A APPENDIX A

This appendix provides pseudo-code to some of the sparsification algorithms as a supplement to the paper.

A.0.1 Rank Degree Sparsifier.

Algorithm 1 *Rank Degree* sparsifier

```
procedure RankDegreeSparsifier(G)
    Input: G: Graph to sparsify
    Input: ρ: 0 < ρ ≤ 1 selects top ρ * #neighbors for each vertex
    Output: H: Sparsified graph

    seeds = [u_1, u_2, ..., u_s]                          ▷ selects s vertices uniformly at random
    V_H = ∅, E_H = ∅                                      ▷ initialize vertex and edge sets in H
    while |V_H| < |V_G| do
        new_seeds=∅
        for all u ∈ seeds do
            neighs=getNeighborsOf(u)
            ranks = {}                                    ▷ dictionary, key is vertex, value is degree
            for all v ∈ neighs do
                ranks[v] = getDegreeOf(v)
            sort ranks by value
            select top k=ρ×len(neighbors), v_1, ..., v_k
            new_seeds = new_seeds ⋃ [v_1, ..., v_k]
            E_H = E_H ⋃ [(u, v_1), ..., (u, v_k)]
        seeds=new_seeds
    H = {Vertex(V_G), Edge(E_H)}
    return H
```

A.0.2 Local Degree Sparsifier.

Algorithm 2 Local degree score

```
procedure getEdgeScore(G)
    Input: G: Graph to calculate edge scores
    Output: Scores: An array of edge scores for each edge

    scores = [0, ..., 0]                                  ▷ scores is an array of length #edges
    for all v_i ∈ V do                                    ▷ iterate all vertex v in V
        d_i = degree(v_i)
        neighbor_degree = {}                              ▷ dictionary, key is edge id, value is degree
        for all v_j ∈ Neighbor(v_i) do                    ▷ find degrees of all neighboring vertex
            d_j = degree(v_j)
            eid = e_ij.edgeID
            neighbor_degree[eid] = d_j

        sort(neighbor_degree)                             ▷ sort by degree

        last_rank, last_degree, num_same = 0
        neighbor_rank = {}                                ▷ dictionary, key is edge id, value is rank
        for all item ∈ neighbor_degree do                 ▷ compute the rank of neighbors by degree
            eid = item.key
            d_j = item.value
            if d_j == last_degree then                    ▷ same rank for same degree
                num_same++
            else
                last_rank += num_same
                num_same = 1
                last_degree = d_j
            neighbor_rank[eid] = lask_rank

        for all item ∈ neighbor_degree do
            eid = item.key
            rank = item.value
            s = 1.0 - log(rank)/log(d_i)
            // score for an edge can be updated multiple times, take the max score
            scores[eid] = max(scores[eid], s)
    return scores
```

A.0.3 *t-Spanner.*

Algorithm 3 Greedy algorithm for t-spanner construction

```
1:  procedure Construct t-spanner(G, t)
2:      Input: G: Original graph
3:      Input: t: stretch factor, must be an odd number > 1
4:      Output: H: t-spanner graph
5:
6:      H ← (V, ∅)                                  ▷ Init H to have the same set of vertices, and no edges
7:      for all e_uv ∈ E in non-decreasing order do
8:          if d_H(u, v) > t w(u, v) then
9:              add e_uv to H
10:     return H
```

A.0.4 *Forest Fire.*

The *Forest Fire* model can be described more formally as follows (modified from [**?**]):

(1) A new vertex $\boldsymbol{u}$ chooses an existing vertex $\boldsymbol{v}$ uniformly at random, and forms an edge to it.
(2) Two random numbers x and y are generated geometrically distributed with means $p/(1-p)$ and $rp/(1-rp)$, where p is the forward burning probability, and r is the backward burning ratio.
(3) Vertex $\boldsymbol{v}$ selects x outgoing edges and y incoming edges that are not visited yet, if there are not enough unvisited edges, select all. For undirected edges, every edge can be both an outgoing and incoming edge. Let $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{x+y}$ denote the other end of the selected edges.
(4) Vertex $\boldsymbol{u}$ forms edges to the $\boldsymbol{w}_1, \boldsymbol{w}_1, ..., \boldsymbol{w}_{x+y}$.
(5) Repeat (3) and (4) recursively to each of the $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{x+y}$, until no edge can be added.

Algorithm 4 Forest Fire Score

```
1:  procedure getEdgeScore(G)
2:      Input: G: Graph to calculate edge scores
3:      Input: bp: The probability a neighbor vertex is burnt, from 0.0 to 1.0
4:      Input: targetBurnRatio: In total targetBurnRatio * m edges will be burnt
5:      Output: Scores: An array of edge scores for each edge
6:
7:      burnt_count = 0                              ▷ keep track of total number of burnt edges
8:      scores = [0, ..., 0]                         ▷ scores is an array of length #edges
9:      burnt = [0, ..., 0]                          ▷ burnt is an array of length #edges
10:
11:     while burnt_count < targetBurnRatio * numberOfEdges(G) do
12:         visited = [false, ..., false]            ▷ visited is an array of length #vertices
13:         vertexQ = []                             ▷ a queue for vertex to be visited
14:         vertexQ.add(randomVertex(G))             ▷ pick a random starting vertex
15:
16:         while vertexQ is not empty do
17:             u = vertexQ.pop()
18:             visted[u.id] = True
19:             neighs = getAllUnvisitedVertices()
20:             while neighs is not empty do
21:                 r = randNum()                    ▷ r is a random float from 0.0 to 1.0
22:                 if r ≤ bp then
23:                     break                        ▷ decides to burn the vertex, not propagate further
24:                 v = pickRandom(neighs)           ▷ pick a random vertex to propagate fire
25:                 remove(neighs, v)                ▷ pick a random vertex to propagate fire
26:                 vertexQ.add(v)
27:                 eid = getEdgeID(u, v)
28:                 burnt_count++
29:     max_burnt = max(burnt)
30:     scores = burnt/max_burnt                     ▷ normalize burnt_count to be the scores
31:     return scores
```

A.0.5 *Similarity-based sparsifiers.*

Algorithm 5 G-Spar

```
procedure G-Spar(G)
    Input: G: Graph to calculate edge scores
    Output: H: G-Spar sparsified graph

    scores = {}                                   ▷ scores is a dictionary, key is edge id, value is edge score
    for all e ∈ E do
        eid = e.id
        score = JaccardScore(e)
        scores[eid] = score
    sort scores by value
    pick top s% edges to form H
    return H
```

Algorithm 6 L-Spar

```
procedure L-Spar(G)
    Input: G: Graph to calculate edge scores
    Input: c: Exponent parameter
    Output: H: L-Spar sparsified graph

    for all v ∈ V do
        d = degreeOf(v)
        E'=getEdgesOf(v)
        scores = {}                               ▷ scores is a dictionary, key is edge id, value is edge score
        for all e ∈ E' do
            eid = e.id
            score = JaccardScore(e)
            scores[eid] = score
        sort scores by value
        add top d^c edges to H
    return H
```

Algorithm 7 Edge Triangle Count

```
procedure EdgeTriangleCount(G)
    Input: G: Graph to calculate triangle edge scores
    Output: triangle_count: Triangle count for each edge

    triangle_count = [0, ..., 0]                  ▷ array of length #edges
    incident_triangle_count = [None, ..., None]   ▷ array of length #vertices

    for all u ∈ V do                              ▷ first vertex in triangle
        for all v ∈ getNeighborsOf(u) do
            incident_triangle_count[v] = 0        ▷ mark all neighboring vertices not None
        for all v ∈ getNeighborsOf(u) do          ▷ second vertex in triangle
            for all w ∈ getNeighborsOf(w) do      ▷ third vertex in triangle
                if incident_triangle_count[w] is not None then   ▷ triangle found
                    // count triangles to the vertices first, each triangle is counted 3 times
                    if u ≥ v then
                        incident_triangle_count[v]++
                    if u ≥ w then
                        incident_triangle_count[w]++

        // add local triangle count to global, reset local triangle count
        for all v ∈ getNeighborsOf(u) do
            eid = getEdgeId(u, v)
            if incident_triangle_count[v] > 0 then
                triangle_count[eid]+=incident_triangle_count[v]
            incident_triangle_count[v] = None
    return triangle_count
```

The Edge Triangle Count is not a standalone sparsification algorithm. We list it here because it is used in the calculation of the local similarity score and the SCAN structural similarity score.

Algorithm 8 Local similarity score

```
1:  procedure LocalSimilarityScore(G)
2:      Input: G: Graph to calculate triangle edge scores
3:      Output: scores: Local similarity scores for each edge
4:
5:      scores = [0, ..., 0]                                   ▷ array of length #edges
6:
7:      triangle_count = EdgeTriangleCount(G)
8:      for all u ∈ V do
9:          neighbors_sims = {}                                ▷ dictionary, key is edge id, value is similarity
10:         d_u = getDegreeOf(u)
11:         for all v ∈ getNeighborsOf(u) do
12:             d_v = getDegreeOf(v)
13:             eid = getEdgeId(u, v)
14:             sim = triangle_count[eid]/(d_u+d_v-triangle_count[eid])
15:             scores[eid] = max(scores[eid], sim)
16:     return scores
```

Algorithm 9 SCAN Structural Similarity Score

```
1:  procedure SCANStructuralSimilarityScore(G)
2:      Input: G: Graph to calculate triangle edge scores
3:      Output: scores: SCAN structural similarity scores for each edge
4:
5:      scores = [0, ..., 0]                                   ▷ array of length #edges
6:
7:      triangle_count = EdgeTriangleCount(G)
8:      for all u ∈ V do
9:          neighbors_sims = {}                                ▷ dictionary, key is edge id, value is similarity
10:         d_u = getDegreeOf(u)
11:         for all v ∈ getNeighborsOf(u) do
12:             d_v = getDegreeOf(v)
13:             eid = getEdgeId(u, v)
14:             sim = (triangle_count[eid]+1)/sqrt((d_u + 1) * (d_v + 1))
15:             scores[eid] = sim
16:     return scores
```

A.0.6 Effective Resistance (ER) Sparsifier.

We briefly summarize the derivation of the effective resistance. Interested readers should refer to [**?**] for more details.

We first define the following notations:

$\mathbb{R}$: real number.

G: Input Graph, in this write-up, G must be symmetrical (undirected).

$|\mathcal{V}|$: Number of Vertices in G.

$|\mathcal{E}|$: Number of Edges in G.

$\boldsymbol{A}$: $\in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, Adjacency Matrix of G.

$\boldsymbol{D}$: $\in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, Degree Matrix of G, where i^{th} diagonal entry is the degree of i^{th} vertex, if the graph is weighted, then it's the sum of all edge weights related to vertex i.

