Simulated Evolution of Mass Conserving Reaction Networks

Anthony M.L. Liekens, Huub M.M. ten Eikelder, Marvin N. Steijaert, Peter A.J. Hilbers

Faculty of Biomedical Technology, Technische Universiteit Eindhoven, the Netherlands
anthony@liekens.net

Abstract

With the rise of systems biology, the systematic analysis and construction of behavioral mechanisms in both natural and artificial biochemical networks has become a vital part of understanding and predicting the inner workings of intracellular signaling networks. As a modeling platform, artificial chemistries are commonly adopted to study and construct artificial reaction network motifs that exhibit complex computational behaviors. Here, we present a genetic algorithm to evolve networks that can compute elementary mathematical functions by transforming initial *input* molecules into the steady state concentrations of *output molecules*. More specifically, the proposed algorithm implicitly guarantees mass conservation through an atom based description of the molecules and reaction networks. We discuss the adopted approach for the artificial evolution of these chemical networks, evolve networks to compute the square root function. Finally, we provide an extensive deterministic and stochastic analysis of a core square root network motif present in these resulting networks, confirming that the motif is indeed capable of computing the square root function.

Introduction

In biological organisms, networks of chemical reactions control the processing of information in a cell. A general approach to study the behavior of these networks is to analyze modules that are frequently observed in natural systems. Numerous network motifs that perform computational tasks have been discovered in biochemical reaction networks. Reaction networks are able to compute Boolean operations and implement simple binary computers (Arkin and Ross, 1994; Sauro and Kholodenko, 2004; Steijaert et al., 2007). Cell signaling networks are known to exhibit parallelism, the integration and amplification of signals, bistable behavior and hysteresis through feedback and memory (Bray, 1990; Bhalla and Iyengar, 1999; Bray, 1995; Tyson et al., 2003; Steijaert et al., 2008). Many engineering metaphors have been put forward as analogies to signaling networks, such as neural networks and analog electronic circuits (Bray, 1995; ten Eikelder et al., 2007). As an example, elementary operations such as addition, multiplication, integration and amplification can be found as modules of the MAPK pathway (Bhalla, 2003).

A more recent approach to study computations in biochemical networks is to construct artificial networks or abstractions of real life systems to study the available space of computations that can be performed in cellular reaction networks. *In vitro* molecular computations have been performed with gene expression networks (Benenson et al., 2004). Through the adoption of artificial chemistries (Dittrich et al., 2001; Dittrich, 2001) to implement chemical networks, it has been shown that algebraic functions can be constructed using a bottom-up approach based on motifs that implement elementary mathematical operations (Buisman et al., 2008). Related research employs *in silico* evolutionary algorithms for the discovery of conceptual networks that perform basic computations (Deckard and Sauro, 2004; Paladugu et al., 2006; Lenser et al., 2007).

In the current study, we have developed a comparable genetic algorithm that allows for the evolutionary design of reaction networks with a desired function. For given input molecules and their initial amounts, the desired reaction network needs to process these input molecules and generate an output pool of products whose concentrations correspond to a desired function of the amounts of inputs. In contrast with related work, our approach guarantees networks that respect the law of conservation of mass explicitly. Molecular species in our reaction networks are considered to be strings consisting of imaginary atoms and by satisfying the condition that the total set atoms in reactants and products in a reaction must be equal for all reactions in the network, we can guarantee the conservation of mass in our reaction network. By enforcing this condition upon the construction of reaction networks and within the genetic operators in our genetic algorithm, it is guaranteed that the evolved networks do not violate the law of mass conservation. Other approaches either test for mass conservation at each fitness evaluation, e.g., Lenser et al. or ignore the law of mass conservation completely, e.g., Paladugu et al.

First, we give an overview of the implementation of the reaction networks within our artificial chemistries framework and how they are evaluated with respect to a desired input-output function. Together with a set of genetic oper-

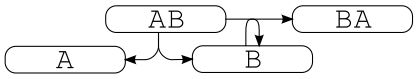

Figure 1: A small reaction network with two reactions.

ators that act on the networks, we can implement a genetic algorithm to artificially evolve such networks. Next, we review some networks that have been evolved with the software implementation, where the target function is the square root, i.e., for an amount X input molecules of a specific type, the reaction network needs to generate an output molecule with desired amount $\sqrt{X}$ at steady state. We review some evolved networks and discuss small network motifs that act as *kernels* in these square root networks. We analytically study one elementary network motif that computes a square root-like function and furthermore examine its behavior with a stochastic approach to model instantiations of the system with small molecular counts.

