The Role of Structure and Complexity on Reservoir Computing Quality

Matthew Dale[1,4], Jack Dewhirst[1,4], Simon O'Keefe[1,4], Angelika Sebald[2,4], Susan Stepney[1,4], Martin A. Trefzer[3,4]

[1]Department of Computer Science, University of York, UK
[3]Department of Chemistry, University of York, UK
[3]Department of Electronic Engineering, University of York, UK
[4]York Cross-disciplinary Centre for Systems Analysis
matt.dale@york.ac.uk

Abstract. We explore the effect of structure and connection complexity on the dynamical behaviour of Reservoir Computers (RC). At present, considerable effort is taken to design and hand-craft physical reservoir computers. Both structure and physical complexity are often pivotal to task performance, however, assessing their overall importance is challenging. Using a recently proposed framework, we evaluate and compare the dynamical freedom (referring to quality) of neural network structures, as an analogy for physical systems. The results quantify how structure affects the range of behaviours exhibited by these networks. It highlights that high quality reached by more complex structures is often also achievable in simpler structures with greater network size. Alternatively, quality is often improved in smaller networks by adding greater connection complexity. This work demonstrates the benefits of using abstract behaviour representation, rather than evaluation through benchmark tasks, to assess the quality of computing substrates, as the latter typically has biases, and often provides little insight into the complete computing quality of physical systems.

Keywords: Reservoir Computing, Unconventional Computing, Echo State Networks, Structure, Complexity

1 Introduction

Reservoir Computing (RC) [26, 29] is a computational model used to train and exploit an increasingly rich variety of dynamical systems, ranging from virtual neural networks to novel physical systems, such as quantum, electrical, chemical, optical and mechanical (see reviews [20, 28]). Every reservoir system is designed to harness the underlying dynamics of the *substrate* it is implemented with, whether that be a physical device or material, a simulated network, or a set of system equations. However, finding a suitable substrate, or designing one, with sufficient dynamics to compute specific tasks is challenging.

Methods to assess the complete range of dynamics a substrate can exhibit are still undeveloped. Therefore, matching substrates to tasks is typically done

through trial and error. This makes for a long and laborious exercise to determine how best to configure, perturb and alter the low-level design of substrates to improve performance.

In [10] we present an alternative method to assess and compare computing systems based on abstract behaviours of dynamical properties. The idea is to map and explore the full dynamical range of substrates and use this to determine the effects of configuration, perturbation and design alteration on substrate "quality".

Here we use that method to investigate how topology and structural complexity in simulated recurrent networks affect dynamical range and quality. Our hypothesis is that, when implementing these networks, compromises in size and structure lead to similar qualities, e.g., larger networks with simple structure are equivalent to smaller networks with complex structures; therefore, simple structures still exhibit complex behaviours, but require larger network size to do so.

In the reservoir computing literature, simple network topologies and regular structures produce competitive performances to complex networks [24, 25, 30]. Simple structures made from multiple processing units are easy to implement and control in hardware. However, complex structures are more abundant in physical systems, but harder to manipulate. Knowing how structure and size affect quality will provide a useful guide to future substrate designers.

2 Reservoir Computing

Embracing the intrinsic properties of physical systems has the potential to offer improvements in performance, efficiency and/or computational power compared with conventional computing systems [5, 18]. However, to do so requires a model of computation that naturally fits the substrate rather than imposing an inappropriate model that fights its implementation [27].

The reservoir computing model's simplicity means that it naturally aligns with many physical systems. However, its simplicity also has drawbacks, for example, it struggles to solve complex tasks requiring high-order abstractions [11, 20].

An input-driven reservoir computer is typically represented and divided into three layers: the input, the *reservoir*, and the readout. The reservoir is the dynamical system, and perturbed and observed as a black-box. The input and readout depend on the chosen method of encoding and decoding of information to and from the reservoir, and the material instantiation of the encoded information. Depending on the implementation, this could be discrete or continuous values encoded in electrical, optical, chemical or other signals.

As with many systems, each reservoir is configured, controlled and tuned to perform a desired function. This often requires the careful tuning of parameters in order to produce working and optimal reservoirs. Most reservoirs are hand-crafted to a task, often requiring expert domain knowledge to design an optimal system. However, the separation between layers allows the reservoir to

be optimised and configured independently of the input and readout layer. Many techniques have been used to optimise virtual reservoirs [1], and more recently, physical reservoirs [7–9].