$\boldsymbol{L}$: $\in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, Laplacian Matrix of G, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$.

$\boldsymbol{B}$: Incidence Matrix, $\in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{V}|}$. Each row in $\boldsymbol{B}$ represents an edge, where the head vertex is -1, the tail vertex is 1, and all others are 0s. The head and tail of an undirected edge are randomly assigned.

$\boldsymbol{W}$: Weight Matrix, $\in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$, is a diagonal matrix, and each diagonal entry represents an edge weight. If the graph is unweighted, then W becomes an Identity Matrix $\boldsymbol{I}$.

χ_u: A unit vector of length $|\mathcal{V}|$, where only the u^{th} element is 1, others are 0s.

R_{uv}: The effective resistance of edge uv.

Now we start the derivation of the effective resistance:

$$\boldsymbol{L} = \boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B} \quad \text{(proof omitted)} \tag{1a}$$

According to Kirchhoff's law, the current flow in is always the same as the current flow out of the vertex,

$$\boldsymbol{B}^T \boldsymbol{i} = \boldsymbol{c_{ext}} \tag{1b}$$

According to Ohm's law,

$$\boldsymbol{i} = \boldsymbol{W}\boldsymbol{B}v \tag{1c}$$

Combing eq. (1a), (1b), and (1c),

$$\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}v = \boldsymbol{L}v = \boldsymbol{c_{ext}} \tag{1d}$$

Let $\boldsymbol{L}^+$ be the pseudo-inverse of $\boldsymbol{L}$, because Laplacian matrix is positive semi-definite, and doesn't have an inverse

$$v = \boldsymbol{L}^+\boldsymbol{c_{ext}} \tag{1e}$$

Now set $\boldsymbol{c_{ext}} = \boldsymbol{\chi_u} - \boldsymbol{\chi_v}$, then eq. (1e) can be written as

$$v = \boldsymbol{L}^+(\boldsymbol{\chi_u} - \boldsymbol{\chi_v}) \tag{1f}$$

Multiply both sides by $(\boldsymbol{\chi_u} - \boldsymbol{\chi_v})^T$,

$$(\boldsymbol{\chi_u} - \boldsymbol{\chi_v})^T v = (\boldsymbol{\chi_u} - \boldsymbol{\chi_v})^T\boldsymbol{L}^+(\boldsymbol{\chi_u} - \boldsymbol{\chi_v}) \tag{1g}$$

Notice that $(\boldsymbol{\chi_u} - \boldsymbol{\chi_v})$ is equivalent to the transpose of i^{th} row in $\boldsymbol{B}$, denoted by $\boldsymbol{B}[i]$, thus,

$$\boldsymbol{B}[i]^T v = \boldsymbol{B}[i]\boldsymbol{L}^+\boldsymbol{B}[i]^T \tag{1h}$$

Eq. (1h) applies to every i, thus can generalized to

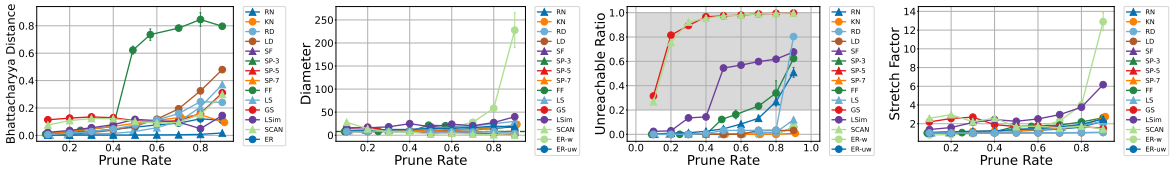
$$\boldsymbol{B}^T v = \boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^T \tag{1i}$$

The l.h.s. of eq. (1g) is the voltage difference between u and v, which can be used to represent the effective resistance of the edge connecting u and v. Thus, the effective resistance is defined as

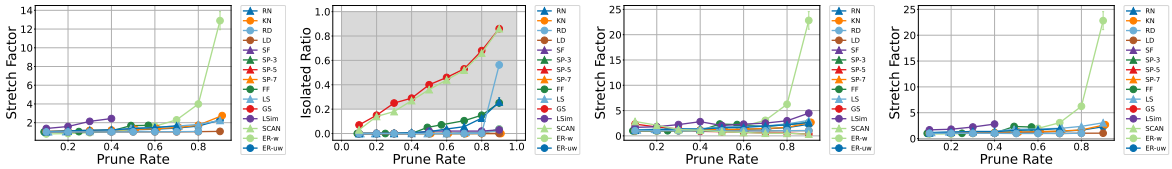
$$\begin{aligned} R_{uv} &= (\boldsymbol{\chi_u} - \boldsymbol{\chi_v})^T\boldsymbol{L}^+(\boldsymbol{\chi_u} - \boldsymbol{\chi_v}) \\ &= (\boldsymbol{\chi_u} - \boldsymbol{\chi_v})^T\boldsymbol{L}^+\boldsymbol{L}\boldsymbol{L}^+(\boldsymbol{\chi_u} - \boldsymbol{\chi_v}) \\ &= (\boldsymbol{\chi_u} - \boldsymbol{\chi_v})^T\boldsymbol{L}^+\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}\boldsymbol{L}^+(\boldsymbol{\chi_u} - \boldsymbol{\chi_v}) \\ &= ((\boldsymbol{\chi_u} - \boldsymbol{\chi_v})^T\boldsymbol{L}^+\boldsymbol{B}^T\boldsymbol{W}^{1/2})(\boldsymbol{W}^{1/2}\boldsymbol{B}\boldsymbol{L}^+(\boldsymbol{\chi_u} - \boldsymbol{\chi_v})) \\ &= ||\boldsymbol{W}^{1/2}\boldsymbol{B}\boldsymbol{L}^+(\boldsymbol{\chi_u} - \boldsymbol{\chi_v})||_2^2 \end{aligned} \tag{1j}$$

B APPENDIX B

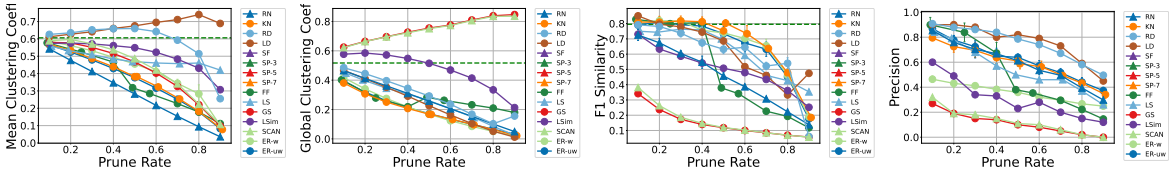
Due to the page limit, we only showed a subset of the results in the paper. This appendix presents the full results generated in this work. We show one dataset on each page, and the sub-captions note which metric each subgraph is measuring. Some figures are missing, and we explain why they are missing in such cases. There are three reasons why certain figures are missing, 1) Some metrics are supported for directed graphs, they are Clustering F1 Similarity, Number of Communities, and Modularity; 2) Some experiments couldn't finish within 24 hours, especially on time-consuming metrics like Eigenvector Centrality and large graphs; 3) Some experiments run out of memory and triggered OOM kill by the OS.



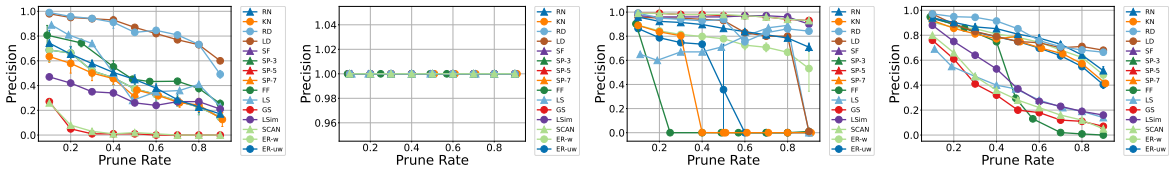
(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor



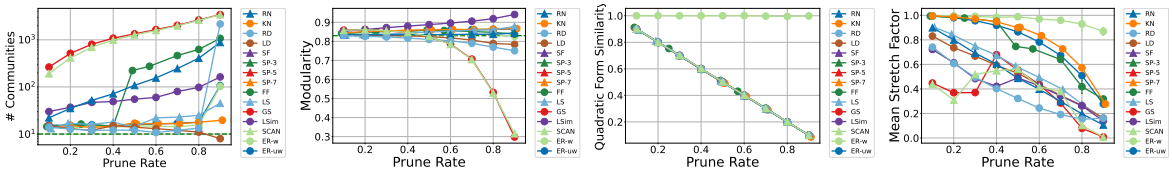
(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint



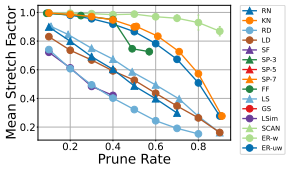
(i) Mean Clustering Coefficient (j) Global Clustering Coefficient (k) Clustering F1 Similarity (l) Betweenness Centrality Precision



(m) Closeness Centrality Precision (n) Katz Centrality Precision (o) Eigenvector Centrality Precision (p) PageRank Precision

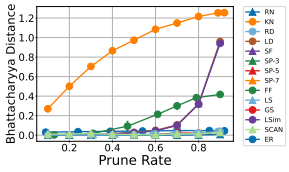


(q) Number of Communities (r) Modularity (s) Quadratic Form Similarity (t) Max Flow Stretch Factor

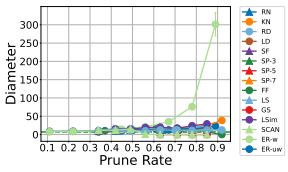


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

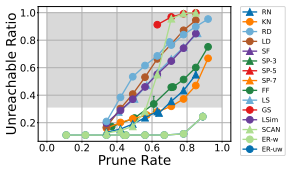
Figure 1: Metric Evaluation on ego-Facebook



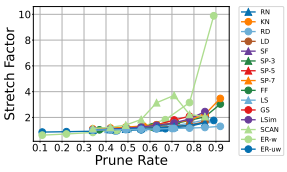
(a) Degree Distribution



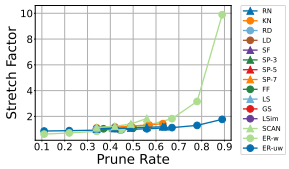
(b) Diameter



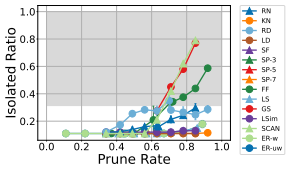
(c) Average SPSP Unreachable Ratio



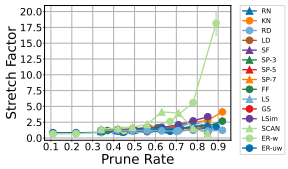
(d) SPSP Stretch Factor



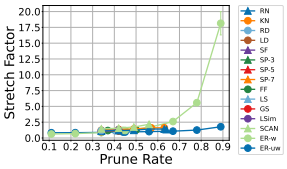
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



