Design, Generation, and Validation of Extreme Scale Power-Law Graphs

Jeremy Kepner$^1$, Siddharth Samsi$^1$, William Arcand$^1$, David Bestor$^1$, Bill Bergeron$^1$, Tim Davis$^2$, Vijay Gadepally$^1$, Michael Houle$^1$, Matthew Hubbell$^1$, Hayden Jananthan$^{1,3}$, Michael Jones$^1$, Anna Klein$^1$, Peter Michaleas$^1$, Roger Pearce$^4$, Lauren Milechin$^1$, Julie Mullen$^1$, Andrew Prout$^1$, Antonio Rosa$^1$, Geoff Sanders$^4$, Charles Yee$^1$, Albert Reuther$^1$

$^1$Massachusetts Institute of Technology, $^2$Texas A&M, $^3$Vanderbilt University, $^4$Lawrence Livermore National Laboratory

Abstract—Massive power-law graphs drive many fields: metagenomics, brain mapping, Internet-of-things, cybersecurity, and sparse machine learning. The development of novel algorithms and systems to process these data requires the design, generation, and validation of enormous graphs with exactly known properties. Such graphs accelerate the proper testing of new algorithms and systems and are a prerequisite for success on real applications. Many random graph generators currently exist that require realizing a graph in order to know its exact properties: number of vertices, number of edges, degree distribution, and number of triangles. Designing graphs using these random graph generators is a time-consuming trial-and-error process. This paper presents a novel approach to using Kronecker products to allow the exact computation of graph properties prior to graph generation. In addition, when a real graph is desired, it can be generated quickly in memory on a parallel computer with no-interprocessor communication. To test this approach, graphs with $10^{12}$ edges are generated on a 40,000+ core supercomputer in 1 second and exactly agree with those predicted by the theory. In addition, to demonstrate the extensibility of this approach, decetra-scale graphs with up to $10^{30}$ edges are simulated in a few minutes on a laptop.

I. INTRODUCTION

Power-law (or heavy-tail) [1], [2] graphs are found throughout a wide range of applications [3], [4]. In such graphs, there are a small number of vertices with a large number of edges and a large number of vertices with a small number of edges. Specific domains where such graphs are important include genomics [5]–[10], brain mapping [11], computer networks [12]–[15], social media [16], [17], cybersecurity [18], [19], and sparse machine learning [20]–[24].

Many graph processing systems are currently under development. These systems are exploring innovations in algorithms [25]–[35], software architecture [36]–[44], software standards [45]–[49], and parallel computing hardware [50]–[56]. The development of novel algorithms and systems to process these data requires the design, generation, and validation of enormous graphs with known properties. Such graphs accelerate the proper testing of new algorithms and systems and are a prerequisite for success on real applications.

This material is based in part upon work supported by the NSF under grant number DMS-1312831. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Many random graph generators currently exist that require creating a graph in order to know its exact properties, such as the number of vertices, number of edges, degree distribution, and number of triangles. Perhaps the most well-known and scalable power-law graph generator is used in the Graph500.org [57]–[59] and GraphChallenge.org [60]–[62] benchmarks. This generator, often referred to as R-MAT, is based on randomly sampling recursive Kronecker graphs. Other highly scalable graph generators are based on randomly specified degree distributions [63]–[65]. Designing graphs using these random graph generators is an iterative process whereby the graph designer selects the parameters of the graph generator, randomly creates the graph with those parameters, and then measures the desired properties. Such a process places certain natural limits on the ability of the graph designer to explore enormous graphs and know prior to graph generation the exact properties of the graph.

This paper presents a complementary approach using Kronecker products that allows the exact computation of graph properties prior to graph generation. In addition, when a real graph is desired, it can be generated quickly in memory on a parallel computer with no interprocessor communication. The paper begins with a review of the relevant properties of Kronecker products. Next, the types of constituent matrices that are suited for generating power-law graphs are described. Various mathematical properties of power-law Kronecker graphs are then derived. Subsequently, a parallel algorithm for rapidly generating large graphs is provided. A variety of performance results and specific examples of various graphs generated using this approach are presented. Finally, the conclusions and a discussion of further research are given.

