Reservoir Computing with Random Chemical Systems

Hoang Nguyen[1], Peter Banda[2], Darko Stefanovic[3] and Christof Teuscher[1]

[1] Portland State University
[2] University of Luxembourg
[3] University of New Mexico
{hoang24, teuscher}@pdx.edu, peter.banda@uni.lu, darko@cs.unm.edu

Abstract

Top-down engineering of biomolecular circuits to perform specific computational tasks is notoriously hard and time-consuming. Current circuits have limited complexity and are brittle and application-specific. Here we propose an alternative: we design and test a bottom-up constructed *Reservoir Computer* (RC) that uses random chemical circuits inspired by DNA strand displacement reactions. This RC has the potential to be implemented easily and trained for various tasks. We describe and simulate it by means of a *Chemical Reaction Network* (CRN) and evaluate its performance on three computational tasks: the Hamming distance and a short- as well as a long-term memory. Compared with the deoxyribozyme oscillator RC model simulated by Yahiro *et al.*, our random chemical RC performs 75.5% better for the short-term and 67.2% better for the long-term memory task. Our model requires an 88.5% larger variety of chemical species, but it relies on random chemical circuits, which can be more easily realized and scaled up. Thus, our novel random chemical RC has the potential to simplify the way we build adaptive biomolecular circuits.

Introduction

Implementing a top-down chemical system relies on reasoning about the functioning of the system parts in sequential causal pathways, well isolated from each other. This rather conservative approach tries to avoid complications arising from non-linear dynamics, inherent parallelism, and concurrency of chemistry, and therefore considers them adversary. Current molecular machines, such as reprogrammable DNA self-assembly (Woods et al., 2019), require the sets of molecules and reactions to be explicitly designed. Chemical systems whose functionalities can be reprogrammed or adjusted by autonomous learning have been explored using abstract *Chemical Reaction Networks* (CRNs) (Blount et al., 2017; Banda et al., 2013, 2014), an enzymatic chemistry (Lakin et al., 2014), as well as buffered DNA strand displacement circuits (Lakin and Stefanovic, 2015, 2016). We argue that a bottom-up approach, where the species and reactions of chemical systems are selected at random, could explore the functional landscape, its phase transitions, and dynamical regimes beyond intuition by embracing the aforementioned properties.

A DNA strand displacement circuit (Soloveichik et al., 2010; Qian and Winfree, 2011) is a plausible choice for a bottom-up constructed CRN. This is due to Soloveichik's proof (Soloveichik et al., 2010) of a universal approximation of mass-action driven CRNs, which showed that complex CRNs can be obtained using DNA-based chemistry. Random chemical networks have been investigated primarily in the origins of life literature (Szathmáry, 2006; Hordijk and Steel, 2018), where several kinds of mostly (auto)catalytic reactions occurring in primordial soup throughout evolution provided the basis for closure and homeostasis. Experimentally, a network of peptides assembled randomly exhibited self-organization for predicted network connectivity (Ashkenasy et al., 2004). Furthermore, random catalytic networks produced self-replication (Segre et al., 1998) and oscillation (Stadler et al., 1993). Nevertheless, compared with non-chemical circuits, such as neural and Boolean networks, randomness in chemistry remains relatively unexplored.

In this paper, we propose that the dynamics of a random chemical system inspired by DNA strand displacement circuits is an ideal candidate for reservoir computing, a recent machine learning approach. A *Reservoir Computer* (RC) (Schrauwen et al., 2007; Lukoševičius and Jaeger, 2009) consists of a fixed, randomly connected recurrent neural network, the reservoir, which acts as a set of high-dimensional filters with fading memory, and a memoryless readout layer, which is trained by supervised learning. RC has a fairly simple mathematical model and can outperform standard machine learning algorithms, especially for temporal tasks (e.g., time series prediction). Without relying on specific species/reaction design or initial concentration, we find that a random chemical circuit can achieve complex dynamics that translate to superior learning performance. The complex dynamics of chemicals are inherently non-linear and several types of dynamical regimes are useful for designing and building an RC. As we will demonstrate, that makes such a chemical circuit an ideal candidate for RC.

In previous work on DNA reservoir computing, Goudarzi *et al.* (Goudarzi et al., 2013) and Yahiro *et al.* (Yahiro et al., 2018) have successfully used deoxyribozyme oscillators (Farfel and Stefanovic, 2006; Morgan et al., 2005) to solve temporal problems. Here we show that a DNA-inspired random chemical system, an alternative chemical building block for RC, can achieve better performance in solving time-series tasks than the deoxyribozyme oscillator RC. In particular, the random chemical RC achieves a 75.5% and 67.2% improvement in performance (NRMSE) compared with the deoxyribozyme oscillator RC in Yahiro *et al.* This RC also successfully learns the Hamming distance between two input bitstreams, with the NRMSE of 0.0246 $\pm$ 0.0063. These results demonstrate the performance and feasibility of the random chemical RC and pave the way for potential wet applications, such as detecting pathogens or gene mutations.

