Evolving Reaction-Diffusion Systems on GPU

Lidia Yamamoto[1], Wolfgang Banzhaf[2], and Pierre Collet[1]

[1] LSIIT-FDBT, University of Strasbourg, France
{Lidia.Yamamoto,Pierre.Collet}@unistra.fr
[2] Computer Science Department, Memorial University of Newfoundland, Canada
banzhaf@mun.ca

Abstract. Reaction-diffusion systems contribute to various morphogenetic processes, and can also be used as computation models in real and artificial chemistries. Evolving reaction-diffusion solutions automatically is interesting because it is otherwise difficult to engineer them to achieve a target pattern or to perform a desired task. However most of the existing work focuses on the optimization of parameters of a fixed reaction network. In this paper we extend this state of the art by also exploring the space of alternative reaction networks, with the help of GPU hardware. We compare parameter optimization and reaction network optimization on the evolution of reaction-diffusion solutions leading to simple spot patterns. Our results indicate that these two optimization modes tend to exhibit qualitatively different evolutionary dynamics: in the former, the fitness tends to improve continuously in gentle slopes, while the latter tends to exhibit large periods of stagnation followed by sudden jumps, a sign of punctuated equilibria.

1 Introduction

In 1952 Alan Turing [27] proposed reaction-diffusion (RD) as a possible mathematical explanation for morphogenetic processes in nature, especially the formation of patterns on the skin of animals, such as zebra stripes and leopard spots. In a reaction-diffusion system (RDS), a number of chemicals (morphogens) diffuse and react in two or three dimensions within a chemical medium. An RDS is a dynamical system usually described by a set of partial differential equations (PDEs) that quantify the speed of diffusion of chemicals and their reactions. Under some conditions, the equilibrium instability in an RDS may lead to spatial patterns such as Turing patterns. Other well-known RD patterns include circles and spirals in the Belousov-Zhabotinsky reaction, self-replicating spots in the Gray-Scott system [10, 23], and diverse patterns on the surface of sea shells [18]. More complex morphogenetic processes such as the formation of net patterns in leafs or veins [7], the formation of insect eyes and various body parts, are only partially explained by reaction-diffusion processes, nevertheless RD remains an important component of morphogenesis in general.

Several applications of reaction-diffusion systems exist in the literature: first, as abstract models of biological pattern formation [6, 17, 18, 21]: feathers, leaves, veins, nerves, body segments, and so on. More recently, they are considered as

L. Antunes and H.S. Pinto (Eds.): EPIA 2011, LNAI 7026, pp. 208–223, 2011.

new substrates for unconventional computation on real chemical media, such as reaction-diffusion computers [1]: algorithms such as image processing *in vitro*, shortest path, robot controllers, and Voronoi diagrams have been proposed on top of such chemical computers. The potential of RD in micro- and nanotechnology is reviewed in [11], such as the fabrication of structures and devices at very small scales. RD can also be used in models of distributed computation inspired by chemistry, with applications to sensor networks (role assignment [22], routing [15]), and robotics (robot controllers [5], swarm robotics [25]).

Although there is a vast literature on the mathematical analysis of various reaction-diffusion systems, their overall design space remains poorly understood: even very simple systems like the Gray-Scott system (composed essentially of two chemicals and one autocatalytic reaction between them) lead to complex analytical solutions and a very narrow region of the parameter space in which interesting patterns occur. Therefore in many cases it makes sense to sweep RD parameter spaces using search heuristics such as evolutionary algorithms.

There are more reasons to evolve reaction-diffusion systems. Given a desired pattern, it is difficult to find a set of chemicals, their reactions and corresponding parameters (speed of diffusion and reaction) that lead to the target pattern. In addition, in the literature sometimes only the PDEs of an RDS are given: deriving the corresponding chemical reactions is straightforward in some cases, but difficult in others. Beyond parameter sweeping with a Genetic Algorithm (GA) or other techniques, it is worth exploring the RDS design space by finding the appropriate set of chemical reactions leading to a given target pattern. This is analogous to finding a program (here, a chemical program) that solves a given problem, therefore it can be considered as a form of genetic programming (GP). A GP-based approach to RD evolution covers a much larger search space than a GA-based one. Whereas the numeric integration of RD PDEs is already computationally expensive, evolving a population of these PDEs is even more demanding. Luckily, both GP and RD numeric integration are well suited for parallelization on top of GPU (Graphics Processing Unit) hardware [2, 16, 20, 24].

In this paper we compare (experimentally) parameter optimization (GA analog) with reaction network optimization (GP analog) to explore the space of RD solutions forming spot patterns. Such comparison is enabled by a hybrid CPU-GPU evolutionary algorithm that makes the problem tractable with a modest investment in computation resources. The paper is structured as follows: Section 2 provides background on RDS, their evolution and parallelization. Section 3 presents our approach to evolving RDS. Section 4 reports our evolution experiments and discusses their results. Section 5 concludes the paper.