Methods

Molecules and reaction networks

In the artificial chemistries deployed by our genetic algorithm, we let an alphabet Σ denote the available *atoms* and strings s with variable length over Σ are possible *molecular species*. The number and the order of atoms in the string defines the uniqueness of a species. We define the *molecular mass* of a species or string s as a vector $\mathbf{m}$ where m_l is the number of characters $l \in \Sigma$ in s.

In order to abstract from natural biochemical networks, we say that an individual – which represents a reaction network – comprises of a number of reactions that take specific reactants from a pool of molecules, execute a specific transformation on these reactants at a predefined rate and return their products to the molecular pool. Each reaction has at most two reactants and two products. This limitation is inspired by nature, where enzymatic reactions with more than two substrates are rare. In order to guarantee mass conservation within the chemical networks, it is required that the total mass of reactants in a reaction equals the mass of the products. All of the reactions in this study are assumed to follow the law of mass-action kinetics according to a rate $k > 0$, i.e., a reaction occurs with a propensity that is proportional to k and the concentrations of available reactant molecules.

As an example, in a setup with alphabet or atom set $\Sigma = \{A, B\}$, the reaction network of an individual as depicted in Figure 1 contains the valid reactions $r_1 = AB \rightarrow A + B$ and $r_2 = AB + B \rightarrow BA + B$, with their respective reaction rates k_1 and k_2. Clearly the total mass, i.e., the number of atoms A and B, is conserved in these reaction.

Network evaluation and fitness

An individual or network is evaluated by providing the network with an input pool of reactants and observing the amounts of participating molecules over time. We let an ordinary differential equation (ODE) model of the reaction network compute the transient behavior of the network and if the system reaches a steady state, we compare the resulting pool of products with a desired output of the network, as a function of the input. As the output concentration of a molecule in the pool is closer to the desired output, for a set of inputs, the fitness of the individual is higher.

In a first step to compute the fitness of an individual, an ODE representation of its reaction network is constructed. Since we have assumed mass-action kinetics, this step is fairly straightforward and results in the following ODE system for our example network with reactions r_1 and r_2 as in Figure 1:

$$\begin{array}{rcl} & d[AB]/dt &= -k_1[AB] - k_2[AB][B] \\ d[A]/dt = & d[B]/dt &= k_1[AB] \\ & d[BA]/dt &= k_2[AB][B] \end{array}$$

where $[s]$ denotes the dimensionless amount or concentration of molecular species s.

In our genetic algorithm, reaction networks are stored as System Biology Markup Language (SBML) objects (Hucka et al., 2003). A network in this format is passed on to the SBML ODE Solver Library (Machne et al., 2006) which constructs an ODE model of the network and solves it with numerical integration. The ODE solver reports the steady state behavior of the network back to the fitness function, when a steady state is detected. If the ODE solver cannot find a steady state within a reasonable amount of time – a network may show, for example, stable oscillatory behavior, keeping it from reaching a steady state – the individual is eliminated from the population.

In order to compute the fitness of an individual in relation to an input-output function that needs to be approximated by the evolutionary algorithm, we iteratively run the ODE model for a set of input and output pairs. At time 0 of the ODE model, we set the initial amount of molecules for specifically designated input molecules. All other molecules in the system are initialized with concentration 0. With these initial values, the steady state of the ODE system is computed with the numerical ODE solver. For each molecule in the system, we compute the squared difference between the desired output and the steady state concentration of that molecule. The molecule with the smallest mean squared error for varying inputs is considered to be the output molecule and the fitness of the individual is inversely proportional to its mean squared error. Consequently, as the steady state concentration of a molecule is closer to the desired output, the fitness of the individual is higher. If an individual is detected not to reach its steady state for at least one input setting, its fitness is set to 0. It should be noted that we select specific molecules to act as input molecules of our system, but we do not select a specific molecule to act as the output molecule of the reaction network. As the evolutionary algo-

rithm is free to let all molecules act as output molecules, it is expected that the genetic algorithm is better able to find approximations of the desired function.

Genetic algorithm

The population of the genetic algorithm is seeded with randomly generated reaction networks, with a fixed number of reactions in each individual. In a random reaction, a random set of atoms is distributed over the two reactants and products, such that the total mass of reactants and products is equal. For all of the experiments in this paper, it sufficed to initialize the reactions with 2 to 7 random atoms. Mutations allow for larger molecules in the reaction networks if these would be required by the evolutionary process, as discussed later. Finally, a reaction rate is assigned to each reaction, uniformly chosen between 0 and 10.

To generate a new individual in the next generation's population, we select two parent individuals from the current population, proportionally to their fitness as defined above. We apply a uniform crossover operator and iterate over the resulting set of reactions with a mutation operator to generate the offspring individual which is evaluated and appended to the new population of the next generation.

