

Evolution beats random chance: Performance-dependent network evolution for enhanced computational capacity

Manish Yadav^{1,*}, Sudeshna Sinha², and Merten Stender¹

¹Chair of Cyber-Physical Systems in Mechanical Engineering, Technische Universität Berlin, Straße des 17. Juni, Berlin, 10623, Germany

²Department of Physical Sciences, Indian Institute of Science Education and Research Mohali, Sector 81, SAS Nagar, 140306, Punjab, India

*manish.yadav@tu-berlin.de

ABSTRACT

The quest to understand *structure-function* relationships in networks across scientific disciplines has intensified. However, the optimal network architecture remains elusive, particularly for complex information processing. Therefore, we investigate how optimal and specific network structures form to efficiently solve distinct tasks using a novel framework of performance-dependent network evolution, leveraging reservoir computing principles. Our study demonstrates that task-specific minimal network structures obtained through this framework consistently outperform networks generated by alternative growth strategies and Erdős-Rényi random networks. Evolved networks exhibit unexpected sparsity and adhere to scaling laws in node-density space while showcasing a distinctive asymmetry in input and information readout nodes distribution. Consequently, we propose a heuristic for quantifying task complexity from performance-dependently evolved networks, offering valuable insights into the evolutionary dynamics of network *structure-function* relationship. Our findings not only advance the fundamental understanding of process-specific network evolution but also shed light on the design and optimization of complex information processing mechanisms, notably in machine learning.

Keywords: evolving networks, reservoir computing, performance-dependent evolution, enhanced information processing, network structure-function

Introduction

Naturally occurring networks process diverse and complex information with limited resources. Biological networks such as neuronal, protein interaction, and gene regulatory networks (GRNs) are the best examples of large-scale complex systems that have grown and evolved by natural selection over millions of years to *efficiently* process complex biochemical signals¹⁻³. These networks exhibit characteristic structural organizations with distinct features such as scaling laws, resulting in robust and efficient functionality-specific information processing^{1,4}.

The growth and evolution of networks have been a central topic of research in the network science community that has provided several scaling laws and unique graph theoretic properties over the years⁵. For instance, the well-known Price's preferential attachment model introduced in the 1960s⁶ provided an understanding of the power-law degree distribution of growing co-citation networks over time. However, the main research focus has remained around the parameter-dependent self-organization of networks showcasing their statistical growth and edge reorganization^{7,8}. The focus of many disciplines in recent years has shifted to the *structure-function* relationship of large complex networks. For instance, the relationship across distinct regions and connectivity was found to scale in the brain^{1,9-12}, and recurrent structures facilitate the memory-dependent temporal information processing in GRNs of *Escherichia coli*¹³, and in several other evolutionary distant species¹⁴. The adaptive dynamic networks theory is a great step toward understanding the functional relationship of a network to its structural organization¹⁵. This theory covers networks that can change their connectivity over time according to their dynamical state. However, the question of how these large-scale natural networks evolve to achieve specific functionality and what should be their natural structural organization to exhibit a function is still missing a general conceptual understanding and explanation.

A different perspective on this problem comes from artificial neural networks (ANNs), which are inspired by biological networks and developed to have specific functionalities for wide-ranging tasks, such as forecasting dynamics. However, the lack of a formal theory explaining process-specific network *structure-function* relationship leaves the researchers with randomly selecting the size and structures of the artificial networks, a process that is mostly based on a trial-and-error strategy or one's

prior experiences¹⁶. There is no guiding principle for network structure selection that can efficiently solve a given task with given resources. Increasing the network size is often attempted in hopes of eventual task-solving¹⁷. It would be of utmost value if there was some framework in which to gauge “how big is big enough” in a general scenario. More importantly, from the fundamental point of view, it is of immense interest to uncover underlying concepts that can lead to the smallest effective networks for information processing.

Motivated by the importance of the problem across multiple disciplines as well as its valuable applications, we are describing the formation of *function-specific network structures* in this paper. For that, we propose a general framework of process-specific network evolution that will help elucidate not just the network *structure-function* relationship from an evolutionary perspective but also in generating task-specific artificial networks. Inspired by biological systems that can solve various tasks with limited resources, we hypothesize that one can reach functionality-specific optimally sized networks more effectively through evolutionary concepts, rather than by random chance. For that, we propose the idea of a performance-informed growth strategy that lets networks naturally evolve into specific and efficient structures for solving a given task with desired performance. This framework will eventually help in the identification of broad principles to grow networks, starting from a few seed nodes, which can potentially allow the synthesis of small networks with enhanced computational capacity.

The proposed idea of functionality-specific network formation can be implemented with an information-processing framework that operates on any network irrespective of its architecture, number of nodes or edges. Such a framework will facilitate the underlying information-processing network with the freedom to grow, morph and evolve into functionality-specific size and architecture. The Reservoir Computing (RC) framework matches such criteria where input signals are mapped into a higher dimensional structure-free *reservoir*^{18,19}. RC is a generalization of earlier neural network architectures such as echo-state networks and liquid-state machines, derived from recurrent neural networks initially introduced to explain information processing in neuronal networks^{20,21}, and this framework has sparked intense wide-ranging interest in recent years^{22–29}. An RC has 3 layers namely the input, the *reservoir* and the output layer. A dynamic input signal $\mathbf{u}(t)$ is fed into the reservoir, and then a simple readout mechanism is trained to obtain a mapping to the desired output. The dynamics of the *reservoir* is iteratively obtained for time $t + 1$ using the Eq. 1:

$$\mathbf{r}(t + 1) = (1 - \alpha)\mathbf{r}(t) + \alpha L\left(W^{res}\mathbf{r}(t) + W^{in}\mathbf{u}(t)\right) \quad (1)$$

$$\hat{\mathbf{y}}(t) = W^{out}\mathbf{r}(t) \quad (2)$$

where $W^{res} \in \mathbb{R}^{N \times N}$ is the adjacency matrix representing all the connections between (N) reservoir nodes. W^{in} specifies which nodes of the reservoir network receive the time-varying input. Each node’s dynamics are defined by the previous state, external stimuli and nonlinear activation function L . The reservoir matrix $\mathbf{r}(t) \in \mathbb{R}^N$ contains information of the reservoir states at a given time t when it is stimulated by an external signal $\mathbf{u}(t) \in \mathbb{R}^T$. Parameter $\alpha \in [0, 1]$ reflects the reservoir information leakage and determines the amount of the past reservoir information being leaked over time. The RC approximated real-time output $\hat{\mathbf{y}}(t)$ is obtained using a linear combination of the reservoir states given by the Eq. 2. A key feature of a reservoir computer is that only the readout is trained while the reservoir network stays fixed. The output weight matrix W^{out} is trained to minimize the difference between the RC output and the actual output $\mathbf{y}(t)$ which is typically obtained using the Ridge regression, also commonly known as *one-step training*.

Reservoir networks are typically chosen to have a random structure with hundreds to a few thousands of nodes^{18,19}. Since there are no specific rules for selecting the network topology of a reservoir, the RC framework provides an opportunity to study the computational capabilities of self-organizing and growing networks vis-a-vis an Erdős-Renyi random networks employed as a reservoir. Therefore, the general problem of network *structure-function* relationship can be studied via the RC framework which will be beneficial for the research community exploring the information processing behavior of RCs as complex dynamical systems. Moreover, a growing number of research has computationally as well as experimentally proved that the *liquid state machines* or RC paradigm can explain the processing of complex biological computations through recurrences^{14,20,30–33}. This suggests and provides another argument to directly employ a recurrent network-based framework to study how networks grow and evolve to achieve function-specific structures³⁴. Therefore, all the questions that we posed above for understanding the network *structure* dependent *functionality* can be directly translated to that of *reservoir* network. Our central idea is to implement an evolving network model into the RC paradigm and observe the graph-theoretic properties of the reservoir networks as they evolve while solving different tasks. The principal focus will be the following: Is it possible to generate optimally sized networks to solve specific tasks? If yes, what is the comparative size of an evolved network that delivers the same performance as a random reservoir? What are the graph theoretic structural differences between evolved networks and random reservoirs? Are there any unifying underlying features in evolved networks grown for different tasks?

