Brief Announcement: Parallel Transitive Closure Within 3D Crosspoint Memory

Alvaro Velasquez

Department of Computer Science
University of Central Florida
velasquez@cs.ucf.edu

Sumit Kumar Jha

Department of Computer Science
University of Central Florida
jha@cs.ucf.edu

ABSTRACT

The infamous memory-processor bottleneck has motivated the search for logic-in-memory architectures. In this paper, we demonstrate how the transitive closure problem can be solved through in-memory computing within a 3D crosspoint memory. The proposed architecture requires only two layers of 1-diode 1-resistor (1D1R) interconnects and external feedback loops.

ACM Reference Format:

Alvaro Velasquez and Sumit Kumar Jha. 2018. Brief Announcement: Parallel Transitive Closure Within 3D Crosspoint Memory. In *SPAA '18: 30th ACM Symposium on Parallelism in Algorithms and Architectures, July 16–18, 2018, Vienna, Austria*. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3210377.3210657>

1 INTRODUCTION

Performance in traditional von Neumann architectures is increasingly being hindered due to the memory wall [1] that arises when the memory and the processing units are segregated. Indeed, a study by NVIDIA [2] demonstrated that a 64-bit arithmetic operation consumes 20 pJ of energy as opposed to the 16 nJ consumed during DRAM read/write operations. This significant gap in energy is due to the distance that must be traversed in order to fetch data from memory. Due to these overheads, there has been a push towards in-memory computing in recent years.

In this paper, we focus on an in-memory solution to the transitive closure problem, which has been studied extensively within graph theory and has ubiquitous applications in many different areas. In the theoretical domain, a classic result of Skym and Valiant demonstrates how problems computable by circuits of polynomial size can be reduced to the transitive closure operation via projections [3]. In more applied domains, this problem underpins many important procedures, such as resolving database queries [4], parsing grammars [5], determining data dependencies at compilation [6], as well as formal verification algorithms such as model checking [7]. Inspired by these applications and motivated by the memory wall problem, we investigate how the inherently parallel flow of information in 3D crosspoint memories can be used to perform graph transitive closure in an efficient manner. In particular, we solve the

transitive closure problem through repeated Boolean matrix multiplication within memory. The method proposed may be suitable for memories similar to Intel's recently unveiled 3D XPoint™ memory architecture [8] – a 3-dimensional crosspoint architecture with two layers of non-volatile memory cells which is orders of magnitude faster than traditional solid-state drives [9]. This technology can already be found on the market in the form of Intel's Optane™ drives [10].

2 3D CROSSPOINT DYNAMICS

Due to advancements in memristor and phase-change memory technologies, as well as concerns over the memory-processor bottleneck, the field of in-memory computing in 2-dimensional memories has garnered much interest in recent years. Many of the proposed digital computing architectures are derived from memristor implication logic [11] and paths-based logic [12] [13]. However, computation within 3D memories has gone largely unexplored.

The computational fabric we adopt in this paper is a 3D crosspoint memory with two layers of cells stacked vertically. These memories consist of sets of parallel wires, with each wire from one set placed perpendicularly to every wire in another set. At every junction of wires, there is a memory cell acting as a switch between the two wires and exhibiting a low-resistance state (LRS) and a high-resistance state (HRS). This is a generalization of modern memristor-based systems and phase change memories, wherein the memory cells have charge-dependent [14] and temperature-dependent [15] resistances, respectively. A selector element is used with each memory cell. This typically consists of a diode, leading to 1-diode 1-resistor (1D1R) structures.

In our approach, the top and bottom rows or nanowires of the crosspoint memory are connected to each other using individual external connections as shown in Fig. 1 (Top). It is mainly in this respect that the proposed architecture differs from existing 3D memories, such as Intel's 3D XPoint™. Definition 2.1 presents an abstraction of the 3D crosspoint architecture.