With uniform crossover, we iterate over the ordered lists of reactions in both parents, where both reactions have an equal probability of ending up in the offspring's reaction network. Consequently, as the population is initialized with reaction networks with a fixed number of reactions, this number of reactions is maintained throughout the evolution of the population. This allows the user to enforce a specific number of reactions in the evolved reaction network and to prevent network bloat.

For the mutation operator, we iterate over the reactions in the offspring reaction network and mutate each reaction with a mutation parameter μ (usually, $\mu = 0.1$). The mutation operator for a reaction consists of two steps, one changing the reactants and products, where the second step mutates the reaction rate of the reactions. Firstly, in order to change the constituent products and reactants of a reaction, a random atom from alphabet Σ is inserted at a random position of a random reactant and a random product with probability $\mu/2$. Similarly, a randomly chosen atom is removed from both a reactant and a product in the reaction, also with probability $\mu/2$. Complete reactions are replaced with new, randomly generated reactions with a probability μ. Through these first mutation operators, the topology of the reaction network changes. Secondly, we multiply the reaction rate with a random number from a Gaussian distribution with mean 1 and standard deviation μ. This latter mutation operator does not change the topology of the reaction network, but it is involved in the parameter optimization of the network.

Typical runs of the genetic algorithm have been seeded with 100 individuals with just a few reactions in each individual. Our primary goal here is to find small networks – with up to 10 reactions – for elementary mathematical operations. However, with a limited reaction network size, the evolutionary algorithm may not be able to find exact implementations of the desired function – not all input-output functions can exactly be represented as a reaction network with a finite number of reactions – such that an approximation of the behavior evolves within the available space of network behaviors.

Parameter optimization

In addition to the above genetic algorithm that evolves both a network topology and reaction rate parameters, we have also adopted a limited version of the genetic algorithm for the optimization of reaction rates in a fixed network topology. In this genetic algorithm, the initial set of individuals is populated with networks of the same topology, but with random reaction rates (uniformly distributed between 0 and 10). The mutation operator in this algorithm is only allowed to mutate the reaction rate parameters of the constituent reactions, according to a normal distribution as described above. With uniform crossover, the reaction rates of the corresponding reactions are exchanged.

Networks that have been found by the main genetic algorithm are further optimized by this genetic algorithm, to obtain a network that behaves optimally for a given topology. Additionally, we have adopted this parameter optimizer to optimize user-defined networks and their corresponding desired behavior. Typical runs of the parameter optimizer assumed populations of 100 individuals for 100 generations.

Results

We have adopted the genetic algorithm to evolve networks that compute elementary mathematical operations. Some of the networks to compute these operations are straightforward. For example, a network that computes the difference $[A] - [B]$ of input molecules A and B, with $[A] > [B]$ can be as simple as the single reaction $A + B \rightarrow AB$ with rate $k > 0$. Each molecule A binds with a B molecule, such that $[A]$ can act as the output in the chemical equilibrium of the system, which is equal to the initial amount of A minus the initial amount of B. A network that is not as straightforward to construct a network that computes the square root of an amount of input molecule.

Square root networks

We have used our genetic algorithm to construct networks that compute the square root of the initial amount of input molecule ABC, with alphabet $\Sigma = \{A, B, C\}$. The ODE model of a candidate network is evaluated by setting the initial amount of molecule ABC to $X = 1, 4, 9, 16, \ldots, 100$ in consecutive runs. The molecule whose amount at steady state is nearest to the desired output $\sqrt{X}$ is designated as the

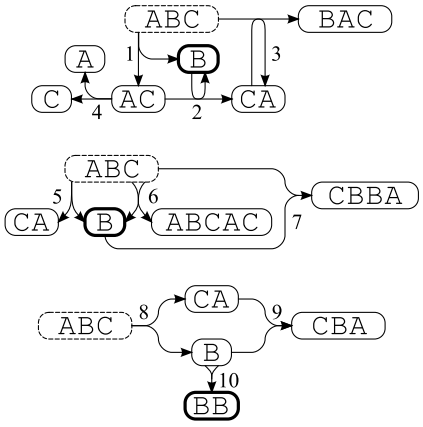

Figure 2: Three evolved networks that compute the square root function. The input molecule is denoted by the dashed outline, where the output is in bold.

output molecule of the network and its mean squared error is reported back to the fitness function.