Model and Approach

In this section, we discuss the set of species and reactions, as well as how to generate a random chemical circuit using its network parameters. We then show how to optimize the random chemical network parameters using a standard genetic algorithm. Lastly, we describe how we use the Gillespie algorithm (Gillespie, 1977) to simulate our system.

In this DNA-inspired random chemical system, we aim to model DNA strands at a functional level; we omit specification of domains or base pairs, and instead we impose general constraints on possible reactions, which in turn should allow transformation to real DNA circuits if needed. In other words, we propose mass-action driven chemical systems (Espenson, 1995; Arnaut et al., 2006) that match real DNA strand displacement systems reasonably well, but the specific sequence design is beyond the scope of this work.

This abstraction enables us to ask questions about the overall capabilities of the DNA strand circuits as a family of chemical systems and draw interesting conclusions about their dynamics. Since each mass-action chemistry is writable as a DNA strand model, we could say that there is no need to limit ourselves to DNA strands only. DNA strand chemistry is universal, just as mass-action chemistry, so everything computable could be expressed in both. However, circuits that are already in DNA strand format could be implemented directly without the intermediate transformation (Soloveichik et al., 2010). Second, we want to discover what mix of random DNA strands produces dynamics promoting information processing and computing.

Random Chemical Circuit Species and Reactions

Our proposed DNA-inspired random chemical circuit is a *Chemical Reaction Network* (CRN) (Arceo et al., 2015), which consists of a set of species and reactions (Dittrich et al., 2001). The model for our random chemical species and reactions inherits those in the DNA strand displacement circuit (Soloveichik et al., 2010; Qian and Winfree, 2011). Here, we postulate four abstract types of species in the network: upper strand, lower strand, partial double strand, and full double strand. Upper and lower strands are similar to the single-stranded DNA molecules such that only opposite strand types can bind. We assume this property is guaranteed by sequence design (Zadeh et al., 2011). What we refer to as full double strand is a species that is inspired by a perfect Watson-Crick DNA double strand, where all base pairs of corresponding upper and lower strands are complementary. Partial double strand is similar to full double strand, but instead of all pairs, only substrings of upper and lower strands match.

These molecules are involved in four types of reactions: binding, displacement, influx, and efflux. Fig. 1 illustrates different variations of strand binding and displacement reactions. In binding reactions, an upper strand binds to its complementary lower strand and forms a full double strand (Fig. 1a), or an upper strand and a lower strand overlap at certain substring and produce partial double strand (Fig. 1b).

A displacement reaction happens between a single strand and a partial double strand. In this type of reaction, there are four scenarios: upper strand displaced by full complement (Fig. 1c), upper strand displaced by another hierarchically stronger (longer) strand (Fig. 1d), lower strand displaced by full complement (Fig. 1e), lower strand displaced by another hierarchically stronger (longer) strand (Fig. 1f).

Overall, the two binding and four displacement reactions in our random chemical system are similar to those binding and displacement reactions that are commonly used in DNA strand circuits. Note that the maximum number of strands that bind together is two, and we forbid formation of triple or higher strands by assuming that a single strand does not bind to a partial double strand, but always displaces its upper or lower part. Again, we assume this can be guaranteed by DNA sequence design. The rationale for doing this is the absence of molecular structure in our model.

Using the above-described model, we can set up and test our random chemical circuit. We assume the system is a microscale continuous stirred-tank reactor (μCSTR) (Farfel and Stefanovic, 2006; Morgan et al., 2005).

Random Chemical Circuit Generation

At this point, we start treating a system of the aforementioned strand species as a network. More specifically, we draw upper and lower single strands as nodes, and connect two nodes wherever full double (solid line) or partial double (dashed line) strand consisting of single strands exists. Note that because each upper single strand has a maximum of one complementary lower strand, there is a maximum of one solid line coming out of each node. We generate random networks using nine parameters. There are five general parameters: the number of single strands n, the ratio of upper to lower strands ρ, the ratio of upper strands with complements γ, the ratio of influx to the overall number of strands

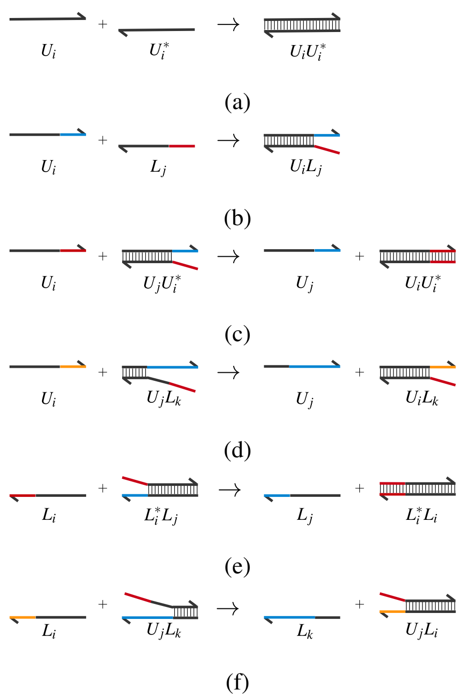

Figure 1: Assumed binding and displacement reactions: a) full double strand creation, b) partial double strand creation, c) upper strand displaced by full complement, and d) upper strand displaced by another hierarchically stronger strand, e) f) lower strand versions of full and partial strand displacements. Upper strands are labeled as U, lower as L. The star ($*$) notation indicates complementarity.

