Evolving Control Metabolisms for a Robot

Jens Ziegler
Wolfgang Banzhaf
Department of Computer Science
University of Dortmund
44221 Dortmund, Germany
{ziegler,banzhaf}
@ls11.cs.uni-dortmund.de

Abstract This article demonstrates a new method of programming artificial chemistries. It uses the emerging capabilities of the system's dynamics for information-processing purposes. By evolution of metabolisms that act as control programs for a small robot one achieves the adaptation of the internal metabolic pathways as well as the selection of the most relevant available exteroceptors. The underlying artificial chemistry evolves efficient information-processing pathways with most benefit for the desired task, robot navigation. The results show certain relations to such biological systems as motile bacteria.

Keywords
artificial chemistry, chemical information processing, autonomous robots, genetic programming

1 Introduction

Artificial life (ALife) first intertwined different research areas into one interdisciplinary attempt to extract the first principles of life. The paradigm of complex systems, which includes living organisms, favors the idea of emergence to describe global properties that result from interactions between subsystems or components. These interactions may follow some very simple locally effective rules causing global behavior of the system not explainable by dividing the whole into parts and investigating the subsystems and components. The system has certain properties not because of the properties of its constituents but because of their *organization* and their *function* in the whole.

The theory of evolution introduced adaptation as a crucial part of sustained existence of complex systems. Abstracting from natural molecular processes, *artificial chemistry* (AC) reproduces the basic properties of such evolving systems and tries to analyze (either with computer simulations or analytically) the dynamics and artificial evolution of complex systems. The idea of evolution has become a key concept in the field of *evolutionary algorithms* [7, 29]. AC systems are part of the area of *emergent computation* [19] and can additionally act as a model of a decentralized and lean control architecture.

In this article, AC deals with information-processing systems consisting of an artificial chemistry as the main processing entity. Artificial chemical computing has been the topic of several contributions in recent years. Hjemlfelt et al. [21], Okamoto et al. [30], and Mills et al. [28], for example, investigated the possibilities of realizing artificial neural networks or Turing machines in vitro; Arkin [5] discovered computational functions in biochemical reaction networks; Adleman realized a DNA-based solution of a combinatorical problem [3]; and Adamatzki et al. [2, 32, 33] proposed the use of oscillating reactions like the Belousov–Zhabotinsky reaction for mobile robot control. Artificial chemical computing has been used for control tasks, especially for mobile robots. Husbands [22] used an AC to get better results with an artificial neural network for robot control. In his model artificial neurons are able to emit substances that diffuse and modulate transmission functions of other neurons. Brooks [14] used an artificial hormone system to model an asynchronous information flow in a distributed parallel control architecture for a humanoid robot. Adamatzky et al. and Ziegler et al. [1, 2, 40] used either an excitable lattice or simplified enzyme-substrate kinetics to control mobile robots. Lugowski, Shackleton et al., and Aoki et al. [4, 25, 26, 35], among others, intended to establish a new view on parallel distributed computer architectures, completely different from the well-known and common von Neumann architecture. Conrad et al. [15, 16, 38, 39] started to implement and analyze a simulator for pattern-recognition-based biochemically motivated complex reaction networks. Astor [6] used AC to evolve artificial neural networks, and Banzhaf [10] used the processing capabilities to implement a "molecular" solution to the traveling salesman problem. Due to the complexity of AC, all approaches used more or less manually designed instances of an AC to arrive at the desired results. Up to now the problem of how to generate ACs with desired properties automatically has remained unsolved (though some discussion can be found in [8, 9, 24]). In the next section follows an introduction into the terminology of ACs, along with a description of their information-processing capabilities. Sections 3 and 4 introduce reaction graphs as a representation of metabolisms and as the basis of the genetic programming system. Section 5 gives the results of experiments with either simulated or real robots. The remainder of the article is devoted to an analysis of the evolved reaction networks and includes a comparison to certain biochemical features of chemotactic bacteria.

2 Artificial Chemistry as a Means of Information Processing

An AC can be described as a tuple $\{S, R, A\}$, where S is a set of all possible symbols that each symbolize a certain molecular species, R is a set of collision rules representing the interaction among the molecules, and A is an algorithm describing the reaction vessel or domain and how the rules are applied to the molecules inside the vessel. The set P (the *population*) is a multi-set (a set that can contain more than one object of a certain type) $P = \{s_1, \ldots, s_M\}$, $s_i \in S \; \forall i$, consisting of M (the population size) molecules. To allow reactions to take place, a reaction vessel is needed, the *reactor*. For a detailed introduction to ACs see [17]. The reactor used in the following experiments has a spatial structure, that is, it is an $n \times m$ grid, folded as a torus. This simple reactor can be seen as a protocell-like structure without any cytoskeleton. The volume of the reactor is $n \cdot m$, the maximum number of molecules it can contain. What kind of reactions occur is determined by the rules $r_i \in R,\ i = 1, \ldots, N$. Each r_i can be written as

$$r_i\colon\ S_i \rightarrow S_i' \tag{1}$$

with

$$S_i = \{s_j \parallel s_j \in S, j = 1, \ldots, N\} \tag{2}$$

and

$$S_i' = \{s_j \parallel s_j \in S, j = 1, \ldots, M\}. \tag{3}$$

```
while ¬$terminate()$ do
    Draw $s$ out of $P$;
    if $\exists S \in$ Neighborhood of $s$ then
        $\forall s_i \in S \subset P$ do
            Draw $s_i$ out of $P$;
        od
        if $\exists$ a rule $r_j \in R$ with $S = S_j$ then
            $S'$ = Set of products according to reaction $r_j$;
        fi
        $\forall s'_i \in S'$ do
            insert $s'_i$ in $P$;
        od
    fi
od
```

Figure 1. Pseudo code of reactor algorithm *A*. This algorithm does not include inflow and outflow of molecules as required for computations (see Section 3 for details).

In our experiments, S_i and S'_i are multi-sets of size 2 (maximum) [$\max(N, M) = 2$], so all possible reactions are of the following four types:

$$s_1 \longrightarrow s'_1 \tag{4}$$

$$s_1 + s_2 \longrightarrow s'_1 \tag{5}$$

$$s_1 \longrightarrow s'_1 + s'_2 \tag{6}$$

$$s_1 + s_2 \longrightarrow s'_1 + s'_2 \tag{7}$$

The algorithm A that creates the dynamics of the system can be written in pseudo code as in Figure 1. The molecules have a *Moore* neighborhood (all eight neighbors on a square lattice). A molecule s is involved in a reaction if there is a multi-set S in the neighborhood that is a right-hand side of a rule $r \in R$ and if $s \in S$. The molecules in S are removed from the reactor and all molecules in S' are put into the reactor. Note that the neighborhood may contain more than one complete set. The diffusion of a molecule is a random move to one of the empty fields in the neighborhood, similar to Brownian motion (see Figure 2). The velocity of the diffusion can be adjusted by an additional parameter that gives the radius of the neighborhood. The Moore neighborhood has a radius $r = 1$ and a size of $s = 8$, that is, only next neighbors are taken into account. Every reaction can be influenced additionally through catalytic or inhibitive effects of other molecules. This will impact the rate constant of the reaction by an additional catalytic/inhibitive factor. To keep the reactor size limited, a dilution flux $\Phi(t)$ removes randomly chosen molecules.

