DOI: http://dx.doi.org/10.7551/978-0-262-32621-6-ch078

An Analog Chemical Circuit with Parallel-Accessible Delay Line for Learning Temporal Tasks

Peter Banda[1] and Christof Teuscher[2]

[1]Department of Computer Science, Portland State University
banda@pdx.edu
[2]Department of Electrical and Computer Engineering, Portland State University
teuscher@pdx.edu

Abstract

Current synthetic chemical systems lack the ability to self-modify and learn to solve desired tasks. In this paper we introduce a new parallel model of a chemical delay line, which stores past concentrations over time with minimal latency. To enable temporal processing, we integrate the delay line with our previously proposed analog chemical perceptron. We show that we can successfully train our new memory-enabled chemical learner on four non-trivial temporal tasks: the linear moving weighted average, the moving maximum, and two variants of the Nonlinear AutoRegressive Moving Average (NARMA). Our implementation is based on chemical reaction networks and follows mass-action and Michaelis-Menten kinetics. We show that despite a simple design and limited resources, a single chemical perceptron extended with memory of variable size achieves 93-99% accuracy on the above tasks. Our results present an important step toward actual biochemical systems that can learn and adapt. Such systems have applications in biomedical diagnosis and smart drug delivery.

Keywords
chemical delay line, chemical perceptron, chemical reaction network, analog asymmetric signal perceptron, temporal learning, chemical computing

Introduction

Learning and adaptation allow organisms to generalize and predict to the ever-changing environment they live in, which leads to a competitive advantage for their survival and reproduction. Artificial neural networks (Rojas, 1996; Haykin, 2009) are the prototypical example of artificial learning, however, their abstraction of biological systems is usually high. Building synthetic or actual molecular systems that can adapt and learn autonomously is a grand challenge for various reasons. First, the integration of the forward-pass (output production) with the backward-pass, where a teacher's action affects the concentrations of biomolecular species that define the system's functionality, requires complicated and timing-sensitive, both positive and negative catalytic feedback loops. Second, there is no natural and immediate mapping of neural networks to chemical systems. So far, only the input-weight integration has been successfully implemented in wet chemistry (Bray, 1995; Mills et al., 1999; Kim et al., 2004; Qian et al., 2011).

In our previous work we proposed several binary and analog chemical perceptrons (Banda et al., 2013, 2014; Banda and Teuscher, 2014). These systems represented the very first simulated chemical systems capable of fully autonomous learning. We used chemical reaction networks (CRNs), which are unstructured macroscopic chemistries where the reactions of symbolic species follow mass-action and/or Michaelis-Menten kinetics. While our previous systems were able to solve simple tasks, the goal of this paper is to propose new chemical learning systems that can tackle much more complex tasks, in particular tasks with temporal dependencies. This challenge is motivated by practical computational as well as biomedical applications. For example, any form of continuous environmental sensing is typically defined temporally, where the target function output depends not only on the actual but also on the previous inputs.

In current chemical reaction networks, it is difficult to coherently store and retrieve values as we are used to in traditional computer architecture. Here, we propose a specific kind of memory widely used in computer and electrical engineering, a delay line (DL), which buffers the past inputs over a sliding window and presents them for reading (consumption) both sequentially but also as a parallel output. Our previous chemical DLs (Moles et al., 2014) lack the ability to scale up and provide solely a sequential access to the content. To address these issues we introduce a new DL called a parallel-accessible DL (PDL), which achieves optimal performance by employing wait queues and operates on the basis of two alternating signals (catalysts).

We then integrate a PDL with an analog asymmetric signal perceptron (AASP), a chemical perceptron trained by supervised learning. In our setup, a PDL feeds an underlying AASP with past input concentrations, and therefore allows to solve tasks defined as time series. We evaluated the performance of our new delayed AASP (AASP-DL) on four temporal tasks: the linear moving weighted average, the moving maximum, and two variants of the Nonlinear AutoRegressive Moving Average (NARMA). We also scale the

system's size to more than two cached inputs to study the effect of memory on learning.

Given the available technology, the AASP-DL may have applications in the area of medical diagnosis and smart medication (LaVan et al., 2003), such as temporal signal processing, monitoring and detection of specific concentration patterns produced, e.g., by cancer cells in a host. Our approach could potentially replace hard-coded solutions and would allow to reuse (retrain) chemical systems without redesigning them. We suggest that our model could be used as a system-level blue-print for actual wet biochemical (substrate-specific) implementations, for example by using DNA-strand displacement (Soloveichik et al., 2010; Zhang and Seelig, 2011) and deoxyribozymes (Stojanovic and Stefanovic, 2003; Liu et al., 2009).

Chemical Reaction Network

A *chemical reaction network* (CRN) (Espenson, 1995; Dittrich et al., 2001) represents an unstructured macroscopic artificial chemistry where the reactions of symbolic species prescribe the behavior of the system. The species are not assigned a molecular structure yet, therefore they serve as placeholders for hypothetical wet chemicals that react in the same way. Since the reaction tank is assumed to be well-stirred, the system's state does not contain any spatial information and is effectively reduced to a vector of species concentrations, which we treat as a dimensionless quantity.