To interpret a substrate as a reservoir, we define that the observed reservoir states $x(n)$ form a combination of the substrate's implicit function and its *discrete* observation:

$$x(n) = \Omega(\mathcal{E}(W_{in}u(t), u_{config}(t))) \tag{1}$$

where $\Omega(n)$ is the observation of the substrate's macroscopic behaviour and $\mathcal{E}(t)$ the continuous microscopic substrate function, when driven by the input $u(t)$. Here, W_{in} symbolises a set of input weights common to all reservoir systems; this is typically random (see [19], a guide for creating reservoir weights). The variable $u_{config}(t)$ represents the substrate's configuration, whether that be through external control, an input-output mapping, or other method of configuration. Typically, $u_{config}(t)$ is not a function of time, but only of a given problem. However, switching between different configurations in time could add additional dynamical complexity.

This formalisation of the reservoir states separates the system into contributing parts, including the observation and configuration method, which as a whole represents the overall reservoir system.

The final output signal $y(n)$ is determined by the readout function g, on the observation $x(n)$:

$$y(n) = g(x(n)) \tag{2}$$

In Eq.1, we give a simple case where no feedback is applied. To add feedback, the input variables y and W_{fb} are added to $\mathcal{E}(.)$, where W_{fb} represents feedback weights.

Note that $\mathcal{E}$, the intrinsic substrate function, is described as being fixed, clamped, or set by u_{config}; only g is adapted. However, depending on the system, $\mathcal{E}$ may change when interacted with or observed, and therefore be non-deterministic.

3 CHARC Framework

The CHARC (CHAracterisation of Reservoir Computers) framework [10] is used to map and compare the computational expressiveness of computing substrates. In this process, the applied computational model, e.g., encoding, decoding and abstract representation of the system, is also assessed, suggesting whether the chosen model is a suitable fit to the physical implementation.

To characterise the computing *quality* of substrates, the framework searches over metrics measuring dynamical properties. The framework shows that carbon nanotube composites possess a lower quality than small recurrent networks when configured and stimulated using current techniques [10]. This supports previous findings [7–9].

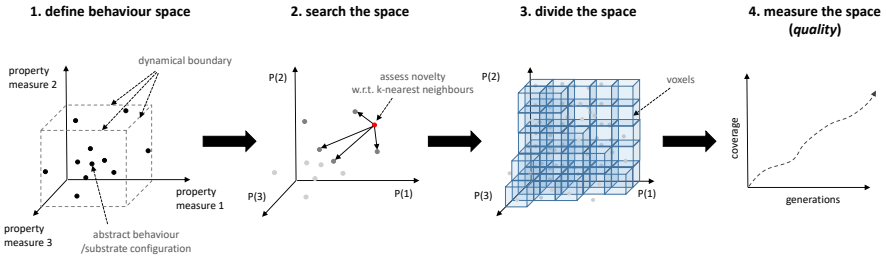


Fig. 1: CHARC framework basic workflow.

In addition to the quality measure, the framework can also be used to model relationships between dynamical behaviour and task performance, which can be used to predict performance across the different substrates, without the need to assess directly [10].

The basic framework is defined in terms of various levels to define and measure quality. The output of lower levels are used by higher levels to model relationships between parameters, dynamics and task performance.

At the base level, the computational model is chosen. The reservoir computing model is applied here: input-driven dynamics recorded as system states and extracted through a trained weighted readout layer. However, other models are possible. In the context of a model, an abstract *behaviour* space is defined. This space represents the dynamical behaviour of the substrate when configured. This space is typically different from the configuration space, where small changes in parameters result in large changes in behaviour, and vice versa.

To define the n-dimensional behaviour space, n independent property measures are used. Here, we define the same three-dimensional space used in [10], using three metrics measuring basic properties required for reservoir computing: Kernel Rank (*separation* property), Generalisation Rank (*generalisation* property), and Memory Capacity (*echo state* property). An example 3-dimensional space is shown in step 1 of the CHARC workflow, fig. 1.

Kernel rank (KR) measures the reservoir's ability to produce a rich non-linear representation of the input u and its history $u(t-1), u(t-2), \ldots$. This measures the *linear separation property*, outlined by Legenstein & Maass [16]. The generalisation rank (GR), proposed at the same time, is a measure of the reservoir's capability to generalise given similar input streams. Reservoirs in ordered regimes typically have low ranking values in both measures, and in chaotic regimes both are high. In general, a good reservoir should possess a high kernel quality rank and a low generalisation rank [4]. However, in terms of matching reservoir dynamics to tasks, the right balance is less clear.