2 Reaction-Diffusion Systems

An RDS is a chemical reaction system in which substances react and diffuse in space. The movement of molecules, their collisions and reactions are stochastic processes at the microscopic level. At the macroscopic level though (for large numbers of molecules) the system can be expressed as a set of PDEs describing

the change in concentrations of substances caused by both reaction and diffusion effects combined:

$$\frac{\partial s_i(\boldsymbol{p},t)}{\partial t} = f_i(s_i(\boldsymbol{p},t)) + D_i\nabla^2 s_i(\boldsymbol{p},t) \tag{1}$$

where s_i is the concentration level at time t of each chemical S_i at position $\boldsymbol{p} = (x, y, z)$. The reaction term $f_i(s_i(\boldsymbol{p}, t))$ describes the reaction kinetics for chemical S_i at each point $\boldsymbol{p}$. The diffusion term $D_i\nabla^2 s_i(\boldsymbol{p}, t)$ tells how fast each chemical substance diffuses in space. D_i is the diffusion coefficient of S_i (a constant scalar in the case of isotropic diffusion), and ∇^2 is the Laplacian operator.

In the simplest case, the dynamics of the reaction term (described by the set of functions f_i) follows the *Law of Mass Action*: it states that, in a well-stirred reactor, the average speed (or rate) of a chemical reaction is proportional to the product of the concentrations of its reactants. For n chemicals and m reactions, the system of differential equations for the reaction terms can be described in matrix notation as:

$$\frac{d\boldsymbol{s}(t)}{dt} = \mathbf{M}\boldsymbol{v}(t) \tag{2}$$

where $\mathbf{M}$ is the stoichiometric matrix of the system, which expresses the net changes in number of molecules for each species $S_i, 1 \leq i \leq n$ in each reaction $R_j, 1 \leq j \leq m$; and $\boldsymbol{v}(t) = \{v_1, ...v_j, ...v_m\}$ is a vector of rates for each reaction, for instance, following the Law of Mass Action:

$$v_j = k_j \prod_{1 \leq l \leq n} s_l^{\mathbf{M}_{e(l,j)}} \tag{3}$$

where k_j is kinetic or rate coefficient of reaction R_j (constant for our purposes), and $\mathbf{M}_e$ is the educt stoichiometric matrix where each element $\mathbf{M}_{e(l,j)}$ expresses the consumption of molecules of S_l in reaction R_j. This provides a simple and automatic way to obtain the system of PDEs from a given set of chemical reactions. The system can then be integrated numerically by discretizing the equation terms in space and time.

2.1 Activator-Inhibitor Models

Activator-inhibitor models [17] are among the simplest reaction-diffusion systems known. The experiments described in this paper focus on the automatic evolution of solutions that fall into this class. The activation-inhibition effect can be achieved by two interacting morphogens: an activator catalyzes its own production and produces an inhibitor that inhibits the autocatalytic action of the activator. Moreover the inhibitor travels faster than the activator, leading to a short-range activation and long-range inhibition effect that under some conditions [21] results in the formation of spot and stripe patterns.

Numerous alternative activator-inhibitor models are available in the literature. A lot is known about specific cases, however much remains to be explored in the vast design space of reaction-diffusion systems that form given patterns.

A well-known activator-inhibitor model is the one by Gierer and Meinhardt [17], described by the following equations:

$$\frac{\partial a}{\partial t} = \frac{\sigma a^2}{h} - \mu_a a + \rho_a + D_a \nabla^2 a \tag{4}$$

$$\frac{\partial h}{\partial t} = \sigma a^2 - \mu_h h + \rho_h + D_h \nabla^2 h \tag{5}$$

where a and h are the concentrations of activator (A) and inhibitor (H), respectively; μ_a and μ_h are the decay rates for each substance; ρ_a and ρ_h are their inflow rates; D_a and D_h are their respective diffusion coefficients.

Note that the inhibition factor $1/h$ does not stem directly from the Law of Mass Action. In [28] the corresponding chemical reactions are derived for the above equations, by introducing a catalyst C that is needed in the autocatalysis of A but is consumed by H, leading to the expected inhibition effect:

$$2A \xrightarrow{\sigma} 2A + H \tag{6}$$

$$C + 2A \xrightarrow{k_1} C + 3A \tag{7}$$

$$C + H \xrightarrow{k_2} H \tag{8}$$

The factor $1/h$ is obtained by assuming that the catalyst C is in steady state with constant concentration, and by setting $\sigma = k_1 \rho_c / k_2$ (where ρ_c is the injection rate of C). C neither decays ($\mu_c = 0$) nor diffuses ($D_c = 0$).

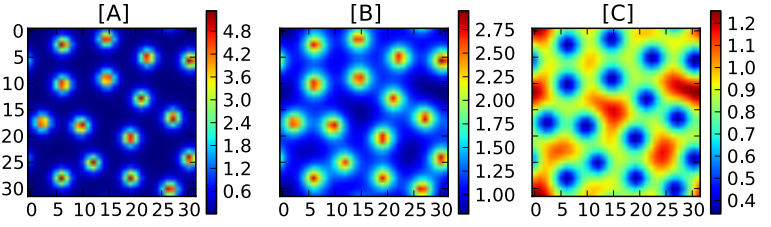


Fig. 1. Activator-inhibitor pattern

Figure 1 shows a typical pattern formed using our implementation of this model, using the parameters from [13, 28], namely: $\sigma = 0.02$, $\rho_a = \rho_h = 0$, $\rho_c = 0.1$, $\mu_a = 0.01$, $\mu_h = 0.02$, $\mu_c = 0$, $k_1 = 0.01$, $k_2 = 0.1$, $D_a = 0.005$, $D_c = 0$, $D_h = 0.2$. [A], [B] and [C] represent the concentrations of activator, inhibitor and catalyst, respectively, plotted as heatmaps (colors indicate concentration levels at each point in space, according to the colorbars to the right of each plot). One can see that the activator peaks coincide with the inhibitor peaks, which are smoother. Moreover the activator/inhibitor peaks coincide with the catalyst valleys, consistent with the fact that the inhibitor consumes such catalyst.