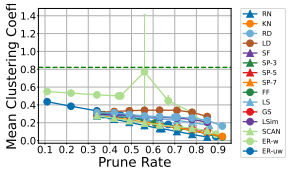
(f) Vertex Isolated Ratio



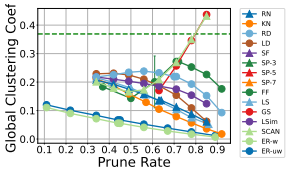
(g) Eccentricity Stretch Factor



(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



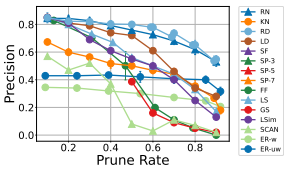
(i) Mean Clustering Coefficient



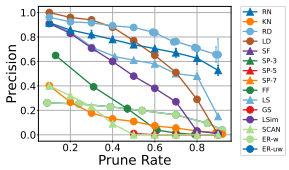
(j) Global Clustering Coefficient

Unsupported for directed graph

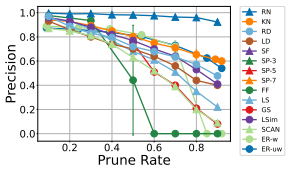
(k) Clustering F1 Similarity



(l) Betweenness Centrality Precision



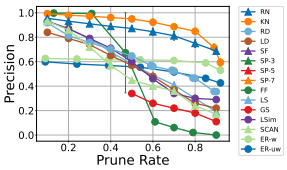
(m) Closeness Centrality Precision



(n) Katz Centrality Precision

Timed out after 24 hours

(o) Eigenvector Centrality Precision



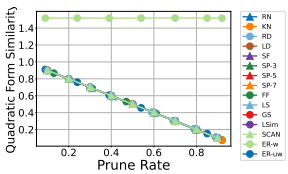
(p) PageRank Precision

Unsupported for directed graph

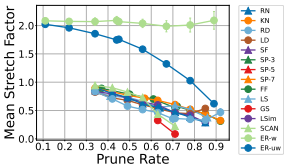
(q) Number of Communities

Unsupported for directed graph

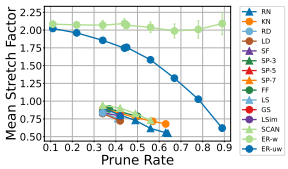
(r) Modularity



(s) Quadratic Form Similarity

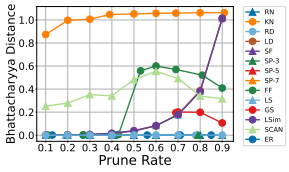


(t) Max Flow Stretch Factor

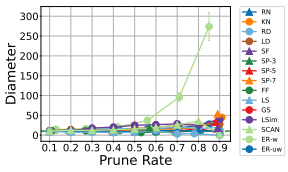


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

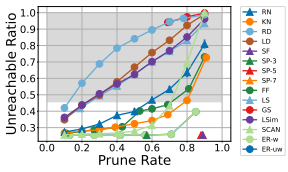
Figure 2: Metric Evaluation on ego-Twitter



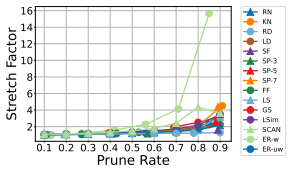
(a) Degree Distribution



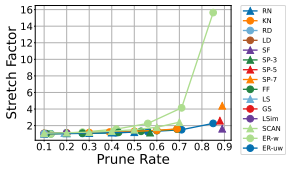
(b) Diameter



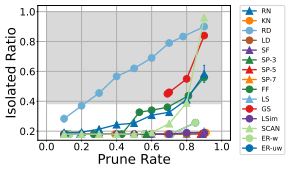
(c) Average SPSP Unreachable Ratio



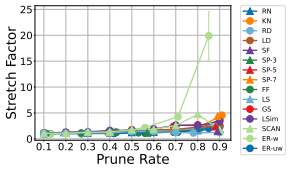
(d) SPSP Stretch Factor



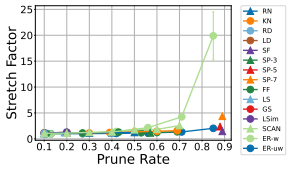
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



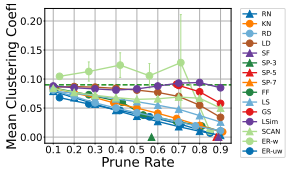
(f) Vertex Isolated Ratio



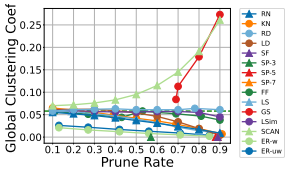
(g) Eccentricity Stretch Factor



(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



(i) Mean Clustering Coefficient



(j) Global Clustering Coefficient

Unsupported for directed graph

(k) Clustering F1 Similarity

Timed out after 24 hours

(l) Betweenness Centrality Precision

Timed out after 24 hours

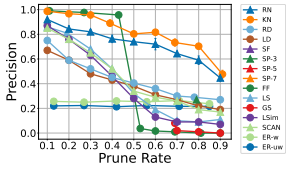
(m) Closeness Centrality Precision

Timed out after 24 hours

(n) Katz Centrality Precision

Timed out after 24 hours

(o) Eigenvector Centrality Precision



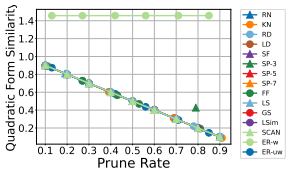
(p) PageRank Precision

Unsupported for directed graph

(q) Number of Communities

Unsupported for directed graph

(r) Modularity



(s) Quadratic Form Similarity

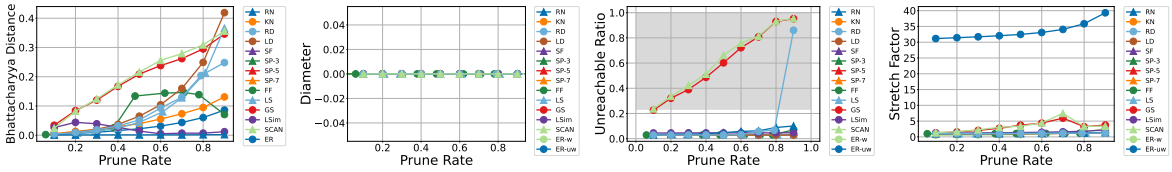
Timed out after 24 hours

(t) Max Flow Stretch Factor

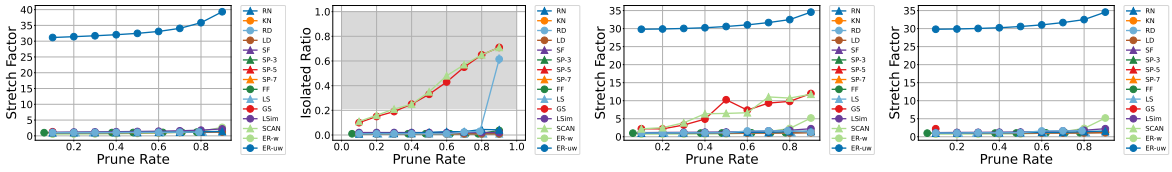
Timed out after 24 hours

(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

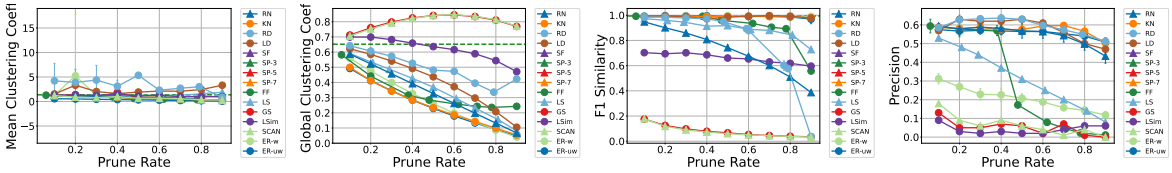
Figure 3: Metric Evaluation on soc-Pokec



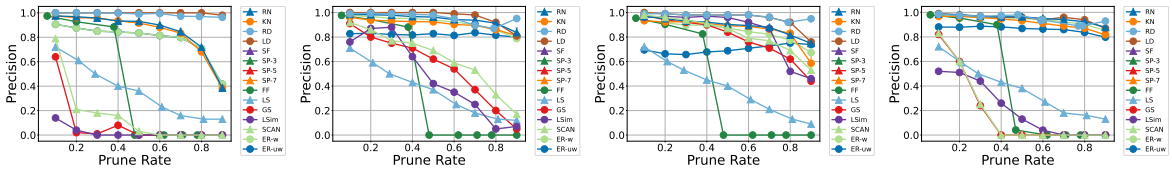
(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor



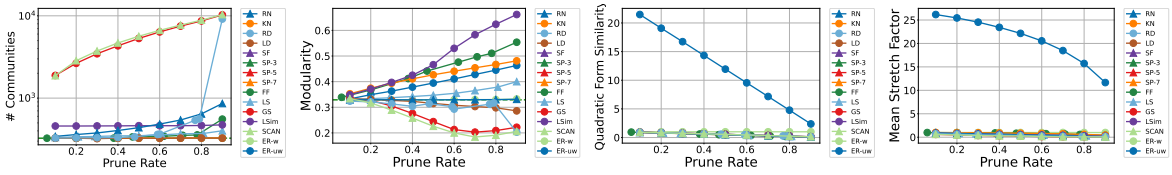
(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint



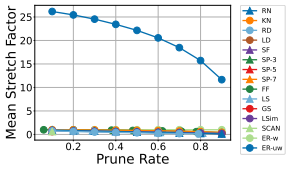
(i) Mean Clustering Coefficient (j) Global Clustering Coefficient (k) Clustering F1 Similarity (l) Betweenness Centrality Precision



(m) Closeness Centrality Precision (n) Katz Centrality Precision (o) Eigenvector Centrality Precision (p) PageRank Precision

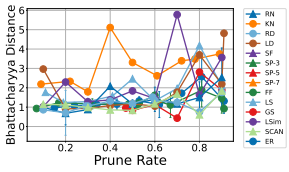


(q) Number of Communities (r) Modularity (s) Quadratic Form Similarity (t) Max Flow Stretch Factor

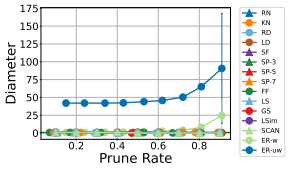


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

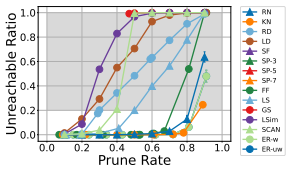
Figure 4: Metric Evaluation on human_gene2