In this work, we propose a novel framework showcasing the functionality-specific network generation via their performance-dependent evolution (see schematic in Fig. 1). We demonstrate that the evolved networks are significantly smaller than their

randomly connected counterparts, suggesting that evolutionary strategies converge to more parsimonious computing structures than random chance. We also suggest a different set of evolution strategies and explore their comparative performances. Further, we uncover scaling relations and graph-theoretic features of the evolved networks marking the formation of optimally sized and computationally efficient network structures across different tasks. These special features have the potential to be used as prescriptive recipes for designing optimal machine-learning algorithms consistent with the scaling relations.

Results

0.1 Performance-dependent network evolution framework

The key idea of generating minimal networks for efficient processing is based on the evolutionary growth of the networks. Such performance-enhancing growth can be achieved by adding nodes to a pre-existing network, as long as they improve the overall information-processing capabilities of the network. Nevertheless, a node improving the performance of the system can also become obsolete in later evolution stages when it no longer serves any useful functionality within the network, e.g. when encoding redundant information processed by other nodes. Therefore, node deletion plays a crucial role, as the presence of redundant nodes makes the entire system inefficient in terms of, for instance, energy, space, and computation time. Therefore, to achieve an evolution-like growth of the networks, the network evolution framework should include a mechanism to continuously *add and delete nodes* to/from a pre-existing network. That framework needs to be embedded into a feedback loop with the specific task to solve, to inform the evolution of the information processing capabilities of the network.

Pursuing this line of thought, we propose a performance-dependent network evolution framework that is provided with *node addition* (A) and *deletion* (D) *modules* interconnected with each other in a feedback loop, schematically shown in Fig. 1(a). The primary function of module A is to add new nodes to the pre-existing network in such a way that will improve the information processing capabilities of the new network. On the contrary, the module D continuously removes the unnecessary nodes from the network. A minimal two-node network G_{s_0} is considered as an initial network at the beginning of the evolutionary iteration s_0 throughout this study. This minimal network reaches the evolution modules A and D initiating the task-specific evolutionary growth. A final evolved network $G_{s_T}^P$ is generated when a desired performance (ϵ^P) for an underlying task P is achieved at iteration $s_i = T$. In a nutshell, the *modules* A and D will orchestrate the growth of the network in such a way that will improve its information processing capabilities by working together to retain only those nodes that assist in enhancing its overall performance while eliminating the ones that do not (Fig. 1(b)).

The network evolved by the performance-dependent network evolution model is considered as the reservoir layer network (W^{res}) of the RC framework at every evolutionary iterative step (s). The reservoir networks will evolve to achieve different functionalities embarked with a range of distinct underlying processes of varying levels of complexity ranging from classical one-to-one mappings, and multidimensional trajectory generation to memory-dependent tasks. In total, we considered seven different tasks: two Sine-Cosine tasks, NARMA-5th, 10th and 15th order tasks³⁵, chaotic trajectory prediction of a prototypical chaotic system and phase-portrait generation of the limit cycle from a nonlinear relaxation oscillator.

The proposed performance-dependent evolution model has been used to solve the aforementioned tasks. The evolution of the size of the networks for different tasks is represented by the number of nodes over evolutionary iterations in Fig. 2(a). Furthermore, the performances as reflected by the Normalized Mean Square Error, denoted by NMSE, of these evolving networks are represented against the number of nodes in Fig. 2(b). The model takes a distinct number of iterations depending on each task (Fig. 2(a)) and stops when the underlying performance (ϵ^P) is achieved. The final evolved networks (represented with different markers) for distinct tasks are different from each other in terms of size, marked by the total number of nodes. It is evident that the Sine-Cosine-1 is the least complex task of mapping the Sine to a Cosine function that can be solved with the least complex networks, some consisting of just 5 nodes. A slightly more complex task of mapping the Sine function to a complex polynomial of Sine and Cosine functions, namely the Sine-Cosine-2 task can also be solved with small networks of just 11 nodes on average.

The networks evolved to solve the input memory-dependent tasks of NARMA-5, 10 and 15th order have average sizes of 20, 55 and 75 respectively. This suggests that the NARMA-type tasks are more complex than Sin-Cosine tasks as they require bigger recurrent networks to achieve specific memory requirements to solve them. Furthermore, it is worth noting that the evolved reservoir networks for NARMA-type tasks are orders of magnitude smaller than typically used, showing that memory-dependent tasks can also be solved by smaller networks depending on the memory requirements. Sample time-series predictions from NARMA and Sine-Cosine tasks from one of the evolved networks are respectively represented in Fig. 2 (c) and (d). NARMA-15 requires the most number of nodes out of all the benchmarks and NARMA tasks which reflects its high complexity. In almost all the cases for the NARMA-15 task, the total network evolutionary iterations limit of the model S^{Evolve} was reached and the final performance was slightly lower than the requisite $\epsilon^{NARMA-15}$ i.e. $E(G_{S^{Evolve}}^{NARMA-15}) \geq \epsilon^{NARMA-15}$. The tasks for multidimensional phase-space trajectory generation have been made more complicated as they have been generated by autonomously working reservoir computers for both tasks. Networks evolved for generating the limit-cycle of the Van der Pol

oscillator contained an average of just 12 nodes while those of the 3D chaotic trajectory of the Lorenz system contained only 27 nodes, resulting trajectories are shown in Fig.2 (e) and (f) respectively.

From this initial study, it is therefore evident that the performance-dependent evolution model demonstrates the existence of small networks with just a few nodes that can efficiently process complex tasks. Another interesting observation is that after starting from the same initial two nodes, the networks evolve to different sizes that are optimally structured to solve specific tasks with different complexity levels, also schematically depicted in Fig.1 (b). Simultaneously, networks evolved to carry out the same process are clustered together with a slight deviation in their sizes. These results are consistent with the observations in naturally evolving networks, such as the presence of distinctly sized and structured brain regions for different functionalities^{1,10}. An example of the latter is the signaling networks responsible for regulating the same cellular processes which are found to be actively present in many different cell types and are conserved over evolutionary scale³⁶. Furthermore, these results are contrary to the common practice of employing randomly connected large networks, typically of size $N \sim \mathcal{O}(10^{2-3})$ to solve complex tasks using reservoir computers^{21,37,38}. Several questions arise from these initial observations: Is it necessary for the network to be grown and contracted continuously to achieve an efficient computational capacity and structure? Is performance-dependency necessary to generate computationally efficient network structures? How are these evolved networks different from the commonly employed randomly connected networks? Is there a scaling law that networks follow while evolving to solve a task?

0.2 Challenging the performance-dependent network evolution with different growth strategies

To answer the aforementioned questions and also, to challenge the results of our performance-based network evolution model, we considered two different network growth strategies. First, to elucidate the requirement of performance dependency for efficient network generation, an uninformed network growth model (N1) is considered as a reference. There will be no feedback to the model for the network performance meaning that the network growth in this model will depend completely on random chance irrespective of any task and performance. To understand the necessity of network growth and contraction a second alternate model is formulated only with growth or the performance-dependent network growth model (N2). The N2 model will only be able to expand a network depending on its performance but will not be able to remove nodes. It can be thought of as a partial network evolution model as it consists only of node *addition module A*. From here onwards N3 will be used to refer to the principal performance-based network evolution model proposed in the Section above.

Starting with the same 2-node initial network, N1 and N2 models were used to grow the networks for all the seven benchmark tasks. The network growth trajectories of the three models N1 (black), N2 (blue) and N3 (pink) for the NARMA-5, 10 and 15th order tasks are displayed together for comparison in Fig.3 (a). The networks generated by the performance-independent N1 model grow linearly over the iterations until the desired performance (ϵ^P) is achieved. In almost all cases, N1 models reach the iteration limit $S^{Evolve} = 500$ for all the three NARMA tasks that were set the same for all the growth models. Also in most cases, the final network did not manage to successfully execute the NARMA tasks (especially the more complex NARMA-15) to the prescribed accuracy. However, the N3 model not only achieved the prescribed accuracy but also generated optimal networks without reaching the maximum iteration limit S^{Evolve} . Similarly, for the Sine-Cosine (Fig.3 (d)), Lorenz's chaotic trajectory and Van der Pol's limit-cycle generation (Fig.3 (g)) tasks, the networks follow a linear growth for the N1 model, while a sublinear growth for the N2 and N3 models. All the networks evolved by the N3 model achieve the desired performance with the least number of iterations consisting of a lesser number of nodes which is consistently observed for all the benchmark tasks. A side-by-side comparison of the distinct network growth strategies (N1 and N2) demonstrates several key aspects of the performance-dependent network evolution (N3). Specifically, examining the final networks generated by the different models for different tasks reveals that the performance-dependent growth strategy generates not only the smaller networks with fewer nodes (Fig. 3 b,e,h) but these evolved networks also exhibit enhanced information processing capabilities reflected by the least error (Fig. 3 c,f,i).