2.2 Evolving Reaction-Diffusion Solutions

Most interesting RDSs are open systems: they rely on a constant inflow and outflow of substances that drives the system away from equilibrium. Moreover they rely heavily on positive and negative feedback (e.g. activation and inhibition), and often result in non-linear PDEs. Combined with the high computation demands on their numeric integration, evolving them automatically becomes very challenging.

The evolution of spot and stripe patterns is presented in [9], via the optimization of parameters of transition rules that emulate Turing patterns on a binary Cellular Automaton (CA). Such CA emulation is simpler and faster than full RD, but sheds no light on the chemical processes behind the observed patterns.

RD parameter sweeping for biology has been demonstrated in [12] using the covariance matrix adaptation evolution strategy (CMA-ES), a variant of evolution strategy suitable for sweeping parameters on difficult fitness landscapes. CMA-ES has also been used for tuning parameters in self-configuring morphogenetic modular robots [19] (a system that is loosely inspired by chemistry), and in the numerical simulation of turbulent fluids [8].

RDS solutions for one-dimensional segmentation (head-tail pattern) were evolved in [26]. Four models were compared: three parameter sweeping variants and the evolution of arbitrary differential equations via GP. The equations obtained by GP performed best, however their actual chemical implementation was not covered.

Stochastic particle-based (discrete) simulations of RDS have been used in [3] to study the molecular evolution of replicators. In [3] molecules are strings that may move in space and react due to complementary substrings, leading to an emergent evolutionary dynamics in space.

RD controllers for robots were evolved in [5], based on the Gray-Scott system in one dimension (ring). Chemicals in the RD ring were used to activate the robot's motors in response to sensor input. The wiring from the sensors to the RD ring were evolved (not the actual RDS).

To the best of our knowledge, the use of evolutionary optimization to discover new chemical reaction networks implementing desired patterns has not been reported so far in the literature (most cases [8, 12, 19] are restricted to parameter sweeping; novel reaction networks may emerge in [3] but with no assigned computation task; arbitrary equations are evolved in [26] but with no guaranteed chemical analog). Our contribution represents a step in this direction.

2.3 Reaction-Diffusion on GPU

GPU approaches to parallel PDE solving for advection-reaction-diffusion equations are discussed and evaluated in [24]. Besides reaction and diffusion, these systems take into account advection phenomena in which chemicals also move by fluid transport, such as pollutants in the air or water. An implementation of reaction-diffusion in three dimensions is reported in [20]. Accompanying source code is publicly available for both [20, 24], however both are far more elaborate

than what we needed for our own experiments. Moreover, none of these implementations covered the execution and fitness evaluation of multiple reaction-diffusion individuals in parallel on the same GPU card. Therefore we have decided to reimplement our own solution from scratch. To the best of our knowledge, the evolution of RDS on GPUs has not been reported so far.

3 RD Evolutionary Algorithm

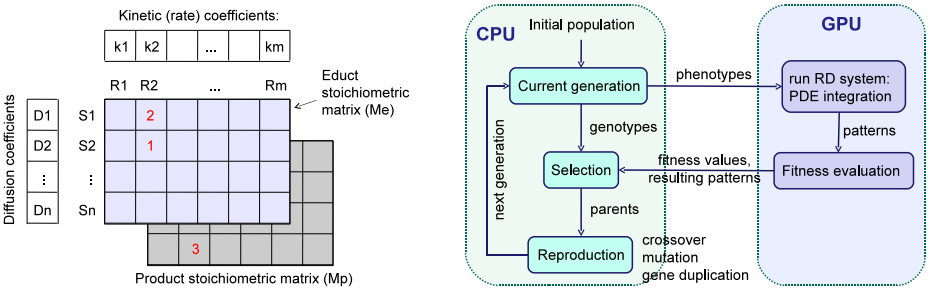
In order to cope with the high computation demands of evolving RDS, we have developed an evolutionary algorithm that runs on the CPU and evaluates the population of candidate solutions (individuals) on the GPU. Two levels of parallelism are exploited: population (multiple individuals on the GPU) and individual (parallel PDE integration on a 2D surface).

As a developmental system, an RDS implies a separation of genotype (the set of reactions that is evolved) and phenotype (the resulting pattern on the surface). The genotype and phenotype encoding are described in Section 3.1, the genetic operations applied to the genotype in Section 3.2, and the fitness evaluation process in Section 3.3.

3.1 Genotype and Phenotype

An individual in the population consists of a genotype, a phenotype and a fitness value. The genotype or genome encodes the reactions and parameters of the RDS. The genome of the individuals is shown in Fig. 2(a). It consists of four elements: the vector of kinetic coefficients $\boldsymbol{k} = \{k_1, ...k_m\}$ for each reaction; the vector of diffusion coefficients $\boldsymbol{D} = \{D_1, ...D_n\}$ for each substance; the educt stoichiometric matrix $\mathbf{M}_e$; and the product stoichiometric matrix $\mathbf{M}_p$. Both matrices have size $n \times m$. An element $\mathbf{M}_e(i, j)$, respectively $\mathbf{M}_p(i, j)$, tells how many molecules of species i are consumed (resp. produced) in reaction j. For example, reaction R_2 in Fig. 2(a) takes two molecules of S_1 and one of S_2 to produce 3 molecules of the last species S_n. The matrices $\mathbf{M}_p$ and $\mathbf{M}_e$ together describe all the reactions that occur in the system (without their rates k_j). The net stoichiometric matrix $\mathbf{M}$ (needed for the PDE integration as shown in Eq. 2) can then be simply calculated as $\mathbf{M} = \mathbf{M}_p - \mathbf{M}_e$.

