BioSystems 213 (2022) 104608

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Computing within bacteria: Programming of bacterial behavior by means of a plasmid encoding a perceptron neural network

A. Gargantilla Becerra [a], M. Gutiérrez [b], R. Lahoz-Beltra [c,d,*]

[a] Department of Systems Biology, Centro Nacional de Biotecnología (CSIC), 28049, Madrid Spain
[b] School of Informatics and Telecommunications, Faculty of Engineering and Sciences, Diego Portales University, Santiago, Chile
[c] Department of Biodiversity, Ecology and Evolution (Biomathematics), Faculty of Biological Sciences, Complutense University of Madrid, 28040, Madrid, Spain
[d] Modeling, Data Analysis and Computational Methods in Biology Research Group. Complutense University of Madrid, 28040, Madrid, Spain

ARTICLE INFO

Keywords:
Bacterial computing
Programming bacterial behavior
Neural network computation in bacteria
Bio-artificial intelligence
Perceptron encoded in a plasmid

ABSTRACT

In nature, bacteria exhibit a limited repertoire of behaviors in response to environmental changes. Synthetic biology has now opened up the possibility of programming cells or unicellular organisms in order to enable them to perform certain tasks, which would allow the programming of 'intelligent' bacteria. Many of the theoretical ideas that Liberman proposed last century, for example his seminal idea that a cell is a computer, are now being put into practice with bacterial colonies in both *wet* and *in silico* experiments.These bacteria may one day be used to solve a wide range of problems whose solution requires their adaptation to external changes either within a bioreactor, organ or tissue of a patient or through the design of microbial-synthetic consortia oriented to their use in bioprocesses to produce medicines, biofuels or biomaterials. In this work, we show the possibility of programming synthetic bacteria with a previously trained perceptron neural network. First, we illustrate how a colony of bacteria endowed with a perceptron is able to solve an optimization problem *in silico*. Secondly, we study by means of *in silico* simulations how a perceptron can be applied to program behaviors in bacteria leading to social interactions and to the formation of complex communities that in the future would be useful in biotechnology. Finally, we go a step further, and study how the above perceptron designed to program bacterial behavior is implemented in a genetic circuit designed for this purpose. Once the genetic circuit was obtained, it was engineered into a plasmid.

1. Introduction

Many living beings exhibit behaviors and learning, i.e. they are organisms whose responses to a stimulus are modified based on previous experience. While insects exhibit stereotyped behaviors, primates exhibit a very rich repertoire of behaviors. For years, the idea has been maintained that learning capabilities and behavior are only manifested in multi-cellular living beings endowed with a nervous system of varying degrees of complexity. Although some forms of elementary learning have also been observed in non-neural multicellular organisms, such as plants (Gagliano et al., 2014), it has traditionally been considered that intelligence is an attribute that is absent in microorganisms. However, bacteria, slime molds and other unicellular organisms exhibit behaviors encoded in their genes. The result is that these organisms behave in a manner comparable to sensory-motor agents (Lahoz-Beltra et al., 2014). Nevertheless, at present, experimental evidence suggests that microorganisms are *intelligent* sensory-motor agents. About a decade ago it has been demonstrated that a unicellular organism, the mold *Physarum polycephalum*, is capable of developing a type of learning called "habituation" and it was also able to solve the problem of a maze (Nagaki et al., 2000). Based on the features of *Physarium* Whiting et al. (2016) using protoplasmic *Physarium* tubes were able to build 'slime mold' computers. In a context different from that of natural computation, Lyon (2015) has proposed to reconsider learning capabilities in bacteria by drawing a parallelism between prokaryotic behavior and neural learning. In some way all bacterial responses to environmental changes, e.g. chemotaxis, are the result of the presence of chemoreceptors which detect signals from the environment and process this information through signal transduction networks.

The study of organisms without brains but exhibiting intelligent

* Corresponding author. Department of Biodiversity, Ecology and Evolution (Biomathematics), Faculty of Biological Sciences, Complutense University of Madrid, 28040, Madrid, Spain.
E-mail addresses: agb1894@gmail.com (A.G. Becerra), martin.gutierrez@mail.udp.cl (M. Gutiérrez), lahozraf@ucm.es (R. Lahoz-Beltra).

https://doi.org/10.1016/j.biosystems.2022.104608
Received 10 September 2021; Received in revised form 7 December 2021; Accepted 11 January 2022
Available online 19 January 2022

