Nat Comput (2014) 13:247–256
DOI 10.1007/s11047-013-9384-7

# Symbol representations and signal dynamics in evolving droplet computers

**Gabi Escuela · Gerd Gruenert · Peter Dittrich**

Published online: 16 July 2013


**Abstract** We investigate several evolutionary computation approaches as a mechanism to "program" networks of excitable chemical droplets. For this kind of systems, we assigned a specific task and concentrated on the characteristics of signals representing symbols. Given a Boolean function as target functionality, 2D networks composed of $10 \times 10$ droplets were considered in our simulations. Three different set-ups were tested: Evolving network structures with fixed on/off rate coding signals, co-evolution of networks and signals, and network evolution with fixed but pre-evolved signals. Evolutionary computation served in this work not only for designing droplet networks and input signals but also to estimate the quality of a symbol representation: we assume that a signal leading to faster evolution of a successful network for a given task is better suited for the droplet computing infrastructure. Results show that complicated functions like XOR can evolve using only rate coding and simple droplet types, while other functions involving negations like the NAND or the XNOR function evolved slower using rate coding. Furthermore we discovered symbol representations that performed better than the straight forward on/off rate coding signals for the XNOR and AND Boolean functions. We conclude that our approach is suitable for the exploration of signal encoding in networks of excitable droplets.

**Keywords** Chemical computer · Droplet network · Evolutionary algorithm · Logic gate · Signal encoding · Symbol representation

Gabi Escuela and Gerd Gruenert have contributed equally.

G. Escuela · G. Gruenert · P. Dittrich (✉)
Bio Systems Analysis Group, Department of Computer Science, Friedrich Schiller University Jena, Ernst-Abbe-Platz 2, 07743 Jena, Germany
e-mail: peter.dittrich@uni-jena.de

G. Escuela
e-mail: gabi.escuela@uni-jena.de

G. Gruenert
e-mail: gerd.gruenert@uni-jena.de

## 1 Introduction

In an excitable medium the propagations and collisions of waves of chemical activity can be used for computation (Adamatzky 2001; Adamatzky et al. 2005; Szymanski and Gorecki 2010; Igarashi and Gorecki 2011; Bull et al. 2013). Such a medium could for example be accomodating the Belousov–Zhabotinsky (BZ) reaction (Zaikin and Zhabotinsky 1970; Noyes et al. 1972; Field et al. 1972; Gyorgyi et al. 1990). We refer to droplets as small amounts of excitable medium floating in oil that are covered with a layer of lipid molecules. The lipids stabilise the droplets against merging but still allow two adjacent droplets to communicate when the lipid molecules form a bilayer similar to that of biological cells (Aghdaei et al. 2008). Excitation waves can be transmitted through droplets but can also interfere with one another, dependent on their timing and on the chemical properties of the droplets and the medium within. Hence, droplets arranged in a network form a potential chemical computer (Gorecki et al. 2003; Szymanski et al. 2011; Adamatzky et al. 2011a, b; Holley et al. 2011). An experimental implementation of such a droplet system is shown in Fig. 1.

While it is clearly possible to rebuild basic and combined logical gates in excitable chemical media (Holley et al. 2011), this might not necessarily use the capabilities of unconventional computers to their greatest extent

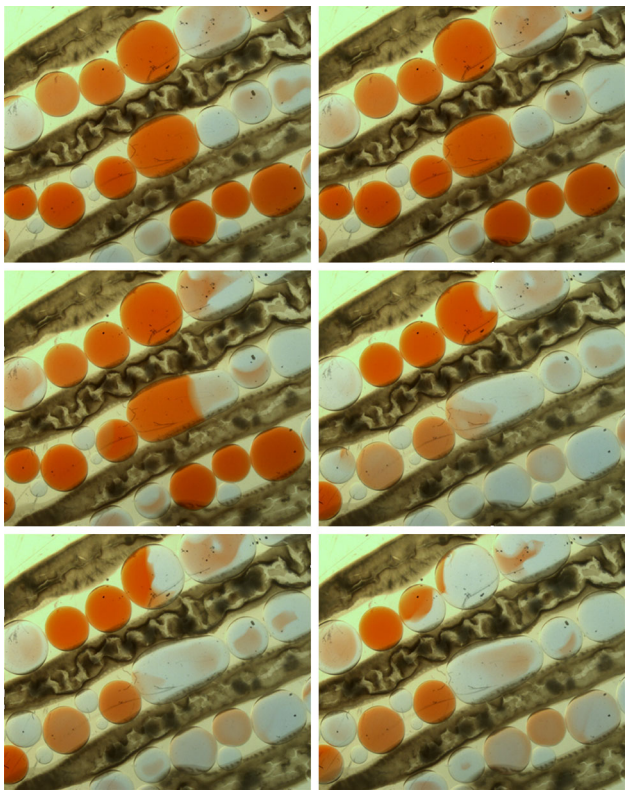


**Fig. 1** Experimental droplet system made from manually aligned droplets of a Belousov–Zhabotinsky medium, taken from Gruenert et al. (2011a, b). In the bright red state, the medium is resting and transparent/white excitation waves propagating between droplets can be observed. Even though droplets remained relatively stable, a merger between the central two droplets is shown in the third frame. (Color figure online)

(Gentili et al. 2012; Stepney 2012). Nonetheless, in this work, we will evolve droplet networks fulfilling basic logical functions because of their simplicity while exploring the impact of different input symbol encodings. So the focus is not mainly on the evolved functionality but rather on the varying difficulty of the evolution process.

In a droplet based computer, the spatiotemporal dynamics of the excitation waves determine the computation, therefore the topology of the coupled droplets plays a decisive role when "programming" such devices. Additionally we can also look at the symbol representation in order to discover an adequate and efficient interpretation for them. Here we refer to "programming" in the broadest sense of specifying the desired functionality of a computing device in contrast to the typically understood exact algorithmic specification of data manipulation. Examples for this unconventional sense of programming could be evolutionary algorithms, functional programming languages, amorphous computing, spatial computing, collision computing, chemical computing, membrane computing, natural computing, neural computing (Banâtre et al. 2005) and liquid state machines (Maass et al. 2002).