The phenotype is the pattern that results from numerically integrating the PDE of the corresponding genome on a 2D surface of $N_x \times N_y$ points (cells) arranged on a regular grid. More specifically, it corresponds to $\boldsymbol{s}(\boldsymbol{p}, t)$, the set of n matrices of size $N_x \times N_y$ containing the concentrations of each of the n species at each point $\boldsymbol{p}$ on the surface, at an observed time t. The phenotype changes in time as the pattern takes shape starting from the initial conditions. After allowing some initial time for a potential pattern to form, the fitness evaluation of the individual starts in periodic rounds. At the end of the evaluation period, the average of all the periodic evaluations is computed, and the obtained fitness value is assigned to the corresponding individual. At every generation, all the individuals are evaluated under new, random initial conditions. The same initial



(a) genome of RD individuals (b) CPU-GPU division of roles

Fig. 2. Elements of the reaction-diffusion CPU-GPU evolutionary algorithm

conditions are applied to all individuals of the same generation, like exposing them to the same environment. Even if an individual may survive from one generation to the next, its fitness is re-evaluated under new conditions. Therefore only those individuals who have a good fitness under all faced situations can survive in the long run.

3.2 Genetic Operators

Three types of genetic operators are used: mutation, crossover and gene duplication. The maximum number of chemicals (n) and reactions (m) is constant, for both parameter and network optimization cases.

For the case of parameter sweeping, only the coefficients in $\boldsymbol{k}$ may be modified by mutation and crossover (gene duplication is disabled). Since these coefficients typically vary by several orders of magnitude, they are expressed in scientific notation, with integer mantissa and exponent fields that may be mutated independently. A mutation increments or decrements one of these fields. A circular crossover is used, in which the vector of coefficients is treated as a ring; two points are chosen on each parent's ring and their segments are swapped.

For the case of reaction network evolution, in addition to $\boldsymbol{k}$, the matrices $\mathbf{M}_e$ and $\mathbf{M}_p$ may be modified too. A mutation may increment or decrement one element of $\mathbf{M}_e$ or $\mathbf{M}_p$, which results in modifying the corresponding chemical reaction. For instance, incrementing $\mathbf{M}_e(i,j)$ (resp. $\mathbf{M}_p(i,j)$) means that now an additional molecule of S_i is needed (resp. is produced) in reaction R_j. Circular crossover can also be applied: in this case, individuals swap the full description of the affected reaction ($\mathbf{M}_e$ and $\mathbf{M}_p$ columns, plus corresponding rate coefficient).

For network evolution, the population is initialized with random genomes containing a small number of reactions. Gene duplication may then be used to create new reactions from existing ones. In this case, the child individual will contain two identical reactions that may diverge in subsequent mutations.

Currently the diffusion coefficients are taken as fixed, i.e. they are not evolved, assuming that the diffusion speed is an inherent property of the mobility of the chemical in the medium due to its shape or size.

3.3 Fitness Evaluation

Since we are interested in the computation properties of RDS, a task-based fitness function was chosen. This function measures the capacity of the RDS to perform a desired computation task. The goal is then to maximize the individual's performance on this task. The RDS is then regarded as a chemical algorithm performing a distributed computation. The goal task chosen for this paper is a distributed cluster head election as described in [28]: the cells where activator peaks are found in spot patterns are "elected" as "cluster heads" (local leaders) of their region in space.

The cluster head evaluation function works as follows. First, four cell types are distinguished: *invalid* (with concentrations outside bounds, `inf` (infinity), or `nan` ("not a number" numerical error)); *peak* (valid cell with activator concentration above a threshold); *alive*: valid non-peak cell sufficiently close to a peak; *dead* (a valid non-peak cell without any nearby peaks). The fitness Φ of a single cell is then computed by respecting the following relations:

$$\Phi(\text{invalid}) < \Phi(\text{dead}) < \Phi(\text{peak}) < \Phi(\text{alive}) \tag{9}$$

Therefore a pattern containing the maximum number of alive cells has the maximum fitness. However, alive cells rely on peak cells, therefore peaks must be present too. Dead cells must be penalized with a low fitness, but invalid cells are even worse, so they should receive a larger penalty. The global fitness of the pattern (individual) is then the sum of the fitness values of all cells on the surface. The actual Φ values adopted in the experiments are reported in Sec. 4.

In RDS, there is a vast region of the search space in which no patterns occur [12], or which lead to misbehaving solutions, for instance, solutions that include autocatalytic reactions that quickly exhaust the available resources and produce infinity values. In order to eliminate these invalid individuals as soon as possible, an incremental fitness evaluation approach is adopted, with three stages: First, the individual is tested for having concentrations within valid bounds. Second, it is tested for the presence of any non-homogeneous pattern. At the third and last stage comes the evaluation against the real task according to the above relations (9). Only when the first stage is overcome can the individual move to the second stage, and so on.

3.4 CPU-GPU Evolutionary Algorithm

In order to cope with the intensive computation requirements of RD evolution, the PDE integration and most of the fitness evaluation are delegated to a GPU card. The resulting hybrid CPU-GPU evolutionary algorithm is shown in Fig. 2(b).