Definition 2.1. An $n \times n$ 3D crosspoint memory with two externally interconnected layers is a 4-tuple $\mathbb{C} = (M^1, M^2, R, C)$, where

$$\bullet M^k = \begin{pmatrix} m_{11}^k & m_{12}^k & \cdots & m_{1n}^k \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1}^k & m_{n2}^k & \cdots & m_{nn}^k \end{pmatrix} \text{ is a Boolean matrix representing the interconnects with } n \text{ rows and } n \text{ columns. Each } m_{ij}^k \in \{0, 1\} \text{ denotes the state of the device connecting row } i \text{ with column } j \text{ in layer } k \in \{1, 2\}. m_{ij}^k = 1 \text{ denotes a 1D1R device in the low-resistance state (LRS) state and } m_{ij}^k = 0 \text{ denotes the same device in a high-resistance state (HRS).}$$

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SPAA '18, July 16–18, 2018, Vienna, Austria

© 2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5799-9/18/07...\$15.00

<https://doi.org/10.1145/3210377.3210657>

- $R = \{r_1, \dots, r_n\}$ is a vector of row nanowire values and $r_i \in \{0, 1\}$ provides the same input voltage to every component $m_{i1}^1, \dots, m_{in}^1$ in the first layer. External feedback loops connect the bottommost row nanowires with the corresponding topmost nanowires. Hence, it is enough to model both the topmost and bottommost rows or nanowires using the same vector of values. See Fig. 1 for an illustrative example.
- $C = \{c_1, \dots, c_n\}$ is a vector of column nanowire values and $c_j \in \{0, 1\}$ provides the same input to every component $m_{1j}^2, \dots, m_{nj}^2$ in the second layer of the 3D crosspoint memory.

Figure 1: (Top) Illustration of a 3D crosspoint memory $\mathbb{C} = (M^1, M^2, R, C)$ with two layers of interconnects and external feedback loops. The dashed arcs denote an external interconnection between the corresponding bottom and top nanowires. **(Bottom)** Given $X = ((1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (1, 0, 0, 1))$, we configure the crosspoint memory in accordance with Theorem 3.1 so that $M^1 = X$ and $M^2 = X^T$. In order to compute the 4th row vector of the transitive closure of X , a voltage bias is applied to r_4 and r_1, r_2, r_3 are grounded. Note that the voltage readings obtained from (r_1, r_2, r_3, r_4) will correspond to $(1, 0, 0, 1)$, as expected.

In Definition 2.1, a nanowire value of 1 denotes a high voltage value (with respect to ground) and 0 denotes a negligible voltage value. The LRS state is analogous to a closed switch, i.e. a component that allows current to flow from one terminal to the next. Conversely, the HRS state functions as an open switch; namely, it impedes the flow

of current. Since each 1D1R component contains a diode, current will only flow in one direction when said component is in the LRS state. Without loss of generality, we assume that m_{ij}^1 and m_{ij}^2 allow current to flow from wire r_i to c_j and c_j to r_i , respectively. For the remainder of this paper, we use the terms interconnect, component, and device interchangeably.

Suppose we have a 3D crosspoint memory $\mathbb{C} = (M^1, M^2, R, C)$. In order to induce a flow of current through the memory, a voltage bias is applied to some row wire r_i and other wire(s) are grounded. The trajectory of current will depend on the configuration of the interconnect matrices M^1, M^2 . Due to the unidirectional flow of current imposed by 1D1R interconnects, current will flow from wire r_i to c_j if m_{ij}^1 is in the LRS state and wire r_i has a high voltage with respect to ground (i.e. $m_{ij}^1 = 1$ and $r_i = 1$). Similarly, current will flow from c_j to r_i when $m_{ij}^2 = 1$ and $c_j = 1$. We can thus define the outputs of components m_{ij}^1 and m_{ij}^2 by $r_i \wedge m_{ij}^1$ and $c_j \wedge m_{ij}^2$, respectively. We can now define the values of r_i and c_j by equations (1) and (2), where $\mathcal{I}_i = 1$ if a voltage source is applied to wire r_i and $\mathcal{I}_i = 0$ denotes a grounded wire.