The reaction rate defines the strength or speed of a reaction application prescribed by kinetics laws. The rate of an ordinary reaction $aS_1 + bS_2 \rightarrow P$ is defined by mass-action law (Espenson, 1995) as

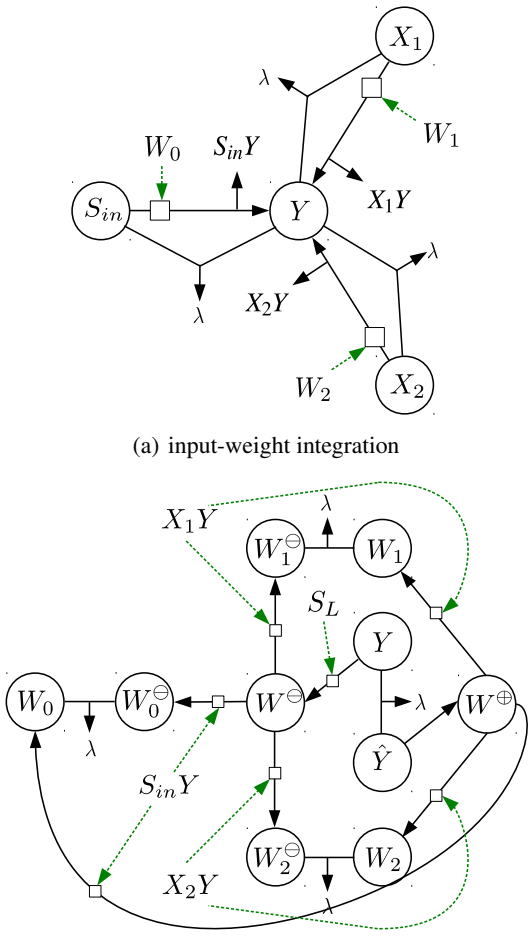
$$r = \frac{d[P]}{dt} = -\frac{1}{a}\frac{d[S_1]}{dt} = -\frac{1}{b}\frac{d[S_2]}{dt} = k[S_1]^a[S_2]^b,$$

where $k \in \mathbb{R}^+$ is a reaction rate constant, a and b are stoichiometric constants, $[S_1]$ and $[S_2]$ are concentrations of reactants (substrates) S_1 and S_2, and $[P]$ is a concentration of product P.

The rate of a catalytic reaction $S \xrightarrow{E} P$, where a substrate S transforms to a product P with a catalyst E, whose concentration increases the reaction rate, is given by Michaelis-Menten kinetics (Copeland, 2002) as

$$r = \frac{d[P]}{dt} = \frac{k_{cat}[E][S]}{K_m + [S]},$$

where $k_{cat}, K_m \in \mathbb{R}^+$ are rate constants. By combining kinetic expressions for all species, we obtain a system of ODEs that we simulate with a 4^{th} order Runge-Kutta numerical integration with step size 0.1.



(a) input-weight integration

(b) learning

Figure 1: (a) The AASP's reactions performing input-weight integration, where cross-weight competition is achieved by the annihilation of the inputs S_{in}, X_1, and X_2 with the output Y. Three species $S_{in}Y, X_1Y$, and X_2Y represent the contributions of the inputs S_{in}, X_1, and X_2 with associated weights in the output Y. (b) The AASP's reactions responsible for learning: comparison of the output Y with the target-output $\hat{Y}$ by annihilation $Y + \hat{Y} \rightarrow \lambda$, and subsequent positive and negative adaptation of the weights W_0, W_1, and W_2. Nodes represent species, solid lines are reactions, dashed lines are catalysts, and λ stands for no or inert species.

Analog Asymmetric Signal Perceptron

In this section we briefly describe the analog asymmetric signal perceptron (AASP) introduced in (Banda and Teuscher, 2014). The AASP is a CRN consisting of 17 species and 18 reactions that mimics a formal two-input analog perceptron (Rosenblatt, 1958). The input species X_1 and X_2 correspond to the formal inputs x_1 and x_2, the species Y to the output y, and the input signal S_{in} to the constant coefficient x_0 of the bias weight w_0. The AASP represents the weights by species W_0, W_1, and W_2. We optimized the

AASP's rate constants to tune its performance by genetic algorithms.