In this study, we consider evolutionary algorithms (Koza 1989; Weicker 2002; Eiben and Smith 2008) as a mechanism to infer adequate symbol representations when building logic gates with droplet networks. Given an optimisation problem, an evolutionary algorithm selects good individuals in a population of solutions that changes over time via genetic operators. Starting with a randomly generated population and guided by the fitness function, the evolutionary algorithm gives us after several generations an approximating solution to the problem. The use of evolutionary algorithms to design logic gates and circuits has been studied specially in the context of genetic programming (Koza 1989) and evolvable hardware (Miller et al. 2000). Also in the context of excitable media, 2D cellular automata have been evolved to fulfil binary logic fitness functions (Stone et al. 2008).

We are not aiming at building a single droplet network design that could act as a universal computer, solving any kind of computable problem. But it appears feasible and useful to build droplet devices that compute results for different instances of a problem. Therefore, given a problem instance, input data needs to be specified in some way. This could either happen through the initial state of the droplet system or during the runtime, most probably through external stimulation of certain droplets. In either way it is an important design decision which encoding is used to feed inputs into the droplet network. Most probably the optimal encoding will depend on a number of factors like the type of task, the number of used symbols, parameters of the computing substrate, the applied quality measure, and how much computation can be done outside the droplet network to generate the encoding.

Since we cannot influence the amplitude of the excitation spikes, a list of times at which we excite particular droplets should contain all the information that is available to the computing droplet system. Nonetheless, different features of this list of excitations might be of more or less importance. From the neurosciences we know for example the coding techniques *rate coding*, *population coding* and *temporal coding* (Brown et al. 2004). In the case of *rate coding*, the (averaged) oscillation frequency is used to distinguish different meanings while the exact timing of the spikes would be ignored. *Population coding* on the other hand would mean that the activity of different sub-populations of droplets denoted different meanings. For *temporal coding*, the precise timing differences between excitation spikes are utilised as information carrier. These coding schemes might be candidates for excitable droplets as well.

To find adequate symbol representations for droplet computers, we start by considering rate coding and evolution of droplet networks that fulfil a functionality, given by simple Boolean functions. Similar to evolutionary

algorithms or to genetic programming the evolved droplet network topology can be seen as the definition for a program that can be executed on the droplet computing architecture. Then we explore the co-evolution of the droplet network topologies with different symbol encoding options for two symbols and basic Boolean logic functions.

## 2 Methods

### 2.1 Self-exciting droplets

For the simulation studies, we are using a stochastic, discrete state, continuous time simulation model for self-exciting droplets that was described in (Gruenert et al. 2012b). It allows fast simulations while retaining the possibility for fine-tuning of the droplet timing parameters and noise levels. Coarsely, each droplet is simulated as being in one of three states: *excited*, *refractory* or *responsive*. After being *excited*, droplets always stay *refractory* for some time, then become *responsive* for some time and then can self-excite or be triggered into the next *excited* state again to begin a new oscillation. The lengths of the three phases are drawn from a truncated normal distribution in this case. Droplets can interact in the following sense: If a droplet in the *excited* state is adjacent to a *responsive* droplet, the *responsive* droplet is triggered into an excitation as well. This means, even though the droplet would self-excite anyway after some time, the next oscillation cycle is triggered to happen earlier. When a droplet is in the *excited* or *refractory* state, it is not influenced by its neighbours. Additionally, also drawn from a normal distribution, there is a signal propagation delay between an excitation and its influence on an adjacent *responsive* droplet.

We also use variations of the model for less excitable droplet types, such that at least two concurrent excitations are necessary to trigger a droplet into an earlier excitation. Furthermore, to allow for a richer dynamic behaviour, we decided to include one more droplet type that would oscillate slower. The different oscillation period can be achieved by differently composed BZ mixtures. In this case, all timing mean values as well as the standard deviations are multiplied by an arbitrary factor of $\frac{3}{2}$.

In this study we use the following parameters: normal droplets $d_{Norm}$ as well as input and output droplets are modelled with an expected oscillation period of 16 s, which is composed of 10 s responsive time $\tau_{res}$, 1 s excited time $\tau_{ex}$ and 5 s refractory time $\tau_{ref}$. Signal propagation delays $\tau_{prop}$ are 1 s. The exact timing parameters for each phase are sampled using normal distributions with a standard deviation of 0.05 s around the mean values given before. Less excitable droplets $d_{LowEx}$ use the same timing distributions but require at least two adjacent droplets to be excited at the same time to trigger an excitation.

### 2.2 Droplet Networks

We perform in silico experiments of droplet networks in a $10 \times 10$ grid of simulated droplets that are connected in a Moore neighbourhood of radius one, such that all directly adjacent cells can excite each other. These geometric properties of the networks were chosen based on the size of networks that can presumably be achieved experimentally by our collaboration partners in the near future. Up to four different kinds of cells are used, which represent empty cells, normal droplets, droplets of lower excitability and droplets with longer oscillation periods. Furthermore, there are two fixed input droplets and two fixed output droplets defined on the network grid. They can be used to dynamically feed a stream of excitations into and out of the droplet network. The positions of the input and output droplets are fixed to arbitrary values, coarsely in the middle of the left and right hand sides of the grid, as visualised in Fig. 2a.

We represent a specific droplet network instance as an $n$ by $n$ array:

$$N = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & \cdots & d_{n,n} \end{pmatrix}$$

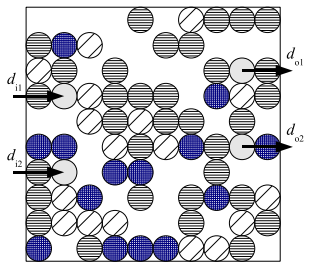
$$d_{i,j} \in \{\varnothing, d_{Norm}, d_{LowEx}, d_{Slow}, d_{In0}, d_{In1}, d_{Out0}, d_{Out1}\}$$

A Moore neighbourhood around each droplet $d_{i,j}$ defines the connectivity of the droplets, i.e. a droplet $d_{i,j}$ is connected to a present droplet $d_{k,l}$ if $|i - k| \leq 1$ or if $|j - l| \leq 1$.

