18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

Multiscale Network Generation

Alexander Gutfraind
School of Public Health
University of Illinois at Chicago
Chicago, IL
Email: agutfrai@uic.edu

Ilya Safro
School of Computing
Clemson University
Clemson, SC
Email: isafro@clemson.edu

Lauren Ancel Meyers
Section of Integrative Biology
University of Texas at Austin
Austin, TX
Email: laurenmeyers@austin.utexas.edu

***Abstract*—Relationships between entities in complex systems could be represented using the paradigm of networks. The network representation can then reveal the evolution, structure and dynamics of those systems. Frequently, obtaining the required scientific data about the networks is expensive or infeasible. In other words, the amount of available empirical data is insufficient for simulation, validation, verification, and other scientific tasks. In these situations, empirical data should be augmented by synthetic data generated from models in such a way that properties of the system are preserved in the synthetic data, even when those properties are unique to the system or not fully known, but existing methods only reproduce a limited set of specified network properties. Here we introduce a novel strategy for synthesizing artificial networks that can realistically model a variety of network properties and that is termed multiscale network generation, or as a specific algorithm, MUSKETEER. This strategy first creates a hierarchy of aggregated representations of the original network, and then reformulates the network generation problem at all levels of this hierarchy in order to take into account properties at multiple scales of the system. The network is then edited at any or all scales, depending on the desired variability in the ensemble of synthetic networks. We find that for many complex networks taken from real-world systems, the strategy is able to preserve important properties with little statistical bias while achieving high degree of variability and arbitrary difference from the original.**

I. Introduction

Across the sciences, networks can represent connections between entities, and thus provide insight into the function, dynamics, and evolution of natural and man-made systems. However, high-quality, large-scale network data is often not available, because of economic, legal, technological, or other obstacles [1]. In particular, government agencies that develop real-time and large-scale network applications name network modeling among their key challenges [2], [3]. For example, the human contact networks along with infectious diseases spread are notoriously difficult to estimate, and thus, we can misunderstand the dynamics and control of epidemics stems from models that make highly simplifying assumptions or simulate contact networks from incomplete or proxy data [4], [5]. In another domain, the development of cybersecurity systems requires testing across diverse threat scenarios and validation across diverse network structures that are not yet known, in anticipation of the computer networks of the future [2]. In both examples, the systems of interest cannot be represented by a single exemplar network, but must instead be modeled as ensembles of networks which represent the range of diversity of networks found in cybersystems. Realistic network models are vital for evaluation of different components in information fusion systems (IFS) such as of the methods for situational awareness [6], and anomaly/threat detection [7]. Such models can be used for parameters' training, and missing data completion that are typical requirements in IFS. These cases point to the importance of data-driven methods for synthesizing networks that capture both the essential features of a system and realistic variability in order to use them in such tasks as simulations, verification, and decision making.

A good synthetic network must meet two criteria - realism and diversity. First, it should be realistic with respect to structural features that govern the domain-specific processes of interest (such as system function, dynamics, and evolution), as the following examples illustrate: (a) models of social networks should be able not only to reproduce structural features (such as the small-world property), but also, and perhaps more importantly, emulate emergent sociological phenomena such as interactions between individuals in a community, as driven by their psychological needs and daily routines (that is, the generated network should show similar interactions by its artificial individuals, as determined by implicit psychological and social rules); (b) models of interdependent infrastructure systems (such as power grids, water, and transportation networks) should demonstrate realistic resilience, joint performance, and potential mutual failures; and (c) models of metabolic interactions should ultimately reflect biochemical properties of a cell. Second, a synthetic network should reflect normally-occurring diversity in a system, but without systematic biases that departs from reality. This feature is particularly important for benchmarking and evaluating the robustness of network-based algorithms, anonymizing networks, and generating plausible hypothetical scenarios.

A number of network generation methods have been developed, and these fall into two classes: parametric generative models and editing methods (see surveys in [8], [1], [3], [9], [2]). The first set of methods produces (by using randomization and replication) a network from a small initial seed network (sometimes empty). The goal of such generation is to produce the structure that matches real data in *prespecified* properties, such as the degree distribution [10], [11], clustering [12], and the number of small subgraphs [13]. These methods are attractive because they often produce networks with the desired features and are grounded in well-developed theory

(e.g., [14]). Some of these network generators mechanistically model network growth [15], [16], whereas others incorporate domain-specific information such as geographic location [17] and cyber networks topological properties [18], [2]. One of the most successful generative strategies is based on Kronecker graphs [19] (including stochastic Kronecker graphs; see also related work [1], [20]). Networks generated by this model preserve properties such as degree distribution, diameter, and eigenvalues. Such methods often describe an evolutionary process that can potentially lead to the original network; however, the probability that it will lead to the structure that is approximately isomorphic to a particular real system is negligible, given the high dimensionality of the space. This makes generative methods ill-suited for studies such as simulations when one may need to work with synthetic networks that are similar to real systems in a broad range of properties and not just in a small number of prespecified properties. The other class of network generators is based on editing [21]. These methods start with a given (real or empirical) network and randomly change its elements until the network becomes sufficiently different from the original network. These are designed to introduce variability while preserving a few prespecified structural properties, such as the degree distribution. While promising, such a process cannot be in generally sustained for many iterations because modifications tend to introduce shortcuts in the network that quickly evolve it towards a random graph. For that reason, to avoid this biasing tendency, an editing algorithm must be calibrated based on information about the network structure.