behaviors, i.e. what has come to be known as 'intelligence outside the brain', is a subject that has long been addressed from a variety of viewpoints. In addition to being obviously addressed from a biological perspective, one of the most influential disciplines has been physics. In the scope of biology, several topics have been proposed tackling issues such as the minimum requirements for intelligence or looking for an explanation of the origin of intelligence in non-neural organisms from the point of view of evolution. Among these approaches there are those that adopt the point of view of the physics of open systems. For example, Ben Jacob et al. (2006) considers that intelligence in all organisms, including bacteria, involves several factors related to their ability to 'sense' the environment. These factors include the interpretation of chemical messages or the distinction between internal and external information, connecting such factors with Schrodinger's notion of negantropy, as well as the ability of non-neural organisms to distinguish itself from other organisms. Under the theoretical framework of physics, it is easier to explain from a theoretical point of view behaviors such as bacterial chemotaxis or the communication between bacteria, e.g. via quorum sensing. Thus, based on this conception a bacterial colony would be a multicellular community exhibiting distributed information processing, with plasmids playing the role of 'cassettes', i.e. a tape recording genetic information. Adopting the ideas of Ben Jacob, other studies (Calvo and Baluska, 2015) wonder what are the distinctive signs for a minimal intelligence. Including also bacteria, they consider those organisms that behave intelligently and lacking a brain show elementary forms of learning and memory (Yang et al., 2020), a sensory-motor system, among other features. The result of the communication between 'minimally intelligent organisms' is the cooperation among individuals and, in short, a form of social intelligence which favors survival. An interesting feature of this view is that it is considered that there must be a capacity to transmit information from a receiver to an effector organism as occurs with the nervous system. There are approaches within this general framework, such as that of Trewavas (2016) relating intelligence with Darwinism, in which intelligence is regarded as the ability of an organism to make decisions to that individual. The ability to process information is a relevant ingredient of intelligence, building networks that ultimately lead organisms to become 'self-aware' as it is believed could happen in plants thanks to a region of the root known as the root cap. However, ideas such as the one mentioned above about the intelligence of plants are still controversial for part of the scientific community. Once again, and in the latter case, intelligence is understood as a faculty requiring an environment as a causal agent, including as a novelty that there must be some goal to be optimized via Darwinian evolution. For example, plants respond to the environment and the objective is to maximize the fitnnes, i.e. the number of seeds. The ability of plants to respond to the environment, i.e. to biotic and abiotic factors, is a sign of their ability to solve problems, therefore being intelligent organisms. Moreover, it is believed that the electric field generated by a growing root would be a communication mechanism with the roots of other plants in the neighborhood, bearing plants a resemblance to swarm intelligence in animals (Baluska et al., 2010). The examples mentioned above illustrate the conceptual background in which the present work is presented. According to Baluška and Levin (2016) the reason why the notion of intelligence in 'headless' organisms is disruptive is mainly due to the ubiquitous nature of cognition in aneural organisms, considering like Trewavas (2016) the role of evolution in the different molecular mechanisms that would be involved in processing information, shaping a memory, etc.

However, it is important to point out that some of the ideas mentioned earlier are not entirely new, having their roots in the 1970s. In the seventies of the 20th century, researchers such as Liberman (1972, 1979) established an analogy between the cells of living beings and computers. In this analogy the computer is not a von Neumann machine or the popular idea of a computer, but a physical system in the tradition of Turing machines with which to study information processing, memory, and other features present in the cells of organisms. Liberman and Minina (1996) suggested that while a program is executed in a computer, in the brain, for example, the program code is written at the molecular level in DNA/RNA, with neurons playing the role of molecular computers. In this way living beings endowed with a brain are able to predict the changes and future state of the environment around them. According to this approach, the substitution of a base in DNA, transcription, etc. are interpreted as possible transformations of the molecular text in DNA. In Liberman's view, and within the previous theoretical framework, in living beings the cell would be under the control of a stochastic molecular machine programmed in the DNA. Such molecular machine is made up of enzymes, which would perform operations on the DNA itself as well as on the RNA and proteins. In agreement with this vision neurons could be interpreted as a particular type of macroscopic machine showing a parallelism with a perceptron, i.e. a simple model of a biological neuron in an artificial neural network (Lahoz-Beltra, 2004).

One of the organisms that best illustrates Liberman's ideas are bacteria. Their characteristics have allowed the cross-fertilization of notions coming from two areas that are in principle far from each other such as molecular biology and computer science. The conception that there is more than a metaphor behind the analogy between bacterial cells and computers has led to statements such as that bacteria are computers that build computers (Danchin, 2009). However, bacteria are more than computers as experimental evidence increasingly suggests how bacteria can learn and anticipate future events. Recently it has been shown (Mitchell et al., 2009) bacteria exhibit Pavlovian conditioning, i.e. conditioned classical learning. Moreover, in their ability to sense the environment and make decisions based on chemical gradients, e.g. during chemotaxis, bacteria show different degree of ability and therefore high individuality as demonstrated in experiments with mazes resembling those performed with mice (Salek et al., 2019).

In summary, the arguments cited above have led to the idea of 'bacterial intelligence' and consequently to the fact that bacteria display 'minimal cognition' (Lyon, 2015). Indeed, several researchers are considering cognition in bacteria results in abilities, e.g. decision-making, robust adaptation, anticipation, etc. sharing the bacterial cell some similarities with higher organisms (Westerhoff et al., 2014).

Assuming the ideas conjectured by Liberman as well as the plausibility of minimal cognition capabilities in non-neural organisms at the present time synthetic biology technologies have opened promising opportunities to program cells or single-cell organisms in order to enable them to perform tasks for which 'intelligence' is required (Gargantilla et al., 2021; Ortiz et al., 2021). These technologies have made it possible to program bacterial behavior (Guiziou et al., 2019) by incorporating synthetic DNA into bacteria with sequences that are sensitive to external signals controling through enzyme synthesis, the activation or inhibition of certain genes. The design of cellular behaviors in bacteria through artificial genetic circuits has even allowed the control of the spatial organization of the bacteria and their population density, synchronization between bacteria and finally the creation of artificial ecosystems (Kong et al., 2014). Indeed one of the most significant achievements in synthetic biology has been the design of artificial communities or consortia (McCarty and Ledesma-Amaro, 2018), formed by programmed bacteria. Only recently (Eetemadi and Tagkopoulos, 2019) it has been established a parallelism between genetic circuits and neural networks. In the field of microbiology it is known that communication between bacteria leads to the formation of colonies or more complex societies such as biofilms where their individuals communicate with each other via quorum sensing and even between distant biofilms through electrical signals (Majudmar and Pal, 2017). These kinds of societies have been seen as resembling the neural networks present in higher organisms. At present, it has been experimentally observed that bacterial colonies may have memory through the presence of membrane potentials, supporting to the plausibility of a parallelism between bacteria and neurons.

