# Towards a Multiscale Approach To Cybersecurity Modeling

Emilie Hogan, Peter Hui, Sutanay Choudhury, Mahantesh Halappanavar, Kiri Oler, Cliff Joslyn
Pacific Northwest National Laboratory
Richland, WA
{emilie.hogan, peter.hui, sutanay.choudhury, mahantesh.halappanavar, kiri.oler, cliff.joslyn}@pnnl.gov

***Abstract*—We propose a multiscale approach to modeling cyber networks, with the goal of capturing a view of the network and overall situational awareness with respect to a few key properties — connectivity, distance, and centrality — for a system under an active attack. We focus on theoretical and algorithmic foundations of multiscale graphs, coming from an algorithmic perspective, with the goal of modeling cyber system defense as a specific use case scenario. We first define a notion of *multiscale* graphs, in contrast with their well-studied single-scale counterparts. We develop multiscale analogs of paths and distance metrics. As a simple, motivating example of a common metric, we present a multiscale analog of the all-pairs shortest-path problem, along with a multiscale analog of a well-known algorithm which solves it. From a cyber defense perspective, this metric might be used to model the distance from an attacker's position in the network to a sensitive machine. In addition, we investigate probabilistic models of connectivity. These models exploit the hierarchy to quantify the likelihood that sensitive targets might be reachable from compromised nodes. We believe that our novel multiscale approach to modeling cyber-physical systems will advance several aspects of cyber defense, specifically allowing for a more efficient and agile approach to defending these systems.**

## I. Introduction

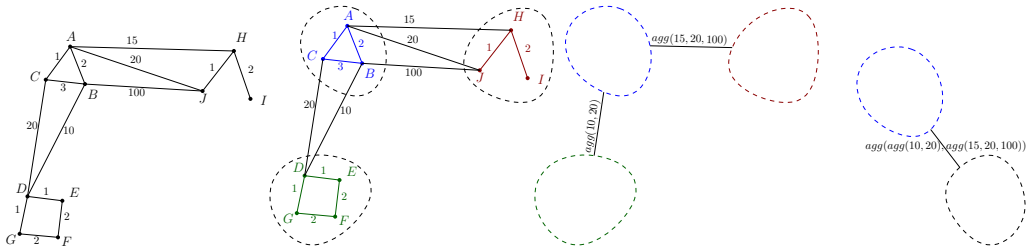
Modeling large cyber-physical systems is an inherently daunting task [7]. With the size of larger enterprise networks approaching the order of millions of nodes or greater, cyber defenders may, in practice, prefer to analyze their network at different scales and different levels of abstraction and granularity. For instance, communications between machines within a subnet might be thought of as comprising the lowest level of granularity, while subnet, local-area-network, and enterprise-level abstractions can be viewed as composing successively higher hierarchical levels. One benefit to abstracting the network at multiple levels of granularity in this manner is the ability to summarize certain network metrics at higher hierarchical levels in a more efficient manner, effectively gaining efficiency while trading off a certain degree of precision in exchange. In particular, under an active cyber attack, defenders of large networks encompassing millions of nodes may, in some cases, be willing to accept a coarse approximation of certain network metrics in lieu of precise measurements if it can be shown that the former can be obtained more efficiently than the latter.

To this end, this paper summarizes some initial results of research performed as part of Pacific Northwest National Laboratory's *Asymmetric Resilient Cybersecurity* research initiative towards using a multiscale view of the network with the intent of achieving an efficient approximation of the system's state while trading off some level of precision in the process, with the intent of gaining an asymmetric advantage over a potential cyber attacker. With this said, it is important to note that the ultimate use of the concepts we develop here is to aid in the efficient analysis of cyber systems for cyber defense. However, before doing so, the theoretical foundations and algorithms for multiscale graphs must be first developed, and to this end, the vast focus of this paper is on laying out these mathematical preliminaries, leaving cyber-based applications of these formalisms for future work. The main contributions of this paper are to present some initial results on foundations and rudimentary algorithms over multiscale graphs, with the underlying intent of ultimately using this theory to model cyber systems. The rest of the paper proceeds as follows: Section II presents multiscale graphs, an analagous definition of paths in these graphs, a multiscale variant of the well-known Floyd-Warshall algorithm [5], and discusses progress towards a software library implementing these concepts as well as preliminary timing results demonstrating the potential efficiency gained when using these methods. Section III discusses the notion of reachability in multiscale graphs, and presents a notion of *probabilistic* reachability intended to quantify the likelihood of a node's compromise within a network under attack. Section IV concludes.