The initial population is generated on the host computer, together with the initial conditions for the patterns. The phenotypes are then transferred to the GPU where they are integrated for pattern formation. PDE integration is implemented as kernel routine on the GPU, where each thread is responsible for numerically integrating Eq. 1 for one point on the grid of cells.

The fitness of the obtained patterns is then evaluated on the GPU, as follows: First, each thread computes the local fitness value of its cell by observing its concentration values and the concentration values in the nearby cells. After all threads have completed this step, the first thread of each block computes the sum of the fitness values of all the cells in the block. These values are then returned to the CPU, where the fitness of all the blocks is summed up to obtain the global fitness of the pattern. This process is repeated for a number of evaluation rounds, at the end of which the fitness of the individual is computed as the average fitness of all the evaluations.

After all the fitness values are computed, a selection process takes place, during which individuals compete in tournaments. The winners of the tournaments gain the right to reproduce via mutation, crossover and/or gene duplication, in order to produce the next generation, which is then re-evaluated on the GPU, and so on, until the maximum amount of generations is reached.

The PDE integration routine deserves special attention since it is the most computationally demanding part of the evolutionary algorithm. It typically consumes over 99% of the total GPU plus CPU computation time. Therefore it is important to optimize its performance. For this purpose, the stoichiometric matrices, reaction and diffusion coefficients are placed in the GPU block's shared memory. This results in considerable performance improvements, as frequently reported in the GPU literature. The experiments were run on a host containing three NVIDIA GTX 480 GPU devices. For the pure PDE integration without evolution, the typical speedups obtained with the GPU implementation range from 100 to 180 times the CPU runtime, for one RD individual filling the whole GPU card (size 128x120 points). A typical evolution run with a population of 150 individuals of 32x32 points over 100 generations still takes around 20 to 50 minutes to complete using the hybrid CPU-GPU evolutionary algorithm presented (against more than 30 hours on a single CPU), depending on the size of the reaction networks involved. Since several runs are usually needed in order to explore the RD design space, such experiments would have been infeasible on a single CPU.

4 Evolution Experiments

We start by sweeping parameter space for spot patterns using the activator-inhibitor system shown in Sec. 2.1. We then explore the space of possible reaction networks leading to such patterns. The results are presented in Secs. 4.1 and 4.2 respectively. Table 1 lists the parameters of the experiments. The population size, number of generations and grid surface were set to the minimum values that still showed some evolutionary behavior, while keeping the runtimes short enough (below one hour per run). The mutation and crossover probabilities were set to the typical ranges used in GP, whereas the gene duplication probability was set low enough according to the results in [4]. The tournament size was set to the minimum, in order to reduce the selection pressure in an attempt to prevent premature convergence, which was very commonly observed in these

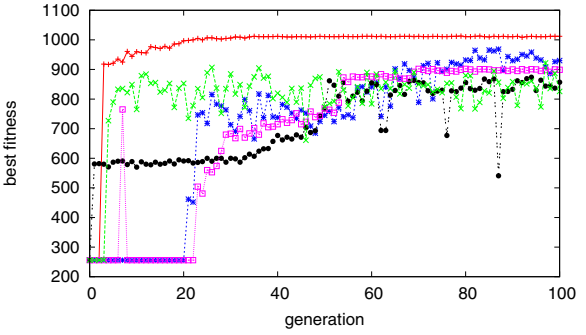


Fig. 3. Selected parameter sweeping runs (each curve corresponds to one run)

experiments. Indeed, we have observed that increasing the selection pressure (up to a tournament of size 7) led more often to premature convergence to non-optimum solutions.

The global fitness of a pattern (sum of the fitness values of each cell) ranges from zero to the total number of grid cells, i.e. 32x32=1024 for the given setup. Peaks are points of maximum activator concentration, above a peak threshold of $a_{peak} = 3.0$. The distance threshold for a cell to be considered as alive is $d_{max} = 6$ cells from the nearest peak. For each cell type, the values of the task-based fitness function per cell are: $\Phi(\text{invalid}) = 0$; $\Phi(\text{dead}) = 0.25$; $\Phi(\text{peak}) = 0.5$ if there is another peak at distance less than d_{max} from itself; $\Phi(\text{peak}) = 0.75$ otherwise; $\Phi(\text{alive}) = 1$.

Table 1. Parameters of the evolution experiments

Evolution parameters	
population size	150 individuals
generations	100
selection method	tournament
tournament size	2 individuals
mutation prob.	0.1 per reaction
mutation prob.	0.2 per mating event
crossover prob.	0.8 per mating event
gene dup. prob.	0.04 per mating event

Phenotype parameters	
grid surface	32x32 points
PDE integration timestep	$\Delta t = 0.1$ s
activator peak threshold	$a_{peak} = 3.0$
max. dist. to nearest peak	$d_{max} = 6.0$
max. valid concentration	$s_{max} = 100$
Fitness evaluation parameters	
n. evaluation rounds	10 per indiv.
evaluation start time	at t=2000 s
evaluation interval	every 200 s

4.1 Sweeping Parameter Spaces

In this set of experiments, we take Reactions 6 to 8 from Sec. 2.1 as given, and let evolution find matching rate coefficients that lead to spot patterns similar to those on Fig. 1. The coefficients are allowed to vary within the interval $0 \leq k < 1000$.