The artificial chemical computing idea uses the following metaphor [11]: The data to be processed (in- and output) is represented by molecules and the processing of data is represented by the molecule–molecule interactions, well known as reactions. To solve a certain computational problem, one needs to model the input (the problem description) and the output (the solution). Therefore, some of the participating molecules have a special meaning. The concentration of input substances mirrors the parameters of the

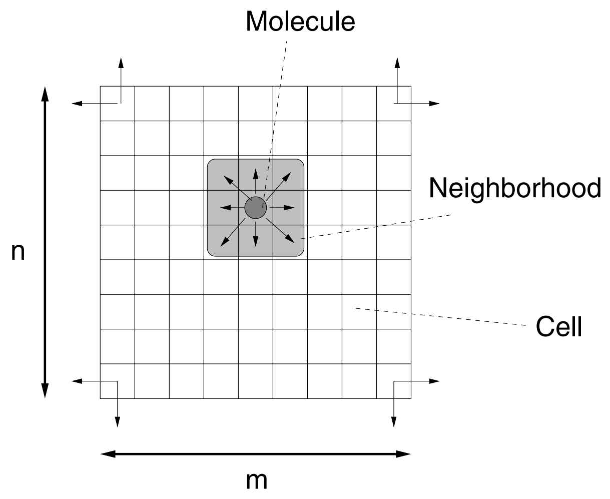


Figure 2. Neighborhood and diffusion of molecules.

problem. Output substances change in concentration according to changes of input flow and due to internal metabolic reactions. To achieve the desired output with a given input one has to set up the right internal reactions, a question that the following sections will discuss.

3 Artificial Reaction Graphs

The most natural way of visualizing reaction mechanisms is with graphs. A chemical reaction such as

$$Ca + H_2SO_4 \longrightarrow CaSO_4 + H_2, \tag{8}$$

where calcium and sulfuric acid react to calcium sulfate and hydrogen, can easily be represented with a special graph as in Figure 3. Artificial reaction graphs are a special case of general graphs, so-called *weighted directed bipartite graphs* [37]. A reaction graph is *directed* because it visualizes transformations. A transformation might be reversible. If it is reversible, it can be displayed with two counter-directed reactions. The graph is called *bipartite* because there are two different types of nodes: those representing molecules and those representing reactions. The graph that has a weight, that is, a numeric value associated with each edge, is a *weighted* graph, which indicates the catalytic or inhibitive activity of a certain substance with respect to a special reaction.

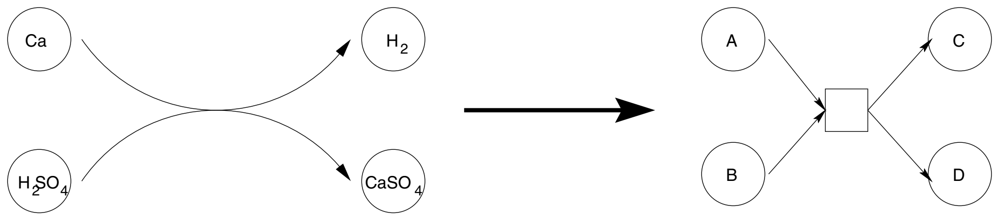


Figure 3. Encoding reaction (8) using a directed bipartite graph.

An artificial reaction network thus has the following properties, written in formal, graph-theoretical notation [20]:

(i) It is a directed graph.

$$G = (V, E) \tag{9}$$

(ii) The set of nodes V is the union of two sets, first the set of molecules S and, second, the set of reactions R.

$$V = R \cup S, \qquad R = \{v_{r_1}, \ldots, v_{r_n}\}, \qquad S = \{v_{s_1}, \ldots, v_{s_m}\} \tag{10}$$

Every node, either reaction or molecule, is assigned a unique name, in this case a number. A node will never be renamed.

(iii) $E \longrightarrow V \times V \times R$ is the set of edges. An edge connects exactly two nodes v and w, so it is a pair of nodes, combined with a numerical value, representing the speed of the reaction.

$$E = \{e_1, \ldots, e_l\}, \qquad e_i = (v_i, w_i, k_i), \qquad \forall\, i \in 1, \ldots, l \tag{11}$$

v_i and w_i must be of different types:

$$v_i \in R \Rightarrow w_i \in S \quad \text{or} \tag{12}$$

$$v_i \in S \Rightarrow w_i \in R. \tag{13}$$

Edges of this type always have a weight k_{default}, which means that this reaction is spontaneous.

(iv) There is a special kind of edge—the edge indicating a catalytic or inhibitive effect of a molecule on a reaction. This edge is always directed

$$v_i \in S \longrightarrow w_i \in R. \tag{14}$$

Edges of this type have weights, from a range $k_i \in [k_{\min}, k_{\max}]$.

(v) A metabolic pathway is a subgraph $P(v, w) = (V', E')$ with $v, w \in S$; $v, w \in V'$; $V' \subseteq V$ and $E' \subseteq E$, so that $P \subseteq G$. P contains only nodes that are encountered during a traversal from v to w on the shortest path in G. If there is no path, P is empty. Additionally, for every reaction node r of the shortest path the following condition must hold:

$$\forall\, s \in S(r)\colon\ (s, r, k_r) \in E' \wedge s \in P \quad \text{and}$$
$$\forall\, s' \in S'(r)\colon\ (r, s', k_r) \in E' \wedge s' \in P, \tag{15}$$

that is, every participant and every product of a reaction node that is part of the pathway must itself be part of the pathway. The length of a pathway is defined as the number of nodes. Together, the metabolic pathways of every node of P form a *metabolic network*, which in turn is a method for visualizing biochemical systems. The complexity of—at first glance—simple biochemical reactions is shown in [27].

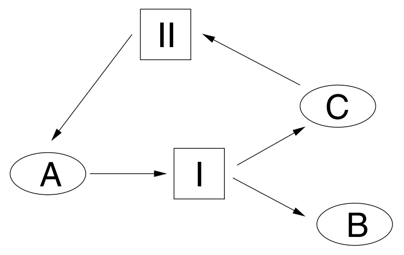


Figure 4. A chemically impossible reaction mechanism.

(vi) The set $I(G)$ is called the *input set* of G, $I(G) \subseteq V$. Every molecule node in I represents a *connected* substance whenever the metabolism is simulated. These substances are connected with external fluxes that, together, represent the state of the environment of the metabolism.

(vii) The set $O(G)$ is the *output set* of G, $O(G) \subseteq V$, $I(G) \cap O(G) = \emptyset$. The nodes in O represent the response of the metabolism to a certain internal state and an external situation.

(viii) The set $F(G) \subseteq V$, the functional set, or *kernel*, of G contains only molecule nodes that fulfill the following conditions:

$$\begin{aligned} &1)\ \forall\, w \in F(G)\colon\ \exists\, e = (v, w) \in \bar{G} \parallel v \in I(G), \\ &2)\ \forall\, w \in F(G)\colon\ \exists\, P(v', w) = (V', E') \parallel v' \in I(G) \wedge \\ &\qquad\qquad \forall\, v \in V'\colon\ \exists\, e = (v'', v) \in \bar{G} \parallel v'' \in I(G). \end{aligned} \tag{16}$$

$\bar{G} = (V, \bar{E})$ is the *transitive closure* of G. $\bar{E}$ is defined as

$$(v, w) \in \bar{E} \Longrightarrow \exists \text{ a path from } v \text{ to } w \text{ in G.} \tag{17}$$

Artificial reaction graphs are natural and intuitive ways of representing the pathways of a metabolism. They build, together with the other constituting parts of the artificial chemistry, a full description of the system's molecular dynamics. Another, formally quite similar approach of visualizing metabolisms is Petri nets [34]. There, the molecular dynamics is created by performing the transactions of the net, starting from an initial state.