## II. Multiscale Graphs and Shortest Paths

The concept of a *multiscale* graph is not particularly complicated, or even a completely new one — see, for instance [1], [3], [4], [6], [8]. At its simplest level, one can view a multiscale graph as a traditional graph paired with a *hierarchical partition* of its nodes. Figure 1 shows a simple example of a 3-level graph. Informally, we view a multiscale graph as a traditional ("single-

978-1-4799-1535-4/13/$31.00 ©2013 IEEE




(a) (b) (c) (d)

Fig. 1: An example constructing a *multiscale* graph.

scale") graph combined with a *hierarchical partition*, in which the nodes are successively partitioned into *supernodes* at each level. In the example shown in Figure 1, the base (*level-0*) graph consists of 10 nodes, $A$ through $J$. In the figure, (a) shows the base (level-0) graph, which can be viewed as a traditional (single-scale) graph consisting of nodes and edges. (b) shows the process of creating a two level multiscale graph by identifying $A, B$, and $C$ into one (blue) node, $D, E, F$, and $G$ into another (green) node, and $H, I$, and $J$ into a third (red) one. (c) shows the resulting 2-level graph. (d) shows the result of identifying the green and red nodes into a single (black) node to create a 3-level multiscale graph. The edges between supernodes are defined by an abstract *aggregation* function, which takes the weights of all edges between component *subnodes*, and returns a single weight between the supernodes (e.g. *min, max, mean*). Formal definitions follow (and are necessarily brief due to space constraints):

**Definition 1** (Hierarchical Partition over $V$)**.** *Let $V$ be a set of vertices. Then a **hierarchical partition over** $V$, $\Pi = \{\Pi_0, \Pi_1, \ldots, \Pi_k\}$, is defined as follows:*

*(i)* $\Pi_i$ *is a partition of $V$ for all $i = 0, \ldots, k$,*

*(ii)* $\Pi_i$ *is a refinement of $\Pi_{i+1}$ for all $i = 0, \ldots, k-1$, i.e., elements of $\Pi_i$ are partitioned, forming elements of $\Pi_{i+1}$, and*

*(iii)* $\Pi_0 = \{\{v\} | v \in V\}$ *and* $\Pi_k = \{V\}$.

**Note.** *A hierarchical partition over $V$ forms a chain in the partition lattice of $V$ [2].*

**Definition 2** (Multiscale Graph)**.** *A **multiscale graph** $\mathcal{G}$ is a tuple $\langle V, E, w, \Pi, f \rangle$, where*

- *$V$ is the set of vertices*
- *$E \subseteq V \times V$ is the set of edges*
- *$w : E \to \mathbb{R}$ is the weight function mapping edges to their weights*
- *$\Pi = \{\Pi_0, ..., \Pi_k\}$ is a Hierarchical Partition over $V$, and*
- *$f : \overline{\mathbb{R}} \to \mathbb{R}$ is the* aggregation *function*

The hierarchical partition $\Pi$ defines a set of graphs $G = \{G_0, ..., G_k\}$, forming a hierarchy as follows:

- $G_0 = (V, E, w) = (V_0, E_0, w_0)$
- For all $i > 0$, $G_{i+1} = (V_{i+1}, E_{i+1}, w_{i+1})$ is the graph formed by identifying all vertices in a set $\pi^j_{i+1}$ (where $\Pi_{i+1} = \{\pi^1_{i+1}, ..., \pi^n_{i+1}\}$) into a single vertex which we name $\pi^j_{i+1} \in V_{i+1}$. Edges $(v, w) \in E_i$, where $v \in \pi^j_i$ and $w \in \pi^k_i$ are contracted into a single edge $(\pi^j_{i+1}, \pi^k_{i+1}) \in E_{i+1}$, with corresponding edge weights are defined by $f$.
- $G_i$ is called the **level-$i$ graph**.