Because networks are used in many diverse scientific and technological domains, constructing a network generator suitable for even a fraction of them is challenging. Indeed, although various models and algorithms have been proposed for synthesizing networks, our analysis suggests that they synthesize networks that fall short of what is desirable. Existing methods (1) involve generative mechanisms or system-specific assumptions that are not plausible across domains, (2) focus on one or a few predefined topological features, such as degree distribution and clustering, at the expense of others that may be critical but are perhaps unrecognized or cannot be incorporated into the generator easily, and/or (3) reliably reproduce a set of target properties but fail to capture naturally occurring stochasticity in those properties.

Here, we introduce the *multiscale network generation* (MNG) approach that is cross-domain, captures many features of real networks and incorporates an arbitrarily large or small stochasticity. MNG is inspired by the observation that many real-world networks exhibit a multiscale structure in the sense that network elements can be naturally assigned into aggregates or communities which are themselves parts of larger aggregates, and so on in a hierarchical fashion [22], [23]. Starting from a single known or hypothesized network from any domain, it synthesizes ensembles of networks that preserve, on average, a diverse set of features at multiple scales of its structure, including several measures of centrality, degree assortativity, path lengths, clustering, and modularity, while introducing unbiased variability across the ensemble in many of these properties at *multiple scales*. The most important property that distinguishes MNG from all other existing strategies is related to the realism of a synthetic network that can be obtained at the required fine-, and coarse-grained resolutions. The synthetic network will be similar to the original one at multiple scales of coarseness. Moreover, the MNG framework makes it easy to control the similarity across the scales, i.e., if desired the coarse-grained scales can differ, whereas the fine-grained scales can still be similar, and vice versa. Additionally, MNG could be combined with other methods, such as methods for anonymization and thus, it opens a new research direction in this class of methods.

The MNG framework is inspired by adaptations of multigrid (and general multiscale) methods [24], [25] to combinatorial optimization on networks [26], [27]. The multiscale method starts with a formulation of an optimization problem on a network G, and constructs a hierarchy of networks $G_0 = G, G_1, ..., G_k$ (progressively smaller) via aggregation procedure [26], solves a simpler optimization problem at the coarsest scale when G_k is small enough, and then disaggregates the solution by gradual prolongations from coarse to next-finer scales. Similarly, we create a hierarchy of networks; but, in contrast to multiscale methods for the computational and optimization problems, we do not optimize any objective on the network but rather edit the topology of the network preserving some features. To prevent the editing process from generating unrealistic structures, at any moment, we allow only local editing because the global changes (if required) are deferred to coarser levels. In other words, the problem of network editing is sequentially formulated and solved at several (possibly at all) scales.

Analogously to multiscale methods for computational problems [24], by using an appropriate aggregation method, that could be domain-specific, MNG is able to detect and exploit the "geometry" behind the original network at multiple scales. This geometry is not captured by other network generation methods. Indeed, network properties that are usually preserved by the existing generation methods (such degree distribution, or clustering) on the original network (i.e., at its finest scale G_0 only) can differ significantly at coarser scales. Moreover, it is known that the topology of many complex networks is hierarchical [22], [23], and therefore it naturally lends itself to reproduction through iterations of generative laws at multiple scales. In general, such generative laws often can be different at different scales, as evidenced by the finding that complex networks are self-dissimilar across scales [28], [22], [29].

The MNG method has been implemented in the software tool, the Multiscale Entropic Network Generator (MUSKETEER), which is made publicly available [30].

II. Notation

For both a graph and the underlying network we will use the same notation G that will be clear from the context. A graph G is represented by a pair (V, E), where V is a set of n nodes, and E is a set m of edges. We consider simple,

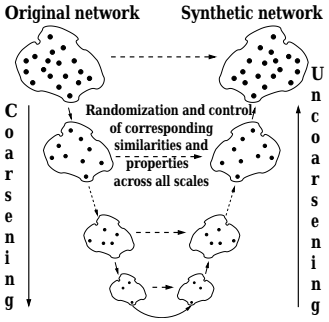

Fig. 1. The V-cycle scheme underlying multiscale network generation. The original network is first aggregated into smaller and smaller networks, and then the process is reversed. The number of levels is usually greater than the four shown here and depends on the structure and size of the original network, the aggregation algorithm, as well as the user's input.

undirected, edge-, and node-weighted graphs, although the MNG strategy can be easily generalized to other networks. The edge-, and node weighting functions are $\omega : E \to \mathbb{R}_{\geq 0}$, and $c : V \to \mathbb{R}_{\geq 0}$. By $d_p = \sum_k \omega_{pk}$ we denote the weighted degree of a node p. A Laplacian of a graph G is denoted by $L = D - A$, where D is a diagonal matrix with entries $\{d_p\}_{p=1}^n$, and A is a weighted adjacency matrix with entries ω_{pq}. The normalized Laplacian of a graph is denoted by $\mathcal{L} = D^{-1/2}LD^{-1/2}$. The Moore-Penrose pseudo-inverse of a matrix M is denoted by $M^\dagger$. The i-th unit vectors in $\mathbb{R}^n$ are denoted by e_i. Throughout this paper, we will use subscript $(\cdot)_i$ to denote the corresponding quantity $(\cdot)$ for level i in the multilevel hierarchy. The levels of the network are indexed by $i \in \{0, 1, \dots\}$, where $i = 0$ corresponds to the input network prototype. For example, G_0 denotes the input network and G_1 correspond to network at the first level of coarsening.