The measure for memory capacity (MC) captures the linear short-term memory capacity of a reservoir. This measure was first outlined in [14] to quantify the *echo state* property. For the echo state property to hold, the dynamics of the

input driven reservoir must asymptotically wash out any information resulting from initial conditions, i.e., produce a fading memory.

For an outline of how to implement these measures consult [10]. As mentioned there, these three measures by themselves do not capture all the information about the reservoir's dynamical properties. To improve the accuracy of the model and the quality measure, more independent property measures are required. However, these three measures suffice to demonstrate the use of the framework to evaluate various substrates here.

The *Exploration & Mapping* level defines the search method used to construct the mapping between abstract reservoir and substrate configuration. This is shown as step 2 in fig. 1.

Exploration of the space is done using an adapted implementation of novelty search [17]. Novelty search is an open-ended genetic algorithm that navigates the behaviour space searching for novel solutions, until some user-defined termination criterion is met. Novelty search discovers a wider range of behaviours than does random search; it explores the behaviour space until it reaches the dynamical boundaries of the system. The mapping process can therefore determine the practical use of the substrate, or whether the computational model and configuration method is appropriate.

The *Evaluation* level defines the mechanisms to evaluate quality. This constitutes the final level for measuring quality. To assess quality, the behaviour space – representing abstract reservoir x, given configuration y – is divided into voxels/cells. Step 3 in fig. 1, demonstrates how the behaviour space is divided. Counting how many voxels are occupied by behaviours builds an approximation of the dynamical freedom: how many distinct reservoirs the substrate can instantiate. This acts as the measure of quality to compare across systems.

The maximum quality of a substrate measured in this way is bounded by number of search evaluations: quality $\leq$ number of evaluations. For example, given a 1000 iterations of the evolutionary search, the maximum number of voxels occupied is 1000. The method therefore requires a sufficiently large number of evaluations to get a good measure of quality.

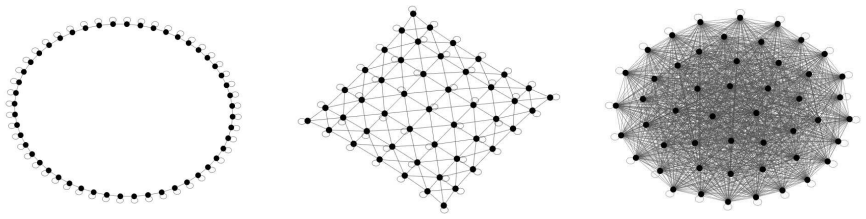
4 Simulated Network Topologies

In [23, 24], it was shown that simple and deterministic connection topologies tend to perform as well as, or better than, standard (fully-connected) randomly-generated reservoir networks on a number of benchmark tasks.

In the experiments described below, we use the CHARC framework to investigate the effect of network topology, by evaluating three simulated recurrent network topologies: *ring*, *lattice*, and *fully-connected* networks.

The ring topology (fig.2a) has the least complexity. Each node has a single self-connection and one connection to each of its neighbours to its left and right, resulting in every node having three connections.

A basic ring topology is the simplest network to implement in physical hardware as the number of connections required is small. Ring structures with various



(a) ring (b) lattice (c) fully-connected

Fig. 2: Network structures investigated here.

connection types have been applied to reservoir computing systems, including minimum complexity echo state networks (ESN) [24], DNA reservoirs (deoxyribozyme oscillators) [12], Cycle reservoirs with regular jumps (CRJ) [23], and delay-based reservoirs using a single non-linear node with a delay line [3, 21].

The lattice topology (fig.2b) has greater connection complexity. Here, we define a square grid of neurons with each connected to its Moore neighbourhood (as commonly used in cellular automata like Conway's Game of Life [2]). So each node (except for the perimeter nodes) has eight connections to neighbours and one self-connection, resulting in each node having a maximum of nine connections.

Lattice networks/models are common in computational physics, condensed matter physics and beyond, modelling physical interactions, phase transitions and structure [15]. Examples include: discrete lattices like the Ising model with variables representing magnetic dipole moments of atomic spins, and the Gray-Scott reaction-diffusion model to simulate chemical systems [22]. Also, physical substrates often have a regular grid of connections. Lattice networks are therefore more realistic representations of many physical systems that would be considered for reservoir computing.