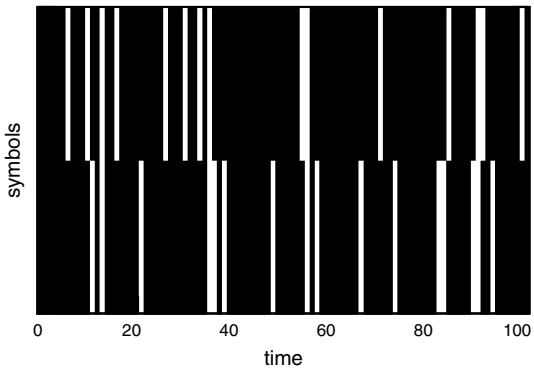
### 2.3 Signal encoding

When representing binary signals by rate coding, we stimulate droplets as much as possible for a symbol '1' and not at all for a symbol '0'. When droplets are maximally stimulated, the oscillation time will be $\tau_{ex} + \tau_{ref} = 6$ s. Normal droplets that are left alone do not stop oscillating but their frequencies are lower with periods of $\tau_{ex} + \tau_{ref} + \tau_{res} = 16$ s.

To allow more complex symbol representations, we use a timing pattern that determines which input droplet is stimulated from the outside at which times as visualized in Fig. 2b. We divide the length $T$ of the stimulation pattern up into $m$ small intervals $\{I_1 \ldots I_m\}$, each of the length $\triangle t = \frac{T}{m}$. Hence, interval $I_j$ is defined between the times $(j-1) \cdot \Delta t$ and $j \cdot \Delta t$. Considering a single droplet only, we




**(a)** Droplet Network




**(b)** Input Symbol Representation

**Fig. 2** Individuals for the Evolutionary Algorithm, taken from Gruenert et al. (2012a, b): **a** Rendering of an evolved 10 × 10 droplet network instance. Each *square* represents a droplet on a two dimensional array. Not all positions of the array are filled with droplets. *Horizontally*, *diagonally* and chequered *striped circles* represent normal, less excitable and long period droplets respectively. The input droplets ($d_{i1}$, $d_{i2}$) and output droplets ($d_{o1}$, $d_{o2}$) on arbitrarily fixed positions are indicated by arrows. Touching droplets can excite each other, defining the connectivity for the droplet simulation. **b** Example of two symbols that evolved together with a network instance to realise the XOR function. The *lower row* of the image represents symbol '0' while the *upper row* represents symbol '1'. Time advances left to right over 100 frames with a time step of 0.5 s, leading to a total length of 50 s per symbol. The input droplets are stimulated only in the intervals that are represented by *white vertical bars* and are left alone where the *black vertical bars* are rendered. The *symbols* are fed into the droplet network repeatedly, recapitulating the stimulation pattern every 50 s. At least three oscillation cycles are completed per symbol repetition because the simulated droplets' self-excitation periods are around 16 s. Since droplets are modelled with refractory times, not every white stimulation bar will actually lead to an excitation in the droplet but can as well be disregarded in the droplets refractory phases, especially when two excitations follow each other closely

define a pattern $S_{(1)}$ as a Boolean vector, which states if the droplet is stimulated in the interval $I_j$ or not. To describe meaningful symbols, $\Delta t$ should be small in comparison to a droplet's oscillation period, resulting in a fine temporal resolution. Meanwhile, the total length $T$ of the symbol should probably be long in comparison to the droplet's oscillation to allow symbols to consist of more than a single excitation.

$$S_{(1)} = (a_{I_1}, a_{I_2}, \ldots, a_{I_m}),\ a_i \in \{0,1\}, \quad S_{(1)} \in \{0,1\}^m,$$
$$m \cdot \Delta t = T$$

Because typically more than one input will be used, multiple droplets will have to be stimulated, e.g., both inputs for an XOR gate. Furthermore, thinking about population coding, a single symbol like a logical '1' could affect multiple droplets with individual stimulation patterns. In contrast, for the sake of redundancy, a common stimulation pattern might be supplied to multiple droplets. Here we use the notion of the *droplet channel* $c_i$ to signify a set of droplets that receives the same stimulation pattern $S_{(1)}^{c_i}$. Two droplet channels $c_i$ and $c_j$ could now either be used as components of a single symbol or as independent inputs. Nonetheless, in the experiments shown in this work, a symbol will only consist of a single droplet channel.

For stimulation patterns that are composed of many channels $C = \{c_1, c_2, \ldots, c_{|C|}\}$, we can extend the pattern definition $S_{(1)}$ to an array $S_{(|C|)}$ that stores the activation state $ac_i,I_j$ of each channel $c_i \in C$ for each interval $I_j$:

$$S_{(|C|)} = \begin{pmatrix} a_{c_1,I_1} & a_{c_1,I_2} & \cdots & a_{c_1,I_m} \\ a_{c_2,I_1} & a_{c_2,I_2} & \cdots & a_{c_2,I_m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{c_{|C|},I_1} & a_{c_{|C|},I_2} & \cdots & a_{c_{|C|},I_m} \end{pmatrix}$$

## 2.4 Task definition

To evaluate the quality of a droplet network and of different symbol encodings, we define Boolean functions that should be fulfilled in terms of their truth tables. As displayed in Table 1, we tested seven different functions with up to two input and output channels.

**Table 1** Boolean functions that were used as fitness criteria in evolution

| Expected output $\tilde{o}_{cp}$ | | | | |
|---|---|---|---|---|
| Task | Input | | | |
| | 0 0 | 0 1 | 1 0 | 1 1 |
| Identity | 0 0 | 0 1 | 1 0 | 1 1 |
| OR | 0 | 1 | 1 | 1 |
| AND | 0 | 0 | 0 | 1 |
| NAND | 1 | 1 | 1 | 0 |
| XOR | 0 | 1 | 1 | 0 |
| XNOR | 1 | 0 | 0 | 1 |
| Half-adder | 0 0 | 1 0 | 1 0 | 0 1 |

Two input and up to two output channels were used

### 2.5 Fitness evaluation

Ultimately, the aim of these experiments is to find symbols that can be used by the network internally as input as well as for output. But to evaluate the fitness of a droplet network for binary operations using arbitrary symbols, a metric that determines the similarity between an input symbol and a recorded output excitation stream would be necessary. As discussed in Sect. 1, choosing an appropriate metric is not trivial. Consequently, we are evolving complex symbol representations to feed into the network but we do not yet expect the network to reproduce these complicated symbols as outputs. Instead we use simple rate coding for the outputs: high activity is interpreted as symbol '1' and low activity as symbol '0'. Here again, as for the rate coding input, high activity means droplets are entering the next oscillation cycle very shortly after leaving the *refractory phase*, resulting in a high spike frequency. Low activity, in contrast, means that droplets are rarely triggered into early excitations by their neighbours and mostly self-excite, resulting in a low spike frequency.