III. Multiscale Network Generation Method

The MNG framework is illustrated schematically in Fig. 1 and its pseudocode is presented in Algorithm 1. The MNG can be expressed as a recursive algorithm where the recursion stops based on user-defined criteria (line 1, for example when the coarsest scale is reached or when the changes are not required). MNG starts with a known network $G_0 = G$, and its first phase recursively constructs a hierarchy of network's coarse representations. Namely, the network is aggregated repeatedly (Algorithm 1, line 2) in order to create the hierarchy $\{G_i\}_{i=1}^k$. A hierarchy of aggregations could be thought of (in the simplest case) as computing consecutive restrictions using Galerkin-like operators:

$$L_{i+1} \leftarrow (P_i)^T L_i P_i, \tag{1}$$

where L_i is the Laplacian of G_i, and the $P_i \in \mathbb{R}^{n_i \times n_{i+1}}$ is the restriction matrix that describes the strength of connectivity between ith-level (fine) nodes and corresponding coarse aggregates at level $i + 1$.

While the aggregates are sometimes well-known from the application (as, for example, in [31]), in most cases the aggregated structure has to be discovered computationally, possibly without any subject-matter input for the particular class of networks. In the most general case, we do not rely on a particular method for detecting the coarse representation of a network and diverse aggregation methods may be used, including schemes like community detection, hierarchical clustering and others.

Algorithm 1. MNG(G_i)

1: **if** perturbations in levels $> i$ **then**
2: $G_{i+1}, P_i \leftarrow$ create aggregated network from G_i
3: $G_{i+1}^g \leftarrow$ MNG(G_{i+1})
4: $G_i^d \leftarrow$ interpolate unedited aggregates from G_{i+1}^g
5: $G_i^d \leftarrow$ disaggregate edited aggregates from G_{i+1}^g
6: **end if**
7: $Q_i \leftarrow$ measure properties of G_i
8: $G_i^g \leftarrow$ editing G_i^d preserving Q_i
9: **Return** G_i^g

Next, MNG recursively calls itself (line 3) to solve the network generation problem on the coarse network G_{i+1}. Generally, the recursion terminates (or some scales can be skipped) when changes at more coarse levels are not requested by the user. If requested, such coarse level changes allow the user to make changes to the larger-scale structure of the network, such as its backbone, and may or may not be desirable depending on the intended use of the synthetic network data. In any case, aggregation would stop automatically if the aggregated network is already very small or very dense.

In the next steps (lines 4 and 5) the edited network G_{i+1}^g from level $i + 1$ is disaggregated at level i, i.e., the newly generated network G_i^d will be created from its coarse representation G_{i+1}^g. Processing of *unedited aggregates* (line 4) is a straightforward lossless reverse projection operation that reverses aggregation based on stored data P_i (line 2). A unique aspect of MNG as compared to conventional multigrid is the need to construct (i.e. fill-in) fine-level structure in nodes and edges which were added to G_{i+1}^g but are not found in G_{i+1}. In order to make the structure realistic, MNG measures the fine-level structure of a random unedited node or edge from G_i, respectively, and copies this structure (with randomization) into the newly-created node or edge. When the reverse projection is ready, the network is fully disaggregated (line 5). An alternative scheme for disaggregation (line 5) may instead use information from other networks in the same domain, rather than necessarily G_i. It is critical for realism that any new structures retain properties of a comparable network at the ith level of aggregation.

Once disaggregation is complete, MNG measures the properties of the unmodified network G_i (line 7) at the i level of coarsening (a specific fast measurement approach based on random walks is described in more detail below.) Next, the disaggregated coarse "replica" network G_i^d receives controlled perturbations (line 8) to obtain G_i^g. These perturbations take into account properties of the clustered networks from the corresponding scales and preserve similarities with them if needed (dashed arrows in Fig. 1). Measurement and editing

of networks in MNG can be informed by various domain-specific considerations. In general, editing at level i should be appropriate to that level, that is, avoid edits that inadvertently modify the hierarchical structure at some higher level $j > i$. The disaggregate-and-edit process is repeated all the way up until the original coarseness level is reached. At that point the final result is generated, namely, a new network G_0^g.

The editing process performs only simple operations, but those operations can generate high-entropy changes because they could be applied everywhere in the entire hierarchy $\{G_i\}_{i=0}^k$. As a result, when nodes are added to the fine levels of the hierarchy, they have the effect of just adding nodes to the replica; but when this same addition occurs at deeper levels, the new nodes will correspond to several nodes or large communities in the final synthetic network. The deeper the network in which the change occurs, the more significant is change to the topology. In this way, one can make large, nontrivial changes to the topology of the replica through a simple random process at deeper levels. Usually the creation of a node is accompanied by the creation of edges to it, and the ultimate effect of those edges on the replica depends on the level: at the finest level, edges are just edges, whereas at deeper levels of aggregation, an edge represents many interconnections across entire communities of aggregated fine nodes. The algorithm constructs those structures using a deep copying processes, where the contents of newly-created the nodes and edges exactly copies the structure of existing aggregates at all levels of clustering. This copying process can also preserve annotation of the nodes including any categories (e.g. student vs. teacher) and relationship type.