The fully-connected topology (fig.2c) has the most connections and is considered the most complex. This type of network is challenging to implement in physical hardware. It is typically used in recurrent neural network models, such as echo state networks [13]; however, its biological plausibility is debatable.

In early work on echo state networks (ESN) it was believed these networks often work best when sparsely connected [13, 19], decoupling dynamics into smaller subsystems. However, the reservoir community is still undecided on the actual benefits of sparsity on performance.

Fully-connected networks possess the most parameters (weights) and degrees-of-freedom in the configuration space, but how this translates to dynamical behaviour is undeveloped.

In the following experiments, each network's dynamics is given by the state update equation:

$$x(t) = (1 - \alpha)x(t - 1) + \alpha f(W_{in}u(t) + Wx(t - 1)) \tag{3}$$

where x is the internal state, f is the neuron activation function (a *tanh* function), u is the input signal, W_{in} and W are weight matrices giving the connection weights to inputs and internal neurons respectively. The parameter α is the *leak rate*, controlling the time-scale mismatch between the input and reservoir dynamics; when $\alpha = 1$, the previous states do not leak into the current states.

The final trained output $y(t)$ is given when the reservoir states $x(t)$ are combined with the trained readout weight matrix W_{out}:

$$y(t) = W_{out}x(t) \tag{4}$$

5 Experiment Parameter Settings

To quantify how network structure affects dynamical behaviour, that is, reservoir *quality*, we investigate the three topologies over multiple network sizes and two internal connection types.

These two connection types define whether a connection in the reservoir is *undirected* – the weights on a link are the same in both directions, and so the weight matrix W is symmetric – or *directed* – the weight w_{ij} from node x_i to x_j may be different from the weight w_{ji} from node x_j to x_i, and so the weight matrix W is not symmetric. Considering these two connection types provides additional limits on the complexity of each network. For the fully-connected topology, we investigate only the directed type, making them equivalent to echo state networks.

The network sizes assessed for each topology are: 25, 50, 100, 200 nodes. For the lattice, these are networks with the nearest square values: 25 (5×5), 49 (7×7), 100 (10×10) and 196 (14×14). By comparing different sizes, we can determine what relationships exist between network structure and quality independent of size, and therefore whether relationships scale with network size.

Each network has many local (weight) and global (scaling) parameters under manipulation in the novelty search process. Individual weights in the input layer matrix W_{in} and the reservoir matrix W are mutated between $[-1, 1]$. Global weight scaling parameters of both matrices are also evolved. For W scaling this is a value between $[0, 2]$, and for W_{in} scaling between $[-1, 1]$. The leak rate parameter α, controlling the "leakiness" of past states into current states, is restricted between $[0, 1]$. Input and internal connectivity, the weight distribution and sparseness of W_{in} and W, are evolved by mutating between zero-value weights and non-zero weights.

The output weight matrix W_{out} for each network is used only for the memory capacity measure, as both KR and GR are calculated using only the reservoir states. When the readout layer W_{out} is in use, training is carried out using ridge regression (see [19] for training details) to minimise the difference between the network output y and the target signal y_t.

The parameters used for the novelty search algorithm are those used in [10]: *population size* = 200, *deme* = 40, *recombination rate* = 1, *mutation rate* = 0.2, $\rho_{min} = 3$, and ρ_{min} *update* = 200 generations.

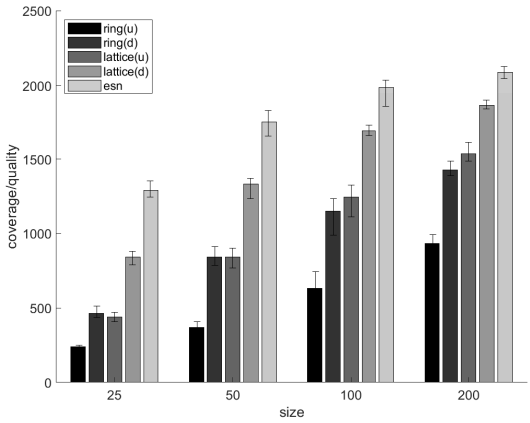


Fig. 3: Behaviour space coverage of all networks.

10 runs are conducted for each network topology, size and connection type, with 2000 generations for each run. In our implementation of novelty search (see [10]), only one new behaviour is possible every generation.