Figure 3 shows the best fitness for the 5 runs that gave the best results. Each curve corresponds to one run. The other runs got stuck at poor fitness values near

the bottom, and are not shown for better readability of the plots. These poor runs usually overcome the first fitness stage but either reach flat homogeneous surfaces with no patterns, unstable patterns, or patterns that do not match the fitness criteria well enough.

Note that the fitness curves vary widely across runs, hence showing averages over several runs would make little sense. This is why we have selected a few runs to plot such as to achieve a compromise between the readability of the plots and the amount of information conveyed by them.

The best evolved patterns are shown in Fig. 4, where [A], [B] and [C] represent the concentrations of activator, inhibitor and catalyst, respectively. The rate coefficients for the best solutions found are shown in Table 2, compared to the human-designed parameter values shown in Sec. 2.1. The solution producing Fig. 4 presents tall and narrow activator peaks, and depletes the catalyst almost entirely. It does that by increasing the rate of autocatalytic production of A, and increasing the rate of consumption of C by the inhibitor, when compared to the hand-made case. These narrow peaks are very good for the cluster head task, since the number of alive cells is maximized. This solution is probably more efficient than the human-designed case.

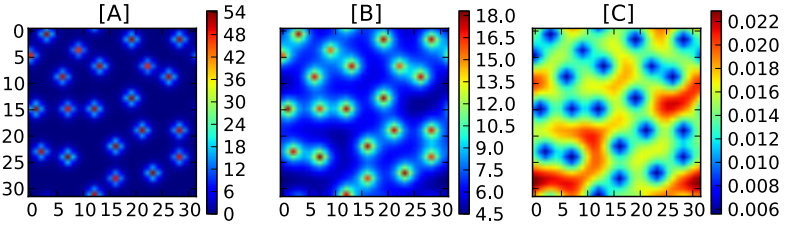


Fig. 4. Evolved patterns for parameter sweeping

Table 2. Evolved vs. hand-made rate coefficients

Reactions	hand-made	Fig. 4
$2A + C \rightarrow 3A + C$	1.0e-2	9.4e-2
$B + C \rightarrow B$	1.0e-1	3.0e-1
$2A \rightarrow 2A + B$	2.0e-2	1.9e-3

Reactions	hand-made	Fig. 4
$\emptyset \rightarrow C$	1.0e-1	3.1e-2
$A \rightarrow \emptyset$	1.0e-2	1.5e-2
$B \rightarrow \emptyset$	2.0e-2	2.0e-2

4.2 Exploring New Reaction Networks

We now let the reaction networks evolve, for a set of $n = 3$ chemicals and up to $m = 4n = 12$ reactions, including inflow and decay reactions. All the individuals have at least n inflow reactions and n decay reactions (one inflow and one decay reaction for each chemical). These reactions are statically defined and cannot be added, deleted nor mutated. Only their rate coefficients may change during evolution. Some coefficients may be imposed. For instance, we disable the

injection of activator and inhibitor by setting their coefficients to zero, in order to avoid solutions that "cheat" by artificially injecting these chemicals. We refer to the remaining up to $2n$ reactions as the set of *evolvable* reactions. In order to avoid unfeasible high-order reactions, when mutating evolvable reactions, the maximum stoichiometric coefficient per molecule is kept at 3: $\mathbf{M}_e(i,j) \leq 3$ and $\mathbf{M}_p(i,j) \leq 3\ \forall i,j$; moreover the maximum number of molecules that participate in a reaction (on either side) is 4: $\sum_i \mathbf{M}_e(i,j) \leq 4$ and $\sum_i \mathbf{M}_p(i,j) \leq 4$ for all reactions R_j.

In the initial population, each individual is initialized with a genome containing a single evolvable reaction (plus the default $2n$ static inflow and decay reactions). More reactions may appear later as genes duplicate and diverge. In this way, we start by exploring the search space of smaller, parsimonious solutions, and then let evolution grow more complex solutions if they happen to be fitter.

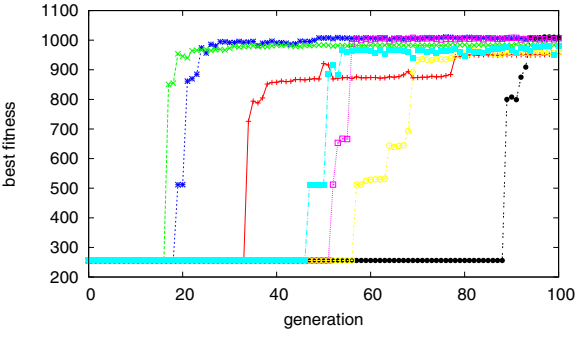


Fig. 5. Selected reaction network evolution runs (each curve corresponds to one run)

Again, a set of 10 runs were performed. Figure 5 shows the best fitness over the generations, for the 7 runs that made progress. The other 3 runs got stuck at the bottom of the plot and have been omitted. Compared to parameter evolution (Fig. 3) a remarkably different qualitative behavior can be noticed: Here the fitness exhibits periods of stagnation followed by steep jumps. Such profile is characteristic of indirect phenotype-genotype-fitness encodings, of which RD is an example, so it should come at no surprise. However, it seems to become significantly more apparent in network evolution than in parameter evolution. A tentative explanation for this might be that network evolution has a much larger range of possibilities of neutral mutations (mutations that have no impact in the fitness) than parameter evolution. These neutral mutations may accumulate over time, until they start to play a role in the fitness. This topic deserves a deeper investigation.