Artificial intelligence is a field of computer science aimed at

designing algorithms where a computer performs tasks in which a human being would use its intelligence (Lahoz-Beltra, 2004). In the 1940s, the first models of artificial neural networks were proposed (McCulloch and Pitts, 1943), a class of models inspired by biological neural circuits. According to Nesbeth et al. (2016) genetic circuits and enzymes are able to support adaptive behaviors, and certain kinds of learning. For instance, Pavlov's associative learning could be implemented with synthetic genetic networks. Previously Armitage et al. (2005) in an article entitled 'Neural networks in bacteria: Making connections' proposes a holistic approach inspiring the design of models that aid to understand how bacteria respond to changes. Moreover, Lissek (2017) conducted a theoretical study in which he coined the term 'neurogenomic computers' and conjectures how nucleic acids could implement the activity of neural circuits. Using elementary synthetic genetic circuits (Nesbeth et al., 2016) showed how it is possible to build a simple perceptron, a memory or to simulate associative learning. However, in their work they did not encode the proposed genetic circuits in a plasmid DNA sequence. In order to implement an artificial neural network within a bacterium, the neural circuit should be translated into a genetic circuit encoding the latter into a DNA sequence. This possibility is shown by Lissek (2017) and Qian et al. (2011) illustrating how the technique known as 'DNA strand displacement cascades' could be used to design linear threshold gates and with the latter a Hopfield neural network. In this same direction Kim et al. (2004) also devised a Hopfield network but in this case using transcriptional circuits *in vitro*.

One of the goals and novelties of the current work is the proposal of a protocol (Fig. 1) which makes it possible to endow a bacterium with a perceptron neural network. Nowadays, synthetic biology gives us some clues about how to design such a protocol. According to Macia and Solé (2014) if we establish a parallelism between electronic and genetic circuits one of the difficulties that arises is the 'wiring problem'. Although it is already known how to implement in a genetic circuit the components of a digital circuit (e.g. switch, oscillator, memory, state machine, logic gates, etc.) the difficulties arise when we want to connect these components together. This problem does not exist when designing an electronic circuit, being a restricting factor in the design of genetic circuits. A possible solution is based on the use of microbial consortia (Shong et al., 2012), thus a community of programmed synthetic cells, e.g. bacteria, with the output resulting from the distributed and parallel computation of all cells in the consortium. An advantage of this procedure is that we add the intercellular space, i.e. the medium in which the cells live and multiply, to the information processing carried out inside a cell (Macia et al., 2016). For this purpose it is possible to design a microbial consortium using two possible strategies, either by controlling the input and output of the genetic circuit via quorum sensing (Tamsir et al., 2011) or alternatively by distributing the function of a genetic circuit among the different cells by applying a method referred to as 'distributed multicellular computation' (Macia and Sole, 2014).

Recently Li et al. (2021) by using the ideas described above designed a microbial consortium in which sender and receiver bacteria through quorum sensing emulate a perceptron neural network capable of 3x3-bit pattern recognition.

In the present work the implementation of a perceptron neural network has been carried out by another route. We embedded a neural network inside a single bacterial cell and not in a microbial consortium as discussed above. Once the cell embeds a perceptron the bacterium will over time multiply resulting in a colony of cells programmed with the neural network.

In this paper, as mentioned above, we have already explored the plausibility of programming a perceptron neural network in synthetic bacteria. We conducted two simulation experiments for different purposes. First, we show how a colony of bacteria endowed with a perceptron is able to solve an optimization problem *in silico*. The goal of this experiment was to learn how to program a perceptron neural network in a cellular programming language, specifically in Gro language (Jang et al., 2012; Oishi and Klavins, 2014; Gutiérrez et al., 2017), as well as to evaluate the advantages and disadvantages of our methodology applied on experiments *in silico*. Secondly, we study by means of simulation experiments how a perceptron could be used to program behaviors in bacteria. In these experiments we observed a remarkable feature: programming bacteria with different behavioral patterns led to the emergence of social interactions and the formation of complex consortia or communities. Consequently, and although the set-up and objective of our simulation experiments are different to those described above in the field of synthetic biology with microbial communities, our experiments conducted to observations and findings related to the aforementioned experiments. In our opinion microbial communities of programmed bacteria and regardless of the method by which they have been programmed would have many and varied applications in biotechnology. Finally, we go one-step further, studying how the aforementioned perceptron designed to program bacterial behaviors could be engineered into a genetic circuit. Once the genetic circuit was obtained, we show how using Cello - a tool for the design of genetic circuits - the perceptron could be translated into a plasmid.