There is a minimal bound to the number of nodes required for a certain task, irrespective of the network structures imposed by different evolution strategies as the initial 2-node network did not manage to solve any of the given tasks. This minimal bound is a proxy for the task complexity, i.e. a universal or task-agnostic quantity to characterize learning tasks which today has not been proposed elsewhere. Furthermore, if a network grows uninformed without any performance-dependent strategy, the number of nodes is expected to increase linearly with iterates. Another interesting observation is that while uninformed node addition (N1), does indeed on average, give rise to linear growth of nodes with iterations, performance-dependent node addition (N2) yields sub-linear growth of nodes. Additionally, the number of nodes needed to achieve a performance level saturates after a certain number of iterates, i.e. one obtains a converged network that does not need to be bigger to solve the task. This implies that not all additions are beneficial to performance, and an emergent network smaller than that obtained by the uninformed growth (N1) is sufficient.

Further, the performance-dependent evolved networks allowing node addition as well as removal (N3) yield the slowest growth of nodes with iterations. Importantly, the number of nodes needed for the achievement of a prescribed level of

performance is significantly lower for networks evolved using a performance-dependent addition and removal strategy (N3), often being an order of magnitude smaller than uninformed growth (N1). In this case, the number of nodes saturates very quickly to a small value. This leads to the important conclusion: these evolved networks can perform tasks efficiently and parsimoniously, with small networks yielding very low errors on a wide range of tasks. Another distinguishing feature is that the spread in the number of nodes of the evolved network is the smallest for the performance-dependent network with node addition and removal. In contrast, if we consider a growing network only with node addition (N2), the variance in the number of nodes of the converged network is very high. This direct comparison of distinct network growth strategies therefore answers that it is indeed necessary for the networks to performance-dependently evolve with continuous growth and contraction to achieve a structure of specific size for enhanced information processing.

These distinct network growth strategies were also compared with the Erdős-Renyi random networks (N0). The size of a random network yielding the same level of performance varies over a much larger range. The average number of nodes needed for successful performance for a random network is markedly more than that of an evolved network. However, though a majority of small random networks will not yield the prescribed performance, since the spread in sizes is large (Fig.3), there exist certain small random networks that can perform efficiently. The search for these special random networks is hard, and there is no over-arching principle to guide the search in the extremely large space of network realizations. This is one of the drawbacks of reservoirs comprised of random networks. This observation leads to a prompt question, why do the majority of the ER random networks fail to efficiently solve the benchmark tasks even after having as many nodes as compared to the performance-dependently evolved networks? Furthermore, is there any similarity between the evolved networks with those few ER random networks that managed to, if not perfectly but reasonably well solve the tasks?

0.3 Network growth vis-a-vis random chance: emergent performance-dependent network structure

Graph theoretic properties of the networks such as density and node-degrees are evaluated to understand the similarities and differences between the networks generated by the diverse growth strategies (N1, N2 and N3) with that of Erdős-Renyi random networks (N0).

$$\delta = M/N(N-1) \quad (3)$$

The density (δ) of a directed network is given by Eq.3, where M is the total number of links in the network³⁹. Densities of networks generated by the three different network growth models for each of the benchmark tasks are plotted with respect to the nodes in that network in Fig. 4. The color of each symbol represents the performance of the generated network for that task, where the purple color exhibits the best while red represents the worst performance for the corresponding task. Furthermore, 500 Erdős-Renyi random networks (N0) are also plotted along with their performances on the $N - \delta$ space in Fig. 4 for the respective tasks. The random networks are generated to have nodes and densities in the range corresponding to that of different tasks.

The final networks generated by the distinct growth strategies (N1, N2 and N3) are perfectly aligned along a rectangular hyperbolic curve, reflecting an inverse relationship between density and nodes. Interestingly, in the space of all possible random network reservoirs, the ones that yield the best performance, as exemplified by low errors, lie in the vicinity of the $\delta - N$ scaling function of the networks generated by distinct growth models. This feature is exhibited by the presence of the deep blue symbols for N0 (reflecting low errors) *only* on the curve displaying the inverse proportionality of density with respect to number of nodes.

Another aspect of elucidating the observed $\delta - N$ scaling of the networks generated by the different growth-based models is in terms of the degrees of the nodes. The number of edges (M) in a growing network can also be expressed as a product of the average degree of the network (μ) and nodes (N) i.e $M = \mu N$. Plugging it back to Eq.3 reveals that the density of the growing or evolving networks scale as $\delta \approx \mu/N$. When a network grows, each new node can make $M^{New} \in [M_{min}^{New}, \dots, M_{max}^{New}]$ number of new edges. This means that over several growth iterations, the network will have its average degree equal to \overline{M}^{New} or $\mu = \overline{M}^{New}$. Therefore, the density of a growing network should scale as $\delta = \overline{M}^{New}/N$ shown with a navy-blue line in $\log(\delta) - \log(N)$ in Fig.5(a) and (b). M_{min}^{New} , M_{max}^{New} and \overline{M}^{New} are 1, 5 and 3 respectively in this study. Furthermore, such a growth will always stay bounded by $\delta_{min} = M_{min}^{New}/N$ and $\delta_{max} = M_{max}^{New}/N$, both are represented by grey dotted-dashed lines in Fig.5(a) and (b). In the case of N1 model, the average degree of the generated networks $\mu^{N1} \rightarrow \overline{M}^{New}$ (Fig.5(c)), for large networks. This implies that there is no preference for the degree of the added nodes, and all degrees are chosen with equal probability, leading to a simple statistically averaged quantity.

The only small difference between μ^{N1} and \overline{M}^{New} appears because of the lower average degrees of the very small networks obtained for the SinCos-1 task. Since this task is not complex enough, the networks converge rapidly within a few iterations (Fig. 2(a) and Fig. 3(d)) implying that new nodes are added with fewer new edges i.e $m < \overline{M}^{New}$.

However, the networks evolved by model N3 have average (in-)degree $\mu^{N3} < \overline{M}^{New}$ (Fig.5 (d)). This explains why all the evolved networks fall below the $\delta = \overline{M}^{New}/N$ (blue) line in Fig.5 (a). In other words, these networks are specifically evolved by the N3 model to have lower node degrees in order to achieve enhanced computational capacity. This shows another interesting feature of the networks generated by the performance-dependent network evolution strategy.

In summary, the significant features of the networks evolved with different strategies vis-a-vis uninformed growth and the random network is as follows: The first feature is that the average degree is smallest when networks evolved with the strategy of performance-dependent node addition and removal. That is, the networks that yield the best performance are sparser as compared to the ones with poor performance generated by alternate growth strategies. The second important feature is that the networks that emerge under performance-dependent node addition, as well as removal, have not only the lowest pre-factor in the scaling relation, but they also converge to low network size N . This is clearly evident in the fact that N3 networks occupy the low N end of the hyperbolic arm of the scaling function in Fig.3.

0.4 Asymmetric distribution of input and readout nodes facilitate enhanced information processing

In the traditional reservoir computing framework, it is a common practice to randomly select any number of input-receiving as well as output (or readout) nodes from the pool of N reservoir nodes. However, there is no criterion for the selection and number of input and output nodes for efficient information transmission, processing, and reading-out. Therefore, all the distinct network growth models (N1, N2 and N3) described in the sections above are equipped with a degree of freedom for the selection of input (I) and output (O) nodes. A new node added by the *node addition module A* can also be an input node with probability P^I , and it can independently also be an output node with probability P^O . We set $P^I = P^O = 0.5$ in this study, meaning that a new node attaching to the network will independently have a 50% chance of being an input or output node. There is a 25% chance that this new node is both an input-receiving as well as a readout node, and from here onwards such nodes are denoted as *Common Nodes* ($C = I \cap O$). Also, there is a 25% chance that a new node is neither an input nor an output node, and these nodes are denoted as *Unique Nodes* ($U = N - (I \cup O)$).