3.1 Is a Random Graph a Reaction Graph?

A random graph of the elements above is *not always* a reaction graph, because there are some conditions that it may violate [37]. For instance, the graph displayed in Figure 4 is not a reaction graph because a small amount of A would produce an infinite amount of B, a reaction that would realize *creatio ex nihilo*. This is a biochemically impossible reaction mechanism. To arrive at reaction networks that do not produce or consume matter, the reaction graphs have to fulfill the condition of *material balance*.

3.1.1 The Material Balance in Artificial Reaction Networks

Let an elementary reaction of a reaction network be written as

$$A + B \rightarrow C + D \tag{18}$$

This equation implies that the weights of the species A, B, C, and D fulfill the following equation:

$$m_A + m_B = m_C + m_D \tag{19}$$

Note that this reaction does *not* imply that

$$m_B = m_D \quad \text{or} \quad m_A = m_C \tag{20}$$

Random artificial reaction networks may contain elementary reactions that violate the material balance. For example, the following equations of an artificial reaction network R (the graph G of R is shown in Figure 4)

$$\begin{aligned} &\textit{i)}\ A \rightarrow B + C \\ &\textit{ii)}\ C \rightarrow A \end{aligned} \tag{21}$$

are inconsistent because the following equation system

$$\begin{aligned} &\textit{i)}\ m_A = m_B + m_C \\ &\textit{ii)}\ m_C = m_A \end{aligned} \tag{22}$$

cannot be solved in positive integers. If A is chosen as the independent variable, $m_A = m_C$ and $m_B = 0$; that is, species B has weight zero. This is not possible. To test whether an artificial reaction network is consistent, the following algorithm must be applied:

Step 1: Construct the $m \times n$ stoichiometric matrix $M(R)$ of the reaction network R. The total number of reactions in the network is n and m is the number of different species taking part in the mechanism.

Step 2: Calculate the rank of M [rank($M(R)$)].

Step 3: Calculate the number of independent variables for species weights using

$$m - \text{rank}(M(R)) \tag{23}$$

Step 4: Declare n species weights as independent variables and set their weights to unity.

Step 5: Solve the system of equations

$$M(R)\dot{m} = 0 \tag{24}$$

with m being the unknown species weights.

Step 6: Check if all weights (all components of $\dot{m}$) are positive. If this is not the case, the artificial reaction network contradicts the laws of thermodynamics.

3.1.2 A Small Example

This can easily be illustrated with the following example. Let the graph in Figure 4 be our artificial reaction system R. Then the 2×3 stoichiometric matrix $M(R)$ is

	A	B	C
(1)	−1	1	1
(2)	1	0	−1

The rank of $M(R)$ is 2, so the number of independent species is, according to Equation 23, $m - 2 = 3 - 2 = 1$. If we now choose A as the independent species with $m_A = 1$, Equation 24 becomes

$$\begin{pmatrix} -1 & 1 & 1 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ \dot{m}_B \\ \dot{m}_C \end{pmatrix} = 0 \tag{25}$$

This results in the following system of equations:

$$\begin{aligned} m_B + m_C &= 1 \\ m_C &= 1 \end{aligned} \tag{26}$$

which has the solution $m_A = m_C = 1, m_B = 0$. Due to the zero weight of B, the artificial reaction network should be discarded. As a result, the reaction network of species A, B, and C with only symbolic names is shown to violate the material balance even with the unknown composition of A, B, and C.

3.2 Variation of Artificial Reaction Networks

Genetic programming (GP) [13, 23] is an evolutionary algorithm that represents an individual as an algorithm that can be executed to get its fitness value. It is a common interpretation of the above definition that GP evolves *computer programs*. But there is a different and more general interpretation of the definition. According to this interpretation, GP systems [13] evolve *structures* of variable size and shape. These structures may be representations of dynamic systems that are not necessarily isomorphic to computer programs. In our case, the execution of an individual amounts to the simulation of the artificial reaction system represented by the individual, that is, the reaction graph. The genetic operators of the GP system now have to handle the graph representation. Additionally, they need to obey the restrictions of reaction graphs. This stands in contrast to the work of Lohn et al. [24], who neglect biochemical plausibility in their graphs. Thus a reaction graph that has undergone a modification by a genetic operator (either mutation or crossover) has to be a real reaction graph, that is, needs to fulfill material balance.

3.2.1 Mutation

A mutation can now be one of the following operations:

1. Adding/deleting inhibitors or catalysts. An existing reaction can be changed by adding an inhibitor/catalyst if none is present, or by deleting the inhibitor/catalyst. A new inhibitor/catalyst is chosen randomly from all present molecules in the

reaction graph. The new inhibitor/catalyst is assigned a randomly chosen rate constant $k \in [k_{\min}; k_{\max}]$.

2. Variation of inhibitive/catalytic rates. The inhibitive/catalytic activity, expressed by the rate constant k, can be changed. Therefore, k is multiplied by a random number from a log normal distribution.
3. Changing the type of a reaction. The type of a reaction can be changed by adding or deleting participants of the reaction, either as a reactant or as a product. If a new participant is added, it is chosen randomly from all present molecules in the reaction graph.
4. Adding/deleting reactions. A reaction is deleted by removing the node from the reaction graph. All edges to and from the reaction node are deleted, too. A reaction is added by choosing all participants randomly from all present molecules in the reaction graph.
5. Adding/deleting molecules. A molecule is deleted by deleting the node from the reaction graph. If the molecule participates in reactions, all edges and all reactions are deleted accordingly. Elements of the sets I and O must not be deleted by a mutation.

The resulting new graph must obey the material balance. If the mutation hurts this condition, a high penalty value is assigned. Formally, the mutation of a graph can be written as follows:

$$\begin{array}{lll}
 & G \longrightarrow G: & \textbf{Mutation} \\
1) & (V,E) \longrightarrow (V',E), & V \subset V' \quad \text{adding a node (Cases 4,5)} \\
2) & (V,E) \longrightarrow (V',E), & V' \subset V \wedge I \subset V' \wedge O \subset V' \quad \text{deleting a node (Cases 4,5)} \\
3) & (V,E) \longrightarrow (V,E'), & E \subset E' \quad \text{adding an edge (Cases 1,3,4)} \\
4) & (V,E) \longrightarrow (V,E'), & E' \subset E \quad \text{deleting an edge (Cases 1,3,4,5)} \\
5) & (V,E) \longrightarrow (V,E'), & E' = \{e_i \| e_i \in E, \forall i \neq j, \quad e'_j = (v_j, w_j, k'_j), k'_j = k_j \cdot e^{N(0,1)}\} \\
 & & \text{with } N(0,1) \text{ being a standard Gaussian-distibuted random variable} \\
 & & \text{mutating the rate constant (Case 2).}
\end{array} \tag{27}$$