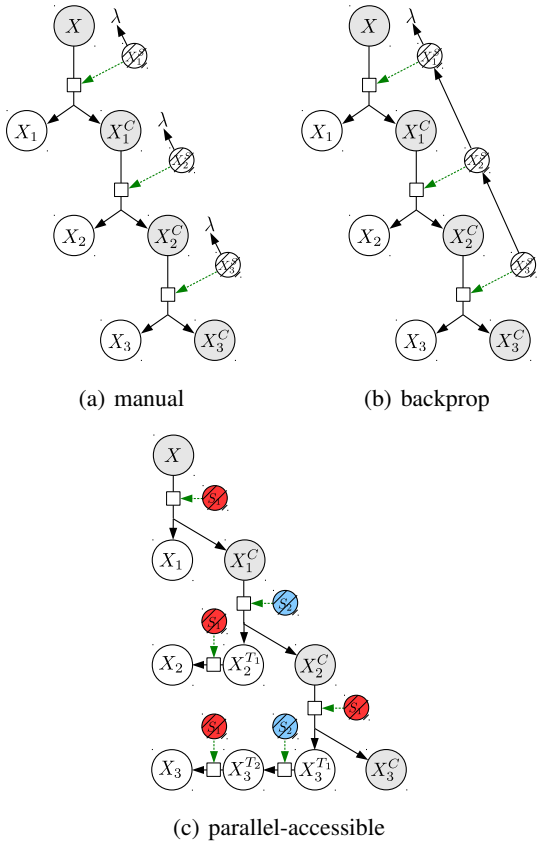
During the nonlinear input-weight integration (Figure 1(a)) each weight catalyzes a transformation of the input X_i to the output Y, and races with its input's annihilation $X_i + Y \rightarrow \lambda$ and the other weights since the output Y is shared. Naturally the output concentration cannot exceed all the inputs provided. When the weight concentration is close to zero, its branch could consume more from the output than it contributes, hence a low weight concentration could impose a negative pressure on a different weight branch. Therefore, a weight species coupled with its annihilatory reaction can act effectively as a catalyst and inhibitor.

The AASP is trained by classical supervised learning (Rojas, 1996) triggered by an injection of the target output $\hat{Y}$ provided some time after the injection of the input species. The AASP compares the output Y and the target output $\hat{Y}$ by a rapid annihilation. A leftover of the output Y implies that the weights need to be decreased, hence a negative weight changer $W^{\ominus}$ is produced from Y. In the opposite case, $\hat{Y}$ transforms to a positive weight changer $W^{\oplus}$ to increase the weights. For the positive adaptation, $W^{\oplus}$ is distributed by concurrent catalytic reactions $W^{\oplus} \rightarrow W_i$ among the weights proportionally to their input-weight contributions $S_{in}Y, X_1Y$, and X_2Y. Similarly, the negative adaptation splits $W^{\ominus}$ to intermediates $W_0^{\ominus}, W_1^{\ominus}$, and $W_2^{\ominus}$, which annihilate with the weights. Note that $S_{in}Y, X_1Y$, and X_2Y are three auxiliary species that represent the contributions of the inputs S_{in}, X_1, and X_2 with associated weights in the output Y. Because of the annihilatory reactions, $[Y] \leq [S_{in}Y] + [X_1Y] + [X_2Y]$. Also, each learning iteration the input-weight contributions species are flushed.

Parallel-Accessible Delay Line

A chemical delay line stores past input concentrations in a queue and provide them to a connected system. We previously introduced two variants of a chemical DL (Moles et al., 2014). The core copy reaction $X_i \xrightarrow{X_i^S} X_{i+1} + X_{i+1}^C$ controlled by the signal (catalyst) X_i^S splits the cached input X_i (or injected input X) into one copy X_{i+1}, which is available for consumption, and another copy X_{i+1}^C, which is buffered and propagated to the next stage.

The manual signalling variant is fully sequential and relies on injections of signals X_i^S for each copy phase in a backward order, as shown in Figure 2(a). First, an injection of the X_3^S signal produces X_3 from cached X_2^C, then the X_2^S signal copies the cached X_1^C into X_2 and X_2^C, etc. Even though this model is error-free because of its sequential operation and a separation of the copying stages, it requires a significant amount of external "help" since the number of injections grows linearly with its size. The signal backpropagation DL keeps all the reactions from the manual version, but adds the transformations of signals $X_i^S \rightarrow X_{i-1}^S$, which replaces the signal injections (Figure 2(b)). Because the backpropagation reactions are not instantaneous, there is always an overlap of consecutive stages, where both signals X_i^S and X_{i-1}^S are present. That produces an error that accumulates with every stage. Therefore, this model could be used only for small sizes, such as $n = 2$ or $n = 3$.



(a) manual (b) backprop

(c) parallel-accessible

Figure 2: Three variants of a chemical DL of size $n = 3$. The species X_1, X_2, and X_3 represent the input X cached at time/cycle $t, t-1$, and $t-2$ respectively. The manual and backpropagation signalling models are sequential, hence they produce the cached values one after another. The parallel model with a wait queue X_i^T for each stage and two alternating signals S_1 and S_2 produces all the values simultaneously. Nodes represent species, solid lines are reactions, dashed lines are catalysts, and λ stands for no or inert species.

Here we propose a new and optimized variant of a chemical DL with a minimal error, minimal latency, minimal number of injections, and a low number of signals. We call it the parallel-accessible DL (PDL) because several non-concurrent stages are executed in parallel as opposed to blocking sequential stages employed before. The PDL utilizes the same copy reaction $X_i^C \xrightarrow{S_1/S_2} X_{i+1} + X_{i+1}^C$, but also introduces a wait queue $X_i^{T_j} \xrightarrow{S_1/S_2} X_i^{T_{j+1}}$, where the

past concentrations are buffered for the required number of cycles.