The final networks generated by the performance-independent node addition (N1) model exhibit an equal fraction of input (I/N) and output nodes (O/N) for all the tasks, as shown in Fig.6 (a) and (b) respectively. Consequently, the fractions of *Common* (C/N) and *Unique* (U/N) nodes are normally distributed around 0.25, as shown in Fig.6 (c) and (d) respectively. These results were expected from the N1 model because the networks grow without the deletion of any nodes and also without any feedback for the selection of specific nodes. As a reference, the I and O nodes in the Erdős-Renyi random networks (N0) are also selected with the same values of the probabilities P^I and P^O respectively. The means of all these fractions over different tasks for the Erdős-Renyi random networks (N0) (grey solid line) are always close to that of N1 networks (black dashed line in Fig.6). Therefore, the fraction of I , O , C and U nodes of the Erdős-Renyi networks also have the statistically same distributions as that of N1 networks.

However, the fraction of readout nodes in the networks evolved with performance-dependency (N3) is surprisingly high as compared to that of N0 and N1 networks. The number of O nodes is consistently high over all the different benchmark tasks. The mean fraction of output nodes (O/N) over all the tasks is ~ 0.74 (Fig.6 (a)). The mean fraction of input nodes (I/N) is ~ 0.53 which is just a slight shift as compared to a 50% rise in output nodes. Interestingly, the fraction of *Common* nodes (C/N) rose to $\sim 40\%$. This means that not only the readout nodes are rising but the nodes that are both I as well as O are getting selected when networks are evolving with performance-dependent node addition and removal (N3). As a result, the fraction of *Unique* nodes (U/N) falls to ~ 0.1 on average for all the tasks.

Furthermore, the networks generated by the N2 model also have a higher percentage of O and C nodes, that is $\sim 58\%$ and $\sim 29\%$ respectively. These fractions are in-between those of N3 and N1/N0 models. Since the N2 model represents partial evolution where networks can only grow depending on their performance without deletion, therefore, this slight rise in the fraction of O and C nodes was expected.

This is an important finding that shows that the performance-dependent network evolution is shifting the readout nodes to a higher fraction that eventually enhances its computation capabilities. This shift toward the more readout nodes means that the ansatz vector space for the linear regression becomes larger and richer increasing the overall accuracy. The network size prescribes how complex the latent space dimensions can be where the larger networks reflect the more complex dynamics of the reservoirs and the readout node number indicates how many of those states are essential for the output generation. Such an asymmetric distribution of readout and common nodes is therefore a unique feature that is required in networks to efficiently solve a given task with limited resources, i.e. number of nodes. This shift is a causal effect of performance-dependently addition and removal of nodes in an evolving network contrary to all different network growth strategies as well as Erdős-Renyi random networks.

Discussion

We have proposed the concept of performance-dependent network evolution for efficient information processing and demonstrated that complex dynamical tasks can be efficiently and accurately solved with smaller evolved networks. Specifically, we showed, through the execution of a range of tasks of varying levels of complexity, that evolved networks exhibit enhanced computation capacity when compared to the Erdős-Rényi random networks. Furthermore, evolution through performance-dependent node addition and deletion outperformed all the networks generated with alternate network growth strategies, that are either uninformed or allow only performance-dependent node addition. This demonstrates the importance of performance dependency, together with continuous adaptive expansion and contraction of the network, to achieve an optimal size for its elevated performance. A prior study by P. François explored oscillatory responses by the biochemical networks with specific biochemical dynamics on evolving phenotypic networks^{40,41}. However, it was confined to smaller biochemical modules and lacked graph-theoretic measures for complex tasks.

Performance-dependently evolved networks exhibit several unique features over their course of evolution such as a saturation in the growth of the network size, as compared to the linear and sub-linear growths observed in the other inefficient network growth strategies. The performance-based node addition/deletion evolution tends to generate sparser networks by preferentially expanding the networks with the nodes having lower degrees while deleting the ones otherwise. The densities of all the networks are found to follow a scaling in the $N - \delta$ space, irrespective of their performance dependency. Furthermore, performance-dependent evolution leads to a unique organization of the networks in the $N - \delta$ space, according to different task complexity. The evolved networks of simpler processes such as Sine-Cosine tasks are smaller and denser, and therefore are present on the top end of the scaling curve in contrast to the ones evolved for complex tasks such as NARMA-10 and 15, which have more nodes and occur on the lower end with low density. Significantly, this correlation offers a new approach to measuring the complexity of a computational task, based on the network size necessary for requisite performance and its corresponding density.

The asymmetric distribution of input-receiving and readout nodes in the evolved reservoir network is yet another unique feature of the performance-dependent network evolution, with these networks having a high percentage of readout nodes and common nodes as compared to the input-receiving ones. This is an important finding suggesting the extraction of maximal and sufficient information from a network with a high fraction of readout nodes, which is contrary to the common practice of random readout node selection. Such an emergent asymmetric distribution of I and O nodes has been observed in biological networks as well, which naturally evolve for more complex processes. For instance, GRNs across different species have been observed to have a high fraction of readout nodes as compared to the *recurrent* part of the networks¹⁴.

The results presented here also shed light on the evolutionary path and scaling followed by the biological networks shown to have *recurrent* structures to carry out biological computations. This general framework can be further extended to understand the formation of specific network structures for biologically relevant processes, node dynamics and evolutionary constraints such as mutations and spatial restrictions.

The unique features of the task-specifically and performance-dependently evolved networks hold promise for the machine-learning community to efficiently solve various classes of problems with smaller, specialized networks. Furthermore, recall that a few Erdős-Rényi random networks also managed to solve some tasks with satisfactory performance. However, since the majority of the random networks failed to solve the tasks, therefore generating efficient random networks is another painstaking task in itself. This search is especially onerous when one does not know either the network density or the fraction of readout nodes that need to be selected. Our results precisely provide pointers to those few Erdős-Rényi random networks that yield satisfactory performance by pinning their location in the vicinity of $\delta - N$ scaling of evolved networks. This enormously narrows down the search space offering a recipe for generating high-performing "designer" random networks. The asymmetric distribution of input-receiving and readout nodes can be further explored to enhance the efficiency of such Erdős-Rényi networks.

The performance-dependent network evolution framework presented here is a first-of-its-kind and comprehensive approach that addresses critical questions regarding efficient information processing networks. It offers insights from graph theory, complex dynamical systems, and machine learning perspectives, providing a formal description of process-specific network *structure-function* dependencies and emergent scaling laws in evolving networks. This framework facilitates the generation of minimal and task-specific networks while elucidating their resulting graph-theoretic properties and scaling laws for efficient information processing. Our exploration lays the groundwork for advancing the field of network science by uncovering guiding principles for generating minimal and efficient task-specific networks and understanding their unique emergent properties.

References

1. Lynn, C. W. & Bassett, D. S. The physics of brain network structure, function and control. *Nat. Rev. Phys.* **1**, 318–332, DOI: <https://doi.org/10.1038/s42254-019-0040-8> (2019).