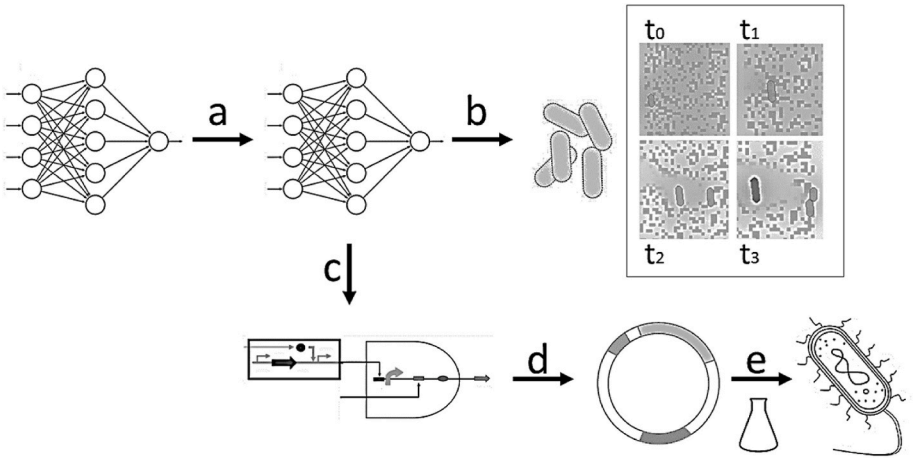

Fig. 1. Experimental protocol for the programming of synthetic bacteria.- First stage: (a) Neural network *offline* training. A neural network is trained by adjusting the weights of the connections between the neurons in the network to appropriate values. The result is that given an input, e.g. 0100, the network will provide a response or output that is the desired one, e.g. 1001. (b) *In silico* simulation experiment (try out). Once the neural network has been trained, the network is endowed inside bacterial cell by writing a script with Gro language. In this way the synthesized bacteria are programmed, studying their behavior, outputs, etc. that a bacterial colony exhibits at different times t_1, t_2, t_3, …t_i.- Second stage: (c) Design of a genetic network. If the results of the simulations work as expected, then the neural network is translated into a genetic network whose performance is similar to the neural network. (d) Implementation of the neural network in a plasmid. Using the Cello platform we encoded in Verilog language the genetic network, obtaining the plasmid. (e) Bacterial transformation is carried out for experimentation in the wet-lab. Stages a-c are performed using a computer, being therefore simulation experiments or *in silico* tasks, and are the stages that we have studied in this work. Bacteria in a future study could be programmed with the plasmid, observing how their behaviors reproduce those observed in the simulation experiments (b).

Our work shows how using principles and tools of synthetic biology in combination with artificial intelligence models it is possible to design *in silico* bacteria with *artificial bio-intelligence*.

2. Materials and methods

As previously described, we performed a first simulation experiment in order to implement and tune the protocol proposed in Fig. 1.

The experimental protocol comprises two stages (Fig. 1). Once the neural network is trained *offline*, the classifier or trained neural network will be embedded inside the bacterial cell, implementing the neural network in a script coded in a cellular programming language named Gro. Gro 4.0 is a cell programming language (Jang et al., 2012; Oishi and Klavins, 2014; Gutiérrez et al., 2017) oriented to the simulation of experiments in synthetic biology. Specifically, the simulator can model different biological phenomena, such as cell division, chemotaxis and signal diffusion, among other physiological phenomena. An extended version of Gro has recently been published allowing the design and recombination of plasmids (Gutiérrez et al., 2017). Consequently, by using this language the synthetic bacteria can be programmed with an artificial neural network. For instance, using this cellular programming language it is possible to endow a synthesized bacterium with the Lac operon, antibiotic resistance and to simulate some details of the chemotaxis mechanism (Gargantilla Becerra and Lahoz-Beltra, 2020).

The Gro language scripts used to program the synthetic bacteria used in the experiments described in this paper can be downloaded from (Lahoz-Beltra and Gargantilla Becerra, 2021).

2.1. Bacterial programming in the PHB optimization problem

In the first simulation experiment we tested the ability of a colony of bacteria to solve an optimization problem. The purpose of the subsequent experiments was to evaluate if the programmed bacteria – i.e. to test whether or not the neural network with the connections already adjusted - were able to solve an optimization problem. The motivation for choosing an optimization problem was because we assumed that it was a suitable scenario to detect the pros and cons of programming bacteria with a perceptron. Consequently, if unexpected bacterial behaviors were observed in the simulations then the programming of the perceptron should be revised. The first stage (Fig. 1a and b) was the only step required to implement the present experiment.

We implemented a feed-forward neural network (FFNN) model, with the aim to optimize the production of polyhydroxybutyrate (PHB) according to the problem described in a previous study (Zafar et al., 2012). However, it is important to note that our aim was not to replicate the above study in all its details and procedures, but to attempt reproducing with programmed bacteria the main results of the optimization experiment cited above. The neural network was designed as is shown in Fig. 2. The values x_1, x_2 and x_3 of the input neurons represent the percentage of molasses and the concentrations of urea and propionic acid respectively. The values of urea and propionic acid were assigned randomly within the experimental intervals used by Zafar et al. (2012) and with the percentage of molasses then normalized before being incorporated into the neural network. To this end, a variable is assigned to each artificial neuron in the intermediate or hidden layer receiving the values of the inputs, using the hyperbolic tangent transfer function. Once the input data, i.e. molasses, urea and propionic acid, were processed by percpetron, we obtained the value y of the output neuron, i.e. the value of the normalized PHB concentration.

In the simulation experiment, and after the topology of the FFNN model has been set up (Fig. 2), we selected the bias values b and bh and conducted the training *offline* of the network adjusting the weights of the network connections by applying the perceptron learning rule (Lahoz-Beltra, 2004). As a result of network training, we obtained a matrix of the adjusted connection weights. Secondly, the weight matrix and the topology of the neural network were translated to a Gro language script.

