I. GRN-WSN MAPPING

The graph attributes discussed in Section II make GRNs an effective choice for design of robust communication networks. In this section, we exploit the aforementioned properties to propose a scalable approach to design WSN graphs.

A. Input and output parameters

Inputs to mapping algorithm are (a) WSN graph G_w where the set of nodes V_w represents sensors and each edge $e(u,v) \in E_w$ exists if two nodes u and v are within transmission range of one another and (b) unweighted directed GRN graph denoted by $G_g(V_g, E_g)$, where V_g and E_g are set of genes and protein interactions between gene pairs, respectively.

We now define a **mapping function** $M : G_{mw} \rightarrow G_g$, s.t. $G_{mw}(V_{mw}, E_{mw})$ is a directed graph, where $V_{mw} \subset V_w$ and $E_{mw} \subset E_w$. Each edge $e(u,v) \in E_{mw}$ exists if and only if there exists a path between $M(u)$ and $M(v)$ in G_g. (We also discuss a variant of this mapping rule in Section I-B2).

Edge weight of undirected WSN graph G_w: Given any pair of nodes u and v in G_w, we define the normalized area subject to interference as follows:

$$A(u,v) = 2r^2 cos^{-1}\frac{d}{2r} - \frac{d}{2}\sqrt{4r^2 - d^2} \tag{1}$$

The weight for edge $e(u,v)$, representing the degree of interference, is calculated as,

$$\omega(u,v) = \begin{cases} 1.0 - \frac{A(u,v)}{2\pi r^2}, & \text{if } d < 2r \\ 1, & \text{otherwise} \end{cases}$$

Here $\omega(u,v) = 1$ indicates maximum separation and therefore *minimum interference* between nodes u and v.

B. Mapping

Let us discuss different facets of the mapping approach.

1) Preprocessing: Initially, V_w and V_g are ranked in non-increasing order of their pageranks. The calculation of pagerank is modeled as an eigenvector problem $T_G.r_G = \lambda.r_G$, where r_G is a rank matrix of graph G and T_G is the transition matrix corresponding to eigenvalue $\lambda = 1$.

A directed graph $G(V,E)$ (such as input GRN G_g) may possess a group of nodes, called *spider trap*, which form a closed loop, causing the random surfer to stay trapped in a closed cycle. We address the problem of trapped nodes by (using Python Networkx library [**?**] and) modifying the transition matrix as $T_G = dT_o r + (1-d) \times I/|V|$, where d is the damping factor, T_o is the original transition matrix, r is the initial rank vector and $I/|V|$ is a matrix of same dimension as T_o where each element has value $\frac{1}{|V|}$.

2) Working: In Algorithm 1, the mapped genes, mapped-WSN nodes and mapped-WSN edges are stored in lists m_g, m_w and e_w, respectively. We map the highest ranking node $n_w = s_w[0]$ to highest ranking gene n_g that meets the condition $G_w.deg(n_w) \leq G_g.deg(n_g)$. Node n_w and gene n_g are added to m_w and m_g, respectively (Lines 4 - 7). In Lines 8 - 21, we iteratively apply *path-to-edge mapping*, where the highest ranking unmapped node n_w is mapped to the highest ranking unmapped gene n_g, if for at least $(100 \times f)\%$ of edges between n_w and already mapped nodes, there exist paths between n_g and corresponding mapped genes. Edges between newly mapped and already mapped nodes are included in e_w. Nodes in m_w and edges in e_w from mapped-WSN G_{mw}.

Algorithm 1: Mapping Algorithm

```
   Data: G_g,G_w,f
   Result: G_mw
1  m_g = {},m_w = {},e_w = {},p_g = {},p_w = {} ;
2  G_mw = (∅,∅) ;
3  Nodes V_w and genes V_g are ranked in non-increasing
     order of page rank ;
4  n_w = s_w[0] ;
5  for each n_g in s_g do
6      If G_w.deg(n_w) ≤ G_g.deg(n_g)
7          m_g = m_g ∪ n_g,    m_w = m_w ∪ n_w ;
8  for each unmapped node n_w ∈ V_w do
9      for each unmapped gene n_g ∈ V_g do
10         If n_w ∈ m_w or n_g ∈ m_g
11             continue ;
12         den = 0.0, num = 0.0 ;
13         for i = 1 to len(m_w) do
14             If e(n_w, m_w[i]) ∈ E(G_w)
15                 den = den + 1 ;
16                 If G_g.has_path(n_g, m_g[i]) or
                     G_g.has_path(m_g[i], n_g)
17                     num = num + 1 ;
18         if num/den ≥ f then
19             m_g = m_g ∪ n_g,    m_w = m_w ∪ n_w;
20             e_w = e_w ∪ e(n_w, m_w[i]);
21 N(G_mw) = m_w,E(G_mw) = e_w ;
```