II. KRONECKER PRODUCTS

The Kronecker product of two square matrices is defined as follows [66]

$$C = A \otimes B =
\begin{pmatrix}
A(1, 1) \otimes B & A(1, 2) \otimes B & \ldots & A(1, m_A) \otimes B \\
A(2, 1) \otimes B & A(2, 2) \otimes B & \ldots & A(2, m_A) \otimes B \\
\vdots & \vdots & \ddots & \vdots \\
A(m_A, 1) \otimes B & A(m_A, 2) \otimes B & \ldots & A(m_A, m_A) \otimes B
\end{pmatrix}$$

This indicates that the Kronecker product of two graphs $A$ and $B$ is a graph whose nodes are the Cartesian product of the nodes of $A$ and $B$, and whose edges are the Cartesian product of the edges of $A$ and $B$.
where $A$, $B$, and $C$ matrices of scalar values $S$

$$\begin{align*}
A & \in \mathbb{S}^{m_A \times m_A} \\
B & \in \mathbb{S}^{m_B \times m_B} \\
C & \in \mathbb{S}^{m_A \times m_B \times m_A m_B}
\end{align*}$$

More explicitly, the Kronecker product can be written as

$$C((iA-1)m_A+iB,(jA-1)m_A+jB) = A(iA) \otimes B(iB,jB)$$

The element-wise multiply operation $\otimes$ can be a variety of functions so long as the resulting operation obeys the standard rules of element-wise multiplication, such as $0$ being the multiplicative annihilator for any value of $s \in S$

$$0 \otimes s = s \otimes 0 = 0$$

Furthermore, if element-wise multiplication and addition obey the conditions of a semiring [67–69], then the Kronecker product has many of the same desirable properties, such as associativity

$$(A \otimes B) \otimes C = A \otimes (B \otimes C)$$

and element-wise distributivity over addition

$$A \otimes (B + C) = (A \otimes B) + (A \otimes C)$$

Finally, one unique feature of the Kronecker product is its relationship to the matrix product. Specifically, the matrix product of two Kronecker products is equal to the Kronecker product of two matrix products

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$$

where matrix multiply

$$C = AB = A \oplus \cdots B$$

is given by

$$C(i,j) = \bigoplus_k A(i,k) \otimes B(k,j)$$

### III. Generating Power-Law Graphs

Generating graphs is a common operation in a wide range of graph algorithms. Graph generation is used in the testing of graph algorithms, in creating graph templates to match against, and for comparing real graph data with models. Given a graph adjacency matrix $A$, if

$$A(i,j) = 1$$

then there exists an edge going from vertex $i$ to vertex $j$ [70, 71]. Likewise, if

$$A(i,j) = 0$$

then there is no edge from $i$ to $j$. The Kronecker product of two graph adjacency matrices is a convenient, well-defined matrix operation that can be used for generating a wide range of graphs from a few parameters [57, 58]. The relation of the Kronecker product to graphs is easily illustrated in the context of bipartite graphs. Bipartite graphs have two sets of vertices, and every vertex has an edge to the other set of vertices but no edges within its own set of vertices. The Kronecker product of such graphs was first looked at by Weischedel [72], who observed that the Kronecker product of two bipartite graphs resulted in a new graph consisting of two bipartite sub-graphs (see Figure [1]).

The essence of a power-law graph is that it has a degree distribution vector $n(d)$ with non-zero entries that follows the relation

$$n(d) \propto \frac{1}{d^\alpha}$$

where $d$ is the number of edges in the vertex of a graph, $n(d)$ is the number vertices with a specific degree $d$, and $\alpha > 0$ is the slope of the power law when it is plotted using logarithmic axes [65]. If the graph is represented as an adjacency matrix, then the degree of a vertex is the number of non-zero (nnz) entries in the corresponding row and column in the matrix.

A star graph is a bipartite graph where one set has only one vertex. Star graphs are always a power-law graph. If a star graph has $m$ vertices, then the number of points in the star is given by

$$\tilde{m} = m - 1$$

with a corresponding degree distribution of

$$n(1) = \tilde{m}$$

$$n(\tilde{m}) = 1$$

which agrees with the power-law relation $\alpha$ given by

$$\alpha = \frac{\log(n(1))}{\log(d_{\text{max}})} = \frac{\log(\tilde{m})}{\log(\tilde{m})} = 1$$

where $d_{\text{max}}$ is the degree of the vertex with the most edges.

The Kronecker product of two star graphs can, under certain conditions, produce another power-law graph. In Figure [1] the
graph of the Kronecker product of two star graphs with \( \hat{m}_A = 5 \) and \( \hat{m}_B = 3 \) has a degree distribution of
\[
\begin{align*}
n(1) &= 15 \\
n(3) &= 5 \\
n(5) &= 3 \\
n(15) &= 1
\end{align*}
\]
which are all points on the curve
\[
n(d) = \frac{15}{d}
\]
The Kronecker product of star graphs can be used to build up extremely large power-law graphs. The degree distributions will follow the power-law relation as long as all of the products of the corresponding \( \hat{m} \) are unique.