Having now defined our concept of multiscale graphs, we shift our attention to the multiscale analogs of two very basic graph concepts — paths and path distance. Informally, the basic idea is that for two nodes $\pi^j_i, \pi^k_i \in \Pi_i$ in the hierarchy, the length of a path between the nodes, $w(\pi^j_i, \ldots, \pi^k_i)$, is the sum of the edge weights along the path, plus the length of the *worst case* (longest) level $i-1$ path through each of the intermediate supernodes (under the assumption that all intermediate supernodes along the path are themselves fully connected— we revisit this assumption in Section III). As usual then, the *distance* between $\pi^j_i$ and $\pi^k_i$, $\delta(\pi^j_i, \pi^k_i)$, is the length of the shortest path between $\pi^j_i$ and $\pi^k_i$. The result is a pessimistic definition, giving a worst-case approximation of the length of the path from any subnode of $\pi^j_i$ to any subnode of $\pi^k_i$. More formally, we give a very natural extension of the traditional definition of path length, incorporating *diameters*, $\emptyset(\pi^\ell_i)$, of the supernodes along a path as given by the following recursive definition:

$$\delta(\pi^j_i, \pi^k_i) = \begin{cases} \min(w(p) : \pi^j_i \overset{p}{\rightsquigarrow} \pi^k_i) & \text{if } \exists \text{ path } p, \\ & \pi^j_i \text{ to } \pi^k_i \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

$$w(\pi^j_i, ..., \pi^k_i) = \sum_{r=1}^{n-1} w(\pi^{p_r}_i, \pi^{p_{r+1}}_i) + \sum_{s=1}^{n} \emptyset(\pi^{p_s}_i) \tag{2}$$

$$\emptyset(\pi^\ell_i) = \max_{\pi^r_{i-1}, \pi^s_{i-1} \in \pi^\ell_i} \delta(\pi^r_{i-1}, \pi^s_{i-1}) \tag{3}$$

Equation 1 defines the length of the shortest path between two nodes, equation 2 defines the weight of a specific path— identical to the traditional definition of pathweight, but now incorporating the *diameters* of the nodes as well, and equation 3 defines the diameter of a node.

These equations, then, give rise to a pair of algo-

rithms, shown in Algorithms 1 and 2, for computing our multiscale version of Floyd-Warshall's algorithm [9], [11]. The algorithm is precisely the standard version of the algorithm, with the following exceptions:

- Algorithm 1, Line 6: we now account for the diameters of the nodes when initializing the $d$ matrix
- Algorithm 1, Line 12: we subtract the diameter of the intermediate node; without doing so, this diameter would be counted twice when calculating the weight
- We include Algorithm 2 to recursively compute the diameter of the supernodes. The recursion terminates at the lowest level of the hierarchy.

**Algorithm 1** Multiscale Floyd-Warshall

1: $d$ = a $|V| \times |V|$ array of minimum distances initialized to $\infty$
2: **for** $i = 1$ to $|V|$ **do**
3: $\quad d[i][i] = 0$
4: **end for**
5: **for** each edge $u, v$ **do**
6: $\quad d[u][v] = w(u, v) + \emptyset(u) + \emptyset(v)$ {$\emptyset$ computed using Algorithm 2}
7: **end for**
8: **for** $k = 1$ to $|V|$ **do**
9: $\quad$ **for** $i = 1$ to $|V|$ **do**
10: $\quad\quad$ **for** $j = 1$ to $|V|$ **do**
11: $\quad\quad\quad$ **if** $i \neq k$ and $k \neq j$ **then**
12: $\quad\quad\quad\quad n = d[i][k] + d[k][j] - \emptyset(k)$
13: $\quad\quad\quad\quad$ **if** $n < d[i][j]$ **then**
14: $\quad\quad\quad\quad\quad d[i][j] = n$
15: $\quad\quad\quad\quad$ **end if**
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end for**
18: $\quad$ **end for**
19: **end for**
20: **return** $d$

Up to this point, we have predicated our discussion on the assumption that by trading off some degree of precision, a multiscale, hierarchical view of the network (graph) provides the opportunity to gain efficiency in exchange. We next show informally how this can be the case in some situations. Due to space constraints, we defer a more thorough analysis for later work.