2. Babu, M. M., Luscombe, N. M., Aravind, L., Gerstein, M. & Teichmann, S. A. Structure and evolution of transcriptional regulatory networks. *Curr. Opin. Struct. Biol.* **14**, 283–291, DOI: <https://doi.org/10.1016/j.sbi.2004.05.004> (2004).
3. Levy, E. D. & Pereira-Leal, J. B. Evolution and dynamics of protein interactions and networks. *Curr. Opin. Struct. Biol.* **18**, 349–357, DOI: <https://doi.org/10.1016/j.sbi.2008.03.003> (2008). Nucleic acids / Sequences and topology.
4. Bassett, D. S. *et al.* Efficient physical embedding of topologically complex information processing networks in brains and computer circuits. *PLOS Comput. Biol.* **6**, 1–14, DOI: [10.1371/journal.pcbi.1000748](https://doi.org/10.1371/journal.pcbi.1000748) (2010).
5. Dorogovtsev, S. N. & Mendes, J. F. F. Evolution of networks. *Adv. Phys.* **51**, 1079–1187, DOI: [10.1080/00018730110112519](https://doi.org/10.1080/00018730110112519) (2002). <https://doi.org/10.1080/00018730110112519>.
6. de Solla Price, D. J. Networks of scientific papers. *Science* **149(3683)**, 510–515, DOI: <https://doi.org/10.1038/s41598-020-71549-y> (1965).
7. Watts, D. J. & Strogatz, S. H. Collective dynamics of ‘small-world’ networks. *Nature* **393**, 440–442, DOI: <https://doi.org/10.1038/30918> (1998).
8. Zhou, B. *et al.* The nature and nurture of network evolution. *Nat. Commun.* **14**, 7031, DOI: <https://doi.org/10.1038/s41467-023-42856-5> (2023).
9. Cabral, J., Jirsa, V., Popovych, O. V., Torcini, A. & Yanchuk, S. Editorial: From structure to function in neuronal networks: Effects of adaptation, time-delays, and noise. *Front. Syst. Neurosci.* **16**, DOI: [10.3389/fnsys.2022.871165](https://doi.org/10.3389/fnsys.2022.871165) (2022).
10. Suárez, L. E., Markello, R. D., Betzel, R. F. & Misic, B. Linking structure and function in macroscale brain networks. *Trends Cogn. Sci.* **24**, 302–315, DOI: <https://doi.org/10.1016/j.tics.2020.01.008> (2020).
11. Achterberg, J., Akarca, D., Strouse, D. J., Duncan, J. & Astle, D. E. Spatially embedded recurrent neural networks reveal widespread links between structural and functional neuroscience findings. *Nat. Mach. Intell.* **5**, 1369–1381, DOI: <https://doi.org/10.1038/s42256-023-00748-9> (2023).
12. van den Heuvel, M. P., Bullmore, E. T. & Sporns, O. Comparative connectomics. *Trends Cogn. Sci.* **20**, 345–361, DOI: <https://doi.org/10.1016/j.tics.2016.03.001> (2016).
13. Ma, H.-W., Buer, J. & Zeng, A.-P. Hierarchical structure and modules in the escherichia coli transcriptional regulatory network revealed by a new top-down approach. *BMC Bioinforma.* **5**, 199, DOI: <https://doi.org/10.1186/1471-2105-5-199> (2004).
14. Gabalda-Sagarra, M., Carey, L. B. & Garcia-Ojalvo, J. Recurrence-based information processing in gene regulatory network. *Chaos* **28**, 106312 (2018).
15. Berner, R., Gross, T., Kuehn, C., Kurths, J. & Yanchuk, S. Adaptive dynamical networks. *Phys. Reports* **1031**, 1–59, DOI: <https://doi.org/10.1016/j.physrep.2023.08.001> (2023).
16. Panchal, F. S. & Panchal, M. Review on methods of selecting number of hidden nodes in artificial neural network. *Int. J. Comput. Sci. Mob. Comput.* **3**, 455–464 (2014).
17. Ahmed, S. F. *et al.* Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artif. Intell. Rev.* **56**, 13521 – 13617, DOI: <https://doi.org/10.1007/s10462-023-10466-8> (2023).
18. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Networks* **115**, 100–123 (2019).
19. Nakajima, K. & Fischer, I. *Reservoir Computing: Theory, Physical Implementations, and Applications* (Natural Computing Series, Springer; Singapore, 2021).
20. Maass, W., Natschlaeger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14(11)**, :2531–60 (2002).
21. Jaeger, H. Short term memory in echo state networks. *Ger. Natl. Res. Cent. for Inf. Technol. GMD Report* **152** (2001).
22. Carroll, T. & Pecora, L. Network structure effects in reservoir computers. *Chaos: An Interdiscip. J. Nonlinear Sci.* **29**, 083130 (2019).
23. Silva, N., Ferreira, T. & Guerreiro, A. Reservoir computing with solitons. *New J. Phys.* **23**, 023013 (2021).
24. Carroll, T. Do reservoir computers work best at the edge of chaos? *Chaos: An Interdiscip. J. Nonlinear Sci.* **30**, 121109 (2020).
25. Jensen, J. H. & Tufte, G. Reservoir computing with a chaotic circuit. *Artif. Life Conf. Proc.* **14**, (MIT Press. 222–229 (2017).
26. Mandal, S., Sinha, S. & Shrimali, M. Machine learning potential of a single pendulum. *Phys. Rev. E* **105**, 054203 (2022).

27. Vlachas, P. *et al.* Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks* **126**, 191 (2020)).
28. Zhang, H., Fan, H., Wang, L. & Wang, X. Learning hamiltonian dynamics with reservoir computing. *Phys. Rev. E* **104**, 024205 (2021)).
29. Rafayelyan, M., Dong, J., Tan, Y., Krzakala, F. & Gigan, S. Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction. *Phys. Rev. X* **10**, 041037 (2020).
30. Vidal-Saez, M. S., Vilarroya, O. & Garcia-Ojalvo, J. Biological computation through recurrence. *arXiv* (2024).
31. Damicelli, F., Hilgetag, C. C. & Goulas, A. Brain connectivity meets reservoir computing. *PLOS Comput. Biol.* **18**, e1010639, DOI: [10.1126/sciadv.abh0693](https://doi.org/10.1126/sciadv.abh0693) (2022).
32. Cucchi, M. *et al.* Reservoir computing with biocompatible organic electrochemical networks for brain-inspired biosignal classification. *Sci. Adv.* **7**, eabh0693, DOI: [10.1126/sciadv.abh0693](https://doi.org/10.1126/sciadv.abh0693) (2021).
33. Suárez, L. E. *et al.* Connectome-based reservoir computing with the conn2res toolbox. *Nat. Commun.* **15**, 1–14, DOI: [10.1126/sciadv.abh0693](https://doi.org/10.1126/sciadv.abh0693) (2022).
34. Seoane, L. F. Evolutionary aspects of reservoir computing. *Phil. Trans. R. Soc. B* **374**, 1–15, DOI: [10.1126/sciadv.abh0693](https://doi.org/10.1126/sciadv.abh0693) (2022).
35. Atiya, A. & Parlos, A. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks* **11**, 697–709, DOI: [10.1109/72.846741](https://doi.org/10.1109/72.846741) (2000).
36. The evolution of signalling pathways in animal development. *Nat. Rev. Genet.* **4**, 39–49, DOI: <https://doi.org/10.1038/nrg977> (2003).
37. Liu Q., L. F. & W., W. Memory augmented echo state network for time series prediction. *Neural Comput. & Appl.* DOI: <https://doi.org/10.1007/s00521-023-09276-4> (2023).
38. Matthias Freiberger, P. B. & Dambre, J. A training algorithm for networks of high-variability reservoirs. *Nature, Sci. Reports* **10:14451**, 1–11, DOI: <https://doi.org/10.1038/s41598-020-71549-y> (2020).
39. Newman, M. E. J. *Networks: an introduction* (Oxford University Press, Oxford; New York, 2010).
40. François, P. & Hakim, V. Design of genetic networks with specified functions by evolution in silico. *PNAS* **101** (2), 580–585, DOI: <https://www.pnas.org/doi/full/10.1073/pnas.0304532101> (2004).
41. François, P. Evolving phenotypic networks in silico. *Semin. Cell & Dev. Biol.* **35**, 90–97, DOI: https://www.sciencedirect.com/science/article/pii/S1084952114001852?ref=pdf_download&fr=RR-2&rr=864b6d3f8aea4528 (2014).

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the Special Priority Program (SPP 2353, project number 501847579). S. Sinha acknowledges support from the J.C. Bose National Fellowship (Grant No. JBR/2020/000004).

Author contributions statement

M.Y. conceptualized the study, performed simulations and prepared the figures. M.Y., S.S. and M.S. interpreted the results and contributed equally in writing the manuscript.

Competing interests

The authors declare no competing interests.