Only two alternating signals S_1 (red) and S_2 (blue) drive the PDL's execution, regardless of its size. As shown in Figure 2(c), an injection of S_1 fires a simultaneous production of all the values X_1, X_2, X_3, and other copy or move-in-wait-queue reactions. The signal S_2 injected some time after S_1 triggers the remaining reactions that move values further to terminals X_i, consumed by an underlying system. The key idea here is that out of each pair of two neighboring species, there is always at least one with zero concentration since no S_1 (or S_2) signals drive the adjacent reactions. The concentration pathway from X to X_i contains $2i-1$ reactions with i red and $i-1$ blue signals.

We set the rates of all PDL reactions to $k_{cat} = 2$ and $K_m = 0.075$. In order to fully proceed the S_1 phase in 25 steps, but at the same time prevent a signal overlap that could produce a transfer error, we set the rate of S_1 and S_2 decay to 0.6. Due to a negligible error and a constant latency, i.e., all past values are available immediately, the PDL could easily integrate with other chemical systems and provide them with a reliable and fast memory. Compared to our previous models, the new model, however, requires more species and reactions. The number grows quadratically for both the species ($\frac{(n+2)(n+1)}{2}$) and the reactions ($\frac{(n+1)n}{2}$). Table 1 summarizes the attributes of all our chemical DL models.

Table 1: Comparison of three different types of a chemical DL.

Attribute	Manual DL	Signal Backprop. DL	Parallel DL
Error	0	exp	0
Injections	$O(n)$	$O(1)$	$O(1)$
Latency	$O(n)$	$O(n)$	$O(1)$
Signals	$O(n)$	$O(n)$	$O(1)$
Species	$O(n)$	$O(n)$	$O(n^2)$
Reactions	$O(n)$	$O(n)$	$O(n^2)$

Perceptron Integration

In this section we will integrate an AASP (Section III) with a PDL (Section IV) to create a new memory-enabled AASP (AASP-DL) by feeding the species X_i produced by the PDL directly into an AASP as shown in Figure 3. As described before, that specific setup will allow us to perform temporal signal processing very effectively.

In order to operate with arbitrary memory we extend the AASP to more than two inputs. For each new cached input X_i we add five reactions: $X_i \xrightarrow{W_i} X_iY + Y$ and $X_i + Y \to \lambda$ for the input-weight integration, and $W^{\oplus} \xrightarrow{X_iY} W_i$, $W^{\ominus} \xrightarrow{X_iY} W_i^{\ominus}$, and $W_i + W_i^{\ominus} \to \lambda$ for the positive and negative weight adaptation. The AASP of size n has therefore $4n+9$ species and $5n+8$ reactions. All new reactions use the same rate constants as the original X_1/X_2 reactions. Since the values X_i are produced rapidly, the AASP's timing, such as the injection of the input signal S_{in}, are unchanged. Also, a single DL operational cycle of 50 time steps is compatible with 500 time steps required for the AASP's training cycle. To demonstrate the scalability, we model AASP-DLs of size 2 to 5.

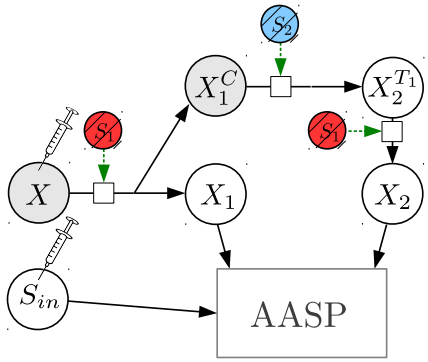


Figure 3: An AASP-DL schematic of size $n = 2$.

Experiments

In this section we describe the tasks, the performance metrics, the training setup, and the learning results of the AASP-DL.

Tasks

We have decided to use the following tasks to evaluate the performance of our new system. The selection and setting of temporal tasks reflect the fact that the expected AASP's output concentration must be between the minimal (zero) and the maximal output concentration, which is equal to the sum of all inputs provided $[S_{in}] + [X_1] + \ldots + [X_n]$.

1) LWMA2: Linear Weighted Moving Average (LWMA) is a time-series of a lagged averages, where each past element is weighted by an arbitrary value. The LWMA of order 2 is defined as

$$y_t = k_1 u_{t-1} + k_2 u_{t-2} + k_0,$$

where $k_1, k_2 \in (0.2, 0.8)$ are randomly drawn constants, $k_0 \in (0.1, 0.4)$ is a constant bias, and u_t is an i.i.d input stream generated uniformly from $(0.2, 1)$. The task is to output y_t based on the past inputs u_{t-i}.

