{\max}$ is the maximum outdegree. This time complexity is the same as that of MEGA (cf. Appendix C), and this is why we choose it as a baseline. In Fig. 17, we see that MEGA with θ^* outperforms the algorithm in [5] on different social networks by averagely 25.3 times faster.

E.1.3 Evaluation on Synthetic Random Networks

We compare the performances of MEGA and the algorithm in [5] on ten large-scale synthetic random networks generated by the BA model and the ER model respectively.

For BA network, we randomly generate the two parameters (m_0, m) for the BA model so as to obtain ten random networks, where each network contains 1 million vertices and $9,999,945$ edges. We use Theorem 1 to compute the optimal threshold θ^* for each BA network such that MEGA can optimally decompose the BA networks in pruning. In Fig. 18(a), we see that although some vertices have a large degree, MEGA can still be able to decompose the networks with θ^* and outperform the algorithm in [5]. Moreover, since we only execute the pruning step with a small threshold ($\theta^* \ll N$), the time complexity of MEGA is in $O(N)$.



(a) facebook (b) CA-CondMat (c) ca-HepPh (d) com-dblp

(e) email-Enron (f) loc-gowalla (g) com-youtube (h) as-skitter

Fig. 15. The performance of MEGA on eight real-world networks with different thresholds θ ranging from 0 to 100. The y-axis is the running time in seconds, and the x-axis is the value of θ. We also compare the performance of MEGA using the degree center (blue) and a random vertex (yellow) as the BFS root for hierarchical clustering.

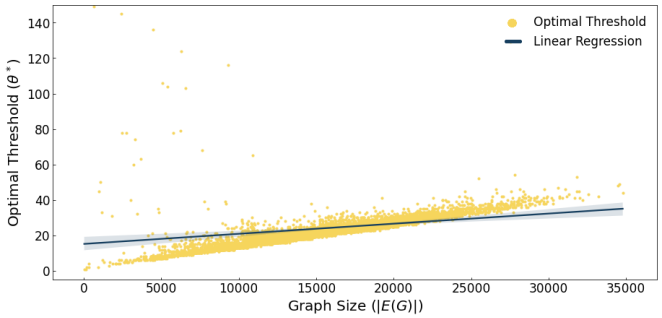


Fig. 16. Optimal threshold tuning on 6,000 synthetic networks based on Lemma 1. The y-axis is the value of the optimal threshold θ^*, and the x-axis is the graph size $|E(G)|$ of each network. The blue line is a linear regression model that fits the results.

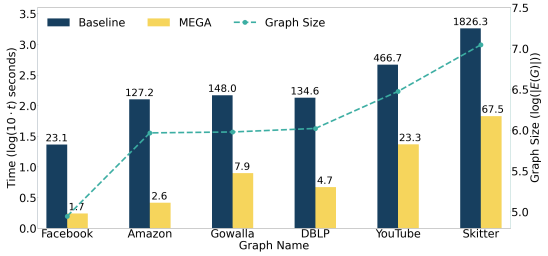


Fig. 17. Comparison between MEGA and the algorithm in [5] (which we set as baseline) on six social networks. The primary y-axis (blue) is the running time t in seconds, and the secondary y-axis (green) is the graph size $|E(G)|$. A log scale is used for these two axes, and we multiply t by 10 to avoid negative values (i.e., primary = $\log(10 \cdot t)$ seconds and secondary = $\log(|E(G)|)$). The number on top of each bar is the exact running time of both algorithms.

Similarly, we generate ten ER random networks with the same size as the BA networks. In Fig. 18(b), we observe that MEGA beats the algorithm in [5] with θ^*. Note that the probability of a given vertex having degree greater than θ^* in the ER networks is less than 8%, which implies that the ER networks are largely decomposed in pruning.

E.2 Distance Center Computation on Social Networks

In this section, we assess the performance of the feature engineering step of MEGA for distance center computation on different social networks provided by SNAP [4].