$$c_j = \bigvee_{i=1}^n (r_i \wedge m_{ij}^1) \quad (1)$$

$$r_i = \mathcal{I}_i \vee \bigvee_{j=1}^n (c_j \wedge m_{ij}^2) \quad (2)$$

To capture the dynamics of the feedback loop connecting the top and bottom row wires, we model the flow of information in the crosspoint memory through a discrete notion of time. Let $r_{i,t}$ and $c_{j,t}$ denote the state of wires r_i and c_j during the t^{th} iteration of the feedback loop. At time $t = 0$, we have $r_{i,0} = \mathcal{I}_i = 1$ if a voltage source is applied to wire r_i and $r_{i,0} = \mathcal{I}_i = 0$ if the wire is grounded. The values of the wires then evolve from $t = 1$ onwards according to equations (3) and (4).

$$c_{j,t} = \bigvee_{i=1}^n (r_{i,t-1} \wedge m_{ij}^1) \quad (3)$$

$$r_{i,t} = \mathcal{I}_i \vee \bigvee_{j=1}^n (c_{j,t} \wedge m_{ij}^2) \quad (4)$$

Equation (3) suggests that column j has a flow of current at time t if and only if a row nanowire r_i has a flow at time $t - 1$ and the corresponding component m_{ij}^1 is in the LRS state in the first layer of the memory. Similarly, equation (4) describes the fact that row r_i has a new flow of current at time t if and only if a column nanowire c_j has a flow at time t and the corresponding component m_{ij}^2 is in the LRS state in the second layer. While the state of the flows in the nanowires changes as time evolves, the states of the interconnects are set at time $t = 0$ and do not evolve. Essentially, the interconnects store the data values in the crosspoint memory while the interconnecting nanowires enable the desired computation in our approach.

3 TRANSITIVE CLOSURE

We solve the transitive closure problem through repeated Boolean matrix multiplication using in-memory computing. The mapping

proposed may be suitable for memories similar to Intel's recently unveiled 3D XPointTM memory architecture [8], provided that feedback loops are added externally (see Fig. 1). Given a graph $G = (V, E)$, where V is the set of vertices in the graph and $E \in \{0, 1\}^{|V| \times |V|}$ is its adjacency matrix, the transitive closure of G consists of the matrix X^* , where $X_{ij}^* = 1$ if $i = j$ or there is a path from v_i to v_j in G , and 0 otherwise.

Let $X^{(k)}$ denote the k -reachability matrix specifying which nodes are reachable by a path of at most k edges. We assume that every node has a path of length 0 to itself, thus $X^{(0)} = I$, where I denotes the identity matrix. $X^{(k)}$ can be defined recursively as $X^{(k)} = (I \vee E)^k = X^{(k-1)}X^{(1)}$. Since the maximum length of a path in G is $|V| - 1$, we have $X^* = X^{|V|-1} = (I \vee E)^{|V|-1}$.

The preceding argument implies that the transitive closure of a graph can be computed using repeated matrix multiplication. While efforts have been made to multiply two Boolean matrices within memory [16], only a single matrix could be stored in memory. An attempt to multiply a set of Boolean matrices within memory has also been presented in [17]. However, the method proposed therein requires a layer of interconnects for each matrix to be multiplied. Thus, given an arbitrary graph $G = (V, E)$, the number of layers required is $O(|V|)$ as opposed to the two layers required by our approach. This is significant not just from a complexity standpoint, but from a practical perspective as well.

Theorem 3.1 below demonstrates how repeated Boolean matrix multiplication can be efficiently computed in an $n \times n$ 3D crosspoint memory with feedback loops. The values of the bits stored at the crosspoints are fixed during initialization and do not change for the entirety of the procedure. See Fig. 1 (Bottom) for a simple example.