2) WMM2: The Weighted Moving Maximum (WMM) is a time-series of maximum lagged inputs. The WMM of order two is defined as

$$y_t = k \max(u_{t-1}, u_{t-2}) + k_0,$$

where similarly to LWMA2 the constants k and k_0 are randomly drawn from the interval $(0.2, 0.8)$ and $(0.1, 0.4)$ respectively, and u_t is an i.i.d input stream generated uniformly from $(0.2, 1)$. The task is to output y_t based on the past inputs u_{t-i}.

3,4) NARMA: The Nonlinear AutoRegressive Moving Average (NARMA) (Atiya and Parlos, 2000) is a discrete time-series, where the current output y_t depends on both the previous inputs and outputs up to a given depth (order). The NARMA task of order n is defined as

$$y_t = \alpha y_{t-1} + \beta y_{t-1} \sum_{i=1}^{n} y_{t-i} + \gamma u_{t-n} u_{t-1} + \delta,$$

where $\alpha = 0.3, \beta = 0.05, \gamma = 1.5, \delta = 0.1$, and u_t is an i.i.d input stream generated uniformly from an interval $(0, 0.5)$. The task is to produce the output y_t based on the previous inputs u_{t-i}. NARMA is widely used as a benchmark task in the neural and reservoir computing literature (Jaeger, 2001; Maass et al., 2002) due to its nonlinearity and dependence on long-term memory.

We tackle two variants, NARMA2 and NARMA10, i.e., the second and tenth order of the problem. Since NARMA10 could be unstable, we bound the series by a non-linear $\tanh$ saturation function. Also, we scale the target stream y_t for NARMA2 by 2 to better fit the AASP's output range.

Performance Measures

We define performance by two standard error measures: symmetric absolute mean percentage error (SAMP) with values ranging from 0% to 100%

$$\text{SAMP} = 100\langle \frac{|y - \hat{y}|}{y + \hat{y}} \rangle$$

and the root normalized mean square error (RNMSE)

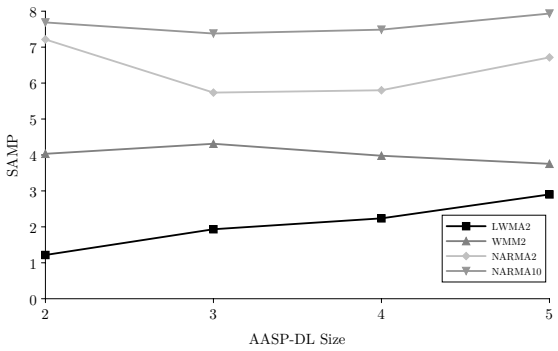
$$\text{RNMSE} = \sqrt{\frac{\langle (y - \hat{y})^2 \rangle}{\sigma_{\hat{y}}^2}},$$

where the square error is normalized by $\sigma_{\hat{y}}^2$, a variance of the target output $\hat{y}$. A RNMSE of 1 therefore corresponds to chance level.

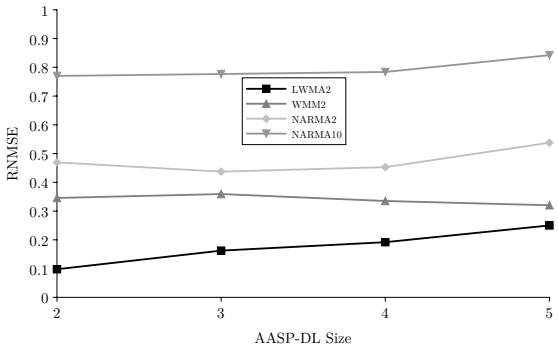
Training Setup

The training of all AASP-DLs starts with a random setting of weight concentrations from $(0.5, 1.5)$. During each training iteration we first inject the input X with a concentration corresponding to u_{t-1} for all tasks. Then we set the bias input S_{in} concentration to 0.5 for LWMA2 and WMM2, and 0.1 for NARMA2 and NARMA10. To trigger the DL operation we also provide the signal S_1, which immediately produces both the current and the cached values X_i. To finish a PDL buffering procedure we inject another DL signal S_2 25 time steps later. Then again after 25 time steps we finally provide both the learning signal S_L and the target output $\hat{Y}$ representing y_t to initiate the weight adaptation.

We run the AASP-DL against $10,000$ training time series of length 800 for each task. We evaluate the RNMSE and SAMP performance over $10,000$ runs for each training iteration, however, we report only the final training error.



(a) SAMP



(b) RNMSE

Figure 4: The relation between the final SAMP and RNMSE error and the AASP-DL size after 800 learning iterations. For each task $10,000$ runs were performed.

Results

The AASP-DL reaches a relatively small error for all temporal tasks, which demonstrates that even a single chemical perceptron, due to its inherent nonlinearity, posses sufficient learning and computing capabilities. The error of the AASP-DL with optimal size settles to the range of $(1.20\%, 7.37\%)$ for SAMP and $(0.10, 0.77)$ for RNMSE as shown in Figure 4. Figures 5 and 6 show SAMP and RNMSE for all tasks over time.