α_{in}, and the ratio of efflux to the overall number of strands α_{out}. There are four random distribution parameters: the normal distribution of rate constant θ, the normal distribution of influx rate θ_{in}, the normal distribution of efflux rate θ_{out}, and the normal distribution of partial double strands per upper strand ϕ.

We then apply the network generation process to generate a random chemical network. Specifically, these steps determine the types and the names of the species in the network, the order of strands for strand displacement reactions, and the influx/efflux of the chemistry, with respect to the network parameters described above.

The ordering corresponds to the length of the overlapping sub-sequences for a given upper or lower strand so only the strands with longer overlap kick out those with shorter overlap from the complex. Note that fully complementary Watson-Crick strands always have the highest order over all competing strands. Now, we impose the ordering of partial double strands globally, where the partial double strands are ordered from the perspective of the entire system. This prevents cyclic displacement.

```
**Input:** Parameters $n$, $\rho$, $\gamma$, $\phi$, $\theta$, $\alpha_{in}$, $\alpha_{out}$, $\theta_{in}$, $\theta_{in}$
**Output:** Random Chemical Network
Determine $n_L = n \div (1+\rho)$, and $n_U = n - n_L$.
Create upper and lower strands as nodes of the network.
Determine $n_F = \gamma \times min(n_L, n_U)$.
Choose $n_F$ complementary upper and lower strands randomly. Connect them with solid lines.
**for** *each upper strand* **do**
    Draw $n_P$ partial double strands from distribution $\phi$.
    Choose randomly given $n_L$ counterparts without repetitions.
    Omit lower strand from selection if already selected as complementary.
**end**
Impose ordering of partial double strands for strand displacement reactions so only strands with higher order displace strands from the complex.
Choose random influx and efflux strands from parameters $\alpha_{in}$ and $\alpha_{out}$.
Generate random rate constants from distribution $\theta$.
Generate random rate constants for influxes and effluxes from distributions $\theta_{in}$ and $\theta_{out}$.
```

Algorithm 1: Random chemical circuit network generation process. n_L, n_U, n_F, n_P refer to the number of lower strands, the number of upper strands, the number of full double strands, and the number of partial double strands.

Chemistry Parameter Optimization

Since the space of possible random chemical networks for our parameter space is large, it would be difficult and time-consuming to sample it blindly in a trial-and-error fashion or by exhaustive search. We therefore employ a standard *Genetic Algorithm* (GA) to optimize the parameters.

Possible values of the network parameters are encoded in a chromosome. The genetic search looks for the classes of the random chemical circuit that have the lowest *Normalized Root-Mean-Square Error* (NRMSE) from the Hamming distance learning task that will be discussed. The fitness is defined as the average NRMSE over 10 experiments and 10 learning epochs of the Hamming distance task. The circuits are randomly generated chemical circuits using parameter values from the chromosome and random initial concentrations drawn from the uniform (0,1) interval. The NRMSE is normalized over all species.

The GA combines elite selection with one point crossover and per-element mutation. Since only certain values of parameters are plausible, we restrict their value range for the mutation and the generation of initial population as shown in Table 1. The setting and constants for the GA are: population size M = 32, elite size E = 16, cross-over probability p_c = 1.0, per-element mutation probability p_m = 0.5, and gen-

Table 1: The bounds of random chemical network parameters used during mutation and the generation of the initial population in the GA.

General Parameter	Bound
n	[5,10)
ρ	[0.5,1)
γ	[0,1)
α_{in}	[0,1)
α_{out}	[0,1)
Normal Distribution Parameter	Bound
θ	[0.05,0.2) $\pm$ [0,0.02)
θ_{in}	[0,0.0006)
θ_{out}	[0,0.0006)
ϕ	[0,4) $\pm$ [0,0.5)

eration limit G = 32.

We then draw classes of random chemical circuits from the parameter bounds as specified in Table 1 during the genetic search. Table 2 gives the optimized parameters and their values that we used to characterize our CRN.

Table 2: Values of network parameters that provide the best performance (lowest NRMSE) in the Hamming distance task.

General Parameter	Value
n	9
ρ	0.846
γ	0.214
α_{in}	0.222
α_{out}	0.0193
Normal Distribution Parameter	Value
θ	0.148 $\pm$ 0.00530
θ_{in}	0.000344
θ_{out}	0.000152
ϕ	2.48 $\pm$ 0.136

It is worth noting that a class of random chemical circuit can be generated using any parameter values within the bounds in Table 1. However, in the experiments that follow, we picked the parameter values from the genetic search to generate the network.

Stochastic Simulation and Perturbation

We use the Gillespie algorithm (Gillespie, 1977) for a stochastic simulation of our random chemical circuits. We chose a stochastic approach because we are working with a small volume and we wanted to capture the random behaviors of the chemistry.