THEOREM 3.1. *Let $\mathbb{C} = (M^1, M^2, R, C)$ be an $n \times n$ 3D crosspoint memory, X denote a Boolean matrix, I be the $n \times n$ identity matrix, and $\gamma \in \{1, \dots, n\}$ be a row index. If $M^1 = X$, $M^2 = X^T$ and $(r_{10}, \dots, r_{n0}) = (I_{\gamma 1}, \dots, I_{\gamma n})$, then $r_{it} = (X^{2t})_{\gamma i}$ and $c_{jt} = (X^{2t-1})_{\gamma j}$ at any given time index t .*

PROOF. The proof is by induction on t . When $t = 1$,

$$\begin{aligned} r_{i1} &= \bigvee_{j=1}^n (c_{j1} \wedge m_{ij}^2) = \bigvee_{j=1}^n \left(\bigvee_{k=1}^n (r_{k0} \wedge m_{kj}^1) \wedge m_{ij}^2 \right) \\ &= \bigvee_{j=1}^n (m_{\gamma j}^1 \wedge m_{ij}^2) = \bigvee_{j=1}^n (X_{\gamma j} \wedge X_{ji}) = (X^2)_{\gamma i} \end{aligned}$$

Assume that $r_{it} = (X^{2t})_{\gamma i}$. We compute $r_{i,t+1}$ as follows.

$$\begin{aligned} r_{i,t+1} &= \bigvee_{j=1}^n (c_{j,t+1} \wedge m_{ij}^2) = \bigvee_{j=1}^n \left(\bigvee_{k=1}^n (r_{kt} \wedge m_{kj}^1) \wedge m_{ij}^2 \right) \\ &= \bigvee_{j=1}^n \left(\bigvee_{k=1}^n ((X^{2t})_{\gamma k} \wedge X_{kj}) \wedge X_{ji} \right) \\ &= \bigvee_{j=1}^n ((X^{2t+1})_{\gamma j} \wedge X_{ji}) = (X^{2t+2})_{\gamma i} \end{aligned}$$

The proof of $c_{jt} = (X^{2t-1})_{\gamma j}$ is similar and therefore omitted. $\square$

From Theorem 3.1, we arrive at the transitive closure of a graph G using an in-memory computing architecture that can implement only repeated Boolean matrix multiplication. Repeating this procedure for each $\gamma \in \{1, \dots, n\}$ yields the n vectors of the transitive closure matrix X^* . We illustrate this with the simple example below, where the matrices $X^{(1)}, \dots, X^{(4)}$ are reachability matrices. Clearly, the transitive closure matrix is $X^* = X^{(4)}$. A visualization of how X^* is computed in a 5×5 crosspoint memory can be seen in Fig. 2.

$$\begin{aligned} X^{(1)} &= \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, & X^{(2)} &= \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \\ X^{(3)} &= \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, & X^{(4)} &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

It follows from Theorem 3.1 that, given a 3D crosspoint memory $\mathbb{C} = (M^1, M^2, R, C)$, $(r_1, \dots, r_n)$ and $(c_1, \dots, c_n)$ will converge on $(X_{\gamma 1}^*, \dots, X_{\gamma n}^*)$ when the wire vector $(r_{10}, \dots, r_{n0})$ is configured according to $(I_{\gamma 1}, \dots, I_{\gamma n})$. This convergence holds independently of timing errors, as is evidenced by equation (5), which is an expansion of (4). This provides a definition for r_{it} that is independent of time.

$$\begin{aligned} r_{it} &= \bigvee_{j_1=1}^n \left(\bigvee_{i_1=1}^n \dots \left(\bigvee_{j_t=1}^n \left(\bigvee_{i_t=1}^n (r_{i_t,0} \wedge m_{i_t j_t}^1) \dots \right) \right) \right) \\ &= \bigvee_{j_1=1}^n \bigvee_{i_1=1}^n \dots \bigvee_{j_t=1}^n (m_{\gamma j_t}^1 \wedge \dots \wedge m_{i_1 j_1}^1 \wedge m_{j_1}^2) \end{aligned} \quad (5)$$