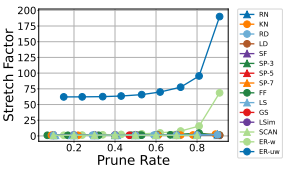
(a) Degree Distribution



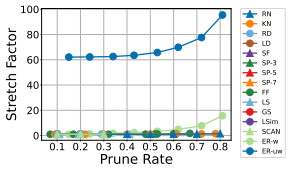
(b) Diameter



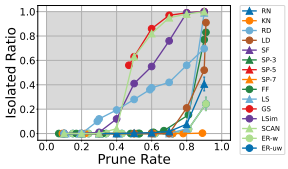
(c) Average SPSP Unreachable Ratio



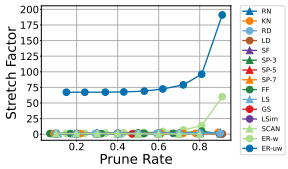
(d) SPSP Stretch Factor



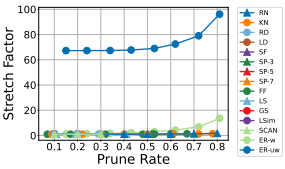
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



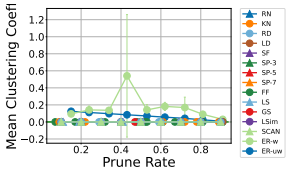
(f) Vertex Isolated Ratio



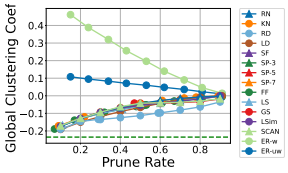
(g) Eccentricity Stretch Factor



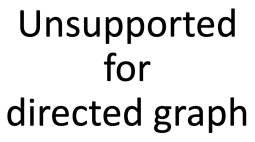
(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



(i) Mean Clustering Coefficient



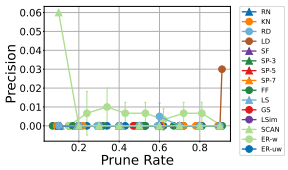
(j) Global Clustering Coefficient



(k) Clustering F1 Similarity

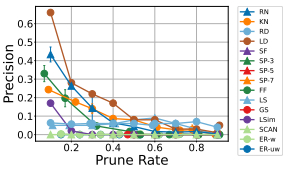
(l) Betweenness Centrality Precision

(m) Closeness Centrality Precision



(n) Katz Centrality Precision

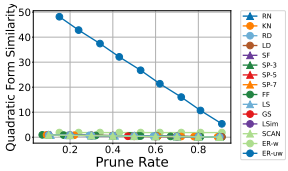
(o) Eigenvector Centrality Precision



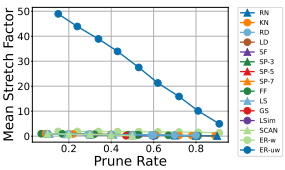
(p) PageRank Precision

(q) Number of Communities

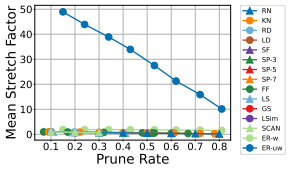
(r) Modularity



(s) Quadratic Form Similarity

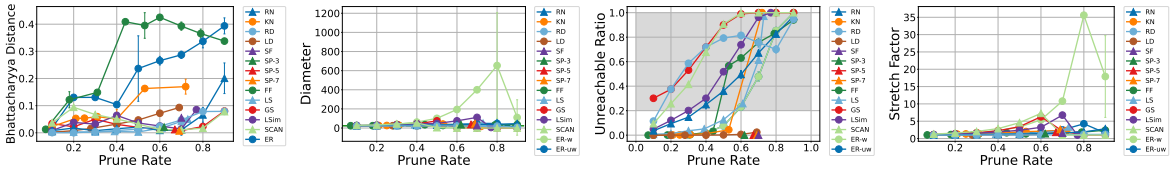


(t) Max Flow Stretch Factor

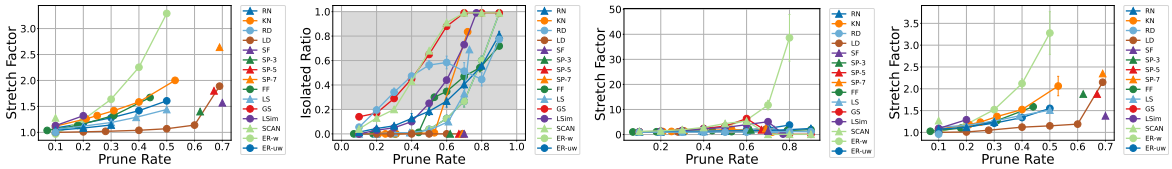


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

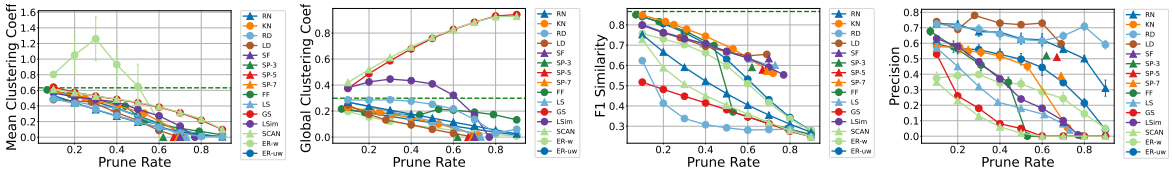
Figure 5: Metric Evaluation on cage14



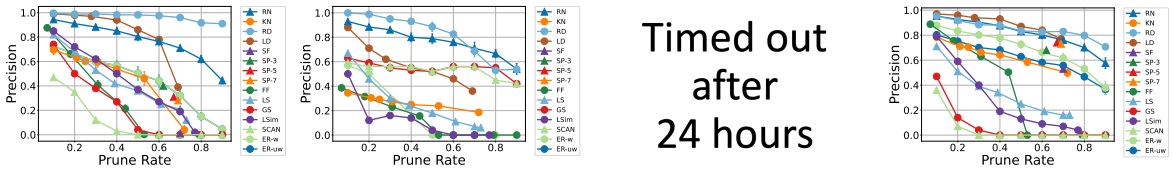
(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor



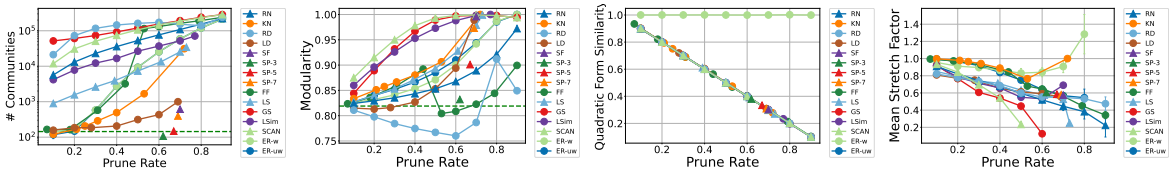
(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint



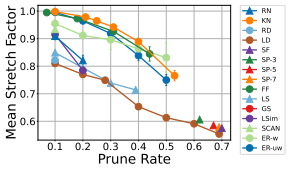
(i) Mean Clustering Coefficient (j) Global Clustering Coefficient (k) Clustering F1 Similarity (l) Betweenness Centrality Precision



(m) Closeness Centrality Precision (n) Katz Centrality Precision (o) Eigenvector Centrality Precision (p) PageRank Precision

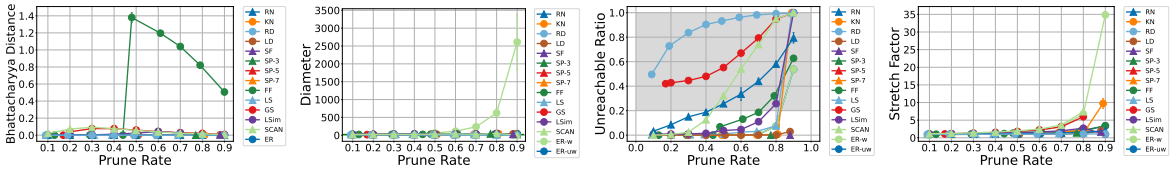


(q) Number of Communities (r) Modularity (s) Quadratic Form Similarity (t) Max Flow Stretch Factor

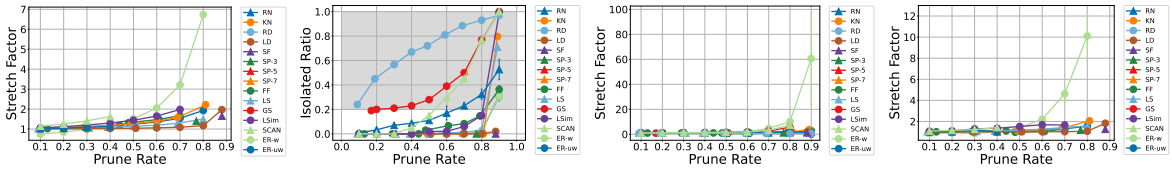


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

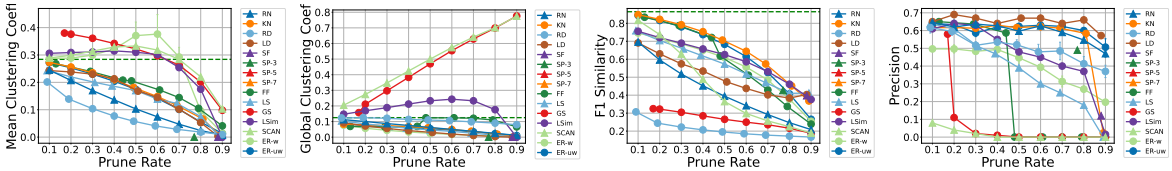
Figure 6: Metric Evaluation on com-DBLP



(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor

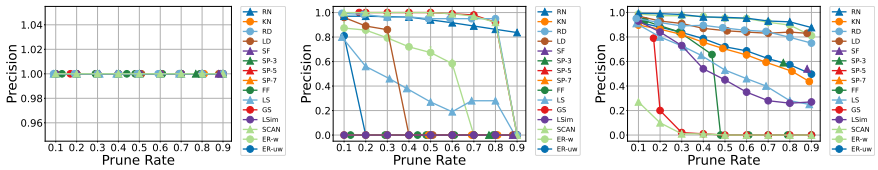


(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint

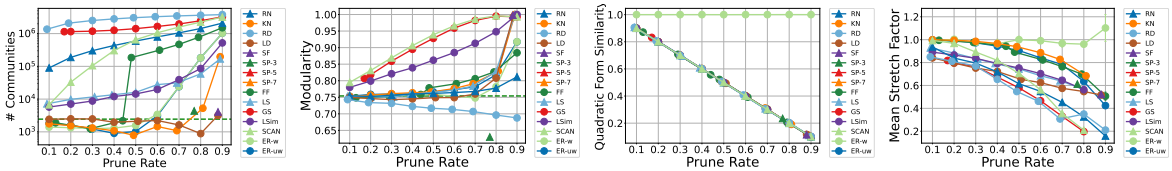


(i) Mean Clustering Coefficient (j) Global Clustering Coefficient (k) Clustering F1 Similarity (l) Betweenness Centrality Precision