It is worth noting that real-world graphs often have approximate power-law distributions when plotted simply, as in this case, or when plotted with logarithmic degree binning, but rarely both. It is possible to use Kronecker products to produce power-law graphs under logarithmic degree binning by placing additional constraints on the values of \( \hat{m} \).

**IV. PROPERTIES OF KRONECKER GRAPHS**

The most powerful feature of Kronecker graphs is that many of their properties can be computed from their constituent matrices without ever having to form the full matrix. It is thus possible to design and analyze extremely large graphs quickly and only actually form the full graph when it is needed.

Let the adjacency matrix of graph \( A \) be constructed by the following Kronecker product
\[
A = \bigotimes_{k=1}^{N_k} A_k
\]
where \( A_k \) are each adjacency matrices of the smaller constituent graphs. The number of vertices in the graphs is equal to the number of rows in \( A \) (or columns since \( A_k \) are square), which can be computed from
\[
m_A = \prod_k m_{A_k}
\]
Likewise, the number of edges in the graph is equal to the number of non-zero entries in \( A \) and is given by
\[
nnz(A) = \prod_k nnz(A_k)
\]
The degree distribution \( n_A(d) \) can be computed from the Kronecker product of the degree distributions \( n_{A_k}(d) \)
\[
n_A(d) = \bigotimes_{k=1}^{N_k} n_{A_k}(d)
\]

**A. Triangles**

The number of vertices, number of edges, and degree distribution are good examples of the core properties of Kronecker products. A more sophisticated example is computing the number of triangles in a graph \([73]-[76]\). Triangles are an important feature of a graph, and counting triangles is a basic property of many graph analysis systems. The total number of triangles in a graph can be computed from the following formula
\[
N_{tri}(A) = \frac{1}{6} I^T (A A \otimes A) I
\]
where \( I \) is a column vector of all 1’s and \( \otimes \) is the element-wise product. The same properties of Kronecker products apply to counting triangles, and the number of triangles can be computed from the component matrices via
\[
N_{tri}(A) = \frac{1}{6} \prod_k I^T (A_k A_k \otimes A_k) I
\]

**B. Case 1: Many Triangles**

Bipartite graphs have no triangles, so the Kronecker product of star graphs will produce a large graph with zero triangles, which can be a useful test case. Fortunately, it is possible to simply modify the \( A_k \) to create a graph with a rich triangle structure. Specifically, if a self-loop is put on the central vertex of the star, the resulting graph will have a large number of triangles. If the central vertex in the star is denoted by vertex \( 1 \), then a self-loop can be created in every constituent graph by setting
\[
A_k(1, 1) = 1
\]
Removal of the self-loop in the final graph is accomplished by setting a single value back to zero
\[
A(1, 1) = 0
\]
The number of vertices is unmodified by the inclusion of the self-loops. The number of edges is computed from the \( A_k \) as before, followed by subtracting 1 from the total to account for the removal of the self-loop
\[
nnz(A) - 1
\]
Likewise, the degree distribution is computed from the $A_k$ as before with the following adjustments
\[ n(d_{\text{max}} - 1) = 1 \]
\[ n(d_{\text{max}}) = 0 \]

The triangle count is computed from the $A_k$ as before with the following correction
\[ N_{\text{tri}}(A) = \frac{1}{2} m_A + 1 \]

Figure 2 (top) shows an example of a graph with 15 triangles produced using this method.

\[ C. \text{ Case 2: Some Triangles} \]

A more modest number of triangles can be generated if one self-loop is put on one of the point vertices of each star, for example by setting
\[ A_k(m_{A_k}, m_{A_k}) = 1 \]

Removal of the self-loop in the final graph is accomplished by setting a single value back to zero
\[ A(m_A, m_A) = 0 \]

The number of vertices is unmodified by the inclusion of the self-loops. The number of edges is computed from the $A_k$ as before, followed by subtracting 1 from the total to account for the removal of the self-loop
\[ \text{nnz}(A) - 1 \]

Likewise, the degree distribution is computed from the $A_k$ as before with the following adjustments
\[ n(2^{N_k} - 1) = 1 \]
\[ n(2^{N_k}) = 0 \]