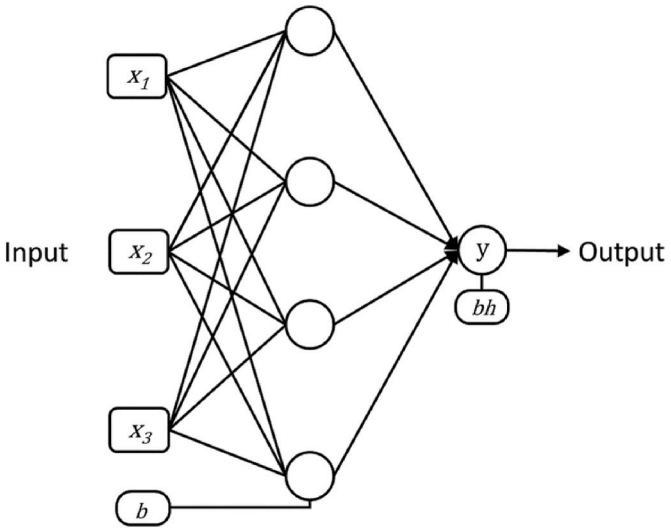

Fig. 2. Perceptron neural network implemented in the synthetic bacteria solving the PHB production optimization problem. In the input layer x_1, x_2 and x_3 represent the percentage of molasses and the concentrations of urea and propionic acid respectively, while in the output neuron y is the concentration of PHB. The parameters b and bh are the bias values in the input and output neurons.

The Appendix describes how a perceptron neural network was coded in Gro language in the bacterium.

The simulation experiments aimed to test the ability of the synthetic bacteria colony to calculate the optimal PHB production under different conditions, particularly the percentage of molasses as well as different concentrations of urea and propionic acid. Once the simulation experiments were conducted the response surfaces were obtained.

2.2. Programming bacterial behavior

A second simulation experiment was carried out but instead of programming the bacteria to solve an optimization problem, the bacteria were programmed to exhibit a repertoire of different behaviors in response to environmental signals. Unlike the previous experiment, the simulation experiments performed now required the use of all the steps shown in Fig. 1. One of the main novelties of the present experiment was to design a genetic circuit that will perform in a manner that is similar to the trained peceptron. In addition, a further step in the experiment is proposed, since by applying Cello platform (Nielsen et al., 2016) the genetic circuit was encoded in a plasmid. Presumably, if a bacterium were transformed with the plasmid its behavior should be close to the behavior observed in the simulation experiments conducted *in silico*. It is important to note that at present we have not conducted experiments in the *wet-lab*.

In the present model, bacteria are able to detect four input signals and to manifest four possible behaviors. Both inputs and outputs were chosen on the basis of their biological plausibility and in order to design a synthetic bacterium that would behave as a sensory-motor agent. The first input signal is galactose (S_{Gal}), which acts as a chemotactic substance for bacteria in the colony. The second is oxygen (S_{O_2}) and the third is the autoinducer AI-2 (S_{AI-2}). This autoinductor is a furanosyl borate diester that is a signal molecule involved in quorum sensing, i.e. AI-2 induces colony gene expression in response to population density. The fourth selected input signal was arabinose (S_{Ara}) which will be the carbon source available to bacteria. Depending on whether one or more of these four signals are present (1) or absent (0), synthetic bacteria may respond with one of these four cellular responses: aerobic response (R_{O_2}), motility or chemotaxis (R_{Mot}), commensalism (R_{Com}) and the stress response or resistance (R_{Str}).

In the event that the bacterium exhibits aerobic adaptation (R_{O_2}) the bacterium emits a signal proportional to the concentration of oxygen in the environment. This signal uses the oxygen present for metabolic purposes. In addition, the growth rate of the bacterium is also set to the environmental oxygen concentration, and consequently the growth not only is regulated by the main carbon source. Now, if the bacterium shows motility (R_{Mot}) or chemotaxis then the cell moves in the direction of a distant signal, i.e. galactose, in order to colonize a possible new ecological niche. Another possible behavior is the commensalism (R_{Com}). In this case and for the purpose of feeding the cell emits a signal that degrades the waste products from other bacteria in the colony. Finally, when the bacteria display a behavior of stress (R_{Str}) or resistance then the cell enters a state of latency in which it stops growing, waiting for better conditions to develop.

A training set was defined. According to this training set different binary strings represented different types of environmental or input signals: presence or absence of galactose, oxygen, AI-2 and arabinose signals, respectively. Once the perceptron was trained *offline*, the matrix of the adjusted connection weights and the topology of the neural network were translated to a Gro language script. The training set describes the programmed behavior that we expect to observe in synthetic bacteria (Table 1). Hence, once the perceptron is trained with Table 1 the mapping of inputs to outputs will be stored in the values of the connection weights adjusted with the perceptron learning rule. Consequently, given a certain input vector, i.e. environmental signals, the behavior of the bacteria will be the result of the processing of the input vector or product of the adjusted weights matrix by this vector:

$$\begin{pmatrix} R_{O_2} \\ R_{Mot} \\ R_{Com} \\ R_{Str} \end{pmatrix} = \begin{pmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \\ w_{40} & w_{41} & w_{42} & w_{43} \end{pmatrix} \begin{pmatrix} S_{Gal} \\ S_{O_2} \\ S_{AI-2} \\ S_{Ara} \end{pmatrix}$$

It is interesting to note that the neural network would play the regulatory role of genetic networks in a real microorganism. In the simulation experiments, we will program synthetic bacteria to develop a commensalistic relationship among themselves, and consequently we expect the synthetic microcolonies will develop patterns that are characteristic of this type of bacterial communities.