The evaluation is divided into distinct phases $p$ by assigning each combination of input symbols to one phase, resulting in four phases for two binary inputs. Then, for each phase, we analyse the output *droplet channels*, i.e., the activity on the designated output droplets, for their similarity to an expected output: for each phase $p$, the system is simulated with the appropriate input signals for a fixed time and the number of received excitations at the droplets of output channel $c$ are stored in $o_{cp}$. We denote the maximal and minimal counted peak numbers as $o_{max}$ and $o_{min}$. The symbol that is expected at the output droplets for the channel-phase pair $(c, p)$ is referenced as $\tilde{o}_{cp} \in \{0, 1\}$ instead.

The final fitness $F$ is influenced by two different aspects, $F_1$ and $F_2$, of the output behaviour. First, the normed difference between highly activated and less activated channel-phase pairs should be maximised to allow some kind of discrimination. We define the difference between the maximum and minimum peak numbers divided by the maximum peak number as $F_1$. $F_1$ is zero if all peak numbers are equal and at most one when the minimum value is zero. Second, the truth table should be fulfilled, leading to a function $F_2$. Here, the worst case channel-phase pair defines the overall fitness. Each channel-phase pair peak number should lie as close as possible to the minimum or maximum peak number, dependent on the expected output $\tilde{o}_{cp}$. Finally, if a minimum discriminability is exceeded and also the Boolean function is fulfilled, the distance between minimum and maximum rates should further be expanded.

$$F = \begin{cases} F_1 & \text{if} \quad F_1 < 0.2 \\ F_2 + 1.0 & \text{if} \quad F_1 \geq 0.2 \quad \text{and} \quad F_2 < 0.9 \\ F_1 + 2.0 & \text{if} \quad F_1 \geq 0.2 \quad \text{and} \quad F_2 \geq 0.9 \end{cases} \tag{1}$$

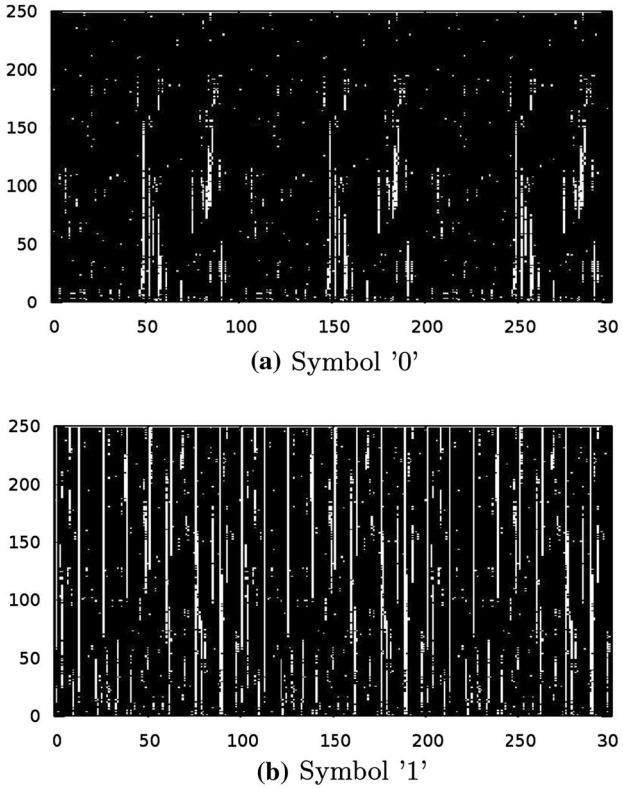



**Fig. 3** Evolutionary trajectory of two symbol representations over 250 generations co-evolution with a droplet network (Gruenert et al. 2012a, b). The y-axis denotes the evolutionary generation while the x-axis represents the stimulation interval for each fitness evaluation similar to the signal plot in Fig. 2b. The regularities that can be observed along the x-axis in both graphics are not evolved regularities but result from the repetition of the pattern: As the pattern of 100 intervals is fed into the simulator during fitness evaluation in a repeated manner, three repetitions of the input signal are plotted over 300 time frames

$$F_1 = \frac{o_{max} - o_{min}}{o_{max}} \tag{2}$$

$$F_2 = \min_{c,p} \begin{cases} 1 - \frac{o_{cp} - o_{min}}{o_{max} - o_{min}} & \text{if} \quad \tilde{o}_{cp} = 0 \\ \frac{o_{cp} - o_{min}}{o_{max} - o_{min}} & \text{if} \quad \tilde{o}_{cp} = 1 \end{cases} \tag{3}$$

### 2.6 Experimental set-up

We employed an evolution strategy of the type (8/2,30)−*ES*, meaning a comma strategy with 8 parents and 30 children, running for 250 generations where the parents of each generation are discarded. Two parents are recombined to produce each child. The best symbol representation of each generation of a single experiment is displayed in Fig. 3. For each experiment, we ran a batch of 50 evolutionary optimisations to build mean values. In total, we conducted 35 experiments for all the combinations of the seven target functions from Table 1 and the five experimental variations: Network only evolution with three or four

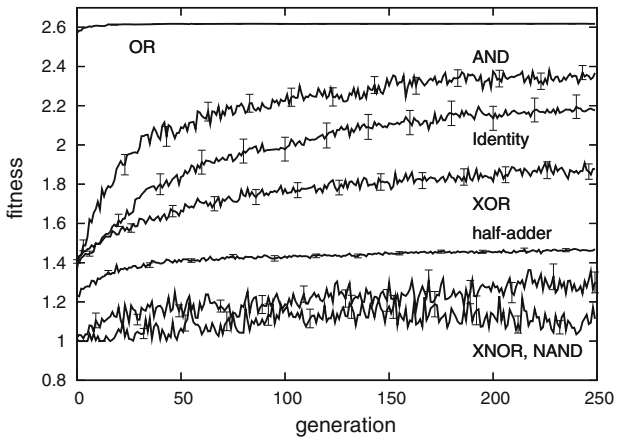
droplet types, network and signal co-evolution with three or four droplet types and network only evolution with pre-evolved symbol representations. The symbol representation for the pre-evolved signals was taken from the co-evolution experiment that achieved the best fitness. Using four droplet types means using empty droplets, normal droplets, less excitable droplets and long period droplets, while the latter is discarded for the three droplet type experiments.