3.2.2 Crossover

With crossover, subgraphs are exchanged between two individuals. Both resulting graphs must hold the reaction graph conditions. Crossover exchanges metabolic pathways of two reaction graphs. A pathway is extracted by a depth-first traversal (15) starting from a molecule node in the reaction graph. The pathway is empty if no path from the starting node to the target node exists. An example for the determination of a metabolic pathway is shown in Figure 5. The two pathways are then exchanged and the resulting reaction graphs are released into the next generation, provided the material balance condition holds. An example for combining a graph with a pathway is shown in Figure 6. Formally, crossover can be written as follows:

$$\begin{array}{lll}
G \times G \longrightarrow G \times G: & & \textbf{Crossover} \\
1) \quad G \longrightarrow G \times P, & \text{cutting a pathway} & \\
\qquad (V,E) & \longrightarrow & (V',E') \times (V'',E''), \\
 & & V = V' \cup V''; E = E' \cup E''; V', V'', E', E'' \neq \emptyset
\end{array}$$

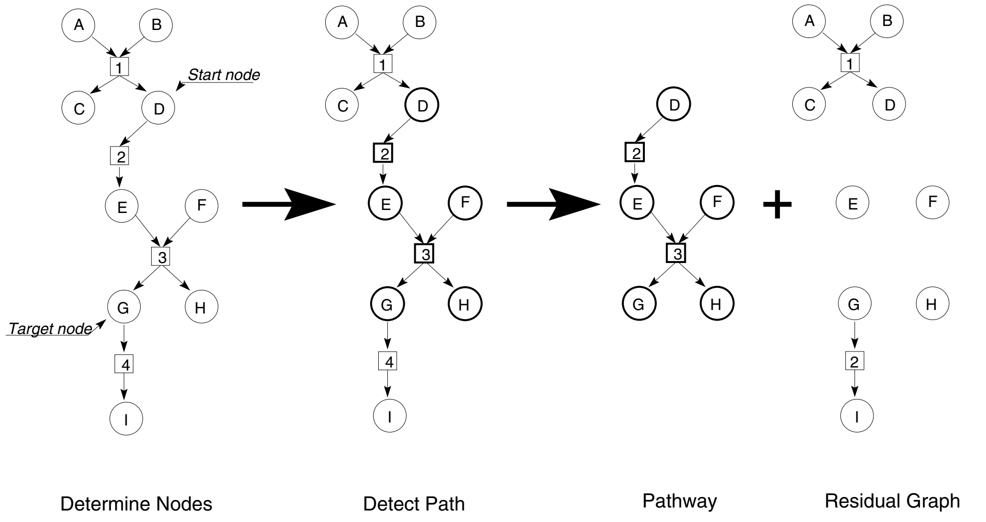


Figure 5. Detecting a pathway in a reaction graph. From left to right: (a) The start node and the target node are set. (b) Depth-first search from start node to target node determines the path. (c) The pathway is extracted from the reaction graph, including all necessary nodes. (d) The pathway is deleted from the reaction graph. Graph and pathway are shown without inhibitive/catalytic edges.

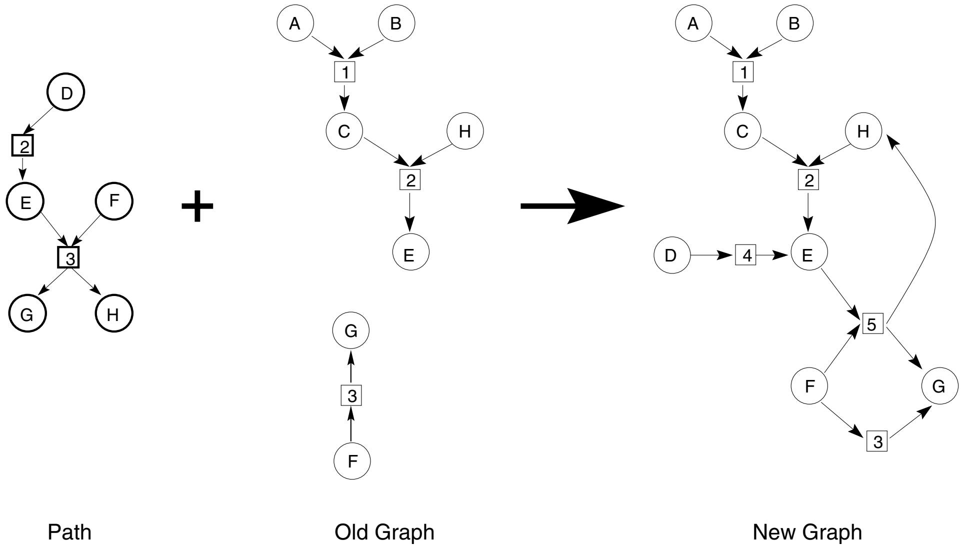


Figure 6. Inserting a pathway into a reaction graph. The reactions of the pathway are inserted into the graph. Missing nodes are created (e.g., node "D"). All existing reactions, molecules, and edges remain unchanged. Graph and pathway are shown without inhibitive/catalytic edges.

2) $G \times P \longrightarrow G$, inserting a pathway

$$\begin{aligned} (V, E) \times (V', E') \quad &\longrightarrow \quad (V'', E''), \\ &V'' = V \cup V'; E'' = E \cup E'; V, V', E, E' \neq \emptyset \end{aligned} \tag{28}$$

A pathway is determined by a traversal on the shortest path between two nodes.

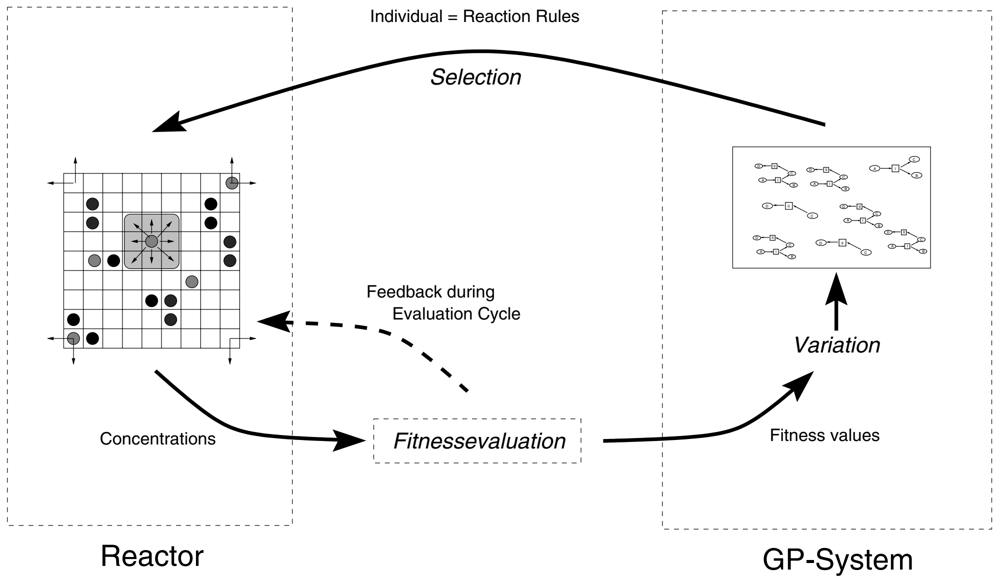


Figure 7. Schematic view of the evolutionary process.