MNG is expected to produce realistic replicas because it reproduces the original network everywhere except at the edited components, and the edited components are copied from the original network. As the edit rate is lowered, the replicas become arbitrarily close to the original in every respect, including properties that are unknown or unspecified, provided the property is not highly sensitive to small perturbations of the network. Furthermore, since all edits in MUSKETEER are constrained by the properties of the network at the appropriate structural level, the replicas are expected to preserve scale-dependent features and can be self-dissimilar across scales (in accordance with the original network). In contrast, random graph model generators are often unable to reproduce unspecified and scale-dependent properties, as demonstrated below.

A. MUSKETEER - a cross-domain implementation of MNG

Our implementation of MNG is a high-performance cross-domain algorithm termed MUSKETEER. In MUSKETEER, aggregation uses a high performance approximate weighted matching algorithm [32] because it gives aggregates of uniform size (typically 2 nodes). In this setting, there is only one nonzero entry in each row of P_i (see Eq. (1)), and at most two nonzeros per column which preserves the sparsity without additional efforts as is required in most optimization solvers [26]. We considered other algorithms such as algebraic distance-based [33] and community detection [34] and they produced qualitatively similar results.

For editing, MUSKETEER uses a higher-performance scheme based on random walks which preserves the small loop structure of the network while avoiding any insertion of edges that bridge nodes of the network which were previously separated by large distances. Before making the edits, it estimates (line 7) the probability distribution of closed random walks in the graph G_i. Starting at a node u with at least two neighbors, it performs a random walk of up to $r_{\max}$ steps which steps to a random neighbor v_1 and then stops if it finds any neighbor v_2 of u where $v_2 \neq v_1$. The probabilities of a random step are proportional to the edge weights (and self-intersect is allowed, but not return to u or to v_1). Thus, at every node p, the random walk selects node q, a neighbor of p, with probability given by $\mathbb{P}[q] = -(L_i)_{pq}/(L_i)_{pp}$. An estimate for $r_{\max}$ is the random walk hitting time upper bound

$$H(v_1, v_2) = 2m_i \left\langle \frac{1}{\sqrt{d_{v_2}}} e_{v_2}, \mathcal{L}_i^{\dagger} \left(\frac{1}{\sqrt{d_{v_2}}} e_{v_2} - \frac{1}{\sqrt{d_{v_1}}} e_{v_1} \right) \right\rangle .$$

The algorithm records the number of steps s in a random walk that found v_2 through this random process, obtaining an empirical distribution of $Q_i[s]$ for s by using several hundreds of starting points u. For graphs with a relatively large number triangles, s would be small and frequently of length 2, while graphs with a high number of longer loops would see a different typical value of s.

As mentioned earlier, the number of edits in terms of fraction of modified nodes and edges is defined by the users of MNG based on their requirements and can be as high as 50% of all nodes and/or edges. In forming new edges, the editing process (line 7) imitates the typical structure found in the measurement of the original topology. The first endpoint of the edge is selected at random, while the second is selected by performing a random walk for s steps, where s is sampled from $Q_i[s]$. The sampling is repeated several dozen times, all the nodes reached at the end of the random walks are recorded. The node found most frequently in terms of the number of hits is selected for the other endpoint. Ties are broken randomly, and existing neighbors are not allowed. Such a random walk approach is a computationally fast local process for constructing new edges consistent with the existing topology. It is possible to replace such path sampling with the estimated hitting times for all nodes but sampling replicates the local structure much more realistically than the global measure of a hitting time.

This sampling process tends to maintain multiple structural properties of the network at multiple levels even at high rates of editing. For example, the clustering coefficient (in the sense of Watts and Strogatz [12]) is preserved because in clustered networks the probability of $s = 2$ is elevated and the edge uv above would construct a triangle. Other distance-based properties such as the average geodesic distance are also preserved under this sampling because the probability of inserting an edge with a large d is low (if inserted, such

an edge would significantly reduce the average distance). When new nodes are inserted, their degree is randomly chosen based on the degree of an existing node in the network The new node is connected to a random existing node, and then subsequent edges from it are inserted by sampling from $Q_i[s]$. Deletion of nodes occurs through random sampling, while edge deletion selects a non-singleton node and deletes an edge to a randomly-selected neighbor. The number of such edits (i.e., the edit rate) can be controlled by the user and can also be tuned for a particular class of networks. MNG applies the same editing process at all scales, unless the user adds new adjustment requirements (line 8 in Algorithm 1) such as control of some invariant (like the connectivity). By default, if the original G_i is connected, the G_i^q is forced to be connected by inserting edges between components.

When the editing rate is large, it is best for realism to split large perturbations over q chained v-cycles (not shown), where each performs a fraction $1/q$ of the required perturbations. This functionality is supported by our implementation as a command-line parameter.

IV. Experimental Evaluation

Our first analysis examined how MNG in general and MUSKETEER in particular model networks at multiple scales of coarseness. The model networks should be able to resemble an existing system except at a selected scale of aggregation, for example, preserving either the system's coarse-level backbone or its fine level structure. To study this characteristic, we applied MUSKETEER to a regular mesh (grid), and power grid network from [17], each edited at two different scales Indeed, as Figures 3 and 2 show, fine-level changes appear to produce small edits while leaving intact the large-scale structure; coarse-level changes change the overall structure without perturbing the local structure (can be clearly visualized in Fig. 3 only).