Note from (5) that r_{it} holds if and only if there is some sequence of interconnects in the LRS state. This assertion is independent of differing delays in wires or interconnects. That is, wire r_i will eventually have a high voltage with respect to ground as long as the right sequence of interconnects is in the LRS state. It follows that delays arising from process variations or external factors can be accounted for by increasing the time interval before reading the outputs of the system. Thus, no additional complexity or circuitry need be introduced in order to deal with such timing errors.

4 COMPLEXITY

Given a graph $G = (V, E)$ with $|V| = n$ and a crossbar $\mathbb{C} = (M^1, M^2, R, C)$, the mapping proposed in Theorem 3.1 uses $O(n^2)$ devices. Since the maximum length of a path is $n - 1$, we need only compute $(I \vee E)^{n-1}$. It follows from Theorem 3.1 that, if $M^1 = (I \vee E)$ and $M^2 = (I \vee E)^T$, then $c_{j, n/2} = (I \vee E)_{\gamma j}^{2(n/2)-1} = (I \vee E)_{\gamma j}^{n-1}$. In the case of transitive closure, this computation must be done for every $\gamma \in \{1, \dots, n\}$. Thus we have a time complexity of $O(n^2)$ for the transitive closure problem. The complexity becomes $O(n \cdot \text{diam}(G))$ if the diameter of the graph is known. The single-source variant of this problem can be solved in $O(n)$ operations since we only need to compute values for exactly one $\gamma \in \{1, \dots, n\}$ corresponding to the source vertex. As a point of comparison, the in-memory approach in [17] achieves the same runtimes of $O(n)$ and $O(n^2)$

Figure 2: Procedure for computing the transitive closure X^* of directed graph $G = ((v_1, v_2, v_3, v_4, v_5), E)$ (Top). The two-layer crosspoint memory with feedback loop $C = (M^1, M^2, (r_1, r_2, r_3, r_4, r_5), (c_1, c_2, c_3, c_4, c_5))$ (Bottom) is unrolled in order to help visualize the computation. Dashed lines represent interconnections between row and column wires, yellow bars denote a wire with a truth value of 1, and solid red lines correspond to interconnects redirecting the flow of current from one wire to another. In this case, we are computing the second row vector $(X_{21}^*, \dots, X_{25}^*)$ of X^* by setting $(r_{10}, r_{20}, r_{30}, r_{40}, r_{50}) = (0, 1, 0, 0, 0)$. Note that $(c_{12}, c_{22}, c_{32}, c_{42}, c_{52}) = (X_{21}^*, X_{22}^*, X_{23}^*, X_{24}^*, X_{25}^*)$.

for single-source reachability and transitive closure, but requires $O(n^3)$ components as opposed to the $O(n^2)$ memory cells used by our method.

It is worth noting that the runtime complexity of our approach, and of the transitive closure operation in general, can be reduced significantly if the underlying graph structure is exploited. Take, for example, the scale-free graphs created by the popular procedure wherein a node is connected to an existing node with a probability that is proportional to the node degree. Simulations [18] and heuristic studies [19] have demonstrated that the diameter of such graphs is approximately $\log n$, where n is the number of nodes. This result also holds for small-world networks [20]. A simple modification to the graph construction model in [21] can significantly change the diameter of the resulting graphs. Indeed, it has been shown that, when a node is randomly connected to two or more existing nodes using degree-proportional probability, the resulting graph has a diameter of asymptotically $\frac{\log n}{\log \log n}$ [22]. For the two cases mentioned, our approach has an expected runtime of $O(n \log n)$ and $O(\frac{n \log n}{\log \log n})$, respectively, using $O(n^2)$ components.