6 Results

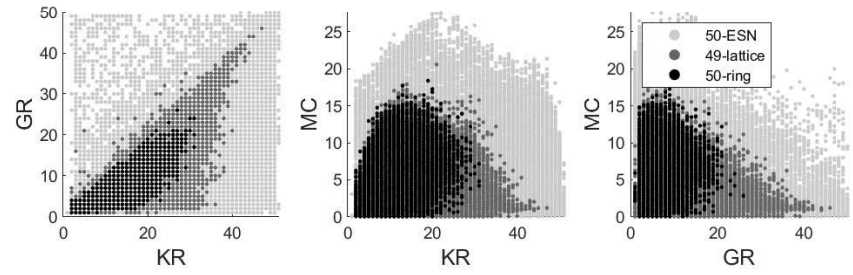
6.1 Size and Structure

Here we compare total coverage of the behaviour space, how many voxels are occupied, by each topology. The argument in [10] is that this number represents a measure of substrate quality. The maximum size of the behaviour space in this work is bound by the largest network size to the power of total number of measures, i.e., total behaviours $= 200^3$. However, a maximum of 2200 behaviours are possible in a single run here, set by the number of generations and initial population size. Hence the maximum possibly quality is 2200.

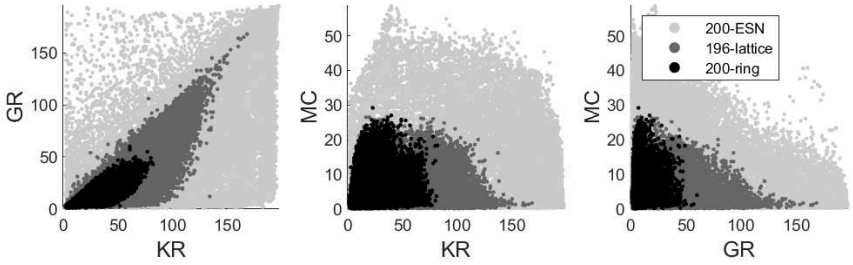
The coverage results (fig. 3) show that the fully-connected (directed) topology, the ESN network, occupies a greater area of the behaviour space, possessing a higher quality than the others, independent of size. This suggests that access to more adjustable parameters typically leads to more dynamical behaviours.

The larger ESN networks reach close to the maximum possible coverage in every run, suggesting more generations are required to better outline their limits. The others, however, struggle to reach the maximum coverage. This suggests that either new behaviours are more difficult to find, requiring additional search time, or the behavioural limits of the networks are nearly reached. A common pattern across all types is each network tends to improve in quality when increased in size. How many new behaviours are discovered by increasing in size, depends on topology and connection type.

Fig. 3 shows that the undirected lattice topology is statistically similar to the directed ring topology, across all network sizes. As implementing the lattice con-



(a) 50-neuron



(b) 200-neuron

Fig. 4: Superimposed behaviour space of: a) 50-neuron network, b) 200 neuron network. Directed topologies only. Showing all 10 runs of 2000 generations each.

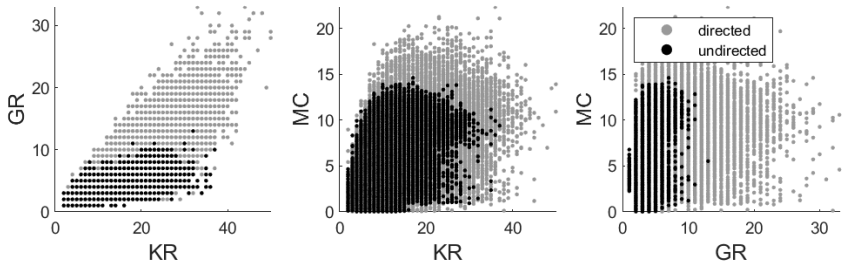
nectivity is more demanding, requiring greater complexity (up to 9 connections per node versus 3 per node), this result suggests a way to get similar dynamical behaviour to an undirected lattice with less connection complexity.

A visualisation of how two network sizes (50 and 200-neuron) cover the behaviour space using each network topology is given in fig. 4. Here, only the directed topologies are shown; for undirected results, see next section. The plot shows that ESNs tend to occupy regions the others cannot, such as chaotic regions, e.g., with high KR and high GR, and regions with larger memory capacities.