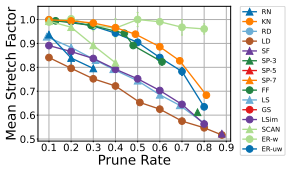
Timed out after 24 hours



(m) Closeness Centrality Precision (n) Katz Centrality Precision (o) Eigenvector Centrality Precision (p) PageRank Precision

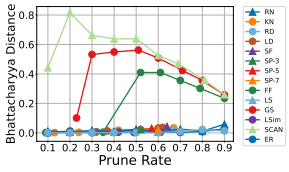


(q) Number of Communities (r) Modularity (s) Quadratic Form Similarity (t) Max Flow Stretch Factor

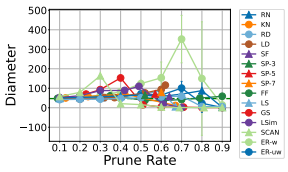


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

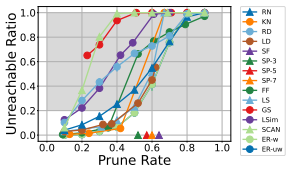
Figure 7: Metric Evaluation on com-LiveJournal



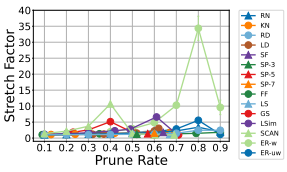
(a) Degree Distribution



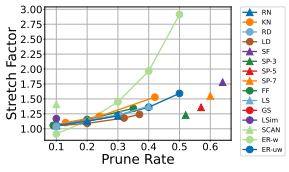
(b) Diameter



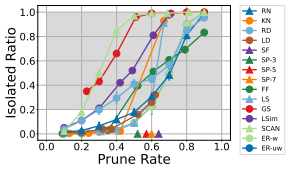
(c) Average SPSP Unreachable Ratio



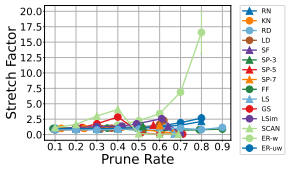
(d) SPSP Stretch Factor



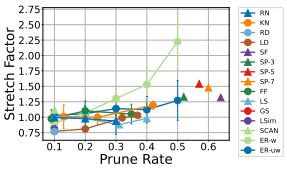
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



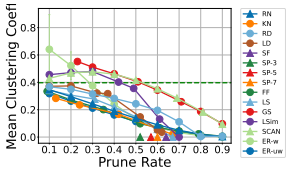
(f) Vertex Isolated Ratio



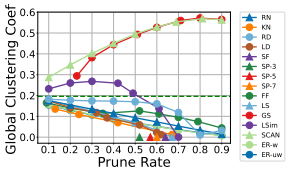
(g) Eccentricity Stretch Factor



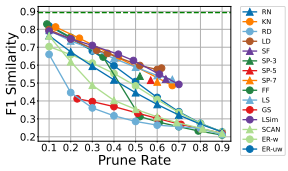
(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



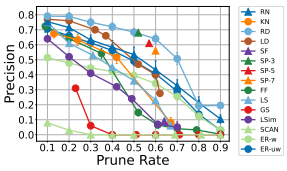
(i) Mean Clustering Coefficient



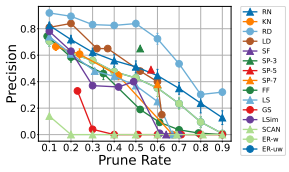
(j) Global Clustering Coefficient



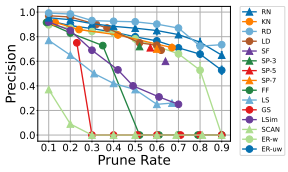
(k) Clustering F1 Similarity



(l) Betweenness Centrality Precision

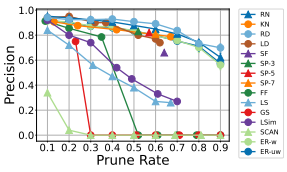


(m) Closeness Centrality Precision

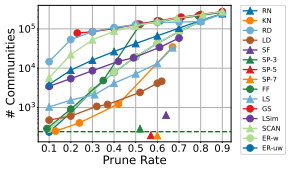


(n) Katz Centrality Precision

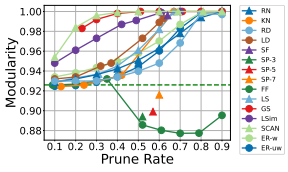
(o) Eigenvector Centrality Precision



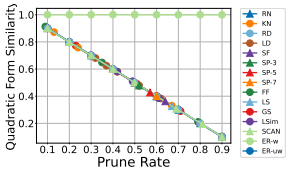
(p) PageRank Precision



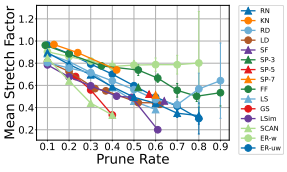
(q) Number of Communities



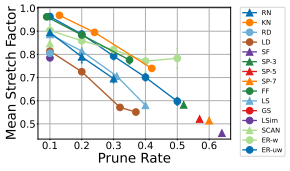
(r) Modularity



(s) Quadratic Form Similarity

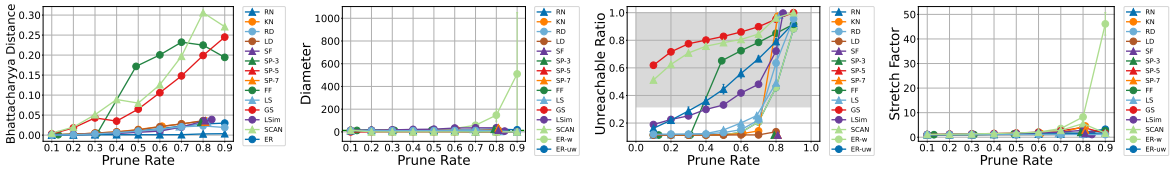


(t) Max Flow Stretch Factor

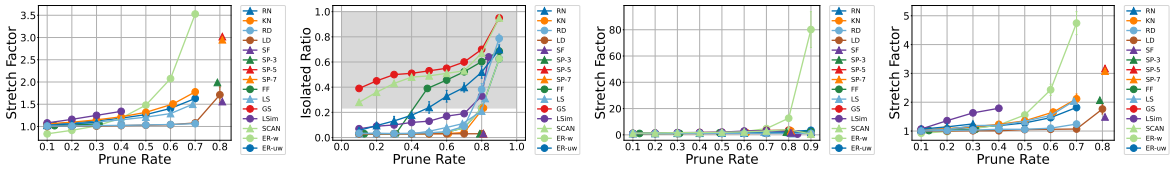


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

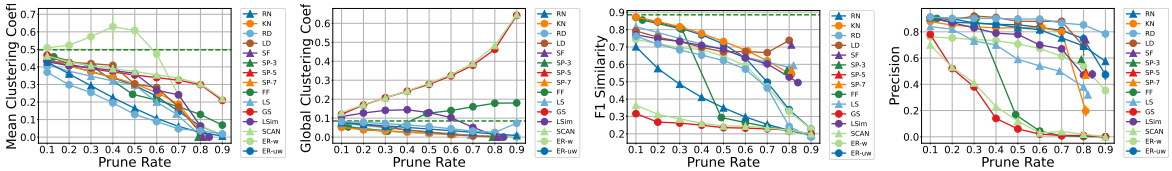
Figure 8: Metric Evaluation on com-Amazon



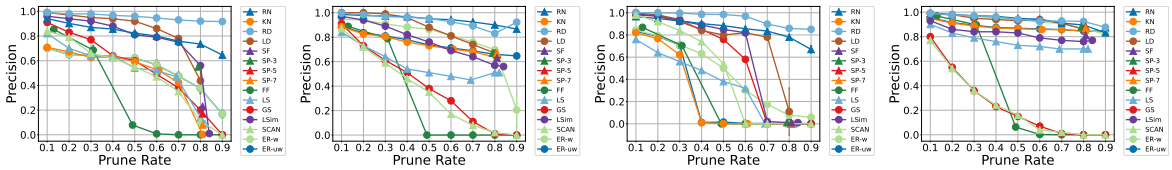
(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor



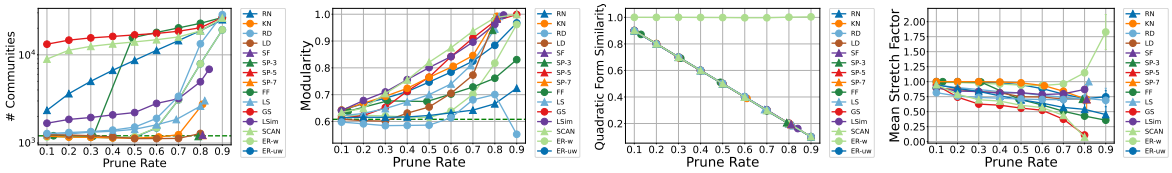
(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint



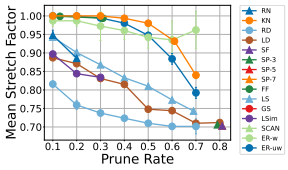
(i) Mean Clustering Coefficient (j) Global Clustering Coefficient (k) Clustering F1 Similarity (l) Betweenness Centrality Precision



(m) Closeness Centrality Precision (n) Katz Centrality Precision (o) Eigenvector Centrality Precision (p) PageRank Precision

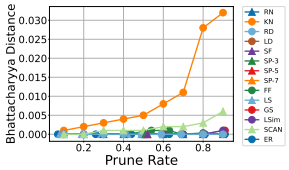


(q) Number of Communities (r) Modularity (s) Quadratic Form Similarity (t) Max Flow Stretch Factor

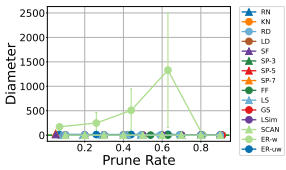


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

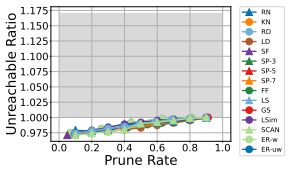
Figure 9: Metric Evaluation on email-Enron