5 CONCLUSIONS

We have presented a new in-memory computing design for obtaining the transitive closure of a graph using a 3D crosspoint memory

with two layers of interconnects. Our proposed design only requires the addition of external feedback loops to already existing 3D memories.

ACKNOWLEDGEMENT

The authors would like to thank the US Air Force for support provided through the AFOSR Young Investigator Award to Sumit Jha and the National Science Foundation for support provided to Alvaro Velasquez through the NSF Graduate Research Fellowship. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0255.

REFERENCES

- [1] Andreas Nowatzky, Fong Pong, and Ashley Saulsbury. Missing the memory wall: The case for processor/memory integration. In *Computer Architecture, 1996 23rd Annual International Symposium on*, pages 90–90. IEEE, 1996.
- [2] Bill Dally. Challenges for future computing systems. <https://www.hipecac.net/events/activities/7151/challenges-for-future-computing-systems/>, January 19 2015. HiPEAC 2015 Keynote speech.
- [3] Sven Skyum and Leslie G Valiant. A complexity theory based on boolean algebra. *Journal of the ACM (JACM)*, 32(2):484–502, 1985.
- [4] Juan L Reutter, Miguel Romero, and Moshe Y Vardi. Regular queries on graph databases. In *18th International Conference on Database Theory (ICDT 2015)*, volume 31, pages 177–194. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.
- [5] Leslie G Valiant. General context-free recognition in less than cubic time. *Journal of computer and system sciences*, 10(2):308–315, 1975.
- [6] Dick Grune, Kees Van Reeuwijk, Henri E Bal, Criel JH Jacobs, and Koen Langendoen. *Modern compiler design*. Springer Science & Business Media, 2012.
- [7] Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. Regular model checking. In *Computer Aided Verification*, pages 403–418. Springer, 2000.
- [8] Intel PR. Intel and micron produce breakthrough memory technology. http://newsroom.intel.com/community/intel_newsroom/blog/2015/07/28/intel-and-micron-produce-breakthrough-memory-technology. Accessed: 28 July, 2015.
- [9] Katherine Bourzac. Has intel created a universal memory technology?[news]. *IEEE Spectrum*, 54(5):9–10, 2017.
- [10] Frank T Hady, Annie Foong, Bryan Veal, and Dan Williams. Platform storage performance with 3d xpoint technology. *Proceedings of the IEEE*, 2017.
- [11] P Kuekes. Material implication: digital logic with memristors. In *Memristor and memristive systems symposium*, volume 21, 2008.
- [12] Alvaro Velasquez and Sumit Kumar Jha. Automated synthesis of crossbars for nanoscale computing using formal methods. In *Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on*, pages 130–136. IEEE, 2015.
- [13] Alvaro Velasquez and Sumit Kumar Jha. Fault-tolerant in-memory crossbar computing using quantified constraint solving. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 101–108. IEEE, 2015.
- [14] R Williams. How we found the missing memristor. *Spectrum, IEEE*, 45(12):28–35, 2008.
- [15] H-S Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010.
- [16] Alvaro Velasquez and Sumit Kumar Jha. Parallel boolean matrix multiplication in linear time using rectifying memristors. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pages 1874–1877. IEEE, 2016.
- [17] Alvaro Velasquez and Sumit Kumar Jha. Computation of boolean matrix chain products in 3d reram. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017.
- [18] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Internet: Diameter of the world-wide web. *nature*, 401(6749):130, 1999.
- [19] Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Random graphs with arbitrary degree distributions and their applications. *Physical review E*, 64(2):026118, 2001.
- [20] Luis A Nunes Amaral, Antonio Scala, Marc Barthélemy, and H Eugene Stanley. Classes of small-world networks. *Proceedings of the national academy of sciences*, 97(21):11149–11152, 2000.
- [21] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
- [22] Béla Bollobás and Oliver Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.