The easiest function is a linear LWMA2 (SAMP of 1.20), with performance decreasing as memory of the past inputs grows. That is expected because LWMA2 depends only on the last two inputs u_{t-1} and u_{t-2}, so any additional information is essentially superfluous. Note that the AASP cannot fully eliminate the contribution or consumption of an extra input-weight branch, hence the input here basically acts as a noise. Figure 7 shows an example of the weight concentration traces and the filtered output and target output for LWMA2.

The WMM2 task's output is also fully prescribed by the last two inputs, however as opposed to LWMA2, the AASP-DL of size 5 performs best (SAMP of 3.75) with a marginal difference to the $n = 2$ instance (SAMP of 4.02). Even

though the extra past inputs do not affect the target output y_t, the AASP might utilize them as a statistical variance source.

Because of its recurrent definition, the NARMA2 performance improves significantly for a longer DL, reaching an optimum for $n = 3$ (SAMP of 5.73) and slightly worse for $n = 4$ (SAMP of 5.81). Since the NARMA2 depends on the last two inputs u_{t-1} and u_{t-2} and recurrently on the last two outputs y_{t-1} and y_{t-2}, its output is fully determined by the last four inputs, which is inline with our results. NARMA10 is the most difficult task (SAMP of 7.37) due to the function dependence on long lags. The performance is fairly constant for $n \leq 4$.

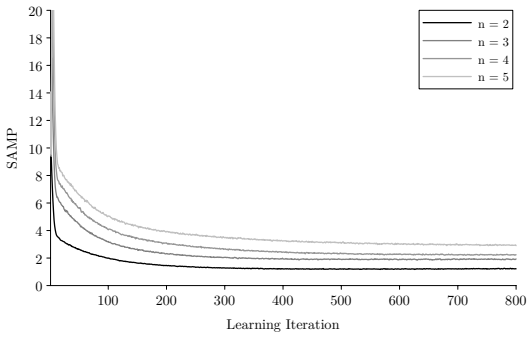
Discussion and Conclusion

In this paper we demonstrated that the idea of chemical learning can be extended to temporal tasks by utilizing a memory of past concentrations provided through a chemical delay line. The AASP-DL successfully learns to produce a weighted moving average as well as a moving maximum. These operations could be applied to monitoring certain substances in a patient's blood, such as the insulin level, and perhaps also to control it by a conditional release of a specific amount of a required substance (cure). To demonstrate more complex nonlinear time dependencies, we also trained the AASP-DL for both the NARMA2 and NARMA10 benchmark tasks.

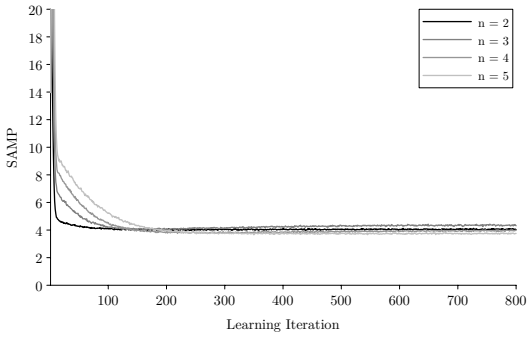
Our results show that memory improves learning for the recursive NARMA2 and nonlinear WMM2 tasks, but leads to lower performance for the simple LWMA2 task. Because the weights compete with each other and the AASP cannot fully eliminate the contribution or consumption of an extra input-weight branch, for a memory larger than the task inherently requires, performance decreases. Note that for the initial weights drawn uniformly from the interval $(0.5 - 1.5)$ the AASP's ideal output region optimized by genetic algorithms (Banda and Teuscher, 2014) is around half of the maximal output concentration, which equals the total amount of input injected $[S_{in}] + [X_1] + [X_2]$. To move the input-output relation closer to that region we scaled the NARMA2 task by two.

Furthermore, since the AASP was optimized for the input range $(0.2, 1)$ with just two inputs, we can speculate that for a larger memory (and number of inputs), some part of the performance decrease is due to the fact that the output surface for the usual weight concentrations moves away from the responsive region and becomes either flat (saturated) or highly rugged, where even a small perturbation of a weight concentration changes the output significantly. Also, since an optimal weight learning region is nonlinear, we can expect that scaling the input and output by the AASP-DL size would not necessarily help. An optimal scaling of the AASP-DL's input and output concentrations is an important topic that is, however, beyond the scope of this paper.

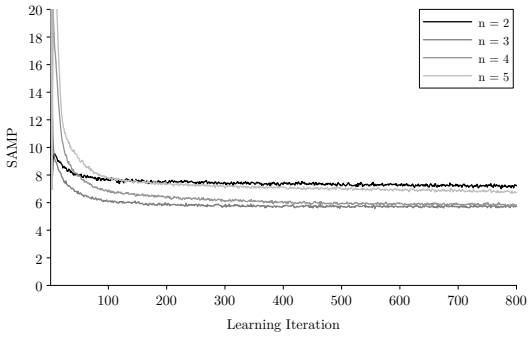
To further improve the performance, the next step would