Figures and legends

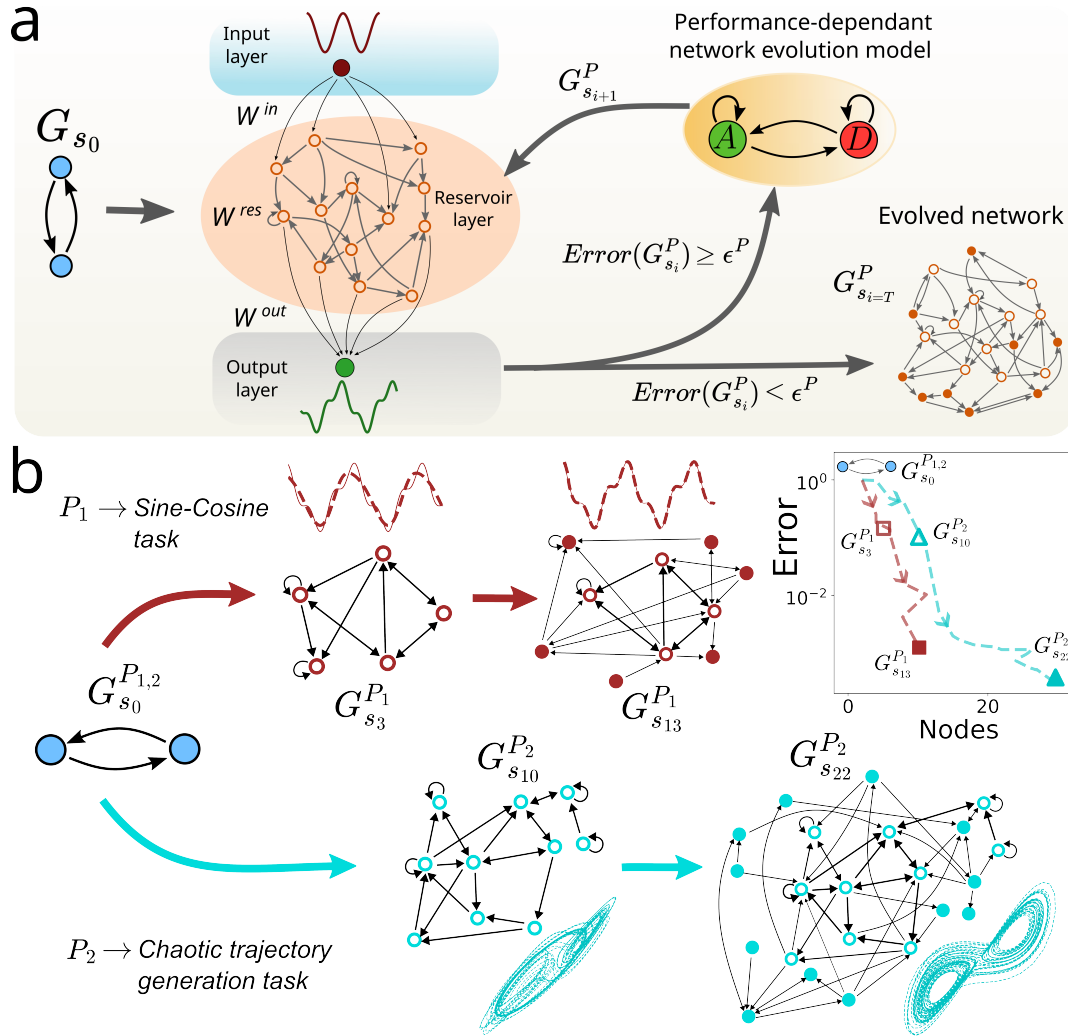


Figure 1. Performance-dependent network evolution. (a) Schematic showing the working mechanism of the network evolution model depending on an underlying process or task (P). The model consists of a node addition module (A) and a node deletion module (D). Module A adds a new node to the pre-existing network while module B removes nodes from the pre-existing network, conditioned on network performance improvement. This procedure continues until the desired performance (ϵ^P) is achieved. Starting with a very small initial network (G_{s_0}), the model generates an evolved and minimal final network ($G_{s_{i=T}}^P$) that can efficiently solve the given task P . (b) Schematic showing the model evolving the same initial network G_{s_0} for two different exemplar tasks; P_1 for converting the sine function to its complex conjugate and P_2 for generating a chaotic trajectory from the Lorenz system. The final evolved networks ($G_{s_{18}}^{P_1}$ and $G_{s_{22}}^{P_2}$) contain some nodes (hollow) from a network at some previous intermediate stage ($G_{s_3}^{P_1}$ and $G_{s_{10}}^{P_2}$). The evolved networks can solve the underlying task with higher accuracy as compared to their intermediate networks.

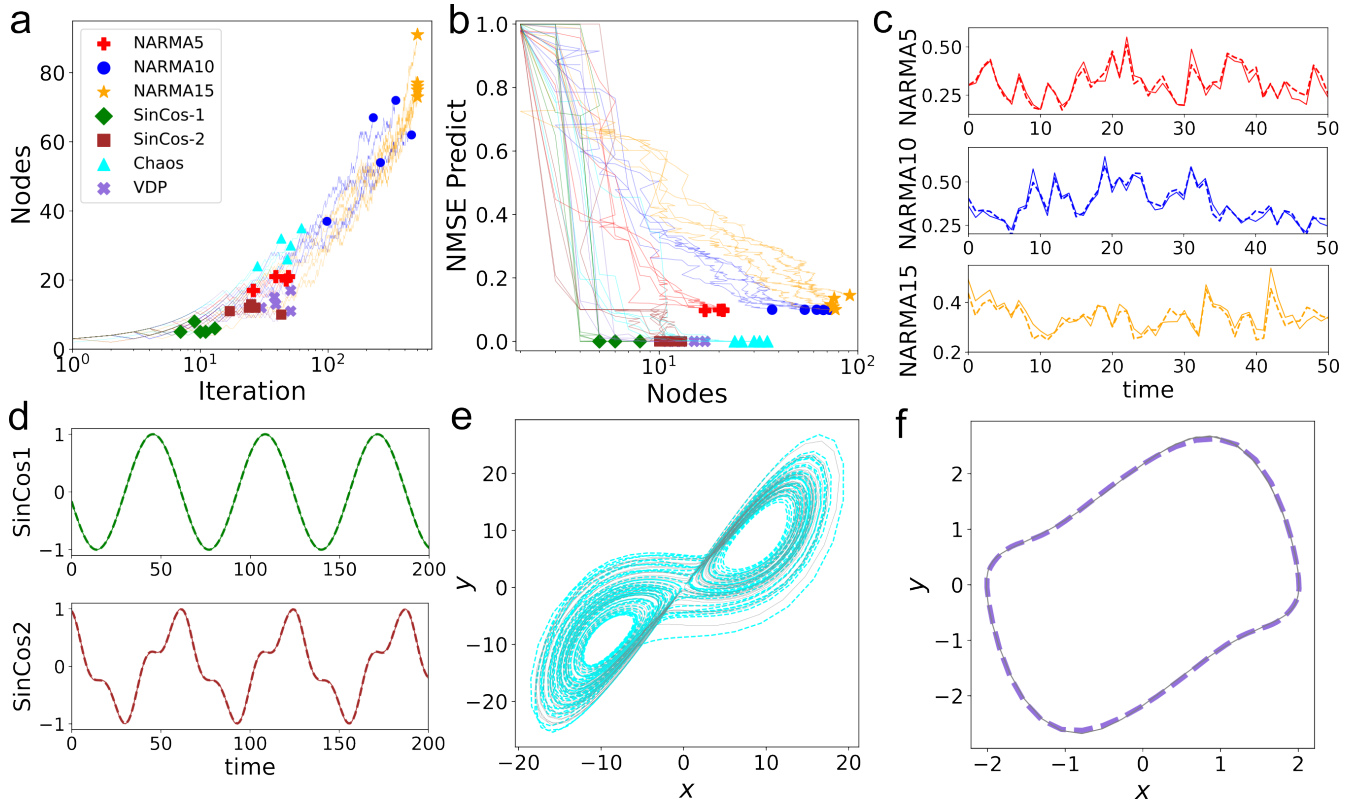


Figure 2. Generalization of the performance-dependent network evolution over diverse benchmark tasks. (a) The number of nodes in the evolving networks is shown over each iteration of the evolution model for different tasks. The total number of nodes in the final evolved networks is represented with different symbols. (b) The performance (Normalized Mean Square Error, NMSE) for the independently evolving networks is represented against the total number of nodes present in the corresponding evolving network. The evolution of 5 representative networks for each task are displayed in (a) and (b). (c) Predicted outputs generated by the evolved networks for the three different NARMA time series prediction tasks of 5th (top), 10th (middle) and 15th (bottom) orders, are plotted with dashed curves and the true values are with solid ones. (d) Predictions of SinCos-1 and SinCos-2 tasks. (e) Chaotic trajectories from the Lorenz system and (f) phase-portrait of the limit-cycle from the Van der Pol oscillator obtained from evolved networks for the respective cases. RC is working autonomously in a closed-loop manner in (e) and (f).