**Notation.** *Let $T(i)$ be the time complexity of Algorithm 1 on a graph at level $i$ of its hierarchy. For the sake of this discussion, we assume that at each level of the hierarchy, nodes are partitioned equally, and let $\mathbf{N}_m = |\Pi_m|$ be the number of nodes at level $m$ of the hierarchy.*

**Example 1.** *The three-level multiscale graph in Figure 1 has $\mathbf{N}_0 = 10$, $\mathbf{N}_1 = 3$, and $\mathbf{N}_2 = 2$.*

Consider a three-level hierarchy. $T(1)$, the time to run Algorithm 1 on the first level of the hierarchy is given by

**Algorithm 2** $\emptyset$: Compute the diameter of the given supernode

$g$ : The supernode of which we are computing the diameter

1: $d = MFW(g)$ {MFW = Multiscale Floyd-Warshall, computed using Algorithm 1}
2: $max = -1$
3: **for** $i = 1$ to $|V|$ **do**
4: $\quad$ **for** $j = 1$ to $|V|$ **do**
5: $\quad\quad m = d[i][j]$
6: $\quad\quad$ **if** $m \neq \infty$ and $m > max$ **then**
7: $\quad\quad\quad max = m$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**

$$T(1) = \underbrace{O\Big(\Big(\frac{\mathbf{N}_0}{\mathbf{N}_1}\Big)^3 \cdot \mathbf{N}_1\Big)}_{(1)} + \underbrace{O(\mathbf{N}_1^3)}_{(2)}$$
$$= O\Big(\frac{\mathbf{N}_0^3}{\mathbf{N}_1^2}\Big) + O(\mathbf{N}_1^3).$$

Quantity (1) gives the time required to run Floyd-Warshall on each of the $\mathbf{N}_1$ nodes, each consisting of $\frac{\mathbf{N}_0}{\mathbf{N}_1}$ vertices, and quantity (2) gives the time required to run Floyd-Warshall at level 1.

Similarly, computing the values at the second level would require a running time of

$$T(2) = T(1) + O\left(\Big(\frac{\mathbf{N}_1}{\mathbf{N}_2}\Big)^3 \cdot \mathbf{N}_2\right) + O(\mathbf{N}_2^3)$$
$$= \underbrace{O\Big(\frac{\mathbf{N}_0^3}{\mathbf{N}_1^2}\Big) + O(\mathbf{N}_1^3)}_{T(1)} + O\Big(\frac{\mathbf{N}_1^3}{\mathbf{N}_2^2}\Big) + O(\mathbf{N}_2^3).$$

And in general, computing the values at the $n$'th level requires a running time of

$$O\left(\sum_{i=1}^{n}\Big(\frac{\mathbf{N}_{i-1}^3}{\mathbf{N}_i^2} + \mathbf{N}_i^3\Big)\right).$$

As one would expect, then, the overall running time depends on how nodes are partitioned into groups at each level. For example, consider a system in which the $\mathbf{N}_i$ nodes at the $i$'th level are partitioned into roughly $\sqrt{\mathbf{N}_i}$ groups. In a three-level system, then, we have that $\mathbf{N}_1 = \Theta(\mathbf{N}_0^{\frac{1}{2}})$, $\mathbf{N}_2 = \Theta(\mathbf{N}_1^{\frac{1}{2}}) = \Theta(\mathbf{N}_0^{\frac{1}{4}})$, and hence that the running time of Algorithm 1 is given by

$$T(2) = O\Big(\frac{\mathbf{N}_0^3}{\mathbf{N}_0}\Big) + O(\mathbf{N}_0^{\frac{3}{2}}) + O\Big(\frac{\mathbf{N}_0^{\frac{3}{2}}}{\mathbf{N}_0^{\frac{1}{2}}}\Big) + O(\mathbf{N}_0^{\frac{3}{4}})$$
$$= O(\mathbf{N}_0^2).$$

which gives an asymptotic improvement over $\mathbf{N}_0^3$, the running time of (traditional) Floyd-Warshall over the base graph. Certainly a more rigorous, extensive anal-

ysis is required to show the precise conditions under which Algorithm 1 gives asymptotic improvement in running time, but due to space constraints, we leave our analysis at this example. The point here is to show that there *are* situations in which a multiscale view of the network, coupled with multiscale variants of algorithms such as Floyd-Warshall, hold the potential for gaining improved efficiency in exchange for reduced precision. We leave a formal analysis of exactly when these conditions hold for future work.