Fig. 2 shows the setup of the random chemical circuit class that has the best learning performance on the Hamming distance task. Since the number of single strands n is 9, and the ratio of upper to lower strands ρ is 0.846, we know that the number of upper and lower strands is 4 and 5, respectively. Also, from the ratio of upper strands with complements $\gamma = 0.214$, and the (positive) normal distribution of partial double strands per upper strand $\phi = 2.48 \pm 0.136$, we can draw the full double (solid line) and partial double strands (dashed line) as shown in Fig. 2. Lastly, the ratio of influx to the overall number of strands α_{in} is 0.222 and the ratio of efflux to the overall number of strands α_{out} is 0.0193, so there are 6 influx species and 1 efflux species in the network.

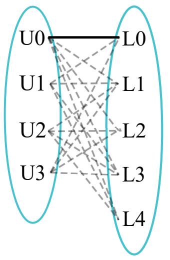

Figure 2: Network representation of the optimized random chemical circuit class. Nodes are single strands, which divide the network into upper strands U (left), and lower strands L (right). Solid lines represent full double strands, dashed lines represent partial strands. For this setup, the species are: Single (U0, U1, U2, U3, L0, L1, L2, L3, L4); Full double (U0L0); Partial double (U0L1, U0L2, U0L3, U0L4, U1L0, U1L1, U1L3, U1L4, U2L1, U2L2, U2L3, U2L4, U3L0, U3L1, U3L2, U3L3).

We can then apply stochastic simulation to this random chemical circuit class. The initial number of species in the chemistry is set at random for all strands. During the simulation, we introduce perturbations to the chemistry by adjusting the base influx rate parameter θ_{in}. This is done by multiplying the base influx rate θ_{in} with a uniformly distributed random number R to create a new influx rate $\theta_{in}^{'} = \theta_{in}R$.

Perturbation starts at time t_p and ends with the stochastic simulation, denoted t_{end}. The amount of time between two consecutive perturbations is the hold time τ. The number of perturbations N then is: $N = \left\lceil \frac{t_{end}-t_p}{\tau} \right\rceil$

In our experiments, perturbation happens from $t_p = 0.01$ seconds to $t_{end} = 1$ second. If the hold time τ is 0.1 seconds, there are $N = 10$ perturbations happening from time $t = 0.01$ seconds to $t = 0.91$ seconds.

Applying RC to Random Chemical Circuit

Reservoir Computing Overview

Reservoir Computing (RC) is a relatively recent machine learning technique that provides real-time computing on reservoir transitions without the need of stable states. RCs outperform the classic recurrent neural networks on time series processing and prediction. A simple RC consists of three components: the inputs of the reservoir, the main reservoir, and the readout layer giving the outputs of the reservoir. Sketches of RCs are shown in Fig. 3 and Fig. 4.

The best known instances of RC are *Echo State Networks* (ESN) (Schrauwen et al., 2008; Büsing et al., 2010) and *Liquid State Machines* (LSM) (Maass et al., 2002). We argue that the basic idea of RC in its most general form—process inputs through a randomly generated component with rich dynamics and then map its states to the output layer by means of simple linear integration—also applies to domains beyond the neural-network or logic circuit formalisms. Models of liquid state machines, for example, were implemented in a bucket of water (Fernando and Sojakka, 2003) and in E. coli (Jones et al., 2007) demonstrated that this approach could extend to spatial or network-based systems.

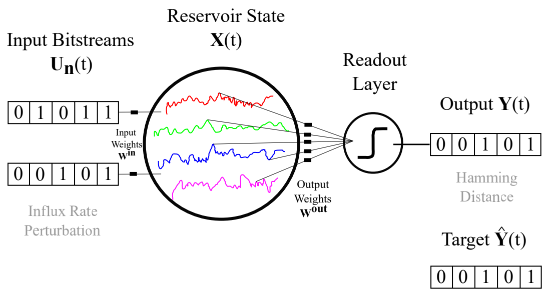

Figure 3: Abstract RC for learning the Hamming distance between two input bitstreams.

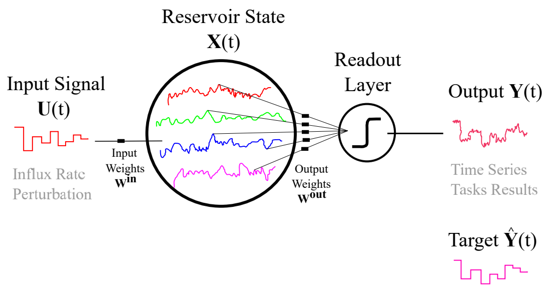
Reservoir State
$\mathbf{X}$(t)
Input Signal
$\mathbf{U}$(t)
Readout
Layer
Output $\mathbf{Y}$(t)
Input Weights $\mathbf{W^{in}}$
Output Weights $\mathbf{W^{out}}$
Influx Rate Perturbation
Time Series Tasks Results
Target $\hat{\mathbf{Y}}$(t)

Figure 4: Abstract RC for learning the short- and long-term memory task.