Three networks that have been evolved with our genetic algorithm are shown in Figure 2. The first network, with mean squared error 0.119 for the 10 desired outputs, whose input is molecule ABC outputs a molecule B and consists of the following 4 reactions

$$\begin{aligned} ABC &\xrightarrow{k_1} B + AC && \text{with } k_1 = 1.159 \\ AC + B &\xrightarrow{k_2} CA + B && \text{with } k_2 = 3.716 \\ ABC + CA &\xrightarrow{k_3} BAC + CA && \text{with } k_3 = 5.590 \\ AC &\xrightarrow{k_4} A + C && \text{with } k_4 = 7.881. \end{aligned}$$

A second network that outputs an amount of molecules B that approximates the square root of molecules ABC (mean squared error equals 0.226) is given by the reaction network

$$\begin{aligned} ABC &\xrightarrow{k_5} B + CA && \text{with } k_5 = 7.359 \\ ABC + ABC &\xrightarrow{k_6} B + ABCAC && \text{with } k_6 = 1.502 \\ ABC + B &\xrightarrow{k_7} CBBA && \text{with } k_7 = 1.468. \end{aligned}$$

The third evolved reaction network with input ABC and output BB is given by

$$\begin{aligned} ABC &\xrightarrow{k_8} B + CA && \text{with } k_8 = 5.56937 \\ CA + B &\xrightarrow{k_9} CBA && \text{with } k_9 = 6.62825 \\ B + B &\xrightarrow{k_{10}} BB && \text{with } k_{10} = 8.34484 \end{aligned}$$

where the mean squared error between the counts of output molecule BB and the desired output is 0.335.

It should be pointed out that the evolved networks have been cleaned up manually to only show the reactions that are essential for the networks' square root behavior. Duplicate reactions have been merged and reactions that further process waste particles have been removed without affecting the output generating behavior of the network. E.g., in the second evolved network, waste molecule $CBBA$ is further processed into smaller waste molecules BB and CA, but this reaction does not interfere with the production of output molecule B from input ABC.

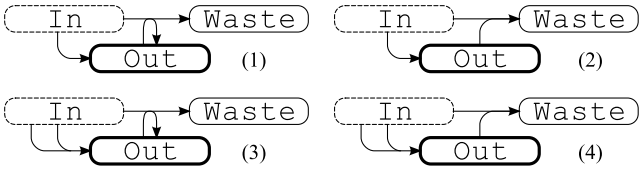

Figure 3: Four network motifs act as the kernels of evolved square root networks.

The relatively small errors show that all three networks provide good approximations of the desired square root behavior, within the range of inputs. Because the fitness is only evaluated within this input range, the networks do not guarantee a generally good approximation for other inputs.

Square root kernels In these and other networks that have been evolved to compute the square root of an input, a common behavioral subnetwork can be observed. In this common motif, a first set of reactions generates output molecules from the input molecules. The output molecule then takes part in reactions that remove remaining inputs from the pool. This behavior is at the core of the first and second evolved square root networks, as in Figure 2.

Figure 3 shows 4 elementary network motifs that are common to most of our evolved square root networks. All of these elementary kernels can be implemented such that they abide to the law of mass conservation. We have constructed these networks and optimized their parameters to study whether the behavior of these network motifs can be related to the square root. Table 1 gives the reaction rates for these networks such that the error for inputs $\{1, 4, 9, ..., 100\}$ is approximately minimal. Networks (3) and (4) behave worst, showing a steady state amount of output molecules that is linearly related to the input. Network (1) provides a good approximation of the square root network, where the performance of network (2) is mediocre. It should be pointed out that network (1) can be further optimized and has a mean squared error of 0.049 when $k_{out}/k_{waste} = 0.579$.

Analysis of a square root kernel

Since network motif (1) in Figure 3 provides the best approximation of the square root, we study the system analytically, in order to understand how the elementary network is capable of computing a good approximation of the square root function. We let x, y and w denote the amounts of input, output and waste molecules, and k_1 and k_2 the reaction rates of the output and waste producing reactions, respec-

kernel	k_{out}	k_{waste}	MSE
(l)	0.931	1.454	0.136
(2)	4.111	0.659	0.608
(3)	0.663	19.157	2.142
(4)	3.039	3.544	2.142

Table 1: Optimized parameters k_{out} and k_{waste} of the corresponding output and waste producing reactions for the square root kernels and their respective mean squared errors

tively. The network can be modeled by the following system of differential equations

$$\begin{aligned} \frac{dx(t)}{dt} &= -k_1 x(t) - k_2 x(t) y(t), \\ \frac{dy(t)}{dt} &= k_1 x(t), \\ \frac{dw(t)}{dt} &= k_2 x(t) y(t). \end{aligned}$$

For the computation of the square root of a number X, the initial concentrations for this system are $x(0) = X$, $y(0) = 0$ and $w(0) = 0$. The value of $y(t)$ for large t then hopefully approaches $\sqrt{X}$. Note that the differential equations are nonlinear, which makes it difficult to obtain analytical results.