The triangle count is computed from the $A_k$ as before with the following correction
\[ N_{\text{tri}}(A) = \frac{1}{2} 2^{N_k} + 1 \]

Figure 2 (bottom) shows an example of a graph with 1 triangle produced using this method.

\[ D. \text{ Incidence Matrix} \]

An incidence, or edge, matrix $E$ uses the rows to represent every edge in the graph, and the columns represent every vertex. There are a number of conventions for denoting an edge in an incidence matrix. One such convention is to use two incidence matrices
\[ E_\text{out}(e, i) = 1 \quad \text{and} \quad E_\text{in}(e, j) = 1 \]

Kronecker products can also be used to construct incidence matrices that satisfy the above adjacency matrix equation. Specifically, let $E_{k,\text{out}}$ and $E_{k,\text{in}}$ be incidence matrices corresponding to $A_k$. The incidence matrices can then be constructed by
\[ E_{\text{out}} = \otimes_{k=1}^{N_v} E_{k,\text{out}} \]
\[ E_{\text{in}} = \otimes_{k=1}^{N_v} E_{k,\text{in}} \]

It is worth noting that the order of edges in the incidence matrices is not uniquely determined. Different realizations of an incidence matrix are only equivalent when comparing their resulting adjacency matrices.

\[ V. \text{ Parallel Generation} \]

Kronecker products allow the properties of a graph to be determined in advance, thus avoiding the iterative approach of other methods. Once the desired graph properties have been determined, Kronecker products also allow large graphs to be generated quickly on a parallel processor. The overall approach is to split the constituent matrices into two matrices $B$ and $C$
\[ A = \otimes_{k=1}^{N_v} A_k \]
\[ = \left( \otimes_{k=1}^{N_B} A_k \right) \otimes \left( \otimes_{k=N_B+1}^{N_C} A_k \right) \]
\[ = B \otimes C \]

The matrices $B$ and $C$ are designed so that both can fit in the memory of any one processor. Let the parallel computer have $N_p$ processors, and each processor is given an identifier $p$ [78], [79]. Each processor reads in $B$ and $C$ and extracts the triples of the non-zero element $B$ into three vectors $i, j$, and $s$, each of length $\text{nnz}(B)$. Each processor then selects a $\text{nnz}(B)/N_p$ of the triples $i_p, j_p$, and $s_p$. If the underlying sparse storage of the matrices is compressed sparse columns (CSC), then the minimum value of $j_p$ is subtracted from $j_p$ and a new matrix $B_p$ is formed from these triples. Each processor can then form the submatrix $A_p$ of the overall matrix $A$ via the Kronecker product
\[ A_p = B_p \otimes C \]

The resulting $A_p$ matrices will have the same number of non-zero entries on each processor. In addition, the resulting graph is free of many of the problematic vertices and edges, such as empty vertices and self-loops, that are found in randomly generated graphs. These problematic vertices and edges often require randomly generated graphs to be reindexed before their properties can be computed.
VI. RESULTS

This section presents a variety of scalability results to demonstrate the properties of the proposed Kronecker graph generation method. Figure 3 shows the rate of graph edge generation as a function of the number of processing cores used in the parallel graph generation technique described in the previous section. In this example, B is a 530,400 vertex graph with 13,824,000 edges constructed from the Kronecker product of star graphs with $\hat{m} = \{3, 4, 5, 9, 16\}$. Likewise, C is a 21,074 vertex graph with 82,944 edges constructed from the Kronecker product of star graphs with $\hat{m} = \{81, 256\}$. The Kronecker product of B and C, produces a graph A with 11,177,649,600 vertices and 1,146,617,856,000 edges and zero triangles. This graph construction was run in parallel on a supercomputer consisting of 648 compute nodes, each with at least 64 Xeon processing cores, for a total of 41,472 processing cores. Using the entire system, the trillion edge graph was generated in 1 second.

Computing the degree distribution of the generated graph can be used to verify that a generated graph agrees with the theory. Figure 4 shows the measured and predicted degree distribution of a graph produced using the parallel graph generation technique. In this example, B is a 530,400 vertex graph with 22,160,060 edges constructed from the Kronecker product of star graphs with $\hat{m} = \{3, 4, 5, 9, 16\}$ and self-loops on the central vertices of the stars. Likewise, C is a 21,074 vertex graph with 83,618 edges constructed from the Kronecker product of star graphs with $\hat{m} = \{81, 256\}$ and self-loops on the central vertices of the stars. The Kronecker product of B and C produces a graph A with 11,177,649,600 vertices and 1,853,002,140,758 edges and 6,777,007,252,427 triangles. This calculation confirms that the predicted and measured graph are in exact agreement.