Let I be the number of reservoir inputs, N the number of nodes inside the reservoir, and O the number of reservoir outputs. Let i the input node index, j and k the reservoir node indexes, and l the output node index. For the input side, we introduce $U(t) = [u_i(t)]$ as the I-dimensional input vector, and $W^{in} = [w^{in}_{ij}]$ as the $N \times I$ input weights matrix. For the main reservoir, we introduce $X(t) = [x_j(t)]$ as the N-dimensional vector of the reservoir state, and $W^{res} = [w^{res}_{jk}]$ as the $N \times N$ reservoir weights matrix. Lastly, on the readout side, we introduce $Y(t) = [y_k(t)]$ as the O-dimensional output vector, and $W^{out} = [w^{out}_{kl}]$ as the $O \times N$ output weight matrix.

The reservoir state vector $X(t)$ can be determined as $X(t+1) = f(W^{res} \cdot X(t) + W^{in} \cdot U(t))$.

Here f is the non-linear transfer function of the reservoir nodes. The output vector $Y(t)$ can also be determined using linear combination of the $X(t)$ vector, $Y(t) = W^{out} \cdot X(t) + w_b$, where w_b is an inductive bias. Since the purpose of reservoir computing is only training the output weights, the input weights matrix W_{in} and reservoir weights matrix W_{res} are kept fixed. The output weights matrix W_{out} can be trained using any linear regression method. Similar to (Goudarzi et al., 2013) and (Yahiro et al., 2018), we employed the simple Moore-Penrose pseudo-inverse method (Penrose, 1955): $W^{out'} = (X'^T \cdot X')^{-1} \cdot X'^T \cdot \hat{Y}'$.

In this equation, $W^{out'}$ is the matrix W^{out} with the bias w_b added. X' is the observation matrix of the reservoir, where each row represents the reservoir state in time and each column represents the state at different nodes such that the last column is a constant 1. Lastly, $\hat{Y}'$ is the target vector.

Random Chemical Circuit RC

Fig. 5 shows an abstract reservoir computer made of random chemical species that form the reservoir. The time-varying inputs of the reservoir are the time-varying influx rate changes caused by our perturbations. Starting at time t_p, we inject a number of these species into the system. The perturbation is held for an amount τ until we inject the next batch of species into the reservoir. The number of reservoir inputs depends on the number of species being perturbed in the random chemical circuit. For instance, if we design a two-input reservoir, then we need to perturb two species during our stochastic simulation.

The reservoir state consists of the time-varying species count of all the strands present in the chemistry during the stochastic simulation. The species count is then weighted with a uniformly-distributed random number, and is projected into a readout layer.

The readout layer of the RC is a single perceptron that uses a feedforward and backpropagation step to map the actual output to a desired target by adjusting the weights. Note that in our setup, the readout layer and its training happen outside of the random chemical systems. Training in chemistry can be done down the road (see e.g., (Blount et al., 2017)) and analogous designs could be integrated, but it is beyond the scope of this work. Here, we instead aim to focus on the computational capacity of a reservoir made of a certain restricted class of chemical reactions.

In our experiments, we set the learning rate to be 0.001 and train the readout layer for 10 epochs. We then use the NRMSE to evaluate the performance of our RC model and compare with other DNA RC models.

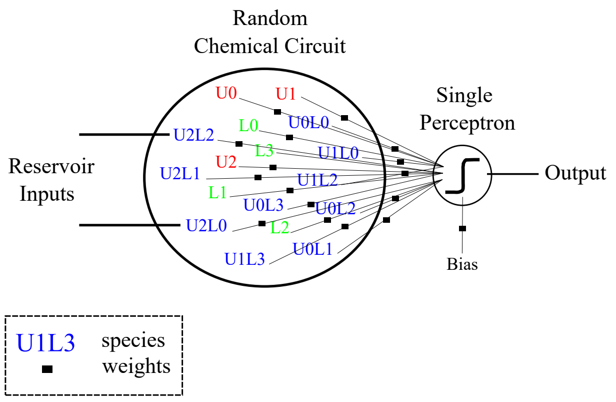

Figure 5: RC model using a random chemical circuit.

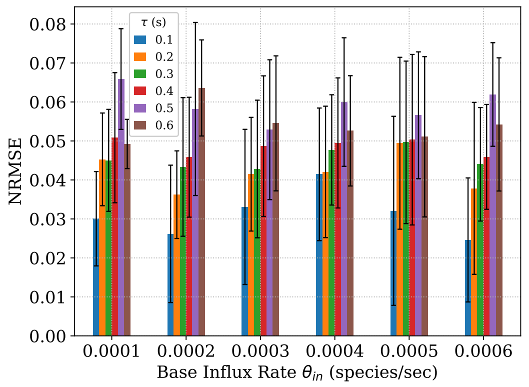

Figure 6: NRMSE of the Hamming distance between two input bitstreams with various lengths, determining by the input hold time τ, learned by the random chemical RC. In general, as the input hold time increases, the error gets higher. The whiskers represent the standard deviation over 20 simulations. This can be reduced by averaging over a larger number of simulations.