We conclude this section with a brief note on an implementation in progress as part of our larger effort towards developing OpenMSG, an open source, multiscale graph library. Specifically, we have an initial C++ codebase which implements Algorithms 1 and 2.

As a compelling argument demonstrating the potential improvement using our proposed multiscale methods, Figure 2 gives a concrete comparison of our implementation on a 3-level multiscale graph to the traditional version on the corresponding flat graph. Specifically, timing measurements were taken over varying sizes of two classes of randomly generated graphs, with the multiscale versions partitioned to three hierarchical levels, with the $\mathbf{N}_n$ nodes at level $n$ partitioned into $O(\sqrt{\mathbf{N}_n})$ supernodes (as in the above discussion).The first of these graph classes is that of a scale-free network (i.e., one whose degree distribution asymptotically follows a power law), and is intended to show performance of our library over a sparse graph. The second class of graphs are Erdős-Rényi random graphs (with parameter 0.5, indicating a probability of 50% of an edge between two vertices), and is intended to show performance over dense graphs.

The results show significant performance gains in this case; the best cases (both sparse and dense graphs of $10^3$ vertices) yield improvements on the order of 3-4 orders of magnitude, while the largest cases (sparse and dense graphs of $10^4$ vertices) complete in approximately 2 seconds in the multiscale case, and take in excess of 24 hours in the flat case. As a side note, these graph sizes are comparatively small in context; for example, cyber networks can conceivably exceed millions of nodes in larger networks.

While these numbers should not be entirely surprising given the comparative time complexity of the respective algorithms, these results nonetheless exemplify the potential drastic improvement in running time attainable when taking a multiscale view of the system as compared to a flat view of the same network. The measurements were taken on a 2.1GHz, 32-core AMD Opteron processor with 64GB RAM, running RedHat Linux.

## III. Reachability

Up to this point, we have focused exclusively on the notion of paths and path distance within a multiscale graph. We now shift our attention to the related notion of reachability. Specifically, by shifting to a multiscale view of the system, the precise definition of reachability within the network is no longer clear.

To this end, we define the question of multiscale reachability in the following manner: Given two supernodes, $\pi_i^j$ and $\pi_i^k$, are all nodes within $\pi_i^k$ reachable from all nodes within $\pi_i^j$? If there is no path from $\pi_i^j$ to $\pi_i^k$ in $G_i$ then clearly the answer is "no". The converse does not hold, however — connectivity between $\pi_i^j$ and $\pi_i^k$ does not imply that all subnodes of $\pi_i^j$ and $\pi_i^k$ are necessarily connected. Figure 3 gives an example — in this figure, $\pi_i^0$ is reachable from $\pi_i^2$, but not all vertices in $\pi_i^0$ are reachable from $\pi_i^2$. Briefly, our motivation behind this question can again be viewed in terms of cyber system defense. For example, if a machine on subnet $A$ is known to have been compromised, a cyber system defender may want to gauge the probability that the intruder could have compromised a system on subnet $B$ as well.

To this end, we formulate the notion of reachability in a probabilistic manner: given supernodes $\pi_i^j$ and $\pi_i^k$ what is the probability that a randomly chosen vertex within $\pi_i^j$ can reach a randomly chosen vertex within $\pi_i^k$?

### A. Probabilistic Reachability

In Section II we did not discuss the possibility of $\emptyset(\pi_i^k) = \infty$. If the set of edges within $\pi_i^k$ forms a disconnected graph then the diameter at that vertex will be infinity. However, it is still possible to have a path that goes through this vertex with finite weight. For example, in Figure 3 the length of the shortest path, as calculated in Algorithm 1, is infinity since $\emptyset(\pi_i^1) = \emptyset(\pi_i^3) = \infty$. For some pairs of vertices in $\pi_i^0$ and $\pi_i^2$ this path length of infinity is correct, but for other pairs there is a finite length path. After making this observation, it seems that a probabilistic argument is needed to be able to say, for example, *The length of the shortest path from an arbitrary vertex in $\pi_i^0$ to an arbitrary vertex in $\pi_i^2$ is infinity with probability $p$ and some finite value with probability $1-p$.*

We propose two approaches to this probabilistic reachability. In both, we restrict to the case where we have three levels in a hierarchical partition over $V$, $\Pi = \{\Pi_0, \Pi_1, \Pi_2\}$. Partitions $\Pi_0$ and $\Pi_2$ are given as in Definition 1. Each supernode, $\pi_1^i \in \Pi_1$, consists of connected components $\{V_1^{i,m}\}_{m=1}^{\ell_i}$. In Figure 3 the connected components within $\pi_1^1$ and $\pi_1^3$ are shown.