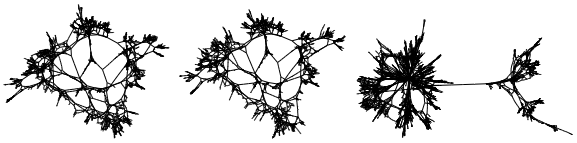

Fig. 2. Illustration of multiscale network editing using a power grid network from [17] (The US Western Interconnection): (a) original network; (b) fine level changes only (node/edge edit rate: 5% at $i = 0$); (c) deep level changes only (node/edge edit rate: $0, 0, 0, 0, 0, 5\%$).

Turning to more detailed practical instance, we evaluate MUSKETEER in terms of both the fidelity and the variability of the replicas, using an empirical example network from infectious disease epidemiology and two well-studied network models. A number of network properties are known to influence the dynamics of infectious disease outbreaks, including the number of nodes (individuals); the degree distribution (contacts) [11], [5]; clustering coefficient (number of triangles out of all possible triangles) [12], [35]; degree assortativity (tendency for nodes to connect to others with similar degree)

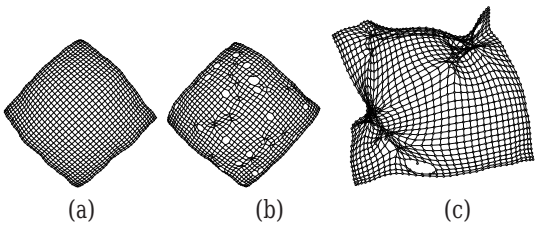

Fig. 3. Illustration of multiscale network editing using a toy input: (a) original network (33x33 mesh); (b) fine level changes only (edge edit rate: 1% at level $i = 0$); (c) deep level changes only (edge edit rate: $0, 0, 0, 0, 1\%$).

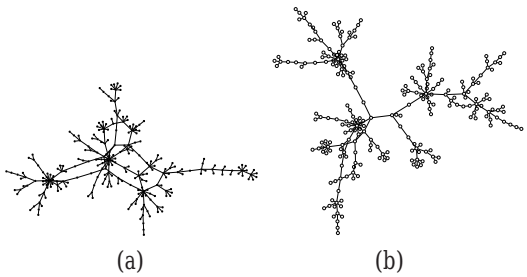

Fig. 4. Replica of an empirical sexual contact network (2a) The original contact network estimated from [41] and its replica generated by MUSKETEER (2b). The replica has 2% difference to the original network, yet still appears visually to retain its general structure. Node and edge annotation of the original (not shown) is also realistically retained in the replica.

[11], [36], which we measure according to [37], [38]; node centrality [11]; average distance, and modularity as measured by [34], which can give rise to multi-wave epidemics [39]. Many of these properties are also highly relevant to system function and dynamics in other domains [40]. One particularly well-studied epidemiological network is based on data collected in Colorado Springs by Potterat et al. ([41] Fig. 4a). It contains 250 individuals who were in contact in the 1980s through sex or injection drug use. In our experiments we edited 8% of the nodes at level $i = 6$ and the same edit rates were assigned to the edges at those levels. This gave approximately 23% topological differences between the original and the replicas, as measured by the Jaccard coefficient on the edge sets. The network and a replica of it are shown in Fig. 4. Generally, large-i editing tends to have higher topological effect and is useful for preserving finer level structures like clustering and degree-degree correlations. In practical applications the appropriate edit rate could be much lower and determined by the application: the edit rate should be high only when the input network is highly different from other networks in its class. The edit rate could be increased to 100%, while still retaining realism, by using the multiple V-cycles capability of MNG. As well, the size of the network could changed to create much larger or much smaller networks than the input data.

To evaluate the performance of MUSKETEER, we generated a large number of networks and compared them to the original network for a variety of local and global structural properties (Fig. 5a). For most properties, the ensemble yields a median value close to the original value and range of

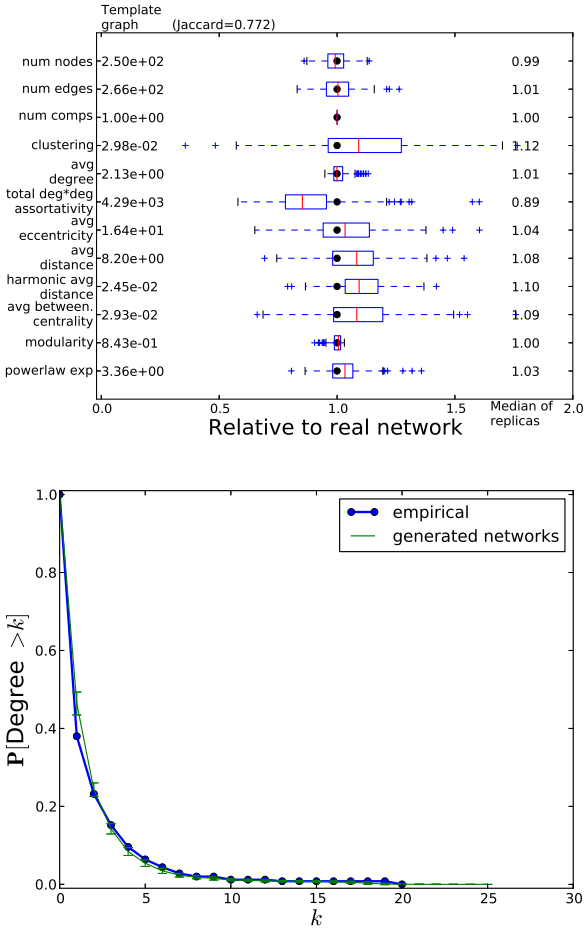

Fig. 5. Performance of MUSKETEER on a sexual contact network (Fig.2A in [41]). (a) Structural properties of 150 generated networks (boxplots) compared with the original network (black dots). All properties of the replicas are normalized so that the value in the original network is mapped to 1.0. The standard boxplots indicate the middle 50 percent of the data (from the 25th to 75th percentile), and whiskers extend to points in either direction whose values are 50% greater than the range of values inside the box (interquartile range). Generally, the replicated networks reflect the properties of the original network with unbiased variation around the median. (b) Average cumulative degree distribution of generated and original networks. Average degree distribution and one standard deviation across replicas (green line and error bars) are compared to original network (blue lines and points).