For mutating the droplet network, the probability of switching an arbitrary position is 0.05. When using four droplet types, the probabilities for changing to an empty cell, to a normal droplet, to a low-excitability droplet and to a long-period droplet are 0.4, 0.4, 0.1 and 0.1 respectively. For the runs without the long period droplet type, the remaining probabilities read $\frac{4}{9}, \frac{4}{9}$ and $\frac{1}{9}$. Single point crossover recombination is applied with an uniformly chosen position in the row-by-row linearised representation of the droplet network. For the input signal, the probability of switching an arbitrary position is 0.025. When a mutation occurs, the probability for generating a '1' is 0.1 while a '0' is generated with probability 0.9. Single point crossover recombination is applied with an uniformly chosen position.

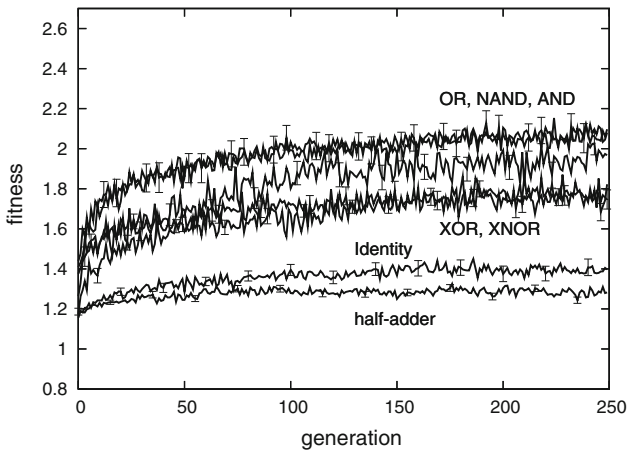
## 3 Results

Small droplet systems of up to 100 droplets were arranged by means of evolutionary algorithms to satisfy the Boolean functions Identity, OR, AND, NAND, XOR, XNOR and half-adder. Based on differentially fast fitness increase, some target functions are easier to evolve than others (cf. Fig. 4). As observed in (Adamatzky and Bull 2009), the reason for this is partially the different problem complexity and partially the properties of the computing substrate that favor and disfavor certain kinds of tasks. In the case of our droplet computing, using rate coding only, the OR and AND functions evolve fastest, followed by Identity, XOR, the half-adder, the XNOR, and the NAND function. Nonetheless, there is a strong qualitative transition between the XOR and the half-adder function. The mediocre fitness of the XOR network is based on the some few evolution runs that produced high fitness XOR networks and many non-functional ones. The XNOR and NAND evolutions using rate coding as well as the half-adder with any coding on the other side did not lead to a single evolution run producing a functional network.

Despite these difficulties, even a complicated function like XOR was evolved, even for single channel rate coding signal inputs, albeit not as fast as a simple OR or AND function (cf. Fig. 5). Interestingly, the identity function,




**(a)** Network only Evolution

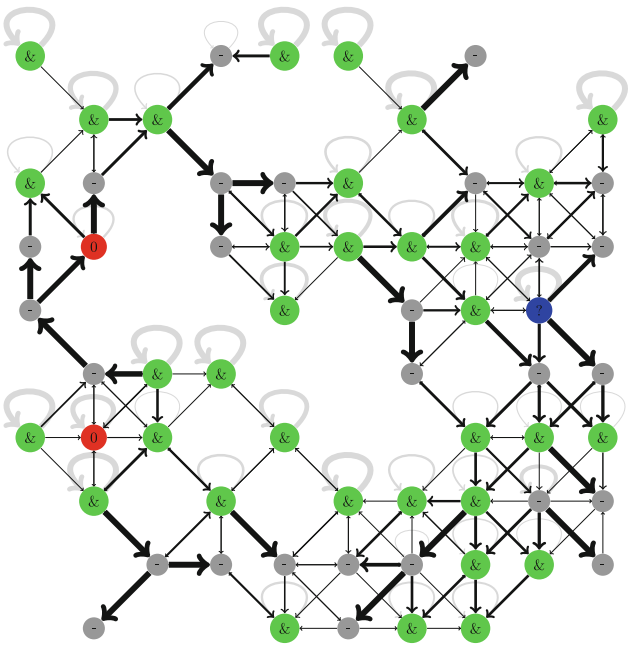



**(b)** Network and Signal Coevolution

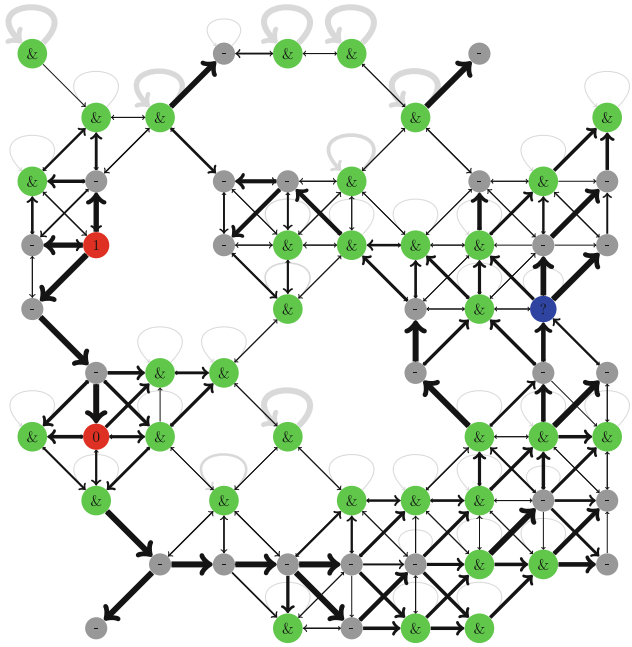
**Fig. 4** Average fitness of population's best individual over 50 experiments for evolving different target functions from Table 1, taken from Gruenert et al. (2012a, b). *Error bars* indicate the standard error of the mean. Generally, all fitness values are lower for the signal and network co-evolution because of the higher dimensional search space. Exceptions are those functions that benefit from a simple swapping of rate coding signals, i.e. the NAND and XNOR functions

meaning a mere connection between both inputs and outputs, is not a simple task compared to AND or XOR when co-evolving input signals (cf. Fig. 4). Apparently co-evolving networks and symbol representations for the identity function is almost as hard as evolving the half-adder. While using rate coding, in contrast, the identity function evolved faster than the XOR function. Evolution with and without the third droplet type with long oscillation periods did not significantly change the speed or final quality of the evolution process.