Kronecker products can allow the rapid design of very large graphs suitable for the world’s largest computers. Figures 5 and 6 show the degree distribution for two graphs with over $10^{16}$ edges. Both graphs are generated from star graphs with $\hat{m} = \{3, 4, 5, 9, 16, 25, 81, 256, 625\}$ and $6,997,208,649,600$ vertices. Figure 5 has $1,433,272,320,000,000$ edges and zero triangles, and the degree distribution exactly follows the power-law degree formula. Figure 6 is generated with self-loops on the central vertices producing $2,318,105,678,089,508$ edges, 12,720,651,636,552,426 triangles, with the degree distribution that follows the power-law degree formula with small deviations above and below the line.

Kronecker products can also enable the exact analysis of graphs that are far beyond the scale of any current or planned computing system. Figures 7 shows the degree distribution of a graph with over $10^{30}$ edges. The graph was generated from star graphs with $\hat{m} = \{3, 4, 5, 7, 11, 9, 16, 25, 49, 81, 121, 256, 625, 2401, 14641\}$ and a self-loop on one point vertex of each star. The resulting graph has exactly $144,111,718,793,178,936,483,840,000$ vertices, 2,705,963,586,782,877,716,483,871,216,764 edges, and 178,940,587 triangles. Most of the points follow the power-law degree line, but there are many points that deviate from this. This degree distribution was computed on a standard laptop computer in a few minutes.

VII. CONCLUSION

Emerging data in metagenomics, brain mapping, Internet-of-things, cybersecurity, and sparse machine learning produce massive power-law graphs and are driving the development of novel algorithms and systems to process these data. The scale and distribution of these data makes validation of graph processing systems a significant challenge. The ability to
Fig. 5. Quadrillion-edge \( (10^{15}) \) power-law Kronecker graph predicted degree distribution. The resulting graph has exactly 6,997,208,649,600 vertices, 1,433,272,320,000,000 edges, and zero triangles.

Fig. 6. Quadrillion-edge \( (10^{15}) \) power-law Kronecker graph predicted degree distribution. The resulting graph has exactly 6,997,208,649,600 vertices, 2,318,105,678,089,508 edges, and 12,720,651,636,552,426 triangles.

Fig. 7. Predicted degree distribution of a decetta-edge \( (10^{30}) \) power-law Kronecker graph. The resulting graph is predicted to have exactly 144,111,718,793,178,936,483,840,000 vertices, 2,705,963,586,782,877,716,483,871,216,764 edges, and 178,940,587,200 triangles.

Create enormous graphs with exactly known properties can significantly accelerate the design, generation, and validation of new graph processing systems. Many current graph generators produce random graphs whose exact properties, such as number of vertices, number of edges, degree distribution, and number of triangles, can only be computed after the graph has been generated. Thus, designing graphs using these random graph generators is a time-consuming trial-and-error process.

Kronecker products of the adjacency matrices of star graphs are a powerful way to create large power-law graphs. The properties of Kronecker products allow many properties of a larger graph to be computed by simply combining the corresponding properties of the constituent matrices. The ability to compute the properties of large graphs using only small graphs allows the graph designer to find these prior to creating the actual graph. Furthermore, real graphs can be created using Kronecker products on a parallel computer with no interprocessor communication. The resulting graphs will have the same number of edges on each processor. In addition, the graph avoids many of the difficulties, such as empty vertices and self-loops, that are found in other graph generators that rely random sampling. These problematic vertices and edges often require randomly generated graphs to be reindexed before their properties can be computed.

To test this approach, graphs with \( 10^{12} \) edges are generated on a 40,000+ core supercomputer in 1 second and exactly agree with those predicted by the theory. In addition, in order to demonstrate the extensibility of this approach, decetta-scale graphs with up to \( 10^{30} \) edges are simulated in a few minutes on laptop. These results indicate that the proposed method can be a powerful tool for enabling the design, generation, and validation of new graph processing systems.