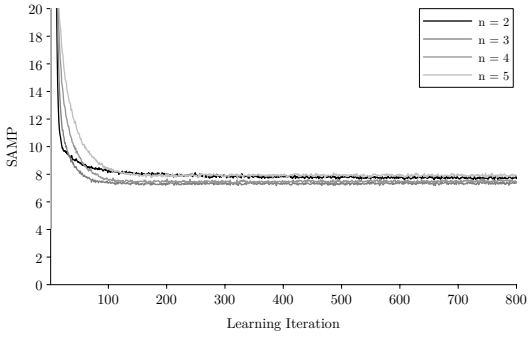
(a) LWMA2



(b) WMM2

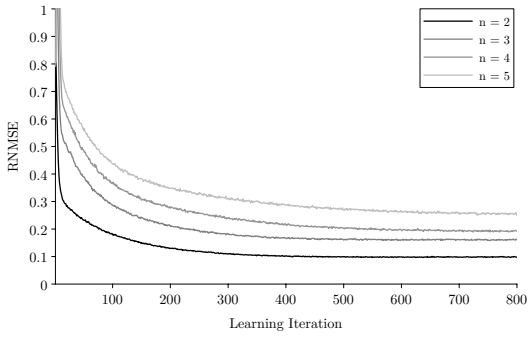


(c) NARMA2

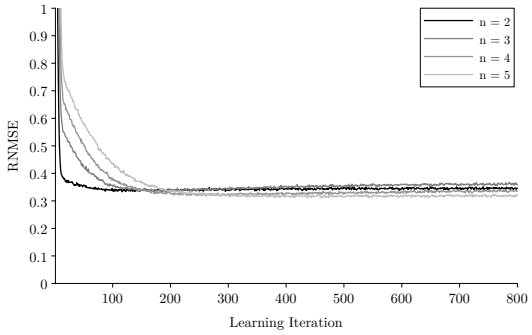


(d) NARMA10

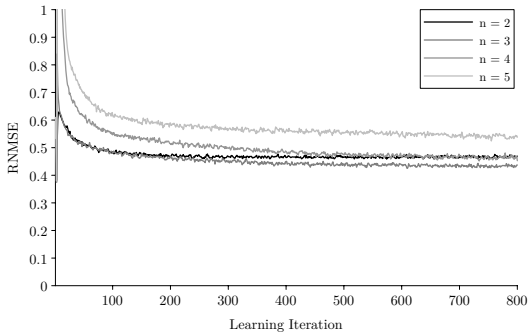
Figure 5: The SAMP error over time for all tasks. Average values over $10,000$ runs.



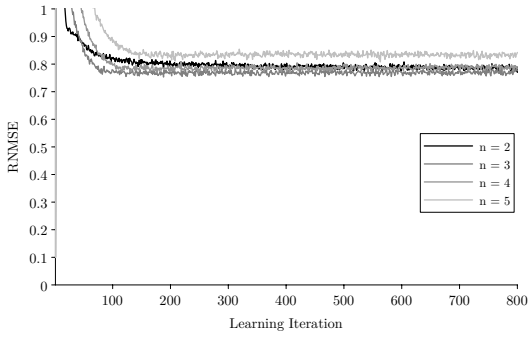
(a) LWMA2



(b) WMM2

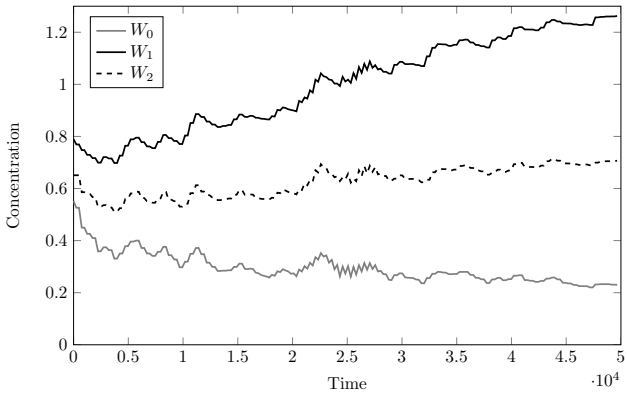


(c) NARMA2

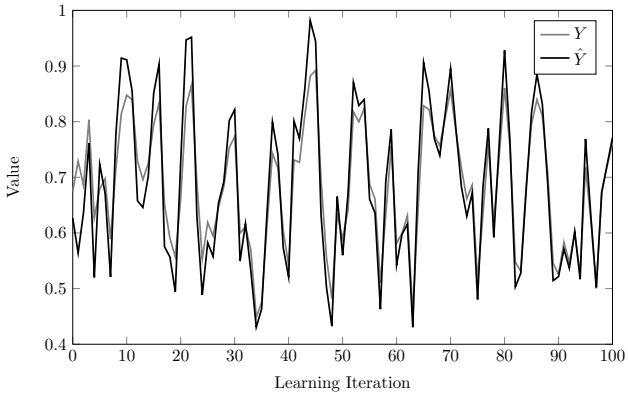


(d) NARMA10

Figure 6: The RNMSE error over time for all tasks. Average values over $10,000$ runs.