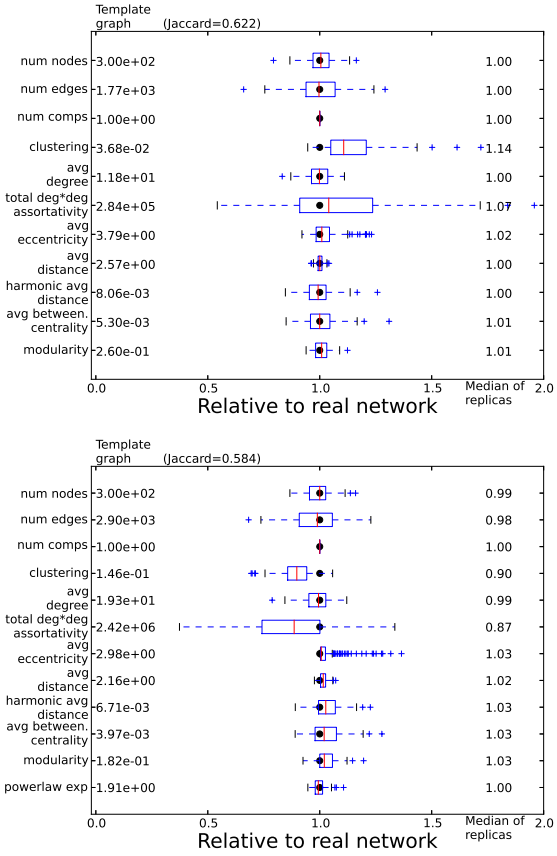

Fig. 6. Comparison of two network models with 150 replicas (top) Erdős-Rényi Graph. (bottom) Barábasi Albert Scale-Free Graph. Lines show the properties of individual networks (black dots: original, boxplot: replicas). In most properties the replicated networks are distributed about the original with desirable variation. There is little or no bias in the properties of the replicas despite the fact that $38-42\%$ of the edges in those networks are new. The generation parameters are $p=0.05$ for ER and $m=10$ in BA and edit rates were 8% for nodes and edges at level $i=4$.

values that is fairly symmetric about the median. The degree distribution as a whole is preserved with slight variation across the generated networks (Fig. 5b).

We applied MUSKETEER to a variety of networks from application domains as diverse as power systems, sociology and biology with positive results (not shown here due to lack of space). As an illustration of the versatility of the algorithm across domains, we applied it to synthesize ensembles of networks starting from 300-node networks generated under two well-understood random graph models: the Erdős-Rényi (ER) model, which yields simple random networks with binomial degree distributions (Poisson in the limit of large networks) [14], and the Barabási-Albert (BA) preferential attachment model, which yields scale free networks, with power law degree distributions [15]. As with the sexual contact network, MUSKETEER produces network ensembles with high fidelity and unbiased variability (Fig. 6).

Synthetic networks should not only reflect realistic structures at multiple scales but also capture emergent properties such as expressing the dynamics or evolution of the system. In the case of the sexual contact network (Fig. 4), we ask whether the replicas produced by MUSKETEER give rise to similar epidemiological dynamics, in terms of the size and timing of a disease outbreak simulated on the network. We compare the MUSKETEER replicas to networks generated by using the configuration model [42] preserving the degree distribution and the BA scale free model [15] calibrated to preserve the density of the network. Both models have been used extensively to model epidemiological processes [5], [43]. Specifically, we simulate a susceptible-exposed-infected-

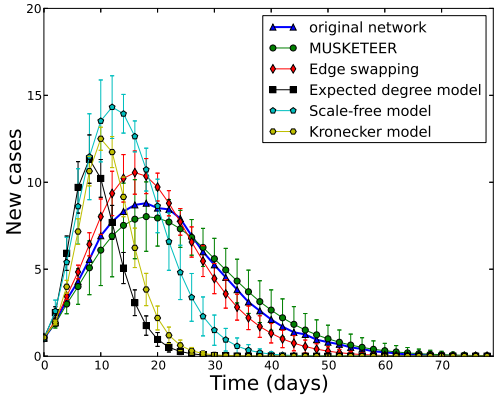

Fig. 7. SEIR epidemics on the original and replica Colorado Springs sexual contact network. Network shows average and one standard deviation (error bars) in incidence on each day of the simulated epidemics, across 1000 runs on the original network and, for each class of synthetic network, 1000 runs on each of 150 replicas (a total of $150,000$ runs per class). Replica networks were generated using MUSKETEER, the configuration model preserving the original degree sequence [42], and the BA scale free model [15].