Two of the best evolved patterns are shown in Fig. 6. Their peaks are sharper than those in the hand-made solution of Fig. 1, and even sharper than in the case of Fig. 4, consistent with the fitness function that rewards for a maximum amount of alive cells that must nonetheless be close to a peak. Note that in these

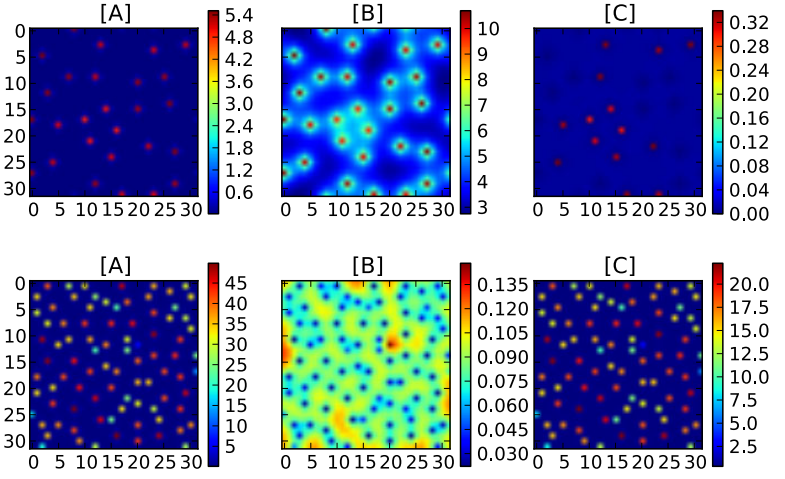


Fig. 6. Patterns stemming from reaction network evolution

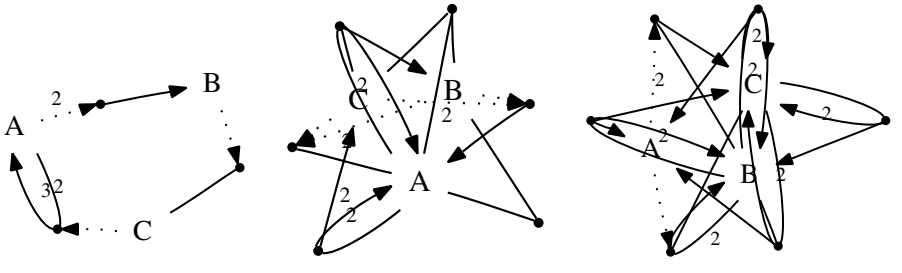
Table 3. Two evolved reaction network solutions

Fig. 6 (top)

Reactions	k
$A + C \rightarrow 2A + B$	6.2
$2A \rightarrow A + 2C$	6.4e-2
$2A + B + C \rightarrow B + C$	2.3e-2
$A + B + C \rightarrow \emptyset$	1.0e-1
$A + B \rightarrow \emptyset$	2.0e-2
$B + 2C \rightarrow A + B + 2C$	1.0

Fig. 6 (bottom)

Reactions	k
$A + 2B + C \rightarrow A + B$	5.0e-3
$2C \rightarrow B + C$	0.0
$B + 2C \rightarrow A + C$	0.0
$2B \rightarrow A + B + C$	1.0
$B + 2C \rightarrow A + 2B + C$	1.0
$2A + B \rightarrow 2A + C$	2.0e-1



Hand-made Fig. 6 (top) Fig. 6 (bottom)

Fig. 7. Best reaction networks evolved (center, right) vs. hand-made solution (left)

solutions the substance C no longer plays the role of catalyst consumed by an inhibitor, since the peaks of C now coincide with those of A.

The corresponding genomes for the patterns of Fig. 6 are listed in Table 3. The respective reaction networks are depicted in Fig. 7, compared with the

hand-made case. Only the evolvable reactions are shown. In this figure, a dot indicates a reaction, continuous lines indicate educts, continuous arrows indicate products, dotted arrows indicate catalysis, and numbers indicate stoichiometric coefficients. We can see that the evolved networks are densely connected and redundant, i.e. there are many ways to produce and consume each substance. This is consistent with the fact that the evolved networks must be robust to mutations in the network topology in order to survive to the next generation.

5 Conclusions

RD evolution is a difficult task with huge computation demands. In this paper we have presented experimental results that compare parameter sweeping and reaction network evolution on the activator-inhibitor model. The experiments were made feasible by the parallelization of fitness evaluation and PDE integration on GPU hardware. The outcome of our experiments indicates that the evolutionary dynamics of RD network evolution seems to present remarkable qualitative differences when compared to RD parameter sweeping. Steep jumps in fitness are very frequently observed during network evolution, while similar jumps are only rarely observed in the case of parameter sweeping. This could be an evidence of punctuated equilibria in the evolution of RD networks, and is a topic that deserves further investigation.

The present paper covered only very simple spot patterns. The evolution of more complex patterns will require a number of extensions to our system. Further investigation into the genotype and phenotype structures is needed, moving towards Genetic Regulatory Networks (GRN), and looking at how modularity could emerge in the corresponding chemical reaction networks [4]. Another envisaged extension is to impose mass and energy conservation constraints to reduce the search space to resource-saving and physically plausible solutions. In order to evolve vein and leaf patterns [7, 18], the active transport of chemicals is desirable, beyond passive diffusion. In order to improve convergence speed and escape more easily from local optima, another research topic would be the adaptation of more sophisticated algorithms, such as CMA-ES [12, 19] or the nested evolution algorithm from [14], to the evolution of reaction networks in the RDS context.