Limiting values We first compute the behavior of the system for $t \to \infty$. Trivially, the limiting value of the input concentration $x(t)$ is given by $\lim_{t\to\infty} x(t) = 0$. Define the limiting values of output and waste by

$$\hat{y} = \lim_{t\to\infty} y(t), \qquad \hat{w} = \lim_{t\to\infty} w(t) \, .$$

We try to compute the value of $\hat{y}$. For all $t \geq 0$ the sum of the concentrations $x(t) + y(t) + w(t)$ is constant. In view of the initial condition this means that

$$x(t) + y(t) + w(t) = X \, . \tag{1}$$

Since $y(t)$ is supposed to approach the square root of X, it is obvious to consider $y^2(t)$. A simple computation gives

$$\frac{dy^2(t)}{dt} = 2k_1 y(t) x(t) = \frac{2k_1}{k_2} \frac{dw(t)}{dt} \, .$$

Using the initial conditions this implies that

$$y^2(t) - \frac{2k_1}{k_2} w(t) = 0 \, . \tag{2}$$

Since (1) and (2) hold for all values of t, we conclude that

$$\hat{y} + \hat{w} = X, \qquad \hat{y}^2 - \frac{2k_1}{k_2}\hat{w} = 0 \, .$$

Elimination of $\hat{w}$ from these equations leads to the quadratic equation

$$\hat{y}^2 - \frac{2k_1}{k_2}(X - \hat{y}) = 0 \, .$$

The non-negative solution of this equation is given by

$$\hat{y} = -\frac{k_1}{k_2} + \sqrt{\tfrac{2k_1}{k_2} X + \tfrac{k_1^2}{k_2^2}} \, .$$

By selecting $k_2 = 2k_1$ we obtain

$$\hat{y} = -\frac{1}{2} + \sqrt{X + \tfrac{1}{4}} \, .$$

Hence for X not too small, the chemical reaction network computes indeed an approximation of $\sqrt{X}$ if $k_2 = 2k_1$. Although all results can also be obtained for the general case, we shall assume in the sequel that $k_2 = 2k_1$. Note that the GA does not find this relation, as it attempts to compensate for the extra $-1/2$ in the steady state relation of our network.

Analytical solution In fact, in this case it is even possible to compute the analytical solution of the system. Using the relations (1) and (2) we can eliminate $x(t)$ and $w(t)$ from the system of differential equations. The resulting equation for $y(t)$ is then

$$\frac{dy(t)}{dt} = -k_1 y^2(t) - k_1 y(t) + k_1 X$$

Since a single first order differential equation can be solved by integration, even if it is nonlinear, we can integrate this equation. This results in

$$y(t) = a \tanh(k_1 a t + C) - \frac{1}{2} \, .$$

where $a = \sqrt{X + \frac{1}{4}}$ and $C = \operatorname{arctanh}(\frac{1}{2a})$. In Figure 4 we give the transient output concentration for the case $X = 400$ and $k_1 = 1/2, 1$ and 2. All three solutions approach $\sqrt{X}$ (more precisely, they approach $-\frac{1}{2} + \sqrt{X + \frac{1}{4}}$), but the speed of convergence increases with increasing k_1.

Stochastic model of a square root kernel

Modelling a chemical reaction system using differential equations that describe the time evolution of concentrations is limited to situations where smoothly varying concentrations exist. If the number of molecules is limited, this assumption does not hold anymore. In that case a discrete stochastic model can be used.

Probability distribution We now describe a simple Markov-like stochastic approach to the square root network. Suppose initially there are X input molecules, and no output and waste molecules. In each reaction of the system one input molecule transforms to either an output molecule or a

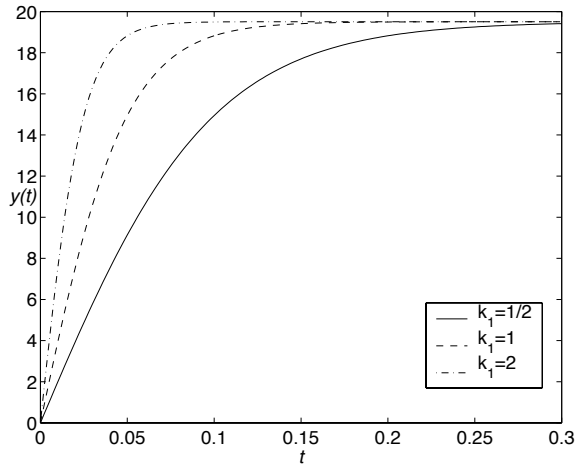

Figure 4: exact solution for $X = 400$ and $k_1 = 1/2, 1$ and 2

waste molecule. Since the total number of molecules is always equal to X, the state of the system can be described by a tuple (q, r) where q is the number of input molecules and r is the number of output molecules. The corresponding number of molecules of the waste compound is then trivially given by $X - q - r$. In general, in a state (q, r) there are two possible state transitions:

1. One of the input molecules transforms into an output molecule. This step happens with propensity $k_1 q$.
2. The other possibility is that an input molecule transforms into a waste molecule. This transition requires an output molecule as catalyst and has a propensity $k_2 q r$.