#### 1) Probabilities based on edge density

The first approach is based on the sizes of the supernodes and connected components within each supernode, and the edge density of the graph. We will compute the probability that an arbitrary $v \in \pi_1^j$ can reach an arbitrary $w \in \pi_1^k$ given that there is an edge $(\pi_1^j, \pi_1^k) \in E_1$. Notice that the subscript here is 1 indicating that we are at level-1. However, this approach

| | | # of Vertices | | |
|---|---|---|---|---|
| | | 1000 | 5000 | 10000 |
| Multiscale | Sparse (Scale-Free) | 0.67s | 1.31s | 2.41s |
| | Dense (Erdős-Rényi) | 0.49s | 0.74s | 2.73s |
| Flat (Single-scale) | Sparse (Scale-Free) | 373.17s | 60000s | $> 24$h* |
| | Dense (Erdős-Rényi) | 2793.59s | $> 24$h* | $> 24$h* |

* *indicates run was terminated prematurely due to excessive running times*

Fig. 2: Timing results comparing traditional (single-scale) Floyd-Warshall vs. an implementation of our multiscale variant on two classes of randomly generated multiscale graphs. Of particular note is the improvement in running time gained when running the multiscale algorithm over the traditional version.

will generalize to an arbitrary level.

$$\mathbb{P}_{j,k} := \mathbb{P}\left[v \in \pi_1^j \overset{p}{\rightsquigarrow} w \in \pi_1^k \;\middle|\; (\pi_1^j, \pi_1^k) \in E_1\right].$$

In order for this event to occur we must have some $v_r$ in the same connected component as $v$, and $w_s$ in the same connected component as $w$ such that $(v_r, w_s) \in E_0$. Say that $v, v_r \in V_1^{j,m}$ and $w, w_s \in V_1^{k,\ell}$. Then the probability above is equal to

$$\mathbb{P}_{j,k} = \mathbb{P}\left[\exists v_r \in V_1^{j,m}, \exists w_s \in V_1^{k,\ell} : (v_r, w_s) \in E_0 \mid (\pi_1^j, \pi_1^k) \in E_1\right].$$

Let's abbreviate the first event, before the conditional, as $E[j,m;k,\ell]$. Now we can calculate $\mathbb{P}_{j,k}$ based on $\mathbb{E}\left[|V_1^{j,m}|\right]$, the expected value of the size of a connected component, for each $j$, $|\pi_1^j|, |\pi_1^k|$, and the edge density $p = \frac{|E_0|}{\binom{|V_0|}{2}}$. First we rewrite $\mathbb{P}_{j,k}$ using Bayes' theorem.

$$\mathbb{P}_{j,k} = \frac{\mathbb{P}\left[(\pi_1^j, \pi_1^k) \in E_1 \mid E[j,m;k,\ell]\right] \mathbb{P}\left[E[j,m;k,\ell]\right]}{\mathbb{P}\left[(\pi_1^j, \pi_1^k) \in E_1\right]}$$

In the numerator, the conditional probability is 1 because of how $E_1$ is created from edges in $E_0$. So, we can rewrite $\mathbb{P}_{j,k}$ as follows.

$$= \frac{\mathbb{P}\left[\exists v_r \in V_1^{j,m}, \exists w_s \in V_1^{k,\ell} : (v_r, w_s) \in E_0\right]}{\mathbb{P}\left[(\pi_1^j, \pi_1^k) \in E_1\right]}$$

$$= \frac{1 - \mathbb{P}\left[\forall v_r \in V_1^{j,m}, \nexists w_s \in V_1^{k,\ell} : (v_r, w_s) \in E_0\right]}{1 - \mathbb{P}\left[(\pi_1^j, \pi_1^k) \notin E_1\right]}$$

$$= \frac{1 - (1-p)^{\mathbb{E}\left[|V_1^{j,m}|\right]\mathbb{E}\left[|V_1^{k,\ell}|\right]}}{1 - (1-p)^{|\pi_1^j||\pi_1^k|}}$$

It's not difficult to show that $\mathbb{E}\left[|V_1^{j,m}|\right]$ is simply the average size of the components in $\pi_1^j$, for any $j$.