Hamming Distance Task

The Hamming distance task consists in learning the Hamming distance between two input bitstreams. We use a two-input reservoir, where we chose two species to be the influx species. The two time-series vectors of influx rates were then converted into two bitstreams that are fed into the reservoir. The number of bits of an input is defined as the number of perturbation N. The target vector contains information whether a bit is flipped between the two inputs, and the sum of all flipped bits is the expected Hamming distance.

Results Fig. 6 shows the NRMSE of the RC for the Hamming distance task. The base influx rate θ_{in} is changed from 0.0001 to 0.0006 species/second with a 0.0001 species/second increment, while the input hold time τ is changed from 0.1 to 0.6 seconds with a 0.1 seconds increment. Each setup is averaged over 20 runs. The random chemical circuit with the best performance has an NRMSE of 0.0246 ± 0.0063.

Time-Series Tasks

We also test the random chemical RC with two time-series learning problems: the short-term memory task and the long-term memory task. These tasks are the simplified version of the popular RC benchmark NARMA and require the reservoir to remember past inputs (Goudarzi et al., 2013). The NRMSE introduced above is used as a means to measure the performance of the RC.

Short-term memory task The target of the short-term memory task is defined as: $\hat{Y}(t) = \theta^k_{in}(t-1) + 2\theta^k_{in}(t-2)$ $\hat{Y}(t)$ is the target vector as a function of time t. $\theta^k_{in}(t)$ is the influx rate $\theta_{in}(t)$ of the kth species, as a function of time.

Long-term memory task The target of the long-term memory task is defined as: $\hat{Y}(t) = \theta^k_{in}(t-\tau) + \frac{1}{2}\theta^k_{in}(t-\frac{3}{2}\tau)$ $\hat{Y}(t)$ is the target vector as a function of time t. $\theta^k_{in}(t)$ is the influx rate $\theta_{in}(t)$ of the kth species, as a function of time. τ is the time between each perturbation (the input hold time).

Results The results of both the short-term and long-term memory task are shown in Fig. 7 and Fig. 8, respectively. We ran 20 simulations for each setting with different input hold times τ and base influx rates θ_{in}. The input hold time τ (seconds) was changed from 0.1 to 0.6 with a 0.1 increment. This allows us to look at the system with different number of perturbations, specifically from 2 perturbations to 10 perturbations. The base influx rate θ_{in} (species/sec) was changed from 0.0001 to 0.0006 with a 0.0001 increment. The range of the base influx rate sweeping was chosen based on the bounds in Table 1.

Model Comparison Fig. 9 shows that our random chemical circuit RC achieves 81.9% and 61.2% better performance than the deoxyribozyme oscillator RC in Goudarzi *et al.* (Goudarzi et al., 2013) for both the short-term and long-term memory task, respectively. Furthermore, our RC model achieves 75.5% and 67.2% better performance than the deoxyribozyme oscillator RC in Yahiro *et al.* (Yahiro et al., 2018). In particular, the deoxyribozyme oscillator RC in Goudarzi *et al.* achieves 0.23 ± 0.05 and 0.11 ± 0.02, the deoxyribozyme oscillator RC in Yahiro *et al.* achieves 0.17 $\pm$ 0.034 and 0.13 $\pm$ 0.036, and the random chemical RC achieves 0.0416 ± 0.0072 and 0.0427 ± 0.0085, in NRMSE for short-term and long-term memory task, respectively.

However, the most important trade-off of using the random chemical RC is the "cost" of implementation, measured in the number of species. Our model requires 26 species, while the deoxyribozyme oscillator RC models only require 3 species. Hence, our system is 88.5% larger in size. For future improvements, we can filter out the species corresponding to the nodes with negligible weights after training. This

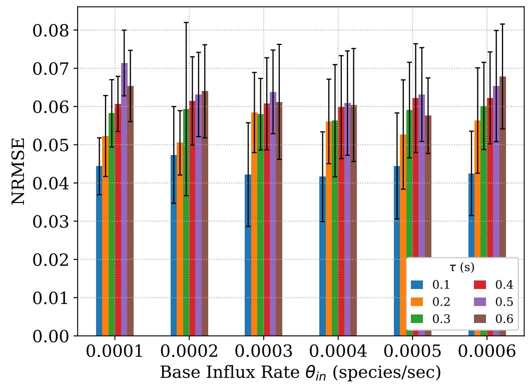

Figure 7: Short-term memory task NRMSE for the random chemical circuit RC. The reservoir achieves the best performance (lowest NRMSE = 0.0416 $\pm$ 0.0072) when the base influx rate θ_{in} = 0.004 species/second and input hold time τ = 0.1 seconds. The whiskers represent the standard deviation over 20 runs. This can be reduced by averaging over a larger number of experiments.