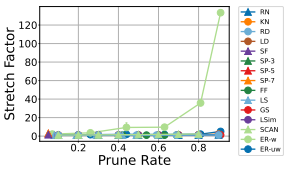
(a) Degree Distribution



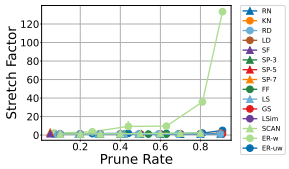
(b) Diameter



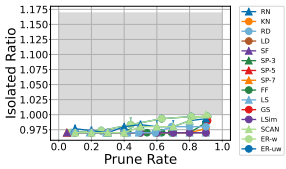
(c) Average SPSP Unreachable Ratio



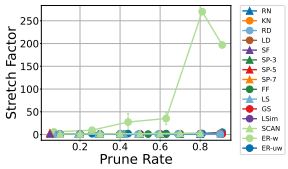
(d) SPSP Stretch Factor



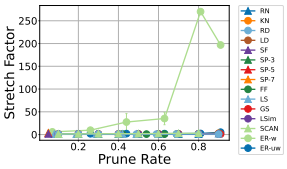
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



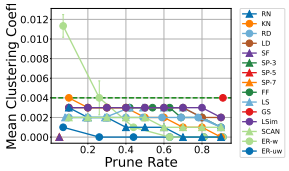
(f) Vertex Isolated Ratio



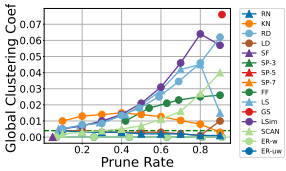
(g) Eccentricity Stretch Factor



(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



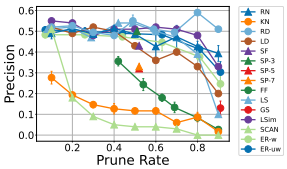
(i) Mean Clustering Coefficient



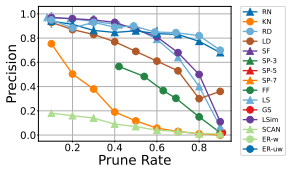
(j) Global Clustering Coefficient

Unsupported for directed graph

(k) Clustering F1 Similarity



(l) Betweenness Centrality Precision



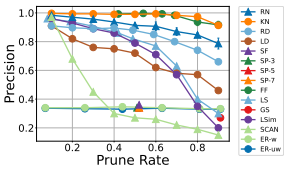
(m) Closeness Centrality Precision

Timed out after 24 hours

(n) Katz Centrality Precision

Timed out after 24 hours

(o) Eigenvector Centrality Precision



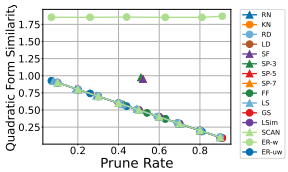
(p) PageRank Precision

Unsupported for directed graph

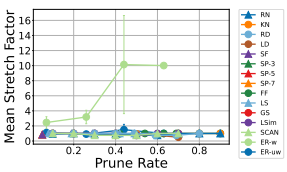
(q) Number of Communities

Unsupported for directed graph

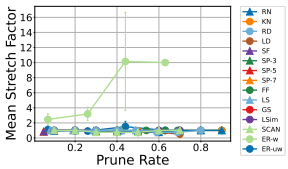
(r) Modularity



(s) Quadratic Form Similarity

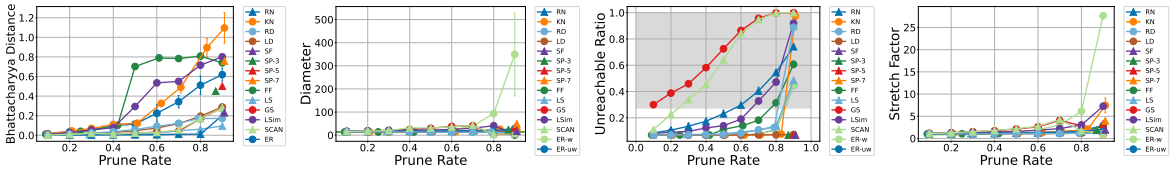


(t) Max Flow Stretch Factor

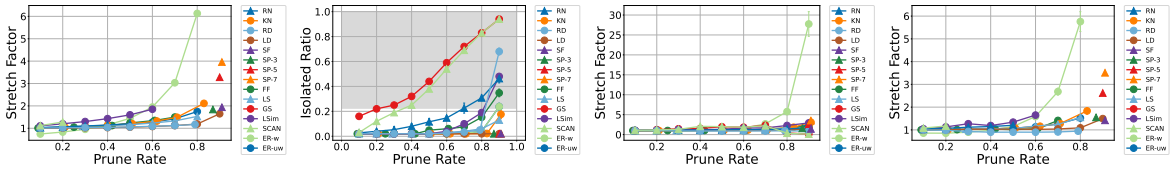


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

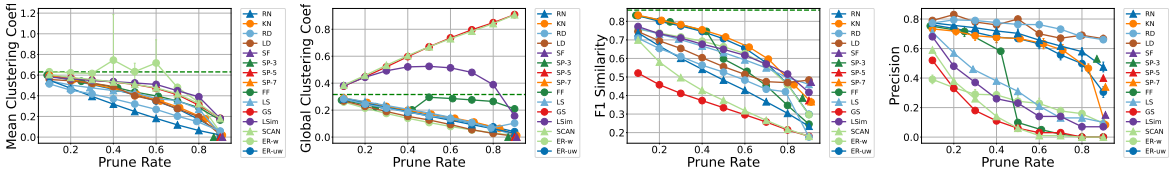
Figure 10: Metric Evaluation on wiki-Talk



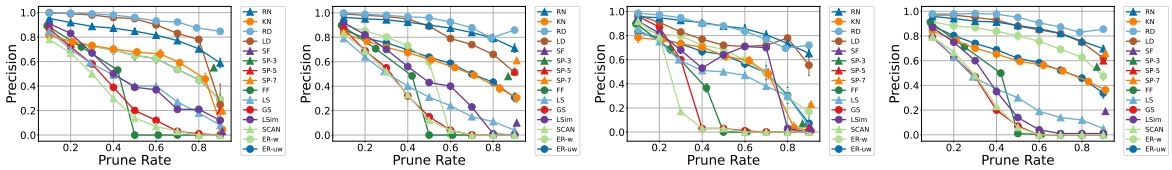
(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor



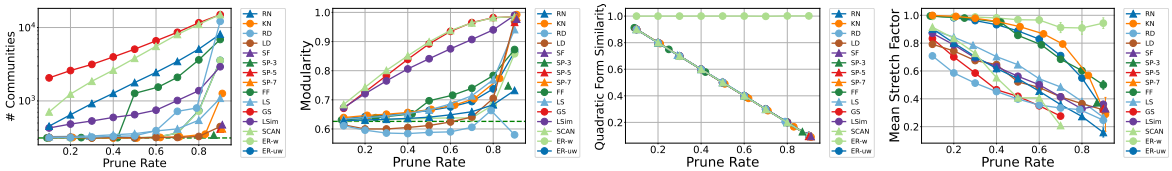
(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint



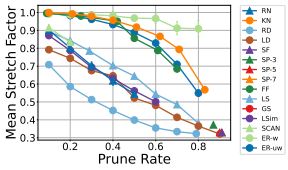
(i) Mean Clustering Coefficient (j) Global Clustering Coefficient (k) Clustering F1 Similarity (l) Betweenness Centrality Precision



(m) Closeness Centrality Precision (n) Katz Centrality Precision (o) Eigenvector Centrality Precision (p) PageRank Precision

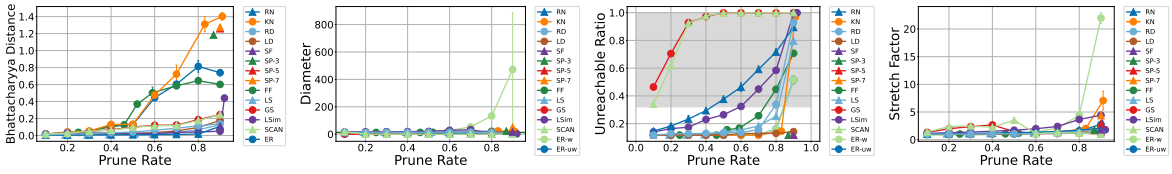


(q) Number of Communities (r) Modularity (s) Quadratic Form Similarity (t) Max Flow Stretch Factor

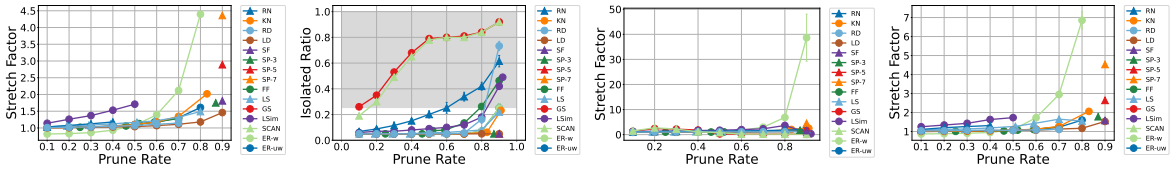


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

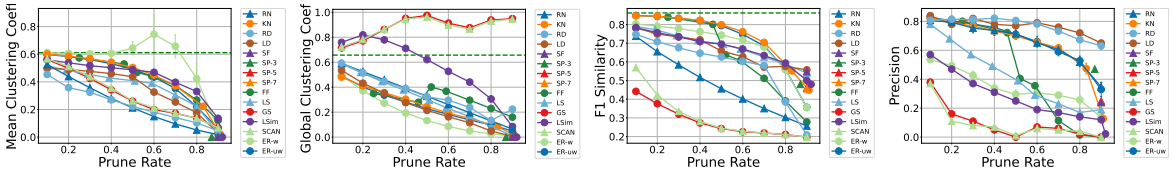
Figure 11: Metric Evaluation on ca-AstroPh



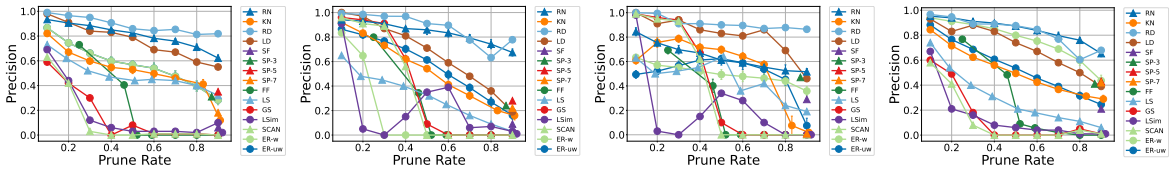
(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor



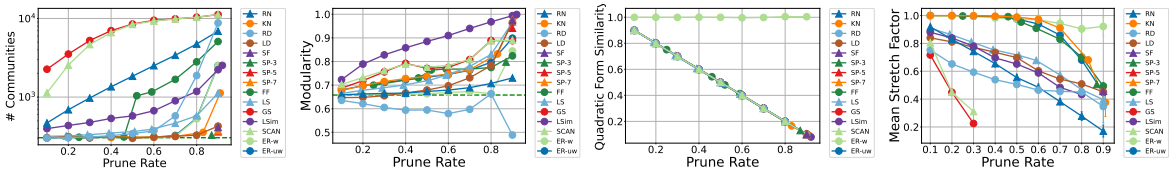
(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint



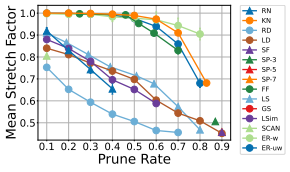
(i) Mean Clustering Coefficient (j) Global Clustering Coefficient (k) Clustering F1 Similarity (l) Betweenness Centrality Precision



(m) Closeness Centrality Precision (n) Katz Centrality Precision (o) Eigenvector Centrality Precision (p) PageRank Precision

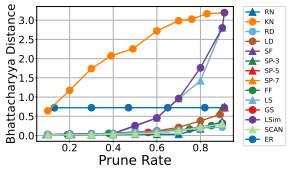


(q) Number of Communities (r) Modularity (s) Quadratic Form Similarity (t) Max Flow Stretch Factor

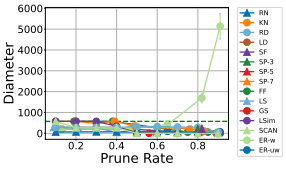


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

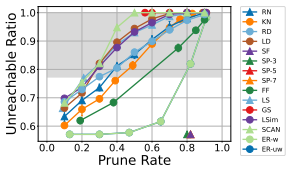
Figure 12: Metric Evaluation on ca-HepPh



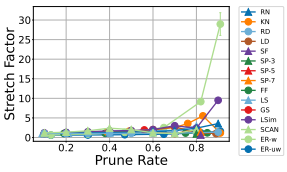
(a) Degree Distribution



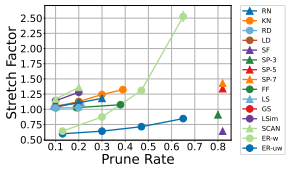
(b) Diameter



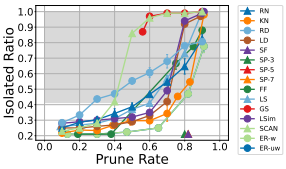
(c) Average SPSP Unreachable Ratio



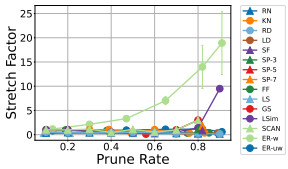
(d) SPSP Stretch Factor



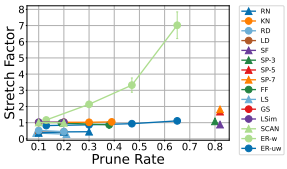
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



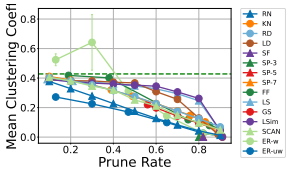
(f) Vertex Isolated Ratio



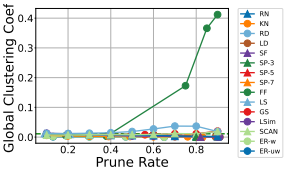
(g) Eccentricity Stretch Factor



(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



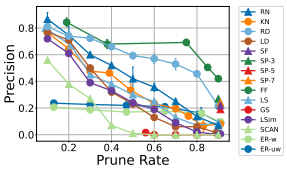
(i) Mean Clustering Coefficient



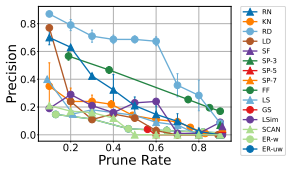
(j) Global Clustering Coefficient

Unsupported for directed graph

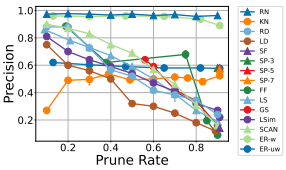
(k) Clustering F1 Similarity



(l) Betweenness Centrality Precision



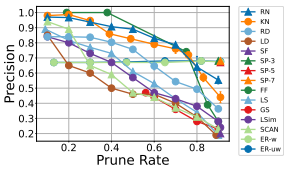
(m) Closeness Centrality Precision



(n) Katz Centrality Precision

Timed out after 24 hours

(o) Eigenvector Centrality Precision



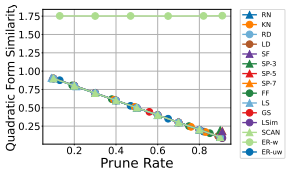
(p) PageRank Precision

Unsupported for directed graph

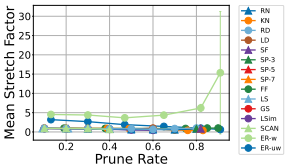
(q) Number of Communities

Unsupported for directed graph

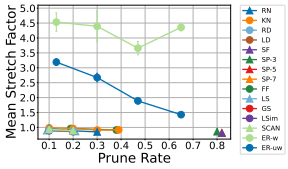
(r) Modularity



(s) Quadratic Form Similarity

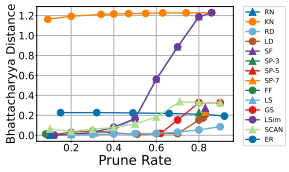


(t) Max Flow Stretch Factor

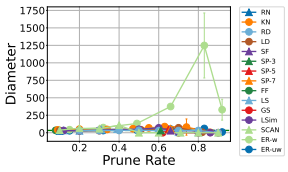


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

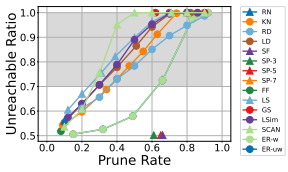
Figure 13: Metric Evaluation on web-BerkStan



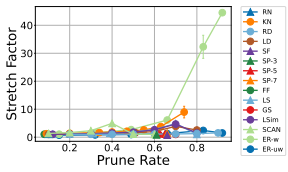
(a) Degree Distribution



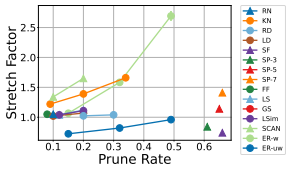
(b) Diameter



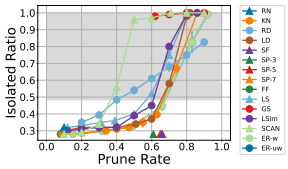
(c) Average SPSP Unreachable Ratio



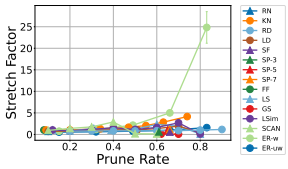
(d) SPSP Stretch Factor



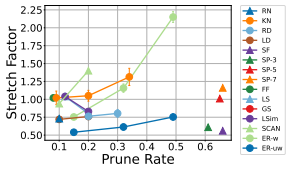
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



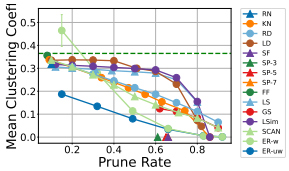
(f) Vertex Isolated Ratio



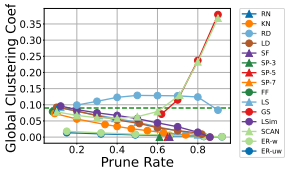
(g) Eccentricity Stretch Factor



(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



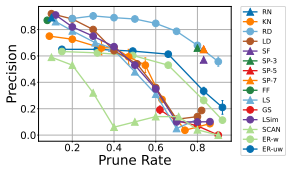
(i) Mean Clustering Coefficient



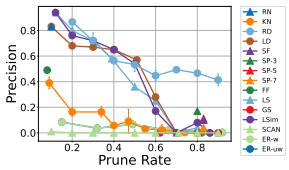
(j) Global Clustering Coefficient

Unsupported for directed graph

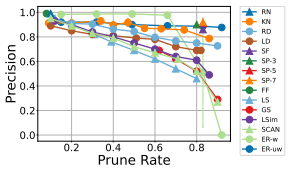
(k) Clustering F1 Similarity



(l) Betweenness Centrality Precision



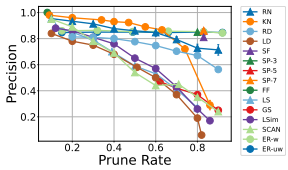
(m) Closeness Centrality Precision



(n) Katz Centrality Precision

Timed out after 24 hours

(o) Eigenvector Centrality Precision



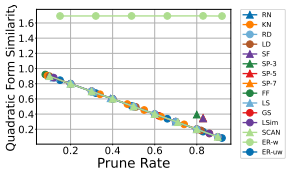
(p) PageRank Precision

Unsupported for directed graph

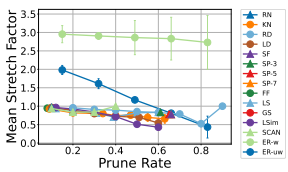
(q) Number of Communities

Unsupported for directed graph

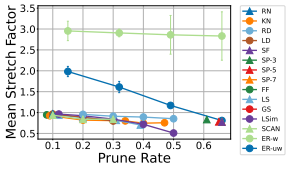
(r) Modularity



(s) Quadratic Form Similarity

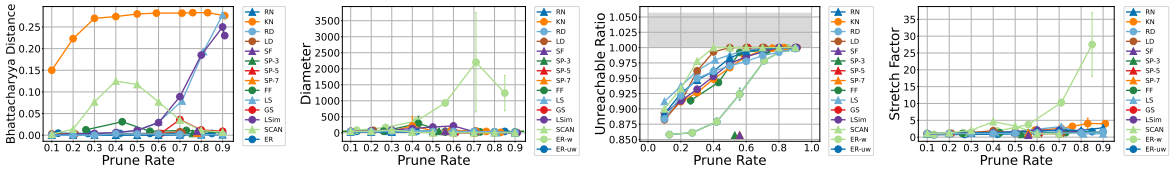


(t) Max Flow Stretch Factor

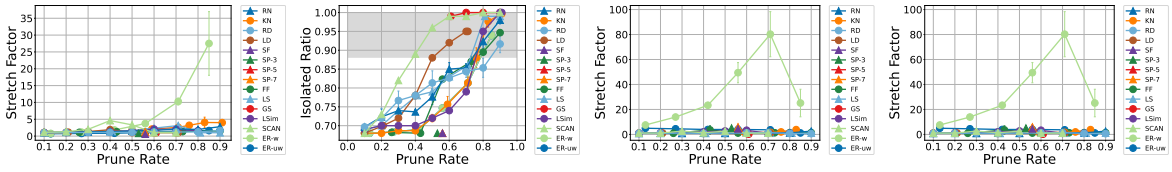


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

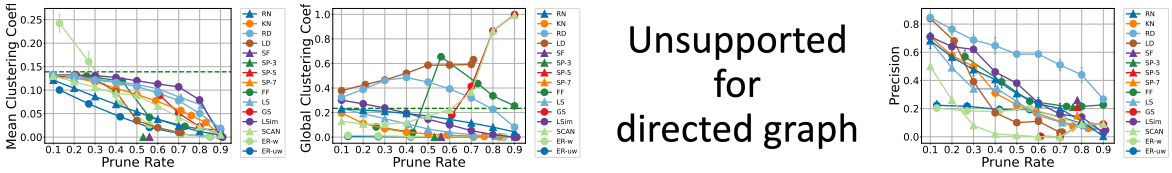
Figure 14: Metric Evaluation on web-Google