From this we can calculate the probability of general reachability by finding a level-1 path and taking the product of the probabilities on each edge. Now, there are likely multiple level-1 paths, and it's possible that the path with the highest probability is not the shortest distance path.

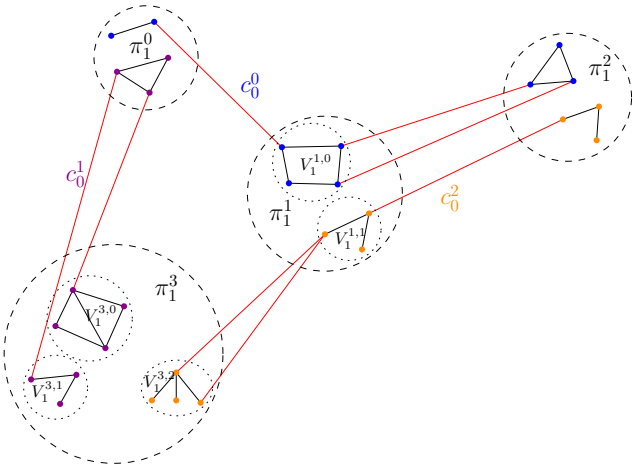



Fig. 3: Example graph for demonstrating reachability option described in Section III-A2.

*2) Probabilities based on connected components in* $\Pi_0$

This approach is based on the interplay between connected components within $G_0$, $\{c_0^\ell\}$, and the partitions $\{\pi_1^j\}_{j=1}^n$. The basic idea is being able to travel along connected components that span multiple partitions. So, we must first precompute connected components on the lowest level of the hierarchy.

1) Compute the connected components of $G$
2) Create a bipartite graph, $B = (\{\pi_1^j\} \cup \{c_0^\ell\}, F)$, where $(\pi_1^j, c_0^\ell) \in F$ if $\pi_1^j \cap c_0^\ell \neq \emptyset$.
3) Edges, $(\pi_1^j, c_0^\ell) \in F$, are weighted with the probability of being in connected component $c_0^\ell$ when in partition $\pi_1^j$.

$$w(\pi_1^j, c_0^\ell) = \frac{|\pi_1^j \cap c_0^\ell|}{|\pi_1^j|}$$

For example, consider the graph in Figure 3 where the partitions are given as the circled sets of nodes, $\pi_1^0, \ldots, \pi_1^3$. The connected components are identified by the colors of the vertices. Vertices colored blue are in $c_0^0$, purple vertices are in $c_0^1$, and orange is $c_0^2$. Given this we can create the bipartite graph described in step 3 above. This graph is found in Figure 4.

Now that we have this bipartite graph we can calculate probabilities for reaching a vertex in one partition

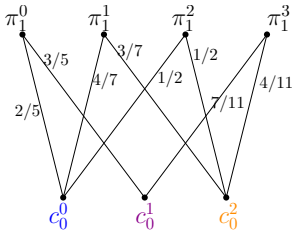



Fig. 4: The bipartite graph representing the intersection between partitions and connected components.

from a vertex in another. Notice that a path of length two in the bipartite graph from $\pi_1^j$ to $\pi_1^k$ represents a possible way of traveling from partition $j$ to partition $k$. For example, consider the path $\pi_1^2 \to c_0^2 \to \pi_1^1$. This represents starting at a vertex in partition $\pi_1^2$, traveling through connected component $c_0^2$, and ending up in partition $\pi_1^1$. We can then use the probabilities on the edges of the bipartite graph to calculate the probability of reaching any vertex in $\pi_1^1$ from any vertex in $\pi_1^2$ on the given path. Any longer path would mean that we transferred traveled through more than one connected component. This cannot happen, therefore we can restrict our attention only to paths through $B$ of length two.