Since the probabilities of these two possible reactions are nothing but normalised propensities, the actual probabilities of the first and second possible reaction are $\frac{k_1}{k_1+k_2 r} = \frac{1}{1+2r}$ and $\frac{k_2 r}{k_1+k_2 r} = \frac{2r}{1+2r}$ respectively, with $k_2 = 2k_1$. Note that these transition probabilities do not depend on the number of input molecules q. Hence the transition probabilities depend only on the number of output molecules r. This means we can describe the system with the transition graph shown in Figure 5. A step to the right in this transition graph corresponds with an output producing reaction. In state r, i.e., with r output molecules, this reaction has probability $p_r = \frac{1}{1+2r}$. A step from state r to state r in the transition graph corresponds with an input to waste reaction. This reaction has probability $1 - p_r = \frac{2r}{1+2r}$.

Initially the system starts with X input molecules and no output and waste molecules. In terms of the transition graph the system starts in $r = 0$. After X steps all input molecules are used and the system can be in any of the states $r = 1, \ldots, X$ Note that, since $p_0 = 1$, the system cannot produce waste particles and is forced to move to state 1.

Let f_s be the distribution of the number of output molecules after s steps. So $f_s(r)$ is the probability that the

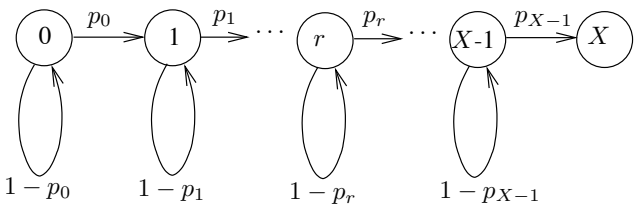

Figure 5: System described by number of output molecules r

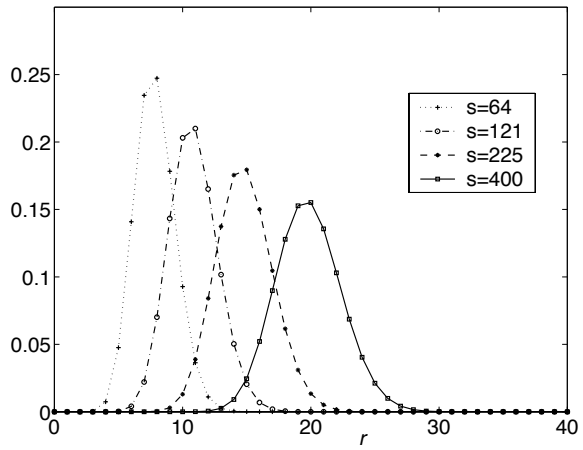

Figure 6: Distribution $f_s(r)$ for $s = 64, 121, 225$ and 400

system is in state r after s steps. Initially $f_0(0) = 1$ and $f_0(r) = 0$ for $r = 1, \ldots, X$. The successive distributions f_s can easily be computed recursively. It is easily seen from Figure 5 that

$$f_{s+1}(r) = \begin{cases} (1 - p_r)\, f_s(r) & \text{if} \quad r = 0 \\ (1 - p_r)\, f_s(r) + p_{r-1}\, f_s(r-1) & \text{if} \quad r \geq 1. \end{cases} \tag{3}$$

With this formula the distributions f_s can easily be computed. In Figure 6 the results are shown for $s = 64, 121, 225$ and 400. As can be seen from Figure 6 the various distributions f_s have their maximum value in $\sqrt{s}$. This means that it is most likely that the stochastic system, started with X input molecules, ends after $s = X$ steps with $\sqrt{X}$ output molecules. However, as Figure 6 shows, other final numbers of output molecules are very well possible.

Mean of the probability distribution The results of the previous subsection suggest that the probability distribution of the number of output molecules after s steps in centered around $\sqrt{s}$. We now try to give a mathematical basis for this observation. Let the mean of probability distribution f_s be given by