(a) Degree Distribution (b) Diameter (c) Average SPSP Unreachable Ratio (d) SPSP Stretch Factor



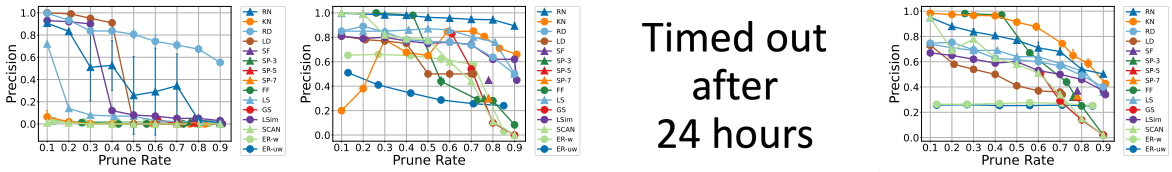
(e) SPSP Stretch Factor with Unreachable Ratio Constraint (f) Vertex Isolated Ratio (g) Eccentricity Stretch Factor (h) Eccentricity Stretch Factor with Isolated Ratio Constraint



(i) Mean Clustering Coefficient (j) Global Clustering Coefficient

Unsupported for directed graph

(k) Clustering F1 Similarity (l) Betweenness Centrality Precision



(m) Closeness Centrality Precision (n) Katz Centrality Precision

Timed out after 24 hours

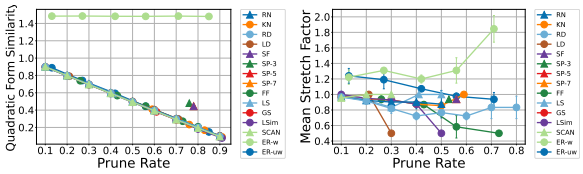
(o) Eigenvector Centrality Precision (p) PageRank Precision

Unsupported for directed graph

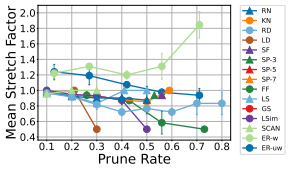
(q) Number of Communities

Unsupported for directed graph

(r) Modularity

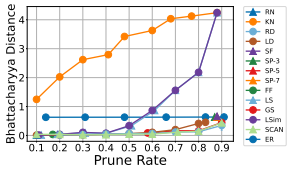


(s) Quadratic Form Similarity (t) Max Flow Stretch Factor

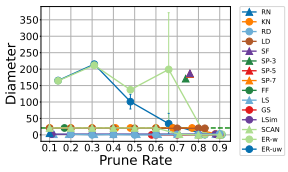


(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

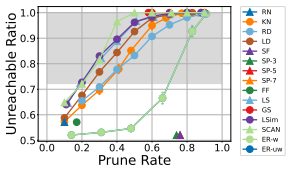
Figure 15: Metric Evaluation on web-NotreDame



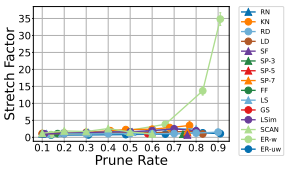
(a) Degree Distribution



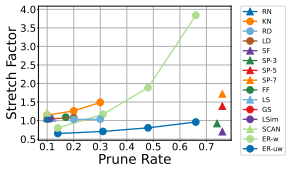
(b) Diameter



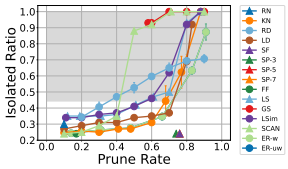
(c) Average SPSP Unreachable Ratio



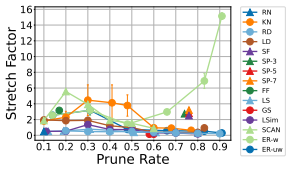
(d) SPSP Stretch Factor



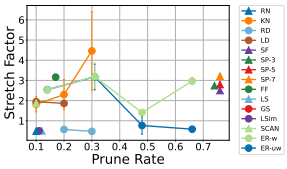
(e) SPSP Stretch Factor with Unreachable Ratio Constraint



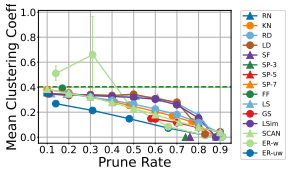
(f) Vertex Isolated Ratio



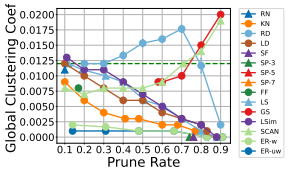
(g) Eccentricity Stretch Factor



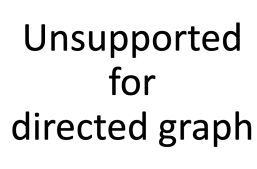
(h) Eccentricity Stretch Factor with Isolated Ratio Constraint



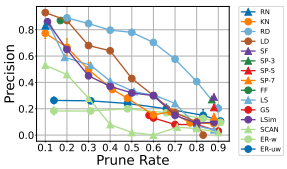
(i) Mean Clustering Coefficient



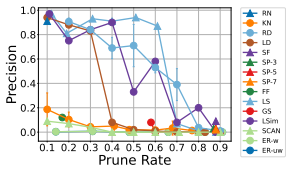
(j) Global Clustering Coefficient



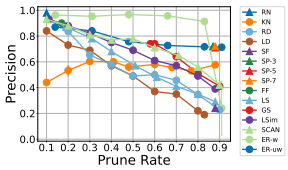
(k) Clustering F1 Similarity



(l) Betweenness Centrality Precision

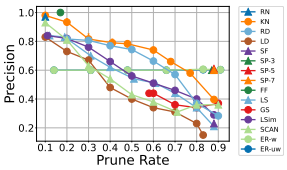


(m) Closeness Centrality Precision

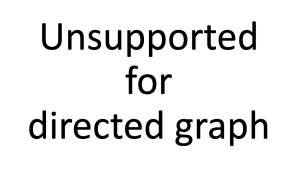


(n) Katz Centrality Precision

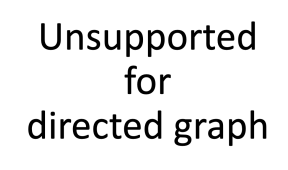
(o) Eigenvector Centrality Precision



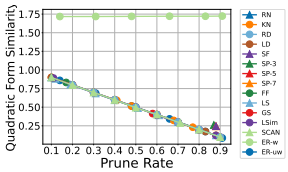
(p) PageRank Precision



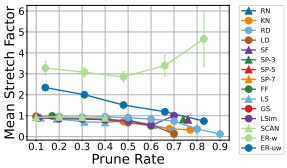
(q) Number of Communities



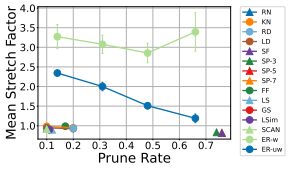
(r) Modularity



(s) Quadratic Form Similarity

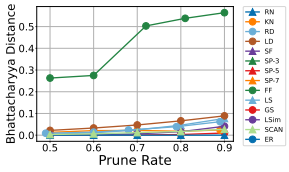


(t) Max Flow Stretch Factor



(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

Figure 16: Metric Evaluation on web-Stanford



(a) Degree Distribution

Out of Memory

(b) Diameter

Timed out after 24 hours

(c) Average SPSP Unreachable Ratio

Timed out after 24 hours

(d) SPSP Stretch Factor

Timed out after 24 hours

(e) SPSP Stretch Factor with Unreachable Ratio Constraint

Out of Memory

(f) Vertex Isolated Ratio

Out of Memory

(g) Eccentricity Stretch Factor

Out of Memory

(h) Eccentricity Stretch Factor with Isolated Ratio Constraint

Out of Memory

(i) Mean Clustering Coefficient

Out of Memory

(j) Global Clustering Coefficient

Timed out after 24 hours

(k) Clustering F1 Similarity

Timed out after 24 hours

(l) Betweenness Centrality Precision

Timed out after 24 hours

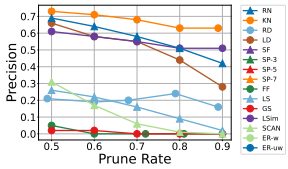
(m) Closeness Centrality Precision

Timed out after 24 hours

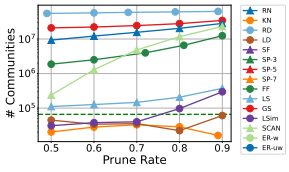
(n) Katz Centrality Precision

Timed out after 24 hours

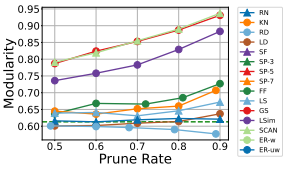
(o) Eigenvector Centrality Precision



(p) PageRank Precision



(q) Number of Communities



(r) Modularity

Timed out after 24 hours

(s) Quadratic Form Similarity

Out of Memory

(t) Max Flow Stretch Factor

Out of Memory

(u) Max Flow Stretch Factor with Unreachable Ratio Constraint

Figure 17: Metric Evaluation on com-friendster

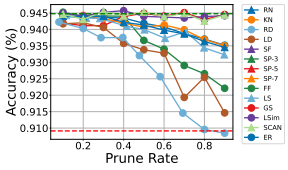


Figure 18: Clustering GCN Accuracy on Reddit

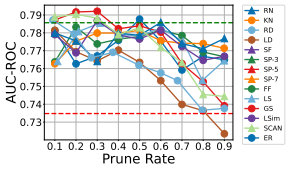


Figure 19: GraphSAGE Accuracy on ogbn-proteins