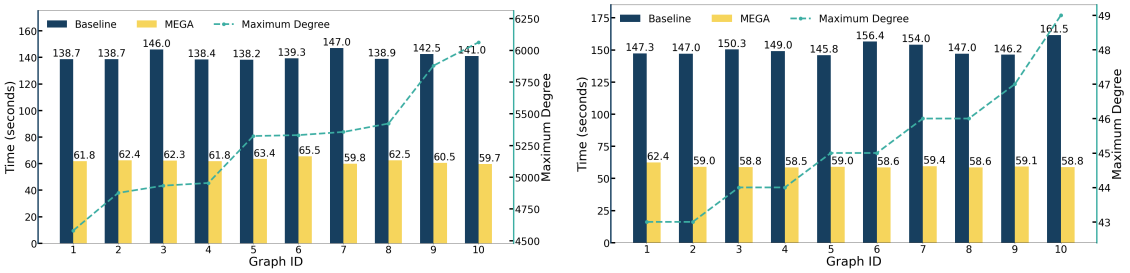
We consider both performance improvement and accuracy on single message source detection. We first compare MEGA with the algorithm presented in [6] based on the speedup which has been defined in Section 7.4.1 and then compare the accuracy of single message source detection with the work in [3].

E.2.1 Evaluation on Social Networks

The graph pruning of our MEGA performs very well on graphs with large diameter, outperforming the NBBound proposed in [6]. However, the efficacy of graph pruning in small-world networks is relatively low since the graph diameter of such networks is small. To further increase the speedup of MEGA, we utilize the *Multi-Source BFS (MS-BFS)* [7]. The idea is motivated by the observation that, when running a large number of BFSs sequentially, most of the edges are visited multiple times, which might deteriorate the overall performance. The MS-BFS algorithm addresses this problem by running BFSs from multiple sources concurrently. When a set of BFSs visits the same vertex, this vertex will be visited only once, and its information will be shared with all BFSs in the set. Thus, a large number of visits can be shared by multiple BFSs when applying the MS-BFS algorithm. In Fig. 19, we see that MS-BFS greatly increases the speedup of MEGA.

E.2.2 Single Message Source Detection in Social Networks

We perform simulations on six social networks and compare the performances of our MEGA framework with the BFS heuristic in [3]. For each graph, we randomly select a message source vertex and let the message spread to 100 vertices. We then apply MEGA and the BFS heuristic in [3] respectively to find a source estimator. Note that the error indicates the graph distance from the source estimator to



(a) BA random networks (b) ER random networks

Fig. 18. Comparison between MEGA and the algorithm in [5] (which we set as baseline) on ten BA random networks (left) and ten ER random networks (right) respectively. Each random network has 1M vertices and 9.9M edges. The primary y-axis (blue) is the running time in seconds, and the secondary y-axis (yellow) is the maximum degree of each random network.