$$M_s = \sum_{r=0}^{X} r f_s(r) \; .$$

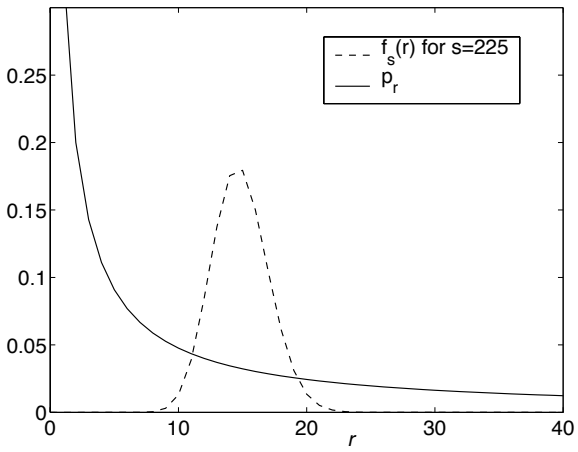

Figure 7: p_r and $f_s(r)$ as function of r for $s = 225$

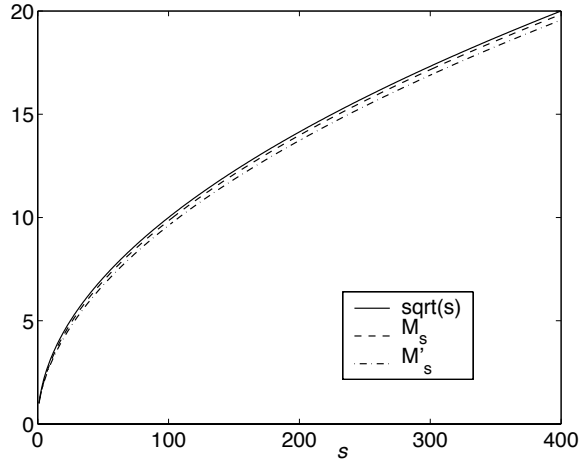

Figure 8: $\sqrt{s}$, M_s and the approximation M'_s

We try to compute M_s. From (3) we obtain for $s < X$.

$$\begin{aligned} M_{s+1} &= \sum_{r=0}^{X} r f_{s+1}(r) \\ &= \sum_{r=0}^{X} r(1-p_r)\, f_s(r) + \sum_{r=1}^{X} r p_{r-1}\, f_s(r-1) \\ &= \sum_{r=0}^{X} r(1-p_r)\, f_s(r) + \sum_{r=0}^{X} (r+1) p_r\, f_s(r) \\ &= \sum_{r=0}^{X} r\, f_s(r) + \sum_{r=0}^{X} p_r\, f_s(r) \\ &= M_s + \sum_{r=0}^{X} p_r\, f_s(r)\ . \end{aligned} \tag{4}$$

The behavior of p_r and $f_s(r)$ as function of r are shown in Figure 7 for the case $s = 225$. This figure shows that the largest contribution to the summation in (4) comes from the r values between 10 and 20. Hence we can approximate the summation in (4) by replacing p_r by a constant value p_{r_0} that gives a good approximation of p_r in the interesting region. In the situation of Figure 7 we could use $r_0 = 15$, thus approximating p_r by the constant p_{15}.

For the general case it is tempting to use $r_0 = \sqrt{s}$, since we conjecture that the maximum of the distribution $f_s(r)$ occurs at $r = \sqrt{s}$. However, since this the goal of this analysis is to compute the mean M_s, it would not be correct to use this conjecture at this point. An alternative is to use the value of M_s for r_0. Since M_s is the mean of the "Gaussian like" distribution $f_s(r)$, the biggest contribution to the summation in (7) originate from r values close to M_s. Thus, approximating p_r in (7) by p_{r_0} with r_0 the (approximate) mean of $f_s(r)$, yields the recurrence relation

$$M'_{s+1} = M'_s + p_{r_0} \sum_{r=0}^{X} f_s(r) = M'_s + p_{r_0} = M'_s + \tfrac{1}{1+2M'_s}\ . \tag{5}$$

It is easily verified that this recurrence relation converges to $\sqrt{s} - \frac{1}{2}$. In fact, it can be shown analytically that the difference $M'_s - (\sqrt{s} - \frac{1}{2})$ is of the order $\mathcal{O}(\frac{\log s}{\sqrt{s}})$. In Figure 8 the exact mean M_s of the distribution, the approximation M'_s and the function $\sqrt{s}$ are shown. Clearly the exact mean of the distribution M_s is close to the "goal" $\sqrt{s}$. So indeed in the stochastic model also the square root is computed. The approximation M'_s obtained from the recurrence relation (5) is a good approximation of the exact mean M_s.

Finally we mention that it is possible to compute the standard deviation of the output, leading to

$$\sigma_X = \mathcal{O}(X^{1/4})\ .$$

This implies that, although the standard deviation increases with increasing input, the coefficient of variation, i.e., the quotient σ_X / M_X, behaves like $\mathcal{O}(X^{-1/4})$ as $X \to \infty$. Consequently the system becomes more and more deterministic as X increases.