3) Fidelity Constant: Fidelity Constant f (ranging from 0 to 1) regulates the *quality vs. quantity trade-off* in mapping. An unmapped node n_w is mapped if, *for at least* $100 \times f\%$ *of edges between* n_w *and already mapped nodes in* G_w, *there exist paths between* n_g *and corresponding mapped genes in* G_g. If f tends to 1.0, we expect greater topological resemblance between the mapped-WSN and GRN; for lower f-value, we expect a greater number of nodes to get mapped.

4) Illustrative example: Let us consider an example to understand the mapping process for $f = 1.0$. As shown in Figures 1a and b, the list of the genes and sensor node labels are annotated by their respective normalized rank scores. Algorithm 1 processes the list of highest ranked gene and

sensor nodes, and maps sensor node 1 into gene C. Since the next highest ranked sensor node having an edge with mapped sensor node 1 is 0, it gets mapped to the next highest ranked gene G (neighbor of gene C). Similarly, sensor node 2 is mapped to gene B. Next highest ranking node, 4, shares edges with mapped nodes 0 and 1. It is mapped to gene E which has paths to corresponding mapped genes G and C. It is noteworthy that the communication links inherit the direction of data forwarding from the direction of interactions between corresponding genes in input GRN, explaining why mapped-WSN is a directed graph. Finally, the last sensor node 3 maps to gene A, because gene A interacts directly with B.

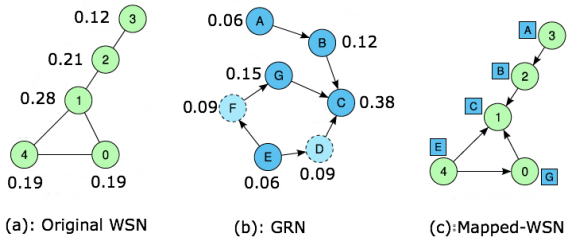
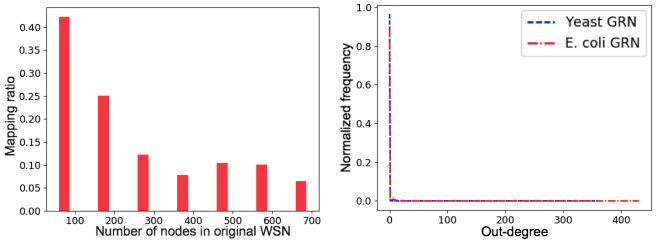


Fig. 1: Working of the mapping algorithm for $f = 1.0$. Value against each node represents its normalized pagerank score.

5) Performance for large graphs: We calculate ratio between the number of nodes in mapped-WSN and input WSN. We generate 20 original WSNs based on ER random graphs each of 100, 200, $\cdots$, 700 nodes and apply the mapping algorithm with fidelity $f = 0.8$. Figure 2a shows that *mapping ratio* decreases with increase in graph order. We analyze the observed decline in mapping ratio for large graphs.



(a) (b)

Fig. 2: (a) Mapping ratio for varying graph order of original WSN (b) Normalized out-degree distribution of GRN

Analysis: ER random graph are initialized with n isolated nodes and existence of directed edges in the graph are regulated by a predefined probability p. Given any node in ER random graph, there are $\binom{n}{d}$ ways of choosing d neighbor nodes, and $p^d \times (1-p)^{n-d}$ denotes the probability of a node with d neighbors. Thus, probability of existence of nodes with degree d is given by binomial distribution $P(d) = \binom{n}{d} p^d \times (1-p)^{n-d}$. Hence, expected mean degree = $\sum_{d=0}^{n} P(d) = np$. Expected mean degree of original WSN is proportional to graph order.

In contrast, GRN has scale free out-degree distribution (Fig. 2b), where very few nodes, called *hubs*, have high out-degree. The hubs, by virtue of high connectivity, get mapped; non-hubs, which have low degree of connectivity, fail to meet the path-to-edge mapping criteria (shown in Lines 8 - 21 of Algorithm 1), resulting in a poor mapping ratio for larger graphs. To address this problem, we propose a scalable version of the mapping algorithm, called *hierarchical mapping*.