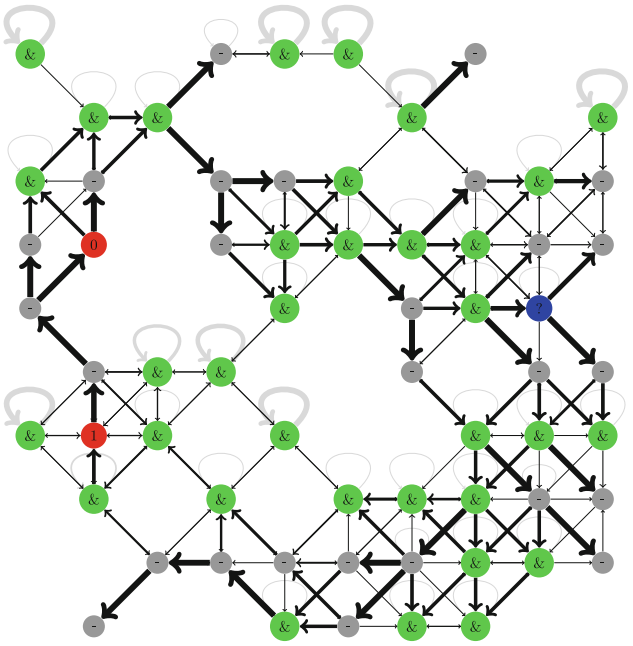
A network successfully implementing the half-adder functionality did not evolve in our experiments so far. The reason for this is most probably the difficulty of crossing over two connections in the two dimensional lattice of



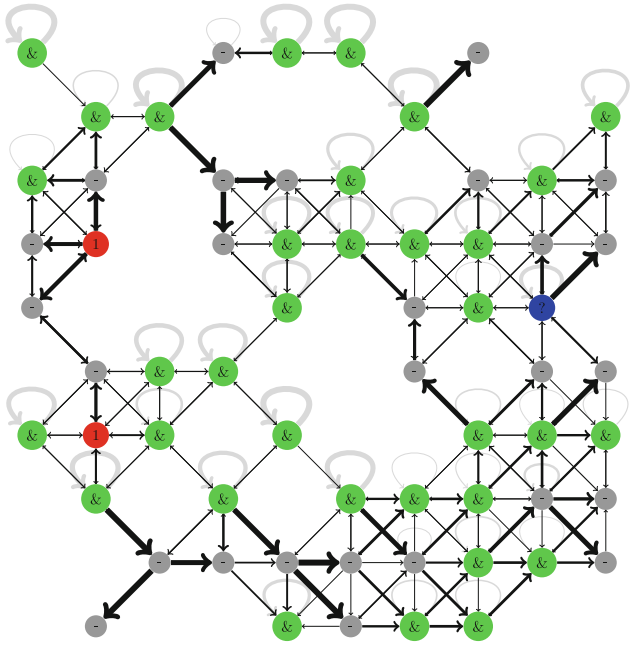
**(a)** Input 0,0



**(b)** Input 0,1



**(c)** Input 1,0



**(d)** Input 1,1

**Fig. 5** Activity plot of an evolved XOR network using different stimulations at the red input droplets on the left side of the network. The nodes in the network represent input droplets (*red*), output droplets (*blue*), normal droplets (*gray*) and less excitable droplets (*green*). The *arrows* in the picture describe the impact of other droplets for the excitations of each droplet. While a *strong loop arrow* on *top* of a droplet means that the droplet mostly self-excited, an *arrow* from a neighbouring droplet means that it was excited by this neighbour many times. While an input of (0,1) or (1,0) leads to a cyclic propagation of pulses through the network in either clockwise or counter-clockwise direction, no excitation at all (0,0) or full stimulation (1,1) results in pulses that propagate from the left to the right. The cyclic propagation results in a higher total spike rate arriving at the *blue* output droplet as thinner self-excitation loop on top of the output droplet can be observed. Hence, the droplet network fulfils the function of an XOR gate when used with rate coding inputs. (Color figure online)

droplets. A half-adder network could be implemented by an XOR gate together with an AND gate, each of which evolved comparatively easy. But to construct the half-adder from the two gates, both inputs would have to be available to each of the gates. Thus at least one of the input signals would need to cross over another one. While we do not exclude the possibility of a signal crossing over another one given a suitable construction of a droplet system and a fitting symbol encoding, we did not observe such a system in our experiments. At least when trying to evolve a rate coding identity function with two inputs and two outputs while crossing over the outputs, the fitness dropped dramatically, such that no satisfying solution has evolved.

Shown in Fig. 6, at least in the case of the AND and XNOR functions, pre-evolved signals exist (cf. Fig. 7) that are clearly leading to a faster evolution of droplet networks than simple rate coding. Here droplet networks and signals were originally co-evolved. Then, one of the best evolved symbol representations was used consistently through a full network-only evolution run.

The evolved symbols look similar (cf. Fig. 7a) to rate coding signals but most probably allow for a better synchronization of arriving spikes. While a single activation peak remained for the '0' symbol, it had no obvious influence on the fitness of the symbol. The synchronization of spikes seems to be important considering that a low excitability droplet is only activated by other droplets, when two spikes arrive in a narrow time window. The length of the window used in our experiments was one second. So while a constant activation, using rate coding symbols, leads to the highest frequency of spikes in the input droplets, the phase of both input droplet oscillations can randomly drift and is dependent on the initial conditions. When using a slightly lower activation rate instead, the phases of both input droplets are controlled by the stimulation, leading to a higher number of concurrent spikes arriving at low excitability droplets. This, in turn, leads to a higher influence of the low excitability droplets on further droplets in the network. We tested an evolved droplet network and stimulated it with a rate coding symbol, a co-evolved symbol and with an additionally engineered symbol. The engineered symbol includes no stimulated at all for symbol '0' and regular spike every seven seconds for symbol '1'. With this spike pattern, the engineered symbol reaches very similar input and output average spike rates compared to the evolved symbol. The measured spike frequencies are summarized in Table 2.