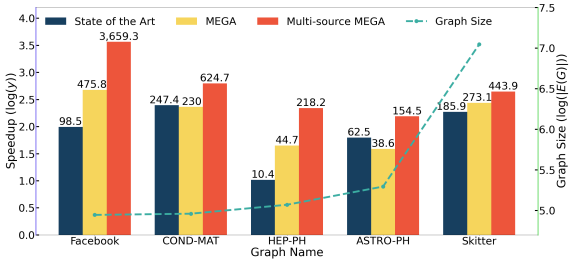
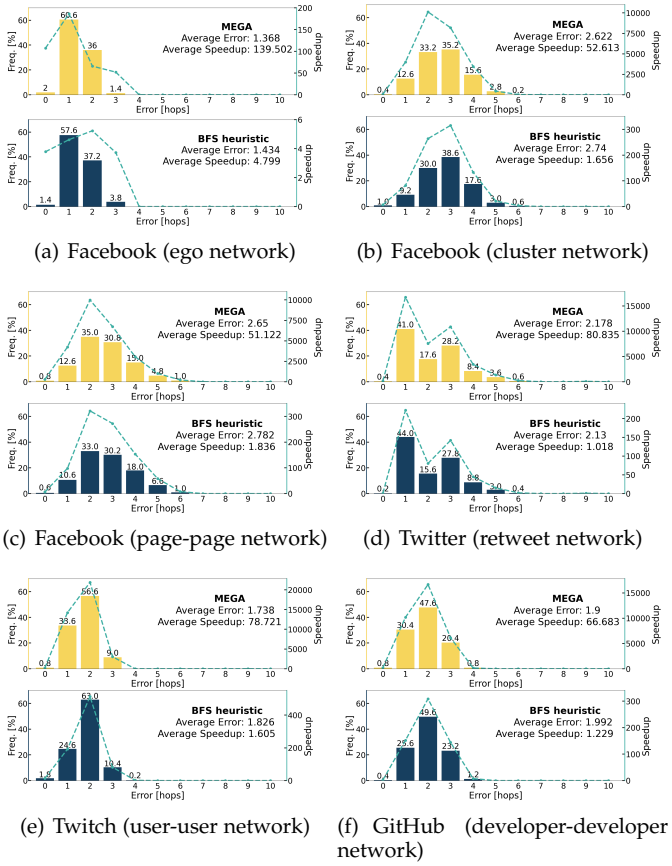


Fig. 19. Comparison between MEGA and the algorithm in [6] (which we set as baseline) on five social networks. The primary y-axis (blue) is the speedup y, and the secondary y-axis (green) is the graph size $|E(G)|$. A log scale is used for these two axes. The number on top of each bar is the exact speedup of both algorithms.



(a) Facebook (ego network) (b) Facebook (cluster network)

(c) Facebook (page-page network) (d) Twitter (retweet network)

(e) Twitch (user-user network) (f) GitHub (developer-developer network)

Fig. 20. Histograms of the error for MEGA (yellow) and the BFS heuristic in [3] (blue) on six different types of social networks with 100 vertices received the message in the spread graph. The green dotted line is the average speedup for each error and the average error is the average distance (in terms of hop count) between the real source and the estimators calculated by both methods over 500 simulations.

the real source vertex. We perform over 500 simulations and calculate the average error and speedup for each graph. In Fig. 20, we can observe that the speedup of our MEGA framework is significantly larger than that of the BFS heuristic in [3] and there is only a slight difference in accuracy (average error) between these two approaches.

APPENDIX F
FINDING THE MOST PROMINENT VERTICES THROUGH SMALL-WORLD ANALYSIS

In the paper, we have identified three parts in which finding the most prominent vertices of the graph is required: (1) finding the root for BFS in hierarchical clustering for triangle counting, (2) finding the root for BFS in hierarchical clustering for distance center computation, and (3) using distance centrality to find influential vertices. To address (1), we suggest using the degree center as the root, as it does not require additional complex computation. In contrast, other approaches could lead to increased computational complexity, and choosing a random root would be meaningless. By choosing the degree center as the root, it's possible to minimize the total number of clusters and potentially reduce computational complexity. However, further analysis is required for (2) and (3), which we will discuss below.

To perform our analysis, we generate five small-world networks using the Watts-Strogatz model, with each network containing 50, 100, 500, 1,000, and 2,000 vertices, respectively. In each network, each vertex is connected to 60% of its nearest neighbors in a ring topology, with each edge having a rewiring probability of 0.5. We identify hubs as vertices with high degree, low distance centrality, or high betweenness centrality. The number of edges for each network is listed below:

50 Vertices	100 Vertices	500 Vertices	1,000 Vertices	2,000 Vertices
750	3,000	75,000	300,000	1,200,000

For (2), we suggest selecting a vertex v with minimum degree in the pruned graph, followed by selecting a vertex with the largest eccentricity (i.e., maximum distance away

TABLE 1
Comparison of the running time (in seconds) between MEGA and the small-world analysis approach using degree center as the root of the BFS in hierarchical clustering. The better-performing method is indicated in bold text.