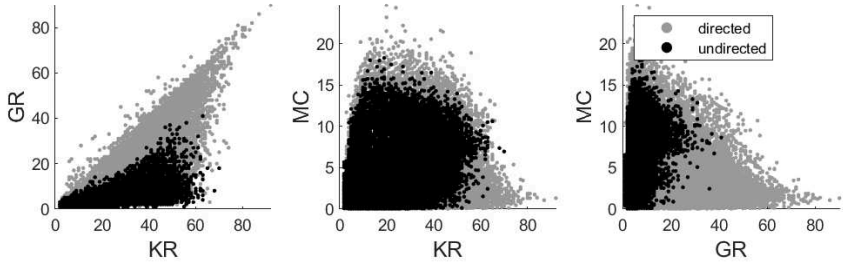
The ring and the lattice topologies have similar maximum memory capacities as each other; however, lattices typically exhibit greater non-linearity and chaotic behaviour (higher KR and GR values) than rings.

6.2 Directed vs. Undirected Networks

Here we compare the difference in quality between directed and undirected connection types. In the previous section, we show that network topology significantly affects quality. Here we show that connection type is equally as important.



(a) 100-neuron ring



(b) 100-neuron lattice

Fig. 5: Directed vs. Undirected: Superimposed behaviour space of: 100-neuron ring and lattice network. Showing all 10 runs of 2000 generations each.

To visualise this, the behaviour space coverage of a 100-neuron ring and lattice is shown in fig. 5. Each plot shows the directed connection type (grey) and undirected type (black). Here, directed connections typically result in broader dynamical behaviour, producing more "challenging" behaviours (high KR and high MC). The difficulty in producing such behaviours exists because non-linearity (KR) and ordered dynamics (MC) are often conflicting. The additional freedom of a non-symmetric weight matrix allows a broader set of behaviours to be realised.

6.3 Parameters vs. Quality

The results show the quality of each topology and connection type tends to scale linearly with the maximum number of weights available in each network configuration (fig. 6). Here, the maximum number of weights is used to represent the total freedom in the network's configuration space. In reality, the actual number of connections – with non-zero weights – required to occupy different regions of the behaviour space may differ significantly.

In fig. 6, several groups exist where networks with the same maximum number of weights produce different qualities. For example, the 25-neuron lattice

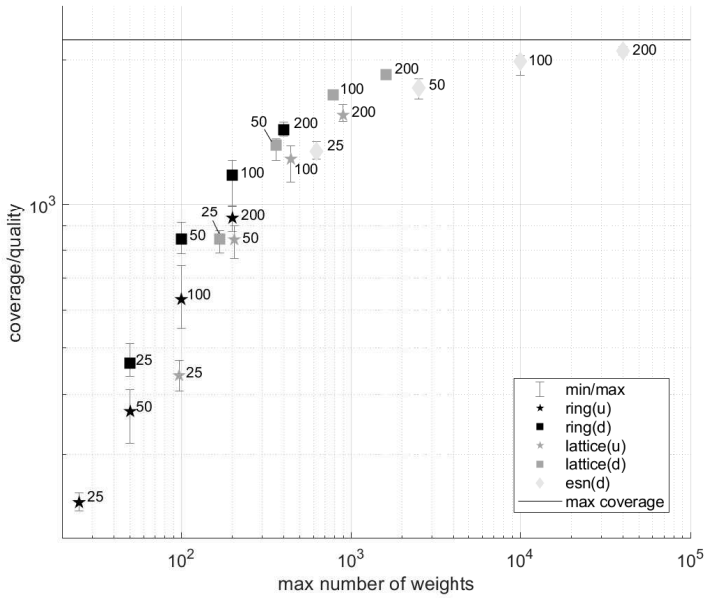


Fig. 6: The quality (coverage) of all network sizes and structures compare to maximum number of weights available by each network.

(undirected), 100-neuron ring (undirected) and 50-neuron ring (directed) are each limited to roughly 100 weights, yet differ significantly in quality/coverage. The 50-neuron ring (directed), with simpler structure, produces many more behaviours than the more complex 25-neuron lattice (undirected). This pattern also continues as both increase in network size.

It is also seen that adding more parameters (weights) does not always lead to more dynamical behaviours, e.g., the 50-neuron ring (directed) and 50-neuron lattice (undirected) produce similar qualities despite the lattice having many more available connections. It then becomes clear that how weights are structured and *directed*, controlling information flow, has a greater affect on quality of the network. This supports similar results using hierarchical networks, where structure and number of parameters also significantly impact performance [6].