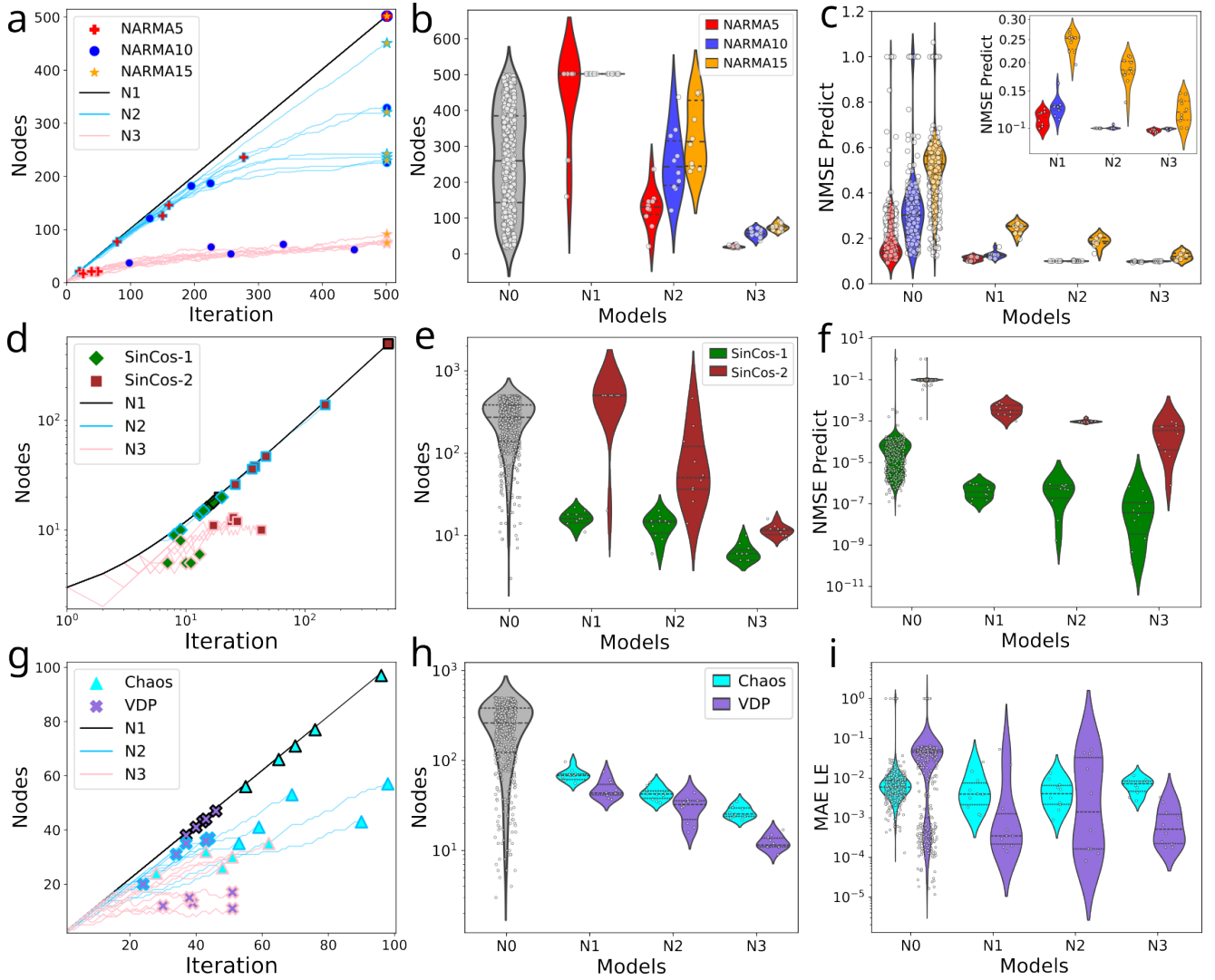


Figure 3. Contrasting networks generated by the performance-based network evolution with alternate network growth strategies. Comparison of the total number of nodes as the networks grow/evolve in the models N1 (black curve), N2 (sky blue curves) and N3 (pink curves) for NARMA-5th, 10th and 15th order tasks in (a), for SinCos-1 and 2 tasks in (d) and Lorenz's chaotic trajectory with Van der Pol's limit cycle generation tasks in (g). The total number of nodes in the final evolved networks is represented with different symbols corresponding to each task. (b), (e) and (h) The nodes in the final evolved networks from N3 model are compared with that of N1 and N2 models along with Erdős-Rényi random networks (N0, grey). (c), (f) and (i) The performance calculated using Normalized Mean Square Error (NMSE) of 10 independently evolved/grown networks by N1, N2 and N3 models for all the tasks are compared alongside the performance of Erdős-Rényi random networks (N0). The Mean Absolute Error of the Lyapunov-Exponents (MSE-LE) between the original and predicted trajectories is used to quantify the error in both the trajectory generation tasks (i).

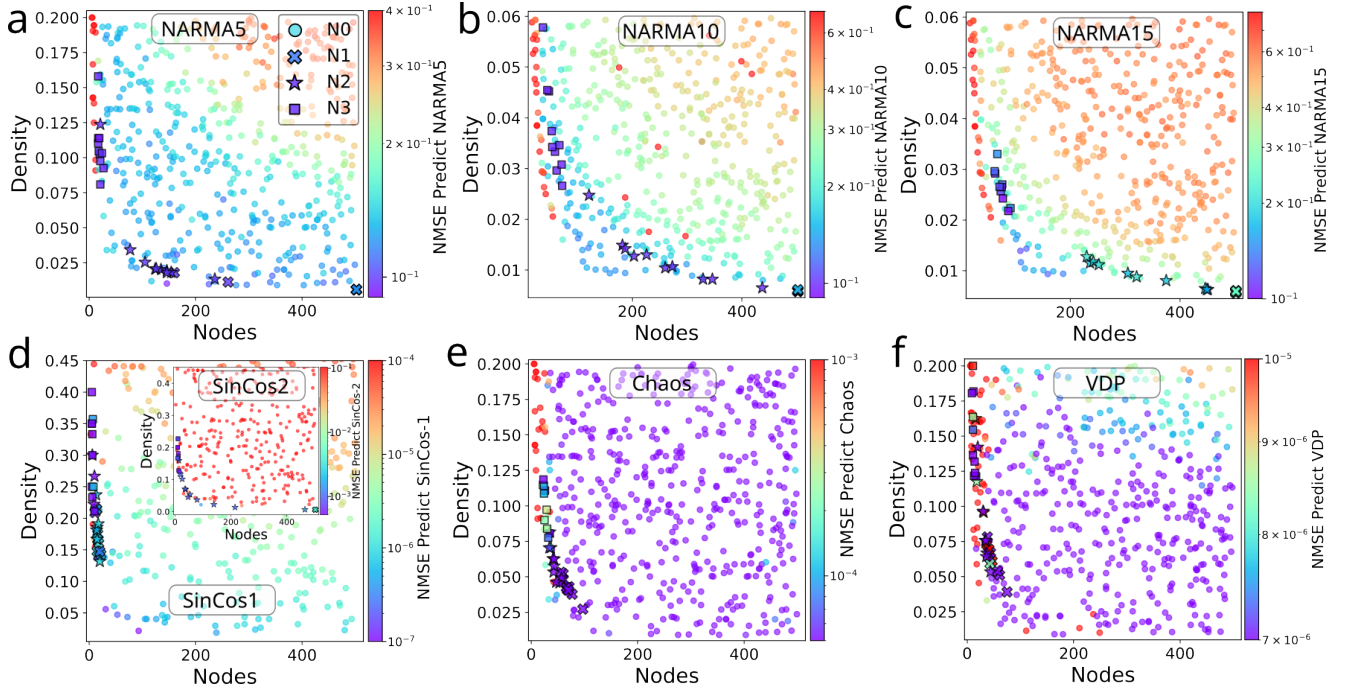


Figure 4. Organization of networks generated by distinct growth model in Nodes(N)-Density(δ) space. Densities of final generated networks are represented as a function of nodes (Eq.3) with symbols crosses, stars and squares respectively for models N1, N2 and N3. Circles covering the entire $N - \delta$ parameter space represent 500 Erdős-Rényi random networks (N0). The symbols are filled with colors corresponding to their performance for solving the tasks NARMA-5, 10 and 15, SinCos-1 (inset SinCos-2), Chaotic trajectory and periodic orbit from Van der Pol oscillator respectively in panels (a) to (f).