On the way through the graph to the target node, every reaction that is encountered becomes part of the pathway, including every necessary preceding molecular node (e.g., node "F" in Figure 5). This pathway is then removed from the original graph and inserted into the crossover partner, from which in turn a pathway has been subtracted before (see Figure 6). Molecules of the pathway whose names (numbers) are already present in the target graph keep their names in the new, resulting graph. Molecules whose names are not present in the new graph are assigned a new name (number). Elements of I and O must not be deleted by a cutting operation.

Mutation and crossover operators cause the reaction network to change its size and constituent parts. This has two effects: (a) The artificial chemistry becomes constructive, which means that it is not limited to the finite number of reactions or molecules already present at the beginning.[1] As a result, the terminal set of the GP system changes during the evolution, because the set of possible molecules is variable; (b) The genome representing a reaction network has variable size and shape, which will result in growing difficulties for the analysis of the kernel.

In the following experiments we will see that the size of the genome grows during the evolutionary phase, showing well-known phenomena such as introns and bloating [12, 13].

4 The Evolution of Metabolisms

The automatic evolution of an AC with desired properties requires (a) a reactor in which the reactions are simulated, (b) a reaction graph representing all possible reactions in the reactor, (c) variation operators for reaction graphs, and (d) a fitness function for the evaluation. A schematic view of the algorithm is shown in Figure 7.

The population of individuals is initialized by the GP system and successively evaluated. The individuals are interpreted by the reactor module as a set of reaction rules. The reactor module produces as an outcome time-dependent vectors of concentrations,

1 In fact, it is a *weakly constructive* chemistry, according to [17, 18].

```
while $\neg terminate()$ do
    Draw graph $g$ out of $P$;
    while $evaluateMetabolism(g)$ do
        Fill in input set substances $I_g(t)$;
        Update reactor using alg. (8) according to $g$;
        Measure output set concentration $\vec{s_t}(O_g)$ at time $t$;
        Use $O_g$ to solve problem;
        Compute outflow $\Phi(t)$;
    od
    Evaluate total fitness $f(g)$;
    Update population (variation);
od
```

Figure 8. Pseudo code visualization of the main loop of the algorithm as shown in Figure 7.

which in turn are taken as input of the fitness evaluation module. This module interprets the concentrations according to the fitness function and assigns a specific fitness value to the simulated individual. The GP system uses its genetic operators to evolve increasingly better individuals until the whole algorithm terminates. The algorithm can now be expressed in pseudo code as in Figure 8.

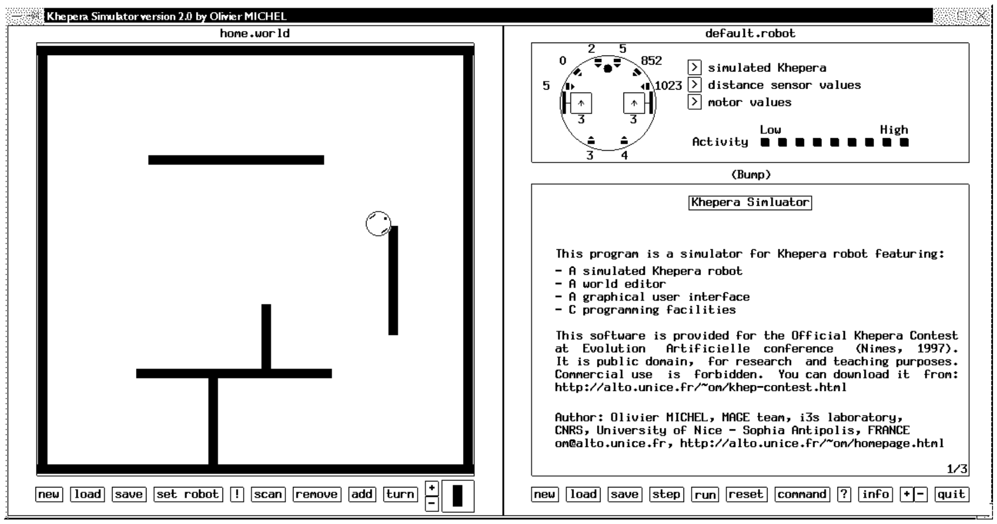
5 Robot Navigation—Experimental Setup

Now we demonstrate the capability of an AC programmed by an evolutionary algorithm to control a miniature robot in real time. Therefore, the simulated AC is connected to both the sensory information and the motors of a robot—either simulated or real existing. The problem of the experiments is the task of navigating an autonomous robot through a maze without hitting any obstacles. To succeed in this task, fast and efficient use of sensory information by means of metabolic information processing shall be evolved. To avoid expensive evaluations on a real robot, the first experiments are carried out with a simulator. Afterward, the evolved artificial metabolisms are tested on a real robot. Figures 9 and 10 show the artificial environment of a small robot in the simulator.[2] This robot simulator acts as the fitness evaluation module in Figure 7. The feedback loop is realized by reducing the motor substance concentration while a motor command is executed by the simulator.

5.1 Sensor Integration

The sensors of the Khepera robot used here are shown in Figure 10. Pairs of proximity sensors (both front sensors and back sensors and in each case the side sensors) cause an inflow of substances, so the concentrations of b, c, d, and e change linearly according to the values of the appropriate sensor pair. The substances b, c, d, and e together form the input set $I \subset V$. To stay with the biological metaphor, several membrane channels into a cellular compartment are realized with this experimental setup. The

2 Written by Olivier Michel et al., University of Nice.



Khepera Simluator

This program is a simulator for Khepera robot featuring:
- A simulated Khepera robot
- A world editor
- A graphical user interface
- C programming facilities

This software is provided for the Official Khepera Contest at Evolution Artificielle conference (Nimes, 1997). It is public domain, for research and teaching purposes. Commercial use is forbidden. You can download it from: http://alto.unice.fr/~om/khep-contest.html

Author: Olivier MICHEL, MAGE team, i3s laboratory, CNRS, University of Nice - Sophia Antipolis, FRANCE
om@alto.unice.fr, http://alto.unice.fr/~om/homepage.html

1/3

Figure 9. The Khepera simulator. A small robot navigating in a random environment.

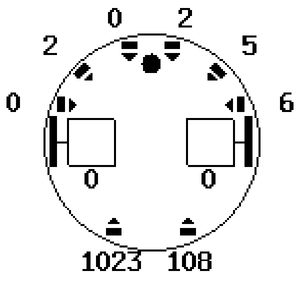


Figure 10. Schematic view of the Khepera robot. The robot has six proximity sensors in the front and at both sides and two backward proximity sensors. It can be controlled by setting the rotation direction and speed of its two motors (currently zero). The numerical values indicate the proximity of obstacles. In the current situation, the robot has an obstacle in front of its left rear sensor (indicated by the high numerical value). This obstacle is also measured by the right rear sensor and determined to be more remote from there.

inflow $\phi(s_i)$ can be described by the following functions:

$$\left.\begin{array}{rcl}\phi(b) & = & f(\text{left}_1, \text{left}_2)\\ \phi(c) & = & f(\text{front}_1, \text{front}_2)\\ \phi(d) & = & f(\text{right}_1, \text{right}_2)\\ \phi(e) & = & f(\text{back}_1, \text{back}_2)\end{array}\right\} \quad \text{with} \tag{29}$$

$$f(s_1, s_2) = \text{cellsize} \cdot \text{maxInflow} \cdot \frac{\max(s_1, s_2)}{1023}. \tag{30}$$

The parameters cellsize and maxInflow limit the inflow of a sensor substance to a maximum value (up to a standard of 10% of cell size per iteration in all experiments). The value 1023 is the maximum sensor value and ensures the linear dependence of the inflow. If the robot is surrounded by obstacles, the sensor-induced inflow per time step of substances would sum up to 0.4 or 40% of the cell volume. Whether or not the metabolism makes use of the inflowing sensor substances is left to its own decision.