In order to ensure that the different responses are not mutually incompatible, a perceptron was designed with an input layer of four neurons and an output layer with four other artificial neurons (Fig. 3). The algorithm governing the try out phase of perceptron was similar to the previous experiment (Fig. 2), as well as the perceptron training phase adjusting the synaptic weights in order to program behaviors in bacteria. The activation function is a step function, carrying an all-or-nothing response in which the action threshold θ was set to 0.5 and the learning rateα was equal to 0.1. Also a detection threshold was introduced for environmental signals which will determine whether a signal will be detected or not by the bacterium. This threshold was set at a value of 0.15 for all environmental signals. Thus, when the synthetic bacterium received a signal with an intensity higher than threshold then the input value of the signal for perceptron is equal to 1. Otherwise, the input value was 0 for a signal value lower than the threshold. In addition, the perceptron input values were updated every 0.5 s, so the output and therefore the behavior of the bacterium was sensitive to the variations of the signals in different locations and times of the artificial environment. The overall scheme of the Gro simulation is depicted in Fig. 4. The general sections of the Gro script have been described in the Appendix.

The conducted simulation experiments involved changes in the intensity of the two environmental signals - oxygen (S_{O_2}) and arabinose (S_{Ara}), i.e. the carbon source -, which have a major influence on the

Table 1
Bacterial behavior training table.

Binary Input	Input Signals	Binary Output	Output Responses
0000	–	0001	R_{Str}
0001	S_{Ara}	0000	–
0010	S_{AI-2}	0011	R_{Com} R_{Str}
0011	S_{AI-2} S_{Ara}	0000	–
0100	S_{O_2}	1001	R_{O_2} R_{Str}
0101	S_{O_2} S_{Ara}	1000	R_{O_2}
0110	S_{O_2} S_{AI-2}	1011	R_{O_2} R_{Com} R_{Str}
0111	S_{O_2} S_{AI-2} S_{Ara}	1000	R_{O_2}
1000	S_{Gal}	0101	R_{Mot} R_{Str}
1001	S_{Gal} S_{Ara}	0000	–
1010	S_{Gal} S_{AI-2}	0111	R_{Mot} R_{Com} R_{Str}
1011	S_{Gal} S_{AI-2} S_{Ara}	0000	–
1100	S_{Gal} S_{O_2}	1101	R_{O_2} R_{Mot} R_{Str}
1101	S_{Gal} S_{O_2} S_{Ara}	1000	R_{O_2}
1110	S_{Gal} S_{O_2} S_{AI-2}	1111	R_{O_2} R_{Mot} R_{Com} R_{Str}
1111	S_{Gal} S_{O_2} S_{AI-2} S_{Ara}	1000	R_{O_2}

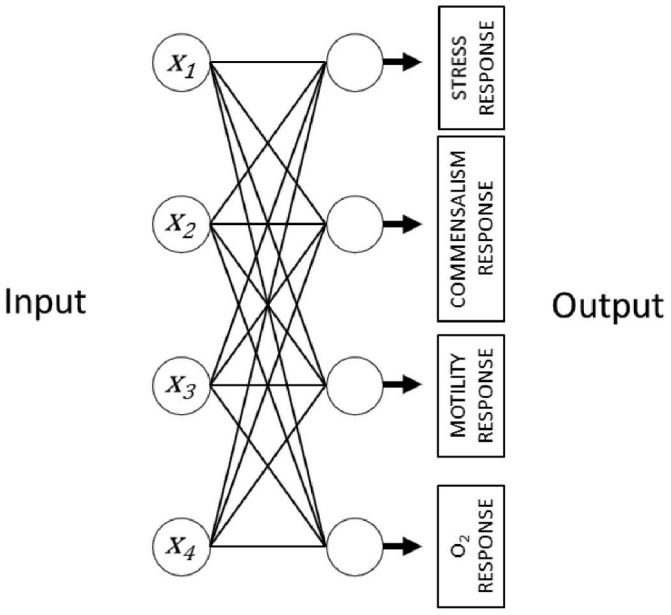

Fig. 3. Perceptron neural network implemented in synthetic bacteria to program their behavior in response to environmental signals. In the input layer x_1, x_2, x_3 and x_4 represent the presence or absence of galactose, oxygen, AI-2 and arabinose respectively, while in the output layer there are four neurons whose activation triggers a particular bacterial response.