Acknowledgments. This work was supported by the French Region Alsace through the EVOL grant.

References

1. Adamatzky, A., Costello, B.D.L., Asai, T.: Reaction-Diffusion Computers. Elsevier Science Inc., New York (2005)
2. Banzhaf, W., Harding, S., Langdon, W.B., Wilson, G.: Accelerating Genetic Programming through Graphics Processing Units. In: Genetic Programming Theory and Practice VI, pp. 1–19. Springer, US (2009)
3. Breyer, J., Ackermann, J., McCaskill, J.: Evolving Reaction-Diffusion Ecosystems with Self-Assembling Structures in Thin Films. Artificial Life 4(1), 25–40 (1998)

4. Calabretta, R., Nolfi, S., Parisi, D., Wagner, G.P.: Duplication of Modules Facilitates the Evolution of Functional Specialization. Artificial Life 6(1), 69–84 (2000)
5. Dale, K., Husbands, P.: The Evolution of Reaction-Diffusion Controllers for Minimally Cognitive Agents. Artificial Life 16(1), 1–20 (2010)
6. Deutsch, A., Dormann, S.: Cellular automaton modeling of biological pattern formation: characterization, applications, and analysis. Birkhäuser (2005)
7. Fujita, H., Mochizuki, A.: The Origin of the Diversity of Leaf Venation Pattern. Developmental Dynamics 235(10), 351–361 (2006)
8. Fukagata, K., Kern, S., Chatelain, P., Koumoutsakos, P., Kasagi, N.: Evolutionary optimization of an anisotropic compliant surface for turbulent friction drag reduction. Journal of Turbulence 9(35), 1–17 (2008)
9. Graván, C.P., Lahoz-Beltra, R.: Evolving morphogenetic fields in the zebra skin pattern based on Turing's morphogen hypothesis. Int. J. Appl. Math. Comp. Sci. 14(3), 351–361 (2004)
10. Gray, P., Scott, S.: Chemical Oscillations and Instabilities: Nonlinear Chemical Kinetics. Oxford Science Publications, Oxford (1990)
11. Grzybowski, B.A., Bishop, K.J.M., Campbell, C.J., Fialkowski, M., Smoukov, S.K.: Micro-and nanotechnology via reaction-diffusion. Soft Matter 1, 114–128 (2005)
12. Hohm, T., Zitzler, E.: A Hierarchical Approach to Model Parameter Optimization for Developmental Systems. BioSystems 102, 157–167 (2010)
13. Koch, A.J., Meinhardt, H.: Biological pattern formation: from basic mechanisms to complex structures. Reviews of Modern Physics 66 (1994)
14. Lenser, T., Hinze, T., Ibrahim, B., Dittrich, P.: Towards Evolutionary Network Reconstruction Tools for Systems Biology. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 132–142. Springer, Heidelberg (2007)
15. Lowe, D., Miorandi, D., Gomez, K.: Activation- inhibition-based data highways for wireless sensor networks. In: Proc. Bionetics. ICST (2009)
16. Maitre, O., Baumes, L.A., Lachiche, N., Corma, A., Collet, P.: Coarse grain parallelization of evolutionary algorithms on GPGPU cards with EASEA. In: Proc. GECCO, pp. 1403–1410 (2009)
17. Meinhardt, H.: Models of biological pattern formation. Academic Press, London (1982)
18. Meinhardt, H.: The Algorithmic Beauty of Sea Shells, 4th edn. Springer, Heidelberg (2009)
19. Meng, Y., Zhang, Y., Jin, Y.: Autonomous self-reconfiguration of modular robots by evolving a hierarchical mechanochemical model. IEEE Computational Intelligence Magazine 6(1), 43–44 (2011)
20. Molnár Jr., F., Izsák, F., Mészáros, R., Lagzi, I.: Simulation of reaction-diffusion processes in three dimensions using CUDA. ArXiv e-prints (April 2010)
21. Murray, J.D.: Mathematical Biology: Spatial Models and Biomedical Applications, vol. 2. Springer, Heidelberg (2003)
22. Neglia, G., Reina, G.: Evaluating activator-inhibitor mechanisms for sensors coordination. In: Proc. Bionetics. ICST (2007)
23. Pearson, J.E.: Complex patterns in a simple system. Science 261(5118), 189–192 (1993)
24. Sanderson, A.R., Meyer, M.D., Kirby, R.M., Johnson, C.R.: A framework for exploring numerical solutions of advection-reaction-diffusion equations using a GPU-based approach. Computing and Visualization in Science 12(4), 155–170 (2009)

25. Shen, W.M., Will, P., Galstyan, A., Chuong, C.M.: Hormone-inspired self-organization and distributed control of robotic swarms. Autonomous Robots 17(1), 93–105 (2004)
26. Streichert, F., Spieth, C., Ulmer, H., Zell, A.: How to evolve the head-tail pattern from reaction-diffusion systems. In: NASA/DoD Conference on Evolvable Hardware, pp. 261–268. IEEE Computer Society Press, Los Alamitos (2004)
27. Turing, A.M.: The chemical basis of morphogenesis. Phil. Trans. Royal Soc. London B 327, 37–72 (1952)
28. Yamamoto, L., Miorandi, D.: Evaluating the Robustness of Activator-Inhibitor Models for Cluster Head Computation. In: Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., Stützle, T. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 143–154. Springer, Heidelberg (2010)