5.2 Actuator Integration

As the task of navigation, a simple chemotactic behavior of the robot should be achieved. Chemotaxis can be described as a frequency modulation of tumbling phases during periods of straight movement [36]. Thus, the rotation direction and speed of the two motors (Figure 10) remain unchanged unless the concentration of the motor control substance a exceeds a certain threshold. Then the rotation direction of one motor is changed, causing the robot to rotate[3] around its z-axis. During the switch of motor rotation direction, a certain amount of a is consumed, so the concentration of a diminishes during a rotation of the robot as specified with the outflow rate α (unless it is produced by the metabolism at a higher rate). The substance a builds the output set $O \subset V$. The outflow of a can then be described by

$$\phi(a) = \begin{cases} \alpha \cdot a & \text{if} \quad a > a_{\min} \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

5.3 Fitness Function

The strategy of the fitness function of the experiments is to achieve long straight movements if no obstacles are present, and short evasive rotations otherwise. The performance of the evaluated metabolism is calculated during a fixed period of execution, by summing up the differences between rotation speed and direction of both motors during the execution. If both motors rotate with the same speed and direction, this difference is zero, so the optimal behavior of a metabolism is to minimize the number and duration of rotation phases.

5.4 Additional Parameters

The difficulty of the correct parameters can be shown by a—at first glance—simple example: If the concentration of a exceeds the threshold $a_{\min}$, the motor rotation is switched and a certain amount of a is consumed. The concentration of a may rise again due to the specific reactions until the next active cycle. The question is now to adjust the consumption rate of a, the number of reactor cycles between two active cycles, the motor speed, and the activation threshold $a_{\min}$ in a way that the robot moves in a reasonable manner. Note that the adjustment of these parameters is strongly influenced by the individual's fitness, although it is fully independent from the actual genome. Another parameter defines the duration of the simulation. In our experiments, a lower bound emerged for the execution time: If the duration was shorter than the time the robot needed to drive straight across the maze, very high fitness values were assigned to the metabolisms, because the uninterrupted straightforward movement of the robot, the default behavior, granted a high reward. So the execution time should be slightly longer to ensure encounters with obstacles.

6 Robot Navigation—Results

The settings of the parameters of the robot experiments are shown in the Koza tableau in Table 1 for the GP system and in Table 2 for the cellular reactor. The development of average fitness and length of the individuals is shown in Figure 11. The number of

3 The rotation direction is either clockwise or counterclockwise and remains unchanged during all experiments.

Table 1. Koza tableau with parameter settings for the GP system in the robot navigation experiment. The size of the terminal set (and therewith the size of the graphs) are potentially infinite.

Parameters	**Values**
Objective	Evolve an artificial reaction network that is able to control an autonomous robot in a maze with a minimal number of collisions (*chemotaxis*).
Terminal set	Species $\{a, b, c, d, e\}$, other species
Function set	Reactions of type 4,5,6, and 7
Selection scheme	(μ, λ)
Population size	$\mu = 50$, $\lambda = 250$
Crossover probability	0.7
Mutation probability	0.9
Termination criterion	Steady state
Max. number of generations	500
Max. size of graphs	Unlimited
Initialization method	Grow

Table 2. Tableau with parameter settings for the cellular reactor for the robot experiments. The number of reactor cycles ensures a sufficiently long execution time per individual. The values of α and $a_{\min}$ proved to be satisfactory for the experiment.

Parameters	**Values**
Reactor size	20×20 toroidal reactor with Moore neighborhood
Neighborhood radius r	1
Initial fill level	50%
Input	10% inflow of each input substance b, c, d, e
Output	Rotation of motors according to concentration of a
Termination criterion	No. of iterations (defined execution time)
Consumption rate α of actuator substance a	0.9
Actuator substance concentration threshold $a_{\min}$	0.1
Reactor cycles between sensor processing	10
Execution time per individual	1,200 reactor cycles

nodes in the reaction graph G determines the length of the genome. The minimum length of the functional code remains constant during the evolution, but the overall length increases. The functional code of the genome is equivalent to the kernel of the representing graph G. A node outside the kernel will never change its concentration due to any reaction, so it can be discarded without any consequences for the dynamics of the metabolism.

7 Verification and Analysis of the Results

The use of a simulator for the evolution of ACs with the desired capabilities may be deemed too simplified a version of the real problem. To address this objection, we tested the solution with a real-world robot in a real world with real time requirements. The evolved graph remained completely unchanged; only some parameters that took the different time scales of the simulator and the real robot into consideration were changed. Above all, due to the faster movement of the real robot we had to amplify

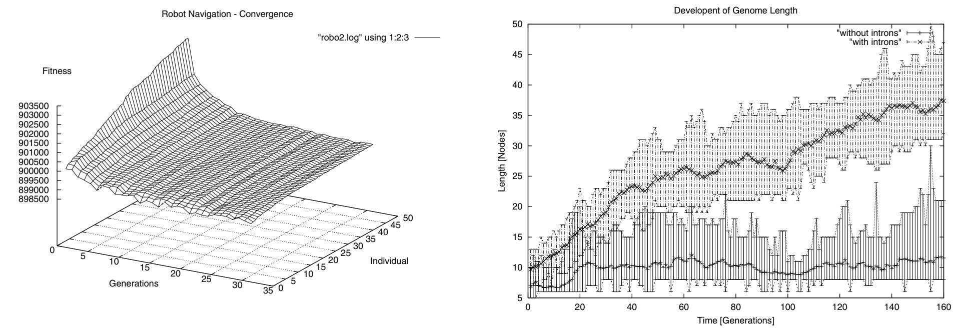


Figure 11. **Left:** Development of fitness values during the run. The individuals are sorted in descending order. **Right:** Development of length of genome. The figure shows the minimal, maximal, and average length per generation of the whole genome and the kernel.

the sensor-induced substance inflow so that the inflow function (Equation 30) got an additional multiplying term. A sequence of photos of a typical object-avoiding turn is shown in Figure 12.