Approach	50 Vertices	100 Vertices	500 Vertices	1,000 Vertices	2,000 Vertices	Retweet Network
Degree Center	**0.001**	0.002	**0.061**	**0.408**	**2.36**	4,223.4
MEGA	0.002	0.002	0.069	0.432	2.597	**4,155.2**

TABLE 2
Comparison of the speedup between MEGA and the small-world analysis approach using degree center as the root of the BFS in hierarchical clustering. The better-performing method is indicated in bold text.

Approach	50 Vertices	100 Vertices	500 Vertices	1,000 Vertices	2,000 Vertices	Retweet Network
Degree Center	765.3	1,515.2	75,150.3	300,300.3	1,200,600.3	247.4
MEGA	765.3	1,515.2	75,150.3	300,300.3	1,200,600.3	**256.8**

from v) as the root. We compare the performance of this approach with using the degree center as the root, as it is not appropriate to use hubs with low distance centrality or high betweenness centrality as the root in this step when our main objective is to identify the distance center. The results below indicate that for small-world networks, using the degree center can reduce the algorithm's running time, even though the speedups of both approaches are the same. As the majority of vertices have low distance centrality and high degree centrality, most neighboring vertices can be reached by a small number of hops. Therefore, the speedups for both approaches should be almost equivalent. Additionally, the efficacy of the graph pruning step of the MEGA framework in small-world networks is relatively low since the graph diameter of such networks is small, and the graph pruning is not triggered. As a result, the framework's average performance on small-world networks is only moderate, although the speedups appear relatively large, they are only proportional to the number of edges in the graphs. However, in the retweet network, our proposed approach outperforms the small-world analysis approach. Therefore, we agree with the reviewer that if the input graph exhibits small-world network properties, the small-world analysis approach can provide better performance in terms of running time. Still, if the input is unlikely to be a small-world network, our proposed approach is more efficient.

For (3), when dealing with small-world networks that have more than 100 vertices, we randomly choose a message source vertex and allow the message to propagate to 100 vertices (at most) for each message spread graph. For small-world networks with 100 vertices or less, we randomly choose a message source vertex and allow the message to propagate to the entire graph. To find the source estimator, we utilize distance centrality (low), degree centrality (high), and betweenness centrality (high). For small-world networks, we use MEGA to compute distance centrality to find the distance center, Stanford Network Analysis Platform (SNAP) to compute betweenness centrality to find the betweenness center and SNAP to find the degree center. In the retweet network, we use the same methods used in small-world networks to identify the top 5,195 vertices with the smallest distance centrality, largest betweenness centrality, and largest degree centrality, respectively. The error indicates the graph distance from the source estimator to the real source vertex. We conduct over 500 simulations and calculate the average error and average running time for each centrality measure in each small-world network. The results below show that in small-world networks, the average errors for the hubs with high degree and high betweenness centrality are slightly higher than those for distance centrality. Additionally, in the retweet network, the performance of using distance centrality is better than that of degree and betweenness centrality. While using small-world analysis could potentially decrease computational costs since degree centrality is the fastest in finding the hubs, betweenness and distance centrality require much longer time, especially for betweenness centrality. The trade-off is that the accuracy will be lower if such a faster approach is applied. Therefore, we choose distance centrality as it is more accurate, but we also optimize the process of finding hubs with low distance centrality to balance the trade-off.

REFERENCES

[1] E. Upfal and M. Mitzenmacher, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[2] P. Yu, C. W. Tan, and H. Fu, "Averting cascading failures in networked infrastructures: Poset-constrained graph algorithms," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 4, pp. 733–748, 2018.

[3] D. Shah and T. Zaman, "Rumors in a network: Who's the culprit?" *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5163–5181, 2011.

[4] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," 2014.