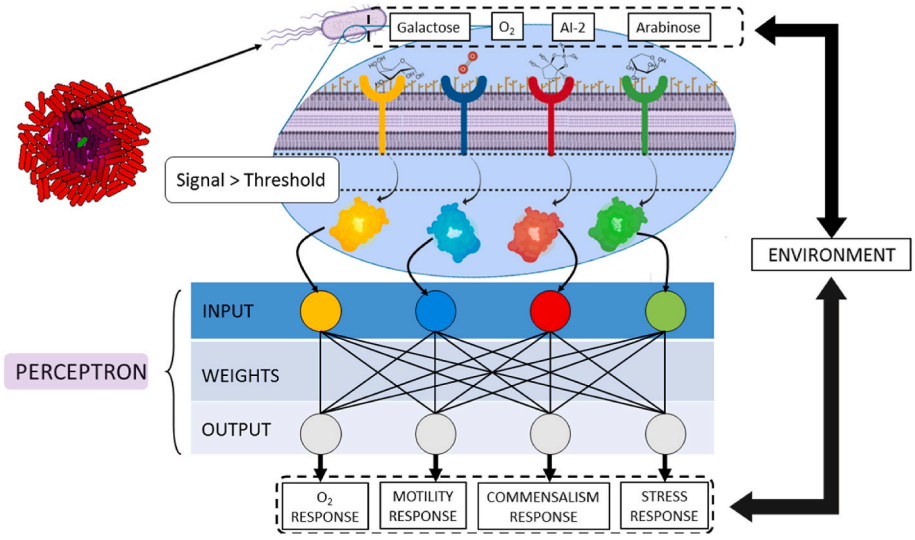

Fig. 4. Bacterial coupling between the receptor system of environmental signals and the perceptron neural circuitry. When the value of an external signal exceeds a threshold value then the input value in the corresponding neuron of the perceptron input layer is 1, otherwise 0. Once the values of the neural circuit input layer have been processed, an output vector is obtained in the output layer, which encodes the behavioral responses of the bacteria.

bacterial colony. For instance, we can simulate an anaerobic environment setting the oxygen signal at a level equal to 0 (S_{O_2} =0) or an aerobic environment setting the signal level $S_{O_2} = 1$. Arabinose signal S_{Ara} was also simulated in above environments controlling the amount of carbon source available in the simulated environment. Values of S_{Ara} equal to 20, 12.5 and 5 were selected for rich, standard and poor environments respectively. According to this conditions six different environments were simulated. Bacteria were programmed to express different reporter proteins and thereby emitting different kinds of fluorescence. In particular, bacteria receiving arabinose as the carbon source signal expressed the green fluorescent protein (GFP) whereas those receiving AI-2, galactose or oxygen expressed the red (RFP), yellow (YFP) and cyan (CFP) fluorescent proteins respectively.

Next, and once the simulation experiments with bacteria programmed in Gro language have been carried out, we go a step further by exploring how to obtain a plasmid whose genetic expression will result in behaviors similar to those programmed with the perceptron neural network.

Finally, we designed a plasmid whose expression inside real bacteria would foreseeably result in the behaviors observed in the previous simulation experiments. However, we did not conducted any laboratory or *wet-lab* experiments.

The possibility of designing a plasmid encoding the responses of a bacterium to the environmental changes has been addressed according to the following procedure:

First, we used Cello platform (Nielsen et al., 2016) developed by MIT (http://v1.Cello.org/), translating the training table (Table 1) to Verilog language (Fig. 5). Note that Table 1 was used to train the perceptron, i.e. to adjust the connection weights in the above simulation experiments. The translation consists in specifying the number of inputs in the genetic network, assigning a specific position in a binary string to each input, meaning the values 1 and 0 the presence or absence of each input respectively. From these specifications, Cello generates a genetically encoded digital circuit replicating in our case the perceptron neural

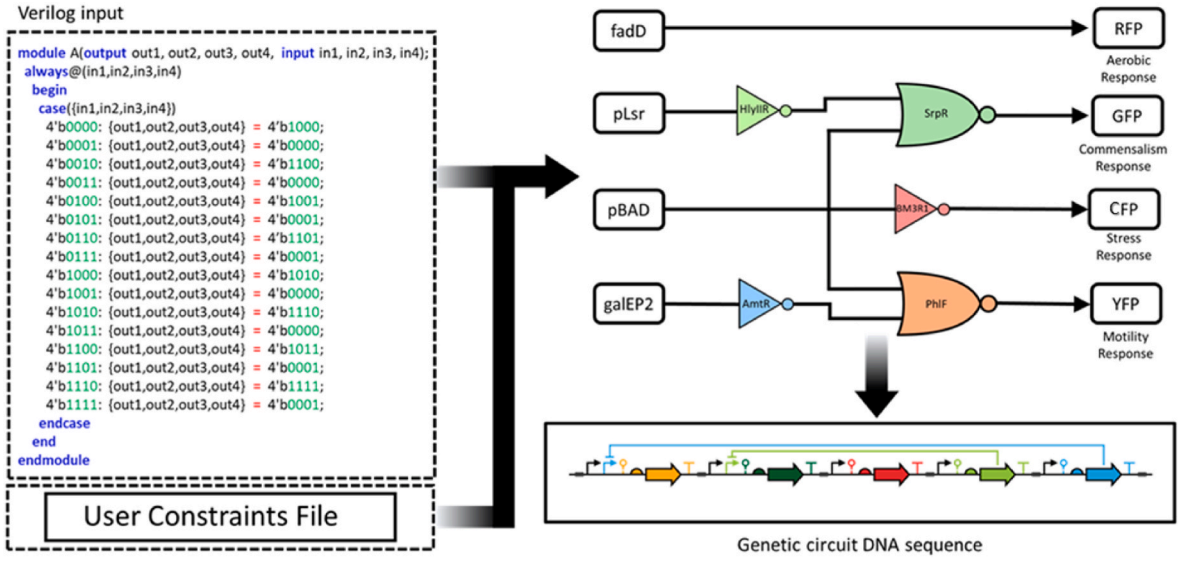

Fig. 5. Script written in Verilog language and its equivalent logic diagram showing the role played by each of the regulatory regions in the response or output of the perceptron. The logic operators have been labeled with the names of the corresponding regions of the regulatory plasmid. Note how the only independent response is the one that corresponds to the fluorescent protein RFP, while all other perceptron responses or outputs are influenced by pBAD.

network.