An analysis of the reaction graph (Figure 13) leads to an interesting correspondence to results from Parkinson and Blair [31]. They report that the motile bacteria *E. coli* has nearly all its exteroceptors at its front end (see Figure 14, right). According to their argument, this is a plausible strategy for placing receptors, because the front end of the cell is exposed to an increased flux of substances from the environment that increases the efficiency of detecting food. In Figure 14, left, the metabolic network is inserted into a schematic view of the robot and connected with the actuators and sensors. Only the front and left sensors as well as the motor are connected with the network. In our robot navigation experiments, it is quite important for the robot (i.e., for the controlling metabolism) to detect obstacles at its front end, because if it

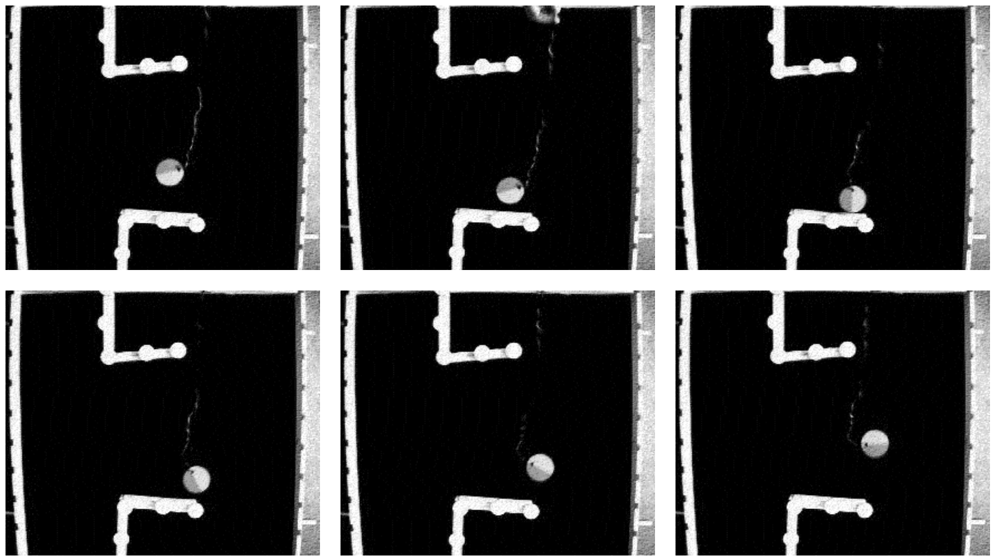


Figure 12. Movement of a real Khepera robot. The Khepera encounters a wall and turns counterclockwise (top left to bottom right). Wide-angle view of a top-mounted camera looking down on a 1-m^2 test maze.

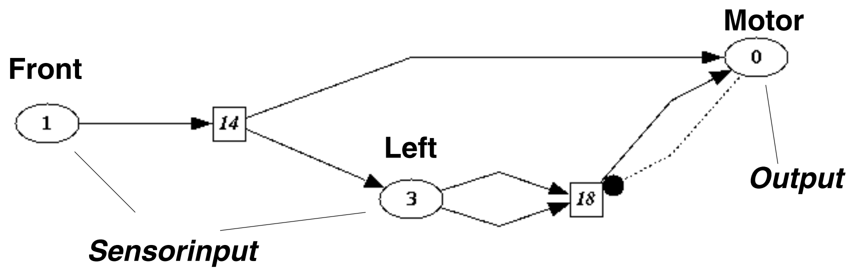


Figure 13. The kernel of the resulting artificial chemistry. The total graph has nearly 50 nodes and 70 edges and is not shown here. The kernel is labeled according to the function of the nodes. The dotted line indicates a catalytic effect.

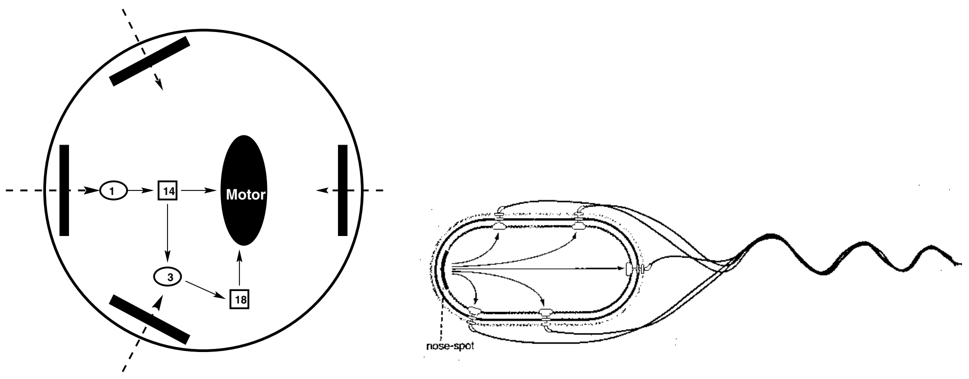


Figure 14. **Left:** The reaction network of the best individual as it is connected with the sensors and actuators of the robot. Note that the nonfunctional parts of the genome are omitted. **Right:** The chemoreceptors of *E. coli* are located at the front of the cell and form a "nose." Picture taken from [31].

is bumping into obstacles, the fitness decreases drastically. The graph in Figure 14 shows that a good metabolism just makes use of the substances representing the front sensors of the robot (represented by the "1"), and, in a superficially strange way, of the left sensors (represented by "3"). In other words, the metabolism increases the concentration of the motor substance if an obstacle is present in front of the robot. The concentration of the substance signaling an obstacle on the left side of the robot is increased, too (though there might not be any obstacle!). However, the left sensors will measure the obstacle during the counterclockwise rotation with a short delay, and the concentration has already been increased by the metabolism, so that a rotation is not interrupted by a short fluctuation of the motor substance concentration below the activation threshold. This results in a continuous turn that lasts long enough to avoid the obstacle.

The evolved metabolism thus only uses the sensors with the maximum benefit and the shortest reaction time to achieve the goal of obstacle avoidance. It realizes this in a very elegant and efficient way. The small kernel is a result of the real time requirements of the problem, because the metabolism maximizes its return on investment (its fitness) if it does not waste its time on computing nonessential metabolites but rather important substances, in this case the motor substance *a*.

8 Conclusion

This article proposes a solution to the problem of programming an artificial chemistry, a special branch of artificial life research. The emergent properties of a highly complex dynamic system are predetermined—or programmed—by artificial reaction graphs, a special representation for a genetic programming system that is used here to evolve increasingly useful artificial metabolisms. Appropriate features of the metabolisms are measured "online" during the system's development and are used to solve a computational problem, here the simulated task of robot navigation. This problem has a strong need for speed and robustness of the solution, so that the algorithm evolves efficient and fast sensor information-processing pathways in the simulated metabolism. Individuals have been successfully transferred into a real-world robotic experiment. We also found intriguing similarities to the use of exteroceptors during chemotaxis in motile bacteria such as *E. coli*.

The use of biochemically plausible reaction graphs, plausible at least at the material balance level, ensures reaction networks, which may act as a model for a potential "wet" chemical implementation of a robot control system. This work can be seen as a model application of an interconnection-free, decentralized, and lean control architecture consisting of an artificial chemistry as the information-processing device.

Acknowledgments
This project is supported by the DFG (*Deutsche Forschungsgemeinschaft*), under grant Ba 1042/2-2 and Ba 1042/6-1. Thanks to Christian Düntgen for his technical assistance.