Conclusion and Discussion

We have developed a genetic algorithm that allows us to evolve artificial mass-conserving reaction networks that compute a function in terms of amounts of input and output molecules. We have evolved networks that compute an amount of output molecules, approximately equal to the square root of the initial amount of input molecules. Several square root kernels have been identified, resulting in one elementary network motif with two reactions that provides a

good approximation of the square root function. Deterministic and stochastic analyses confirm the desired behavior of this network motif.

The artificial chemistries adopted in the reaction networks in this approach provide a rather rudimentary abstraction of biochemical networks. One limitation of the resulting networks is that they are *single shot* networks. Once the system has reached its equilibrium state, it has to be reset – waste and output molecules need to removed from the system – before a new amount of input molecules can be introduced into the system. By only allowing the input molecule to serve as a catalyst in the reactions and by providing sufficient (constant) resource molecules as additional inputs of the systems, this problem can be overcome. In future work, the assumption of mass-action kinetics is to be expanded to Michaelis-Menten kinetics, which provides more realistic reaction dynamics for the enzymatic reactions envisioned by our approach, but prove harder to grasp analytically. The current implementation can also be adapted to evolve transient behaviors, instead of solely involving the limit behavior of the model as the target of the output function. As such, the genetic algorithm can be adopted to evolve for example oscillatory networks or networks with specific transient responses to temporal inputs.

References

Arkin, A. and Ross, J. (1994). Computational Functions in Biochemical Reaction Networks. *Biophysical Journal*, 67(2):560–78.

Benenson, Y., Gil, B., Ben-Dor, U., and Adar, R. (2004). An Autonomous Molecular Computer for Logical Control of Gene Expression. *Nature*, 429:423–9.

Bhalla, U. (2003). Understanding Complex Signaling Networks through Models and Metaphors. *Progress in Biophysics and Molecular Biology*, 81(1):45–65.

Bhalla, U. and Iyengar, R. (1999). Emergent Properties of Networks of Biological Signaling Pathways. *Science*, 283:381–387.

Bray, D. (1990). Intracellular Signalling as a Parallel Distributed Process. *Journal of Theoretical Biology*, 143(2):215–31.

Bray, D. (1995). Protein Molecules as Computational Elements in Living Cells. *Nature*, 376(6538):307–12.

Buisman, H., ten Eikelder, H., Hilbers, P., and Liekens, A. (2008). Computing Algebraic Functions with Biochemical reaction Networks. *Artificial Life Journal, Special Issue on Artificial Chemistries*.

Deckard, A. and Sauro, H. (2004). Preliminary Studies on the In Silico Evolution of Biochemical Networks. *ChemBioChem*, 5(10):1423–31.

Dittrich, P. (2001). *On Artificial Chemistries*. University of Dortmund.

Dittrich, P., Ziegler, J., and Banzhaf, W. (2001). Artificial Chemistries, A Review. *Artificial Life*, 7(3):225–275.

Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., Kitano, H., Arkin, A., Bornstein, B., Bray, D., Cornish-Bowden, A., et al. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models.

Lenser, T., Hinze, T., Ibrahim, B., and Dittrich, P. (2007). Towards Evolutionary Network Reconstruction Tools for Systems Biology. *Proceedings of the Fifth European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO), LNCS 4447*, pages 132–142.

Machne, R., Finney, A., Muller, S., Lu, J., Widder, S., and Flamm, C. (2006). The SBML ODE Solver Library: a native API for symbolic and fast numerical analysis of reaction networks.

Paladugu, S., Chickarmane, V., Deckard, A., Frumkin, J., McCormack, M., and Sauro, H. (2006). In silico Evolution of Functional Modules in Biochemical Networks. *IEE Proceedings System Biology*, 153:223.

Sauro, H. and Kholodenko, B. (2004). Quantitative Analysis of Signaling Networks. *Progress in Biophysics and Molecular Biology*, 86:5–43.

Steijaert, M., Liekens, A., ten Eikelder, H., and Hilbers, P. (2007). Multiple Functionalities of Biochemical Reaction Networks. *International Conference on Systems Biology (ICSB 2007)*.

Steijaert, M., ten Eikelder, H., Liekens, A., Bosnacki, D., and Hilbers, P. (2008). Stochastic Switching Behavior of a Bistable Auto-phosphorylation Network. *12th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2008)*.

ten Eikelder, H., Crijns, S., Steijaert, M., Liekens, A., and Hilbers, P. (2007). Computing with Feedforward Networks of Artificial Biochemical Neurons. *2nd International Workshop on Natural Computation (IWNC 2007)*.

Tyson, J., Chen, K., and Novak, B. (2003). Sniffers, Buzzers, Toggles and Blinkers: Dynamics of Regulatory and Signaling Pathways in the Cell. *Current Opinion in Cell Biology*, 15:221–231.