7 Conclusion

Assessing and comparing how structure affects dynamical behaviour is often challenging. For unconventional substrates, determining what predefined structure and channels of information exist is even more challenging. Even if structure can be decided at creation, what is a suitable or ideal structure is often limited by physical constraints.

Here we show that the CHARC framework provides a method to assess what effect changes in structure have on computing quality. We use simulated recur-

rent networks as a model of physical reservoir systems. Our experiments show that networks with low structural complexity can exhibit similar quality and behaviours to more complex structures. However, in almost every case, this is only possible when network size is increased, resulting in a trade-off between size and structure. This result therefore acts as useful guide to the design of new unconventional reservoir computing substrates, where more complex structures tend to be more challenging to implement than simple structures like the ring topology.

Overall, this work showcases one undeveloped area within a much wider challenge: how to better understand the computing properties of substrates. The work demonstrates how the CHARC framework can be used to evaluate changes in design, configuration and representation of unconventional substrates and relate it to a task-independent measure of computing quality. In future work, we intend to further develop the framework to improve the design and fit of computational models, and even substrate design directly. For example, moving beyond the reservoir computing model, leading to a more generic CHARC framework; using the framework as a test-bed to assess unconventional program constructs; and, using quality as an objective to optimise within the substrate design process.

Acknowledgments

This work is part of the SpInspired project, funded by EPSRC grant EP/R032823/1. Jack Dewhirst is funded by an EPSRC DTP PhD studentship.

References

1. R. I. Abubakar Bala and, Idris Ismail and and S. M. S. . Applications of metaheuristics in reservoir computing techniques: A review. *IEEE Access*, 2018.
2. A. Adamatzky. *Game of life cellular automata*, volume 1. Springer, 2010.
3. L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2:468, 2011.
4. L. Büsing, B. Schrauwen, and R. Legenstein. Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural Computation*, 22(5):1272–1311, 2010.
5. J. P. Crutchfield. The calculi of emergence. *Physica D*, 75(1-3):11–54, 1994.
6. M. Dale. Neuroevolution of hierarchical reservoir computers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 410–417. ACM, 2018.
7. M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. Evolving carbon nanotube reservoir computers. In *International Conference on Unconventional Computation and Natural Computation*, pages 49–61. Springer, 2016.
8. M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. Reservoir computing in materio: An evaluation of configuration through evolution. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.

9. M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. Reservoir computing in materio: A computational framework for in materio computing. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2178–2185, May 2017.
10. M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. A substrate-independent framework to characterise reservoir computers. *arXiv preprint arXiv:1810.07135*, 2018.
11. C. Gallicchio, A. Micheli, and L. Pedrelli. Deep reservoir computing: a critical experimental analysis. *Neurocomputing*, 2017.
12. A. Goudarzi, M. R. Lakin, and D. Stefanovic. Dna reservoir computing: A novel molecular computing approach. In *DNA Computing and Molecular Programming*, pages 76–89. Springer, 2013.
13. H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.
14. H. Jaeger. *Short term memory in echo state networks*. GMD-Forschungszentrum Informationstechnik, 2001.
15. D. A. Lavis. *Equilibrium statistical mechanics of lattice models*. Springer, 2015.
16. R. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, 2007.
17. J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.
18. S. Lloyd. Ultimate physical limits to computation. *Nature*, 406(6799):1047, 2000.
19. M. Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.
20. M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.
21. Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar. Optoelectronic reservoir computing. *Scientific Reports*, 2, 2012.
22. J. E. Pearson. Complex patterns in a simple system. *Science*, 261(5118):189–192, 1993.
23. A. Rodan and P. Tiňo. Simple deterministically constructed recurrent neural networks. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 267–274. Springer, 2010.
24. A. Rodan and P. Tino. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 22(1):131–144, 2011.
25. A. Rodan and P. Tiňo. Simple deterministically constructed cycle reservoirs with regular jumps. *Neural computation*, 24(7):1822–1852, 2012.
26. B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European symposium on artificial neural networks*. Citeseer, 2007.
27. S. Stepney. The neglected pillar of material computation. *Physica D: Nonlinear Phenomena*, 237(9):1157–1164, 2008.
28. G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. Recent advances in physical reservoir computing: A review. *arXiv preprint arXiv:1808.04962*, 2018.
29. D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.
30. Y. Xue, L. Yang, and S. Haykin. Decoupled echo state networks with lateral inhibition. *Neural Networks*, 20(3):365–376, 2007.