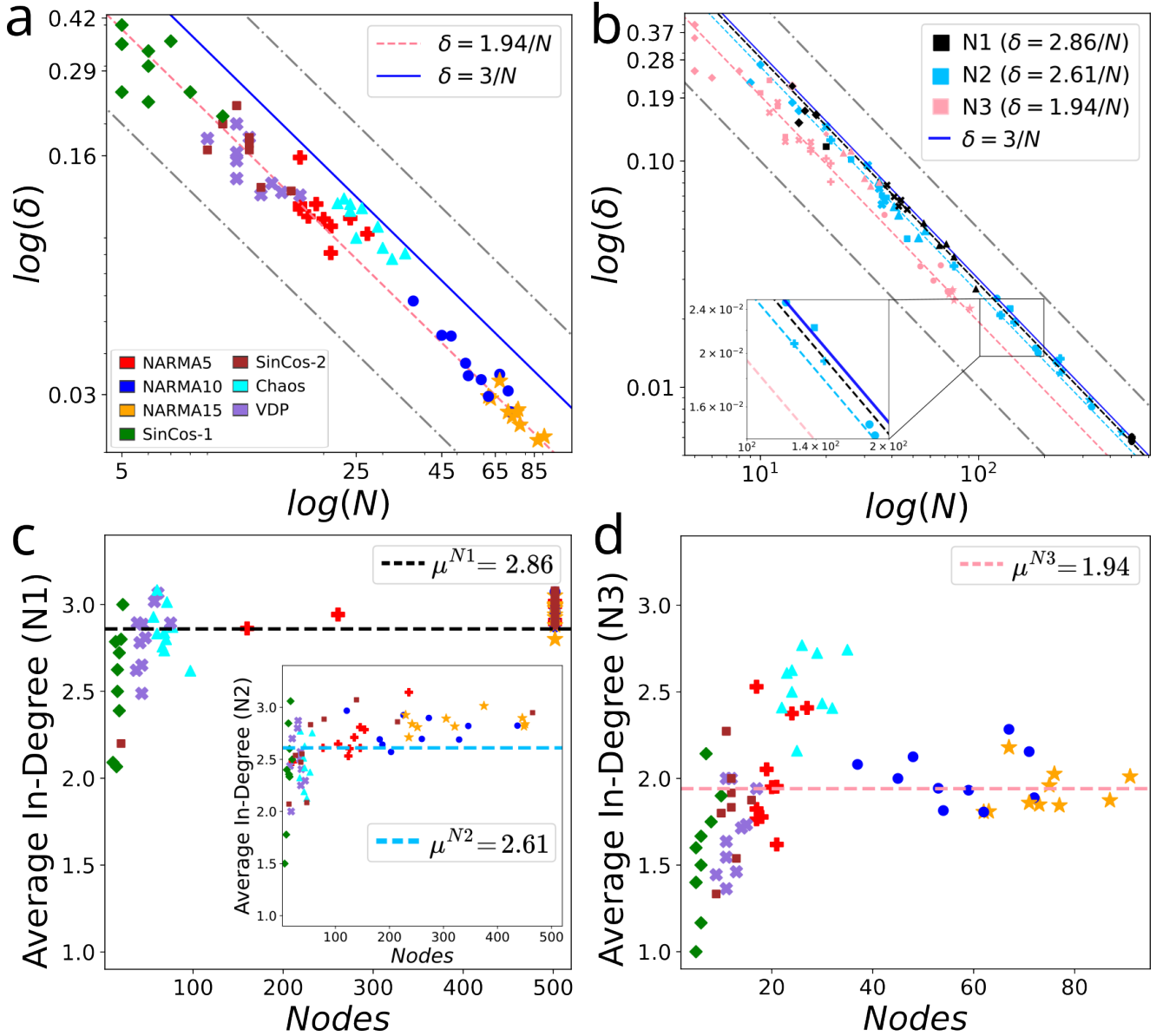


Figure 5. $N - \delta$ scaling over different network growth strategies. (a) The densities (δ) of the final network N3 evolved from performance-dependent node addition and deletion, represented against the number of nodes (N), over $\log - \log$ scale, for 7 different tasks. (b) Comparison of the scaling of the densities as a function of number of nodes, for different network growth models, N1 (black symbols), N2 (sky-blue symbols) and N3 (pink symbols). The lower and upper grey dotted-dashed lines in panels (a) and (b) represent the limiting cases of network growth with $\delta_{min} = 1/N$ and $\delta_{max} = 5/N$, respectively. Average in-degrees of the final networks with respect to the nodes for different tasks are shown for model N1 in (c), N2 in (c, inset) and for N3 in (d) for different tasks. The mean of all the average in-degrees (μ^{N1} , μ^{N2} and μ^{N3}) across different tasks for that particular model are represented with the dashed lines in panels (c) and (d).

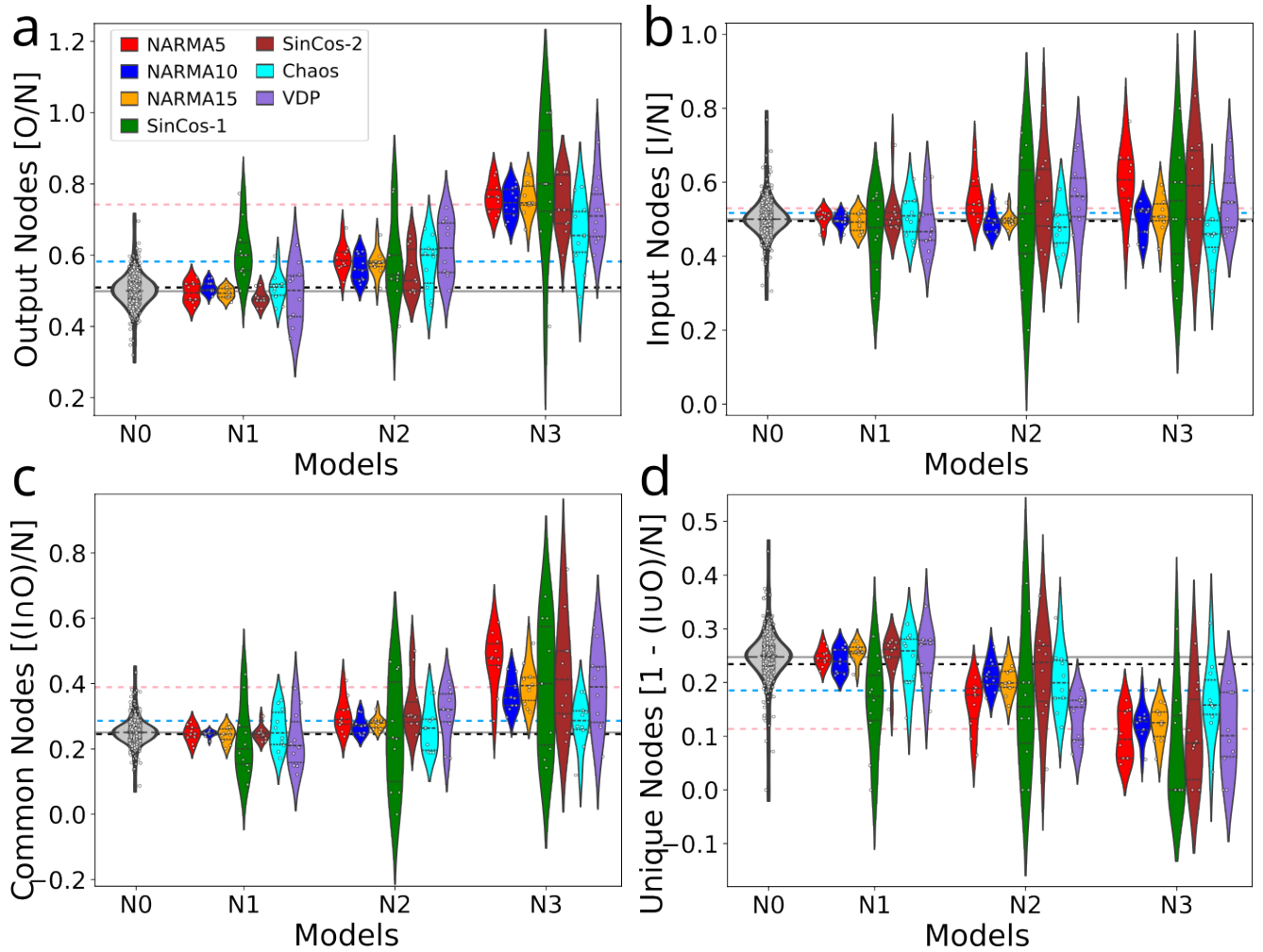


Figure 6. Asymmetric distribution of input receiving and output (readout) nodes in performance-dependently evolved networks. Fraction of output (O/N) (a), input (I/N) (b), common (both input and output nodes, C/N) (c) and unique network nodes (U/N) (d) are compared for different network growth models (N1, N2, and N3) along with that of Erdős-Renyi random networks (N0) for reference. The mean over all the tasks for each of the fractions is represented with colored lines for the models N0 (solid grey), N1 (dashed black), N2 (dashed sky-blue) and N3 (dashed pink).