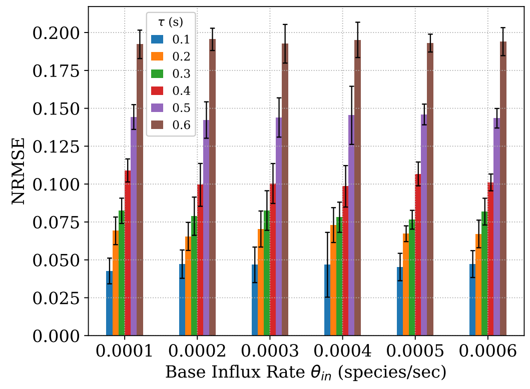

Figure 8: Long-term memory task NRMSE for the random chemical circuit RC. The reservoir achieve the best performance (NRMSE = 0.0427 $\pm$ 0.0085) when the base influx rate θ_{in} = 0.001 species/second and input hold time τ = 0.1 seconds. The whiskers represent the standard deviation over 20 runs. This can be reduced by averaging over a larger number of experiments.

would reduce the size with a small compromise on performance.

It is worth noting that Yahiro et al. performed additional experiments beyond their aforementioned results, where they investigated the reservoirs using non-negative linear regression, explored the effects of the reservoir size on its performance, and concluded the trade-off between size and performance. Using evolutionary optimization, they found the best performances with a NRMSE of 0.06 for a 15-species reservoir under the normal condition, and a NRMSE of

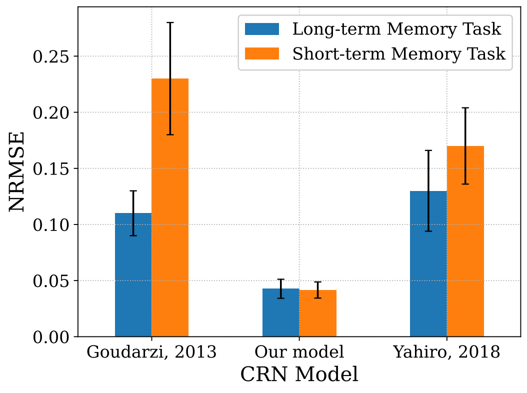

Figure 9: Performance comparison between the deoxyribozyme oscillator RCs and our random chemical RC. The random chemical circuit RC has its performance improved 81.9% and 61.2% compared to the deoxyribozyme oscillator RC in Goudarzi *et al.* for the short-term and long-term memory task, respectively. Also, the performance of our RC model improves by 75.5% and 67.2% in NRMSE compared to the deoxyribozyme oscillator RC in Yahiro *et al.*

0.14 for a 10-species reservoir under the non-negative condition. However, the authors claimed that these performance were not maintained and might be an artifact of over-fitting (Yahiro et al., 2018).

Conclusion

In this paper, we have introduced and simulated a stochastic model of a random chemical circuit that was inspired by DNA strand displacement reactions. The reaction dynamics of the chemical species were used as a reservoir in an RC to learn the time-series tasks. Compared with the state of the art (Goudarzi et al., 2013; Yahiro et al., 2018), our novel approach achieved better performance on both the short-term (best performance NRMSE = 0.0416 $\pm$ 0.0072) and the long-term memory task (best performance NRMSE = 0.0427 $\pm$ 0.0085). Specifically, the random DNA strand circuit RC outperforms the deoxyribozyme oscillator RC in Goudarzi *et al.* by 81.9% and 61.2%, and in Yahiro *et al.* by 75.5% and 67.2%, for the short-term and long-term memory task, respectively.

Our novel random chemical reservoir computing approach has the potential to simplify the way we build adaptive biomolecular circuits. Such circuits could have applications in the area of biomedical diagnosis, pathogen detection, and industrial monitoring.

While our CRN model is still abstract and would require more experiments to construct the actual DNA sequences, we will reserve to future work the design of more realistic DNA strand displacement circuits that can emulate the desire behavior in Figure 1.

Acknowledgements

This material is based upon work supported by the National Science Foundation under grants no. 1518833 and 1518861.

References

Arceo, C. P. P., Jose, E. C., Marin-Sanguino, A., and Mendoza, E. R. (2015). Chemical reaction network approaches to Biochemical Systems Theory. *Mathematical Biosciences*, 269:135–152.

Arnaut, L., Burrows, H., and Formosinho, S. (2006). *Chemical Kinetics: From Molecular Structure to Chemical Reactivity*. Elsevier. Google-Books-ID: 8DBib3zq6LIC.

Ashkenasy, G., Jagasia, R., Yadav, M., and Ghadiri, M. R. (2004). Design of a directed molecular network. *Proceedings of the National Academy of Sciences*, 101(30):10872–10877. Publisher: National Academy of Sciences Section: Physical Sciences.

Banda, P., Teuscher, C., and Lakin, M. R. (2013). Online learning in a chemical perceptron. *Artificial Life*, 19(2):195–219.

Banda, P., Teuscher, C., and Stefanovic, D. (2014). Training an asymmetric signal perceptron through reinforcement in an artificial chemistry. *Journal of the Royal Society, Interface*, 11(93):20131100.