(a) Weights



(b) Output

Figure 7: An example of the AASP-DL $n = 2$ learning LWMA2, showing (a) the concentration traces of the weights and (b) the filtered output and the target output.

be to compose several chemical perceptrons with a delay line into larger (nested) multicompartment networks, where each compartment hosts a single chemical perceptron and where compartments communicate with each other through a channel-mediated exchange of molecular species.

Note that the NARMA task is generally tackled with a magnitude larger and more complicated machine learning approaches than our single memory-enabled chemical perceptron. A common approach is to employ recurrent neural networks (Atiya and Parlos, 2000) or advanced reservoir computing (Büsing et al., 2010), i.e., echo state networks (Jaeger, 2001) or liquid state machines (Maass et al., 2002), consisting of hundreds of nodes (neurons). For instance, minimal complexity echo state networks reported by Tino and Rodan (Rodan and Tino, 2011) needed 50 nodes to achieve a NMSE of 0.16, i.e., RNMSE of 0.4. The primary goal of using NARMA in this paper was to train our simple chemical learner to demonstrate its principal capabilities and to set a baseline.

We also plan to implement our model in an actual wet chemistry. More specifically our goal is to map each cat-

alytic reaction to a deoxyribozyme-based substrate cleavage (Stojanovic and Stefanovic, 2003; Liu et al., 2009). In the core deoxyribozyme reaction $QF \xrightarrow{D} Q + F$, the downstream enzyme (catalytic DNA) D cleaves the oligonucleotide (short, single-stranded DNA) QF into two parts Q and F, where F is a chemical called fluorescein emitting fluorescence, which we can measure. By combining different cascading types of upstream and downstream enzymes and activation and deactivation relations, more complicated scenarios could be obtained. However, the mapping of three enzymes X_1Y, X_2Y, and $S_{in}Y$, as well as the DL operational signals S_1 and S_2, which catalyze two (or more) reactions, would be non-trivial to implement in practice. Hence, we could expect that each of these catalysts would need to have several variants to serve different reactions. Alternatively we could obtain a wet chemical implementation of the AASP-DL by compiling mass-action-driven CRN to DNA-strand displacement reactions (Soloveichik et al., 2010; Zhang and Seelig, 2011). That would, however, produce more than 100 different strands, which would be complicated to produce accurately in the lab. Finally, note that for DNA-strand displacement, we would need to continually add new gate structures (fuel) to the solution to make the circuit reusable many times during training.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant no. 1028120.

References

Atiya, A. F. and Parlos, A. G. (2000). New results on recurrent network training: unifying the algorithms and accelerating convergence. *Neural Networks, IEEE Transactions on*, 11(3):697–709.

Banda, P. and Teuscher, C. (2014). Learning two-input linear and nonlinear analog functions with a simple chemical system (in press). In Ibarra, O. H. and Kari, L., editors, *Unconventional Computing and Natural Computing Conference*, volume 8553 of *Lecture Notes in Computer Science*, pages 14–26. Springer International Publishing Switzerland.

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).

Bray, D. (1995). Protein molecules as computational elements in living cells. *Nature*, 376(6538):307–312.

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.

Copeland, R. A. (2002). *Enzymes: A practical introduction to structure, mechanism, and data analysis*. John Wiley & Sons, Inc., New York, New York, second edition.

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

Espenson, J. (1995). *Chemical kinetics and reaction mechanisms*. McGraw-Hill, Singapore.

Haykin, S. (2009). *Neural networks and learning machines*. Pearson, New Jersey, third edition.

Jaeger, H. (2001). 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.

Kim, J., Hopfield, J. J., and Winfree, E. (2004). Neural network computation by in vitro transcriptional circuits. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 17, pages 681–688. MIT Press.

LaVan, D. A., McGuire, T., and Langer, R. (2003). Small-scale systems for in vivo drug delivery. *Nature biotechnology*, 21(10):1184–91.

Liu, J., Cao, Z., and Lu, Y. (2009). Functional nucleic acid sensors. *Chemical Reviews*, 109(5):1948–1998. PMID: 19301873.

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.

Mills, A. P., Yurke, B., and Platzman, P. M. (1999). Article for analog vector algebra computation. *Biosystems*, 52(1-3):175–180.

Moles, J., Banda, P., and Teuscher, C. (2014). Delay line as a chemical reaction network (under review). *Parallel Processing Letters*, http://arxiv.org/abs/1404.1152.

Qian, L., Winfree, E., and Bruck, J. (2011). Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372.

Rodan, A. and Tino, P. (2011). Minimum complexity echo state network. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 22(1):131–44.

Rojas, R. (1996). *Neural networks: A systematic introduction*. Springer-Verlag, Berlin.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review*, 65:368–408.

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

Stojanovic, M. N. and Stefanovic, D. (2003). A deoxyribozyme-based molecular automaton. *Nature Biotechnology*, 21(9):1069–1074.

Zhang, D. Y. and Seelig, G. (2011). Dynamic DNA nanotechnology using strand-displacement reactions. *Nature chemistry*, 3(2):103–113.