References

1. Adamatzky, A., & Holland, O. (1998). Edges and computation in excitable media. In C. Adami, R. K. Belew, H. Kitano, & C. Taylor (Eds.), *Proceedings of the 6th International Conference on Artificial Life* (pp. 379–383). Cambridge, MA: MIT Press.
2. Adamatzky, A., Holland, O., Rambidi, N., & Winfield, A. (1999). Wet artificial brains: Towards the chemical control of robot motion by reaction-diffusion and exitable media. In D. Floreano, J.-D. Nicoud, & F. Mondada (Eds.), *Advances in Artificial Life. Proceedings of the 5th European Conference on Artificial Life* (pp. 304–313). Berlin: Springer.
3. Adleman, L. M. (1994). Molecular computation of solutions to combinatorical problems. *Science, 266*, 1021.
4. Aoki, T., Kameyama, M., & Higuchi, T. (1992). Interconnection-free biomolecular computing. *Computer, 25*, 41–50.
5. Arkin, A., & Ross, J. (1994). Computational functions in biochemical reaction networks. *Biophysical Journal, 67*, 560–578.
6. Astor, J. C., & Adami, C. (1998). Development of evolution of neural networks in an artificial chemistry. In C. Wilke, S. Altmeyer, & T. Martinetz (Eds.), *Third German Workshop on Artificial Life* (pp. 15–30). Frankfurt: Verlag Harri Deutsch.
7. Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.). (1997). *Handbook of evolutionary computation.* New York: Oxford University Press.
8. Bagley, R. J., & Farmer, J. D. (1992). Spontaneous emergence of a metabolism. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems* (pp. 93–140). Reading, MA: Addison-Wesley.
9. Bagley, R. J., Farmer, J. D., & Fontana, W. (1992). Evolution of a metabolism. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems* (pp. 141–158). Reading, MA: Addison-Wesley.
10. Banzhaf, W. (1990). The "molecular" traveling salesman. *Biological Cybernetics, 64*, 7–14.

11. Banzhaf, W., Dittrich, P., & Rauhe, H. (1996). Emergent computation by catalytic reactions. *Nanotechnology, 7*, 307–314.

12. Banzhaf, W., Nordin, P., & Francone, F. D. (1997, July). *Why introns in genetic programming grow exponentially.* Paper presented at the Workshop on Exploring Non-coding Segments and Genetics-based Encodings at ICGA-97.

13. Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic programming—An introduction. On the automatic evolution of computer programs and its applications.* San Francisco: Morgan Kaufmann, and Heidelberg: dpunkt.verlag.

14. Brooks, R. A. (1994). Coherent behavior from many adaptive processes. In D. Cliff, P. Husbands, J. A. Meyer, & S. Wilson (Eds.), *From animals to animats 3* (pp. 22–29). Cambridge MA: MIT Press/Bradford Books.

15. Conrad, M. (1992). Molecular computing: The key-lock paradigm. *Computer, 25*, 11–22.

16. Conrad, M., & Zauner, K.-P. (1998). Conformation-driven computing: A comparison of designs based on DNA, RNA, and protein. *Supramolecular Science, 5*, 787–790.

17. Dittrich, P., Ziegler, J., & Banzhaf, W. (2001). Artificial chemistries—A review. Manuscript submitted for publication.

18. Fontana, W., Wagner, G., & Buss, L. W. (1994). Beyond digital naturalism. *Artificial Life, 1*, 211–227.

19. Forrest, S. (1991). *Emergent computation.* North-Holland: Elsevier.

20. Güting, R. H. (1992). *Datenstrukturen und Algorithmen.* Stuttgart: B. G. Teubner.

21. Hjelmfelt, A., Weinberger, E. D., & Ross, J. (1991). Chemical implementation of neural networks and Turing machines. *Proceedings of the National Academy of Science USA*, 88, 10983–10987.

22. Husbands, P. (1998). Evolving robot behaviors with diffusing gas networks. In *Evolutionary robotics* (pp. 71–86). Lecture Notes in Computer Science 1468. Berlin: Springer.

23. Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection.* Cambridge, MA: MIT Press.

24. Lohn, J. D., Colombano, S. P., Scargle, J., Stassinopoulos, D., & Haith, G. L. (1998). Evolving catalytic reaction sets using genetic algorithms. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (pp. 487–492). New York: IEEE.

25. Lugowski, M. W. (1986, Sept.) *Computational metabolism.* (Tech. Rep. No. 200). Bloomington: Indiana University Computer Science Department.

26. Lugowski, M. W. (1989). Computational metabolism: Towards biological geometries for computing. In C. G. Langton (Ed.), *Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems.* (Vol. 6 of *Santa Fe Institute Studies in the Sciences of Complexity*, pp. 341–368). Redwood City, CA: Addison-Wesley.

27. Michal, G. (1998). *Biochemical pathways.* Heidelberg, Germany: Spektrum Akademischer.

28. Mills, A. P., Jr., Yurke, B., & Platzman, P. M. (1999). DNA analog vector algebra and physical constraints on large-scale DNA-based neural network computation. In E. Winfree & D. K. Gifford (Eds.), *Proceedings of the 5th DIMACS Workshop on DNA Based Computers* (pp. 65–73). Providence, RI: American Mathematical Society.

29. Nissen, V. (1997). *Einführung in evolutionäre Algorithmen.* Braunschweig/Wiesbaden, Germany: Vieweg.

30. Okamoto, M., Tanaka, K., Maki, Y., & Yoshida, S. (1995). Information processing of neural network system composed of 'biochemical neuron': Recognition of pattern similarity in time-variant external analog signals. In R. Paton (Ed.), *Proceedings of the First International Workshop on Information Processing in Cells and Tissues.*

31. Parkinson, J. S., & Blair, D. F. (1993). Does *E. coli* have a nose? *Science, 259*, 1701–1702.

32. Rambidi, N. G., Kuular, T. O.-O., & Makhaeva, E. E. (1999). Information-processing

capabilities of chemical reaction-diffusion systems I. Belousov-Zhabotinsky media in hydrogel matrices and on solid support. *Advanced Materials for Optics and Electronics, 8*(4):163–171.

33. Rambidi, N. G., & Maximychev, A. V. (1997). Molecular image-processing devices based on chemical reaction systems 6. Processing half-tone images and neural network architecture of excitable media. *Advanced Materials for Optics and Electronics, 7*(4):171–182.

34. Reddy, V. N., Mavrovouniotis, M. L., & Liebmann, M. N. (1994). Modeling biological pathways—a discrete-event-systems approach. *Molecular Modeling, 576*, 221–234.

35. Shackleton, M. A., & Winter, C. S. (1998). A computational architecture based on cellular processing. In M. Holcombe & R. Paton (Eds.), *Information Processing in Cells and Tissues* (pp. 261–272). New York: Plenum Press.

36. Stock, J. B., & Surette, M. G. (1996). Chemotaxis. In *Escherichia coli* and *Salmonella typhimurium.* (pp. 1103–1129). Washington, DC: ASM Press.

37. Temkin, O. N., Zeigarnik, A. V., & Bonchev, D. (1996). *Chemical reaction networks—A graph-theoretical approach*. Boca Raton, FL: CRC Press.

38. Zauner, K.-P., & Conrad, M. (1996). Simulating the interplay of structure, kinetics, and dynamics in complex biochemical networks. In R. Hofestädt, T. Lengauer, M. Löffler, & D. Schomburg (Eds.), *Computer Science and Biology—Proceedings of the German Conference on Bioinformatics* (No. 1 in IMISE Report, pp. 336–338). Leipzig, Germany: Universität Leipzig.

39. Zauner, K.-P., & Conrad, M. (1998). Conformation-driven computing: Simulating the context-conformation-action loop. *Supramolecular Science, 5*, 791–794.

40. Ziegler, J., Dittrich, P., & Banzhaf, W. (1998). Towards a metabolic robot control system. In M. Holcombe & R. Paton (Eds.), *Information processing in cells and tissues*. New York: Plenum Press.