A further extreme rise in evolution efficiency was observed for the NAND function. However, this is most probably only due to a crosswise substitution of the signals

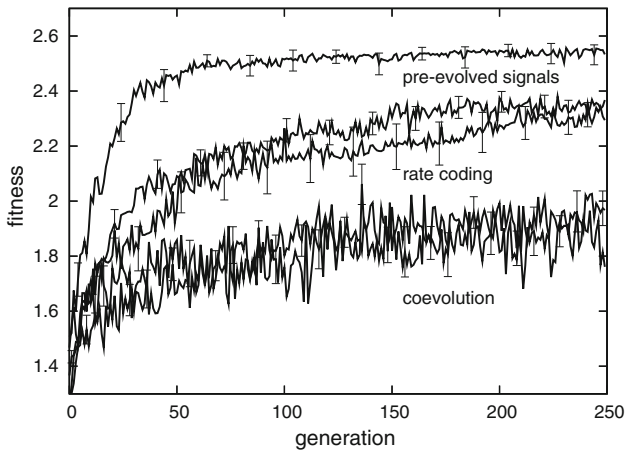



**Fig. 6** Average fitness of populations' best individual over 50 experiments for evolving the AND function using rate coding, co-evolution and pre-evolved symbol representation Gruenert et al. (2012a). *Error bars* indicate the standard error of the mean. For the rate coding and co-evolution experiments, two curves are plotted: The corresponding simulations ran with and without the long period droplet types, but no significant difference was observed

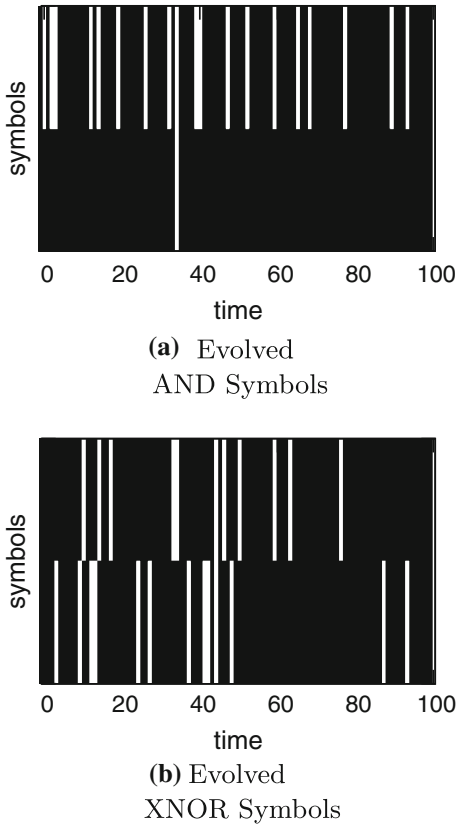



**Fig. 7** Evolved symbol representations for the AND and the XNOR functions that performed better than rate coding, taken from Gruenert et al. (2012a). **a** While the AND symbol looks very similar to rate coding symbols, there is one peak included for symbol '0' that might serve as a helper for synchronisation. **b** For the XNOR signals, both symbols are represented by a series of about 30 s activation followed by ca. 20 s rest. The difference between both symbol representations could be either in the shift of the active phases of about 10–20 s or in the exact pattern of each signal

**Table 2** Summary of the spike frequencies measured in an evolved AND network when stimulated with rate coding, evolved and an engineered symbol input

| | Average spike frequency (spikes/second) | | |
|---|---|---|---|
| | Rate coding | Evolved symbol | Engineered symbol |
| Symbol '0' input droplet | 0.067 | 0.067 | 0.067 |
| Symbol '1' input droplet | 0.167 | 0.140 | 0.143 |
| Supposedly active output droplet | 0.086 | 0.139 | 0.138 |
| Supposedly inactive output droplet | 0.064–0.067 | 0.064 | 0.064–0.066 |

The engineered symbol did not produce any stimulation for symbol '0' and a regular spike pattern of a spike every seven seconds for symbol '1'

for symbols '0' and '1', such that the problem is reduced to a rate coded OR function. Functions that involve a mapping from symbol '0' to low activity like the XNOR, and NAND functions seemed more difficult with pure rate coding. This problem of inverting signals should easily be resolved when using multi-channel symbol representations that would supply a high and a low activity channel for each symbol. Problems that did not benefit significantly from pre-evolved symbol representations were the OR, the XOR, the Identity function and the half-adder. Nonetheless, the pre-evolved symbols never led to worse evolution trajectories in our experiments.

## 4 Discussion and future work

Besides designing droplet network structures and symbol encodings, evolutionary algorithms also served another purpose in this work: To some extent, evolutionary algorithms also offer a measure of complexity, telling us whether a problem is simple or hard to solve (Adamatzky and Bull 2009). Or, given two distinct symbol encodings, which of them makes searching for a solving network structure easier.

A straight forward construction of two adequate symbols, representing '0' and '1', might be to maximise the distance between them. The problem here is to define the distance metric that would heavily influence the result of the maximisation. Ideally these experiments would only depend on the properties of the computing substrate itself and not on arbitrary definitions that are put in from the outside. But any kind of metric like the Hamming distance or the spike train similarity measures from the neurosciences (Dauwels et al. 2008) seem sensible but artificial with respect to the computing droplet substrate. A meaningful alternative would be to run a nested evolution of a droplet network simulation as distance metric—the easier it is to evolve a network that discriminates both signals, the larger the distance between both symbols. Still, the computational efforts for a single evaluation of the fitness function appear immense. This led us to the different approach of co-evolving signals and droplet networks for simple binary problems at first.

Even though simple logic functions were evolved here, the automatic construction of larger, more complex systems might be hard, especially when fitness functions cannot provide enough gradient for the optimisation algorithm to follow. The "multi-step" fitness functions that we used in Eq. 1 tries to focus different aspects of generating the network functions at different times, dependent on how close to perfect the solution is. But since it is generally impossible to find all non-dominated solution candidates by mapping multiple fitness criteria onto a single scalar value, we will move on to using Pareto optimisation for future experiments (Schaffer 1985; Zitzler et al. 2004).

Generally the influence of the droplet network dimensions should be interesting—especially how few droplets can generate the sought-after behaviour, what number of droplet species are essential, is there a preferential length for droplet signal patterns and how many input channels should be used per symbol? For this purpose, we will also consider population coding (Pouget et al. 2000; Averbeck et al. 2006) in forthcoming experiments. Also the aspect of robustness has not yet been in the focus of this work. Nonetheless it appears important if a droplet network and symbol representation led to a high score accidentally or if the performance can be sustained under different initial conditions and with noise.