[5] R. Pearce, "Triangle counting for scale-free graphs at scale in distributed memory," in *High Performance Extreme Computing Conference*, 2017, pp. 1–4.

[6] E. Bergamini, M. Borassi, P. Crescenzi, A. Marino, and H. Meyerhenke, "Computing top-k closeness centrality faster in unweighted graphs," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 5, 2019.

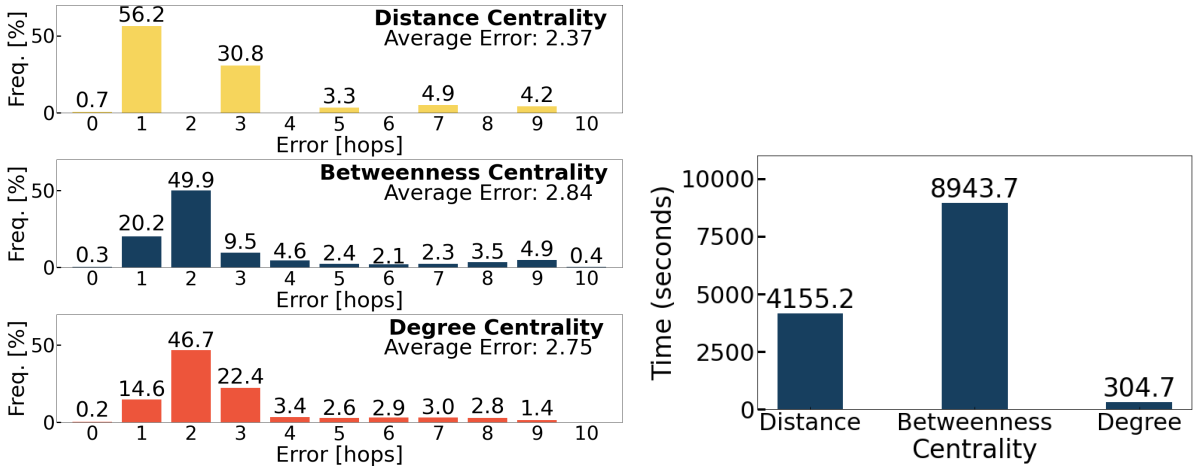
[7] M. Then, M. Kaufmann, F. Chirigati, T.-A. Hoang-Vu, K. Pham, A. Kemper, T. Neumann, and H. T. Vo, "The more the merrier: Efficient multi-source graph traversal," *Proceedings of the VLDB Endowment*, vol. 8, no. 4, pp. 449–460, 2014.

TABLE 3
Comparison of the average error in identifying prominent vertices in small-world networks using distance centrality (MEGA), betweenness centrality, and degree centrality as the metrics. The best approach is indicated in bold.

Approach	50 Vertices	100 Vertices	500 Vertices	1,000 Vertices	2,000 Vertices
Distance Centrality (MEGA)	**1.29**	**2.11**	**2.37**	**2.27**	**2.59**
Betweenness Centrality	1.31	2.21	2.44	2.32	2.63
Degree Centrality	1.32	2.18	2.47	2.36	2.60

TABLE 4
Comparison of the average running time (in seconds) in identifying prominent vertices in small-world networks using distance centrality (MEGA), betweenness centrality, and degree centrality as the metrics. The best approach is indicated in bold.

Approach	50 Vertices	100 Vertices	500 Vertices	1,000 Vertices	2,000 Vertices
Distance Centrality (MEGA)	0.0021	0.0024	0.0028	0.0023	0.0027
Betweenness Centrality	0.0027	0.0031	0.0033	0.0029	0.0030
Degree Centrality	**0.0012**	**0.0012**	**0.0013**	**0.0011**	**0.0012**



(a) Average Error (b) Running Time

Fig. 21. Comparison of the average error and running time (in seconds) in identifying prominent vertices in the retweet network using distance centrality (MEGA), betweenness centrality, and degree centrality as the metrics.