Choose an arbitrary vertex in $\pi_1^2$, then the first edge in our chosen 2-path through the bipartite graph tells us that there is $\frac{1}{2}$ probability that we are on a vertex in connected component $c_0^2$. Then, when we travel along that connected component into partition $\pi_1^1$ the next edge tells us that we have $\frac{3}{7}$ probability to be on an arbitrarily chosen vertex in $\pi_1^1$. Therefore, given $v_2 \in \pi_1^2$ and $v_1 \in \pi_1^1$ we have $\frac{1}{2} \cdot \frac{3}{7} = \frac{3}{14}$ probability of reaching $v_1$ from $v_2$ via connected component $c_0^2$. Similarly, the path $\pi_1^2 \to c_0^0 \to \pi_1^1$ has probability $\frac{4}{14}$. The bipartite graph tells us that these two paths are the only way to reach from a vertex in $\pi_1^2$ to $\pi_1^1$ so we can say that the total probability is

$$\mathbb{P}[v_2 \in \pi_1^2 \to v_1 \in \pi_1^1] = \frac{3}{14} + \frac{4}{14} = \frac{1}{2}$$

We can formalize this process (for undirected graphs) as follows. Given two partitions, $\pi_1^j$ and $\pi_1^k$, find the set of their common neighbors in $B$. This will be the connected components that intersect both partitions. Add together the product of the probabilities along each 2-path $\pi_1^j \to c_0^\ell \to \pi_1^k$ to get the total probability of reachability.

$$\mathbb{P}\left[v \in \pi_1^j \to w \in \pi_1^k\right] = \sum_{c_0^\ell \in \mathcal{N}(\pi_1^j) \cap \mathcal{N}(\pi_1^k)} w(\pi_1^j, c_0^\ell) w(c_0^\ell, \pi_1^k).$$

This method has the advantage of knowing that we have considered all possible paths since we can only look at 2-paths through $B$. However, it relies on being able to precompute connected components in $G_0$, a possibly computationally intensive step.

## IV. Conclusion

We have laid out some initial theoretical foundations underlying multiscale graphs, ultimately intended for use in computing efficient approximations of network metrics in a cyberdefense setting. While the vast bulk of the paper has been devoted to these mathematical formalisms, we have alluded to ways that these methods can be used in such applications. For example, our algorithm for computing multiscale shortest paths at at any level of the hierarchy, could be used to determine distance of a cyber-attacker to a sensitive resource. Furthermore, we have shown empirically that at least in some situations, our methods can lead to drastic improvements in running time. We have also laid the groundwork for computing a probabilistic notion of reachability, providing analysts with a method of gauging the likelihood that a sensitive machine has been compromised. Clearly, this is only the beginning. How can these methods be used on network data? What are some other use case scenarios that could benefit from a multiscale view? And how can these concepts be used to define, for instance, a multiscale notion of clustering coefficients and centrality? These are all important questions which we plan to address as our next steps.

## References


[1] James Abello. Hierarchical graph maps. *Computers & Graphics*, 28(3):345–359, 2004.

[2] Garrett Birkhoff. *Lattice Theory*. Colloquium Publications, American Mathematical Society, 1995.

[3] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Structural inference of hierarchies in networks. In *Statistical network analysis: models, issues, and new directions*, pages 1–13. Springer, 2007.

[4] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Press, 2001.

[6] Sivarama P Dandamudi and Derek L. Eager. Hierarchical interconnection networks for multicomputer systems. *Computers, IEEE Transactions on*, 39(6):786–797, 1990.

[7] Patricia Derler, Edward A. Lee, and Alberto Sangiovanni Vincentelli. Modeling cyber-physical systems. *Proceedings of the IEEE*, 100(1):13–28, January 2011.

[8] S. Ferrari, M. Maggioni, and N.A. Borghese. Multiscale approximation with hierarchical radial basis functions networks. *Neural Networks, IEEE Transactions on*, 15(1):178–188, 2004.

[9] Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, June 1962.

[10] Jeremy Siek, Lie-Quan Lee, and Andrew Lumsdaine. The Boost Graph Library. http://www.boost.org/libs/graph/, June 2000.

[11] Stephen Warshall. A theorem on Boolean matrices. *Journal of the ACM*, 9(1):11–12, January 1962.