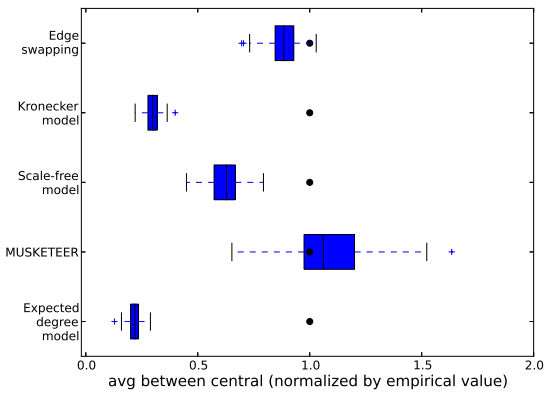

Fig. 8. Average betweenness centrality in networks generated by different strategies. Realism is measured by comparing to the Colorado Springs network ([41]) The average centrality appears to be correctly produced by MUSKETEER, but not alternative approaches.

recovered (SEIR) model [4] on the original and replica networks. For simplicity, we use discrete time steps of 1 day, and infected individuals have a latent period of 2 days (± 1) followed by an infectious period of 9 days (± 1). An infectee has 50% chance per day of infecting each susceptible neighbor.

Another important emergent property of many networks, including epidemiological networks, is the betweenness centrality (BC) of nodes. A node's BC correlates with its likelihood of playing an important role in the network, or of contracting an infection [43]. MUSKETEER appears to be able to generate networks that on average have the appropriate average betweenness centrality (Fig. 8), while alternative approaches produce biased betweenness centrality values. Low average centrality might reflect networks with a many leaf nodes and many hubs, or multiple cycles of equal length.

Theoretically, the running time of MNG scales as $O(m)$, where m is the number of edges of the original graph. Our implementation of MNG, MUSKETEER, uses the functionality of the Python programming language (version 2.7.3) and the NetworkX package [44] (version 1.8.1) both of which sacrifice running time for ease of implementation. We benchmarked the algorithm on a 1GHz CPU (U520 quadcore) with 3.8GB of physical memory. A network of 30000 edges can be generated in less than a minute even under a very high edit rate (such as 10% over several scales); With over $110,000$ nodes, a network can be replicated in under 1 minute. Extensive tuning options are provided that can improve both the accuracy and running time for a specific network modeling problem.

V. Discussion

In this study we suggested a novel practical strategy to model network structures termed multiscale network generation (MNG). The basic step in MNG is to apply multiscale perturbation to the original network, which makes it highly suitable for such needs as verification and simulation of different methods on networks similar to real examples. Unlike network generation approaches that generate a synthetic network starting with an empty network, networks synthesized with MNG retain a large number of latent properties of the input network across multiple scales. This is a key advantage over existing approaches which attempt to imitate a small number of predefined structural properties but cannot efficiently reproduce other properties. In the case of emergent properties, such as epidemic transmission, we saw concretely that the MUSKETEER replicas achieve greater realism in terms of epidemic peak timing than commonly-used existing methods. This finding is attributable to the ability of MNG to avoid establishing bridges, i.e. connections across distant parts of the network that accelerate the breakout of the epidemic.

Overall, the MNG framework has several attractive characteristics as a strategy for modeling networks. First, it allows to control the magnitude of the changes to the input data. Therefore the diversity of the output network ensemble could be both close to and distant from the input as required by the demands on the synthetic data. This means that MNG also accurately preserves many types of annotations, such as node category or edge type found in the original data. Second, MNG can preserve and control the properties of the generated networks at multiple scales. We observed experimentally that *local editing* inherent in MNG preserves many network properties including the different centrality measures, modularity and clustering. Clearly, in the applications with concrete properties, this method will raise theoretical questions of guaranteeing property-friendly network editing. In practice many properties can be preserved across the entire hierarchy by measuring them at all levels during the aggregation and then correcting the network at the disaggregation stage of MNG.

VI. Conclusions

This paper introduced the multiscale network generation (MNG) framework to synthesizing ensembles of networks with realistic properties and the MUSKETEER implementation of MNG. In our evaluation, MUSKETEER was able to reproduce all local and global network properties considered without a priori specification of those properties. Thus, it can be applied right out of the box to generate networks across diverse domains. A software implementing the framework, including the source code, is available for free download [30].

References

[1] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 2, 2006.

[2] D. Dunlavy, B. Hendrickson, and T. Kolda, "Mathematical challenges in cybersecurity," Sandia National Laboratory, Tech. Rep., 2009.

[3] J. M. Brase and D. L. Brown, "Modeling, simulation and analysis of complex networked systems - a program plan," *Lawrence Livermore National Laboratory*, 2009.

[4] M. Keeling and P. Rohani, *Modeling infectious diseases in humans and animals*. Princeton University Press, 2008.

[5] L. Meyers, B. Pourbohloul, M. Newman, D. Skowronski, and R. Brunham, "Network theory and sars: predicting outbreak diversity," *Journal of theoretical biology*, vol. 232, no. 1, pp. 71–81, 2005.

[6] P. Svenson, "Complex networks and social network analysis in information fusion," in *Information Fusion, 2006 9th International Conference on*. IEEE, 2006, pp. 1–7.

[7] B. A. Miller, M. S. Beard, and N. T. Bliss, "Eigenspace analysis for threat detection in social networks," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*. IEEE, 2011, pp. 1–7.

[8] N. Baumann and S. Stiller, "Network models," in *Network Analysis*, ser. Lecture Notes in Computer Science, U. Brandes and T. Erlebach, Eds. Springer Berlin Heidelberg, 2005, vol. 3418, pp. 341–372.