Blount, D., Banda, P., Teuscher, C., and Stefanovic, D. (2017). Feedforward Chemical Neural Network: An In Silico Chemical System That Learns xor. *Artificial Life*, 23(3):295–317.

Büsing, L., Schrauwen, B., and Legenstein, R. (2010). Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural Computation*, 22(5):1272–1311.

Dittrich, P., Ziegler, J., and Banzhaf, W. (2001). Artificial chemistries–a review. *Artificial Life*, 7(3):225–275.

Espenson, J. H. (1995). *Chemical kinetics and reaction mechanisms*, volume 102. McGraw-Hill New York.

Farfel, J. and Stefanovic, D. (2006). Towards Practical Biomolecular Computers Using Microfluidic Deoxyribozyme Logic Gate Networks. In Carbone, A. and Pierce, N. A., editors, *DNA Computing*, Lecture Notes in Computer Science, pages 38–54, Berlin, Heidelberg. Springer.

Fernando, C. and Sojakka, S. (2003). Pattern Recognition in a Bucket. In Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., and Kim, J. T., editors, *Advances in Artificial Life*, Lecture Notes in Computer Science, pages 588–597, Berlin, Heidelberg. Springer.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361. Publisher: American Chemical Society.

Goudarzi, A., Lakin, M. R., and Stefanovic, D. (2013). DNA Reservoir Computing: A Novel Molecular Computing Approach. In Soloveichik, D. and Yurke, B., editors, *DNA Computing and Molecular Programming*, Lecture Notes in Computer Science, pages 76–89, Cham. Springer International Publishing.

Hordijk, W. and Steel, M. (2018). Autocatalytic Networks at the Basis of Life's Origin and Organization. *Life*, 8(4).

Jones, B., Stekel, D., Rowe, J., and Fernando, C. (2007). Is there a Liquid State Machine in the Bacterium Escherichia Coli? In *2007 IEEE Symposium on Artificial Life*, pages 187–191. ISSN: 2160-6382.

Lakin, M. R., Minnich, A., Lane, T., and Stefanovic, D. (2014). Design of a biochemical circuit motif for learning linear functions. *Journal of the Royal Society Interface*, 11(101).

Lakin, M. R. and Stefanovic, D. (2015). Supervised Learning in an Adaptive DNA Strand Displacement Circuit. In *Proceedings of the 21st International Conference on DNA Computing and Molecular Programming - Volume 9211*, DNA 21, pages 154–167, Boston and Cambridge, MA, USA. Springer-Verlag.

Lakin, M. R. and Stefanovic, D. (2016). Supervised Learning in Adaptive DNA Strand Displacement Networks. *ACS synthetic biology*, 5(8):885–897.

Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.

Morgan, C., Stefanovic, D., Moore, C., and Stojanovic, M. N. (2005). Building the Components for a Biomolecular Computer. In Ferretti, C., Mauri, G., and Zandron, C., editors, *DNA Computing*, Lecture Notes in Computer Science, pages 247–257, Berlin, Heidelberg. Springer.

Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413. Publisher: Cambridge University Press.

Qian, L. and Winfree, E. (2011). Scaling up digital circuit computation with DNA strand displacement cascades. *Science (New York, N.Y.)*, 332(6034):1196–1201.

Schrauwen, B., Büsing, L., and Legenstein, R. (2008). On computational power and the order-chaos phase transition in reservoir computing. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'08, pages 1425–1432, Vancouver, British Columbia, Canada. Curran Associates Inc.

Schrauwen, B., Verstraeten, D., and Van Campenhout, J. (2007). An overview of reservoir computing: theory, applications and implementations. *Proceedings of the 15th European Symposium on Artificial Neural Networks. p. 471-482 2007*, pages 471–482.

Segre, D., Lancet, D., Kedem, O., and Pilpel, Y. (1998). Graded Autocatalysis Replication Domain (GARD): kinetic analysis of self-replication in mutually catalytic sets. *Origins of Life and Evolution of the Biosphere: The Journal of the International Society for the Study of the Origin of Life*, 28(4-6):501–514.

Soloveichik, D., Seelig, G., and Winfree, E. (2010). DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398. Publisher: National Academy of Sciences Section: Biological Sciences.

Stadler, P. F., Fontana, W., and Miller, J. H. (1993). Random catalytic reaction networks. *Physica D: Nonlinear Phenomena*, 63(3):378–392.

Szathmáry, E. (2006). The origin of replicators and reproducers. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 361(1474):1761–1776. Publisher: Royal Society.

Woods, D., Doty, D., Myhrvold, C., Hui, J., Zhou, F., Yin, P., and Winfree, E. (2019). Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature*, 567(7748):366–372. Number: 7748 Publisher: Nature Publishing Group.

Yahiro, W., Aubert-Kato, N., and Hagiya, M. (2018). A reservoir computing approach for molecular computing. *Artificial Life Conference Proceedings*, 30:31–38. Publisher: MIT Press.

Zadeh, J. N., Steenberg, C. D., Bois, J. S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M., and Pierce, N. A. (2011). NUPACK: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.21596.