**Acknowledgments** The research was supported by the NEUNEU Project (248992) sponsored by the European Community within FP7-ICT-2009-4 ICT-4-8.3—FET Proactive 3: Bio-chemistry-based Information Technology (CHEM-IT) Program.

## References

Adamatzky A (2001) Computing in nonlinear media and automata collectives. IOP, Bristol

Adamatzky A, Bull L (2009) Are complex systems hard to evolve?. Complexity 14(6):15–20

Adamatzky A, Costello BDL, Asai T (2005) Reaction-diffusion computers. Elsevier Science, Amsterdam

Adamatzky A, Holley J, Bull L, DeLacy Costello B (2011a) On computing in fine-grained compartmentalised Belousov–Zhabotinsky medium. Chaos Solitons Fractals 44:779–790

Adamatzky A, de Lacy Costello B, Holley J, Gorecki J, Bull L (2011b) Vesicle computers: approximating a voronoi diagram using voronoi automata. Chaos Solitons Fractals 44:480–489

Aghdaei S, Sandison M, Zagnoni M, Green N, Morgan H (2008) Formation of artificial lipid bilayers using droplet dielectrophoresis. Lab Chip 8(10):1617–1620

Averbeck B, Latham P, Pouget A (2006) Neural correlations, population coding and computation. Nat Rev Neurosci 7(5):358–366

Banâtre JP, Fradet P, Giavitto JL, Michel O (2005) Unconventional programming paradigms. International workshop UPP 2004, Le Mont Saint Michel, France, September 15–17, 2004. Revised selected and invited papers. Lecture Notes in Computer Science, vol 3566. Springer, Berlin

Brown EN, Kass RE, Mitra PP (2004) Multiple neural spike train data analysis: state-of-the-art and future challenges. Nat Neurosci 7(5):456–461

Bull L, Holley J, Costello BDL, Adamatzky A (2013) Toward turing's a-type unorganised machines in an unconventional substrate: a dynamic representation in compartmentalised excitable chemical media. In: Dodig-Crnkovic G, Giovagnoli R (eds) Computing nature. Springer, Heidelberg

Dauwels J, Vialatte F, Weber T, Cichocki A (2008) On similarity measures for spike trains. In: Proceedings of the 15th international conference on advances in neuro-information processing, vol 1. Springer, pp 177–185

Eiben A, Smith J (2008) Introduction to evolutionary computing. Springer, Berlin

Field R, Körös E, Noyes R (1972) Oscillations in chemical systems II. Thorough analysis of temporal oscillation in the bromate–cerium–malonic acid system. J Am Chem Soc 94(25):8649–8664

Gentili PL, Horvath V, Vanag VK, Epstein IR (2012) Belousov–Zhabotinsky "chemical neuron" as a binary and fuzzy logic processor. Int J Unconv Comput 8(2):177–192

Gorecki J, Yoshikawa K, Igarashi Y (2003) On chemical reactors that can count. J Phys Chem A 107(10):1664–1669

Gruenert G, Dittrich P, Zauner KP (2011) Artificial wet neuronal networks from compartmentalised excitable chemical media. ERCIM News 85(85):30–32

Gruenert G, Escuela G, Dittrich P (2012a) Symbol representations in evolving droplet computers. In: Durand-Lose J, Jonoska N (eds) Unconventional computation and natural computation. 11th international conference, UCNC 2012, Orléan, France, September 3–7, 2012. Lecture Notes in Computer Science, vol. 7445. Springer, Berlin, pp 130–140

Gruenert G, Szymanski J, Holley J, Escuela G, Diem A, Ibrahim B, Adamatzky A, Gorecki J, Dittrich P (2012b) Multi-scale modelling of computers made from excitable chemical droplets. Int J Unconv Comput

Gyorgyi L, Turàànyi T, Field R (1990) Mechanistic details of the oscillatory Belousov–Zhabotinskii reaction. J Phys Chem 94(18): 7162–7170

Holley J, Jahan I, Costello B, Bull L, Adamatzky A (2011) Logical and arithmetic circuits in Belousov–Zhabotinsky encapsulated discs. Phys Rev E 84(5):056110

Igarashi Y, Gorecki J (2011) Chemical diodes built with controlled excitable media. IJUC 7(3):141–158

Koza JR (1989) Hierarchical genetic algorithms operating on populations of computer programs. In: Sridharan NS (eds). Proceedings of the eleventh international joint conference on artificial intelligence IJCAI-89, Morgan Kaufmann, Detroit, MI, USA, vol 1. pp 768–774

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

Miller J, Job D, Vassilev V (2000) Principles in the evolutionary design of digital circuits—part I. Genet Program Evolv Mach 1(1):7–35

Noyes R, Field R, Koros E (1972) Oscillations in chemical systems. I. Detailed mechanism in a system showing temporal oscillations. J Am Chem Soc 94(4):1394–1395

Pouget A, Dayan P, Zemel R et al (2000) Information processing with population codes. Nat Rev Neurosci 1(2):125–132

Schaffer J (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st international conference on genetic algorithms, L. Erlbaum Associates Inc., Hillsdale. pp 93–100

Stepney S (2012) Programming unconventional computers: dynamics, development, self-reference. Entropy 14(10):1939–1952

Stone C, Toth R, de Lacy Costello B, Bull L, Adamatzky A (2008) Coevolving cellular automata with memory for chemical computing: Boolean logic gates in the BZ reaction. In: 10th international conference on parallel problem solving from nature (PPSN). pp 579–588

Szymanski J, Gorecki J (2010) Chemical pulses propagating inside a narrowing channel and their possible computational applications. IJUC 6(6):461–471

Szymanski J, Gorecka JN, Igarashi Y, Gizynski K, Gorecki J, Zauner KP, Planque MD (2011) Droplets with information processing ability. Int J Unconv Comput 7(3):185–200

Weicker K (2002) Evolutionäre Algorithmen. Vieweg, Teubner, Stuttgart

Zaikin AN, Zhabotinsky AM (1970) Concentration wave propagation in two-dimensional liquid-phase self-oscillating system. Nat Biotechnol 225(5232):535–537

Zitzler E, Laumanns M, Bleuler S (2004) A tutorial on evolutionary multiobjective optimization. Metaheuristics for multiobjective optimisation pp 3–37