This paper has presented formulas for a number of properties of Kronecker graphs. There are many additional properties that could be computed in future research, such as eigenvectors, iso-parametric ratios, betweenness centrality, and triangle enumeration. The parallel Kronecker graph generator is ideally suited to the GraphBLAS.org software standard and the creation of a high performance version using this standard is a future goal. Finally, the ability to reason about graphs that are beyond any current or planned computer opens up new possibilities for the theoretical study of phenomena on these large graphs.
ACKNOWLEDGMENTS

The authors wish to acknowledge the following individuals for their contributions and support: Alan Edelman, Charles Leisner, Steve Pritchard, Michael Wright, Bob Bond, Dave Martinez, Sterling Foster, Paul Burkhardt, and Victor Roytburd.

REFERENCES

[1] V. Pareto, Manuale di Economia Politica. Societa Editrice, 1906, vol. 13.
[2] G. K. Zipf, The Psycho-Biology of Language. Houghton-Mifflin, 1935.
[3] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” Science, vol. 286, no. 5439, pp. 509–512, 1999.
[4] D. F. Gleich, “PageRank beyond the web,” SIAM Review, vol. 57, no. 3, pp. 321–363, 2015.
[5] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert, “GeneRank: using search engine technology for the analysis of microarray experiments,” BMC Bioinformatics, vol. 6, no. 1, p. 233, 2005.
[6] B. L. Mooney, L. R. Corrales, and A. E. Clark, “Molecular networks: An integrated graph theoretic and mining tool to explore solvent organization in molecular simulation,” Journal of Computational Chemistry, vol. 33, no. 8, pp. 853–860, 2012.
[7] D. Polychronopoulos, D. Sellis, and Y. Almirantis, “Conserved noncoding elements follow power-law-like distributions in several genomes as a result of genome dynamics,” PLoS one, vol. 9, no. 5, p. e95437, 2014.
[8] S. Dodson, D. O. Ricke, and J. Kepner, “Genetic sequence matching using D4M big data approaches,” in High Performance Extreme Computing Conference (HPEC). IEEE, 2014.
[9] S. Dodson, D. O. Ricke, J. Kepner, N. Chiur, and A. Shcherbina, “Rapid sequence identification of potential pathogens using techniques from sparse linear algebra,” in Symposium on Technologies for Homeland Security. IEEE, 2015.
[10] N. Gouda, Y. Shiwa, M. Akashi, H. Yoshikawa, K. Kasahara, and M. Furusawa, “Distribution of human single-nucleotide polymorphisms is approximated by the power law and represents a fractal structure,” Genes to Cells, vol. 21, no. 5, pp. 396–407, 2016.
[11] A. Fornito, “Graph theoretic analysis of human brain networks,” JMRI Techniques and Protocols, pp. 283–314, 2016.
[12] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” Computer Networks and ISDN Systems, vol. 30, no. 1, pp. 107–117, 1998.
[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos, “On power-law relationships of the internet topology,” in ACM SIGCOMM Computer Communication Review, vol. 29.4. ACM, 1999, pp. 251–262.
[14] G. Yan, G. Tsekenis, B. Barzel, J.-J. Slotine, Y.-Y. Liu, and A.-L. Barabási, “Spectrum of controlling and observing complex networks,” Nature Physics, vol. 11, no. 9, pp. 779–786, 2015.
[15] R. Fontugne, P. Abry, K. Fukuda, D. Veitch, K. Cho, P. Borgnat, and H. Wendt, “Scaling in internet traffic: a 14 year and 3 day longitudinal study, with multiscale analyses and random projections,” IEEE/ACM Transactions on Networking, 2017.
[16] M. Zuckerburg, “Facebook and computer science,” Harvard University CS50 guest lecture, Dec. 7 2005.
[17] H. Kwak, C. Lee, H. Park, and S. Moon, “What is Twitter, a social network or a news media?” in Proceedings of the 19th International Conference on World Wide Web. ACM, 2010, pp. 591–600.
[18] S. Shao, X. Huang, H. E. Stanley, and S. Havlin, “Percolation of localized attack on complex networks,” New Journal of Physics, vol. 17, no. 2, p. 023049, 2015.
[19] S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojevonic, “Malware propagation in large-scale networks,” IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 1, pp. 170–179, 2015.
[20] H. Lee, C. Ekanadham, and A. Y. Ng, “Sparse deep belief net model for visual area v2,” in Advances in neural information processing systems, 2008, pp. 873–880.
[21] M. Ranzato, Y.-I. Boureau, and Y. L. Cun, “Sparse feature learning for deep belief networks,” in Advances in neural information processing systems, 2008, pp. 1185–1192.
[22] X. Ghorb, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in Aistats, vol. 15, no. 106, 2011, p. 275.