[9] M. Kaufmann and K. Zweig, "Modeling and designing real–world networks," in *Algorithmics of Large and Complex Networks*. Springer, 2009, pp. 359–379.

[10] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, 2002. [Online]. Available: http://dx.doi.org/10.1103/RevModPhys.74.47

[11] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167–256, 2003.

[12] S. Bansal, S. Khandelwal, and L. Meyers, "Exploring biological network structure with clustered random networks," *BMC Bioinformatics*, vol. 10, no. 1, p. 405, 2009.

[13] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, "An introduction to exponential random graph (p*) models for social networks," *Social Networks*, vol. 29, no. 2, pp. 173–191, 2007.

[14] P. Erdos and A. Renyi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, pp. 17–61, 1960.

[15] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, October 1999. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/10521342

[16] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proceeding of the 17th international conference on World Wide Web*. ACM, 2008, pp. 695–704.

[17] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440–442, 1998.

[18] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in internet topologies," Boston University, Boston, MA, USA, Tech. Rep., 2000.

[19] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *The Journal of Machine Learning Research*, vol. 11, pp. 985–1042, 2010.

[20] G. Palla, L. Lovász, and T. Vicsek, "Multifractal network generator," *Proceedings of the National Academy of Sciences*, vol. 107, no. 17, p. 7640, 2010.

[21] C. Mihail and E. Zegura, "The Markov chain simulation method for generating connected power law random graphs," in *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments*, vol. 111. Society for Industrial Mathematics, 2003, p. 16.

[22] S. Itzkovitz, R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, and U. Alon, "Coarse-graining and self-dissimilarity of complex networks," *Physical Review E*, vol. 71, no. 1, p. 016127, 2005.

[23] E. Mones, L. Vicsek, and T. Vicsek, "Hierarchy measure for complex networks," *PLoS ONE*, vol. 7, no. 3, p. e33799, 03 2012. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0033799

[24] A. Brandt and D. Ron, "Chapter 1 : Multigrid solvers and multilevel optimization strategies," in *Multilevel Optimization and VLSICAD*, J. Cong and J. R. Shinnerl, Eds. Kluwer, 2003.

[25] U. Trottenberg and A. Schuller, *Multigrid*. Academic Press, Inc., 2001.

[26] D. Ron, I. Safro, and A. Brandt, "Relaxation-based coarsening and multiscale graph organization," *Multiscale Modeling & Simulation*, vol. 9, no. 1, pp. 407–423, 2011.

[27] S. Leyffer and I. Safro, "Fast response to infection spread and cyber attacks on large-scale networks," *Journal of Complex Networks*, vol. 1, no. 2, pp. 183–199, 2013.

[28] J. Carlson and J. Doyle, "Complexity and robustness," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. Suppl 1, p. 2538, 2002.

[29] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[30] A. Gutfraind, L. A. Meyers, and I. Safro, "MUSKETEER - multiscale entropic network generator," 2012. [Online]. Available: http://www.cs.clemson.edu/~isafro/musketeer

[31] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, no. 16, pp. 309 – 320, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128600000839

[32] D. E. Drake and S. Hougardy, "A simple approximation algorithm for the weighted matching problem," *Inf. Process. Lett.*, vol. 85, no. 4, pp. 211–213, 2003.

[33] J. Chen and I. Safro, "Algebraic distance on graphs," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3468–3490, 2011.

[34] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and R. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 10, p. P10008, 2008.

[35] E. Volz, J. Miller, A. Galvani, and L. Meyers, "Effects of heterogeneous and clustered contact patterns on infectious disease dynamics," *PLoS Computational Biology*, vol. 7, no. 6, p. e1002042, 2011.

[36] M. Newman, "Mixing patterns in networks," *Physical Review E*, vol. 67, p. 026126, 2003.

[37] J. Doyle, D. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, and W. Willinger, "The "robust yet fragile" nature of the internet," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 41, p. 14497, 2005.

[38] L. Li, D. Alderson, J. Doyle, and W. Willinger, "Towards a theory of scale-free graphs: Definition, properties, and implications," *Internet Mathematics*, vol. 2, no. 4, pp. 431–523, 2005.

[39] A. Galstyan and P. Cohen, "Cascading dynamics in modular networks," *Phys. Rev. E*, vol. 75, no. 3, p. 036109, Mar 2007.

[40] A. Gutfraind, "Optimizing topological cascade resilience based on the structure of terrorist networks," *PLoS ONE*, vol. 5, no. 11, p. e13448, 2010.

[41] J. J. Potterat, L. Phillips-Plummer, S. Q. Muth, R. B. Rothenberg, D. E. Woodhouse, T. S. Maldonado-Long, H. P. Zimmerman, and J. B. Muth, "Risk network structure in the early epidemic phase of hiv transmission in colorado springs," *Sex Transmit Infect*, vol. 78, no. Supplement I, pp. 159–163, 2002.

[42] F. Chung and L. Lu, "Connected components in random graphs with given expected degree sequences," *Annals of Combinatorics*, vol. 6, pp. 125–145, 2002.

[43] M. Newman, *Networks: An Introduction*. New York, NY: Oxford University Press, Inc., 2010.

[44] A. Hagberg, D. Schult, and P. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference SciPy2008*, vol. 836, no. SciPy, 2008, pp. 11–15.