Secondly, in order to design the plasmid we must include the genetic sequences of the sensors for all environmental signals. The procedure requires that the molecular machinery regulating these sensors be available in *E.coli*. In addition, it must be fulfilled that the molecular machinery has to react in a positive manner, i.e. by triggering transcription once the presence of the target molecules is detected. These target molecules are the different input signals, i.e. galactose, oxygen, AI-2 and arabinose. Note that in above simulation experiments we implemented the input signals as code in the Gro script written to program synthetic bacteria. The following promoters present in *E.coli* were used as sensors:

- fadD region: models the presence of oxygen (Feng et al., 2012).
- pBAD promoter: represents the presence of arabinose, i.e. the sugar that has been chosen as a carbon source.
- pLsr: is part of the quorum sensing system used by *E.coli* in response to the self-inducing molecule-2 (AI-2). In our model it represents the detection of the presence of another bacterium (Abisado et al., 2018; Hauk et al., 2016; Xue et al., 2009).
- galEP2: models the presence of galactose. Galactose was chosen as a chemotactic signal in the simulation experiments conducted above with synthetic bacteria programmed in Gro language (Chilcott and Hughes, 2000; Roggo et al., 2019; Weickert and Adhya, 1993).

The molecular mechanisms regulating the response of the selected promotors relate the presence of the target molecules, i.e. galactose, oxygen, AI-2 and arabinose, with the transcription of the reporter genes which express the synthesis of four different fluorescent proteins (Fig. 5): RFP, GFP, CFP and YFP.

Finally, Cello output is the sequence of a plasmid that if inserted into *E. coli* would program the bacterium to replicate the responses or behaviors observed in the previous simulation experiments in a colony of synthetic bacteria. In other words, the transformation of bacteria with the plasmid would cause bacteria to behave intelligently in response to environmental changes, exhibiting the bacteria predictable behavioral patterns.

In summary, in this section we have described how to design a perceptron that embedded in a bacterium would program its behavior in the style of a senso-motor agent. Simulation experiments are the result of implementing the trained perceptron and its topology in the appropriate Gro language script (Appendix). Finally, we consider the first steps concerning how the perceptron could be replaced by an equivalent genetic network that is encoded in a plasmid. To this end, the neural network was translated into a genetic network using the Verilog language, replacing the input neurons by promoters and the output neurons by the expression of fluorescent reporter proteins. Finally, in order to consider other aspects that in the future would allow us to make the leap from *in silico* to wet-lab experiments, the following section describes how we have addressed the simulation of the microbial environment.

2.3. Simulation of the microbial environment

In the simulation experiments (Section 2.2) we included a model of the environment, e.g. the Petri dish holding the growth medium in which synthetic bacteria could be cultured. The features of the environment were also included in the Gro script. According to Fig. 6 the environmental signals interact with those emitted by the synthetic bacteria, showing normal, commensalistic and aerobic energetic metabolism. In the conducted experiments, the performance of the energy metabolism was set to 20%, although this could be changed according to empirical data obtained in the laboratory. Therefore, the behavior exhibited by the bacteria will be influenced by the spatiotemporal distribution and intensity of the environmental signals.

3. Results

The obtained results show the possibility of implementing *in silico* a trained artificial neural network in a colony of synthetic bacteria. The results obtained in the first experiment opens the possibility of solving optimization and classification problems by using programmed bacterial colonies. In the second experiment we have shown how a bacterium could be programmed to behave as if it were a sensory-motor agent. In this paper we also present a way to translate a neural network into an

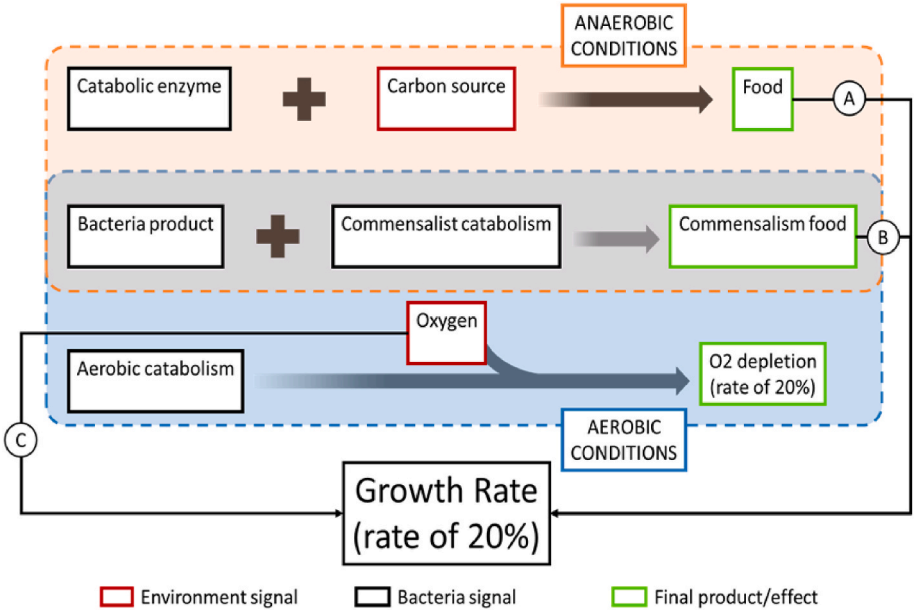

Fig. 6. Environmental model that simulates the environment in which bacterial growth takes place in the experiments. Note how, depending on environmental signals, a bacterium can exhibit three kinds of metabolism: normal (**A**), commensalistic (**B**) and aerobic (**C**) energetic metabolism (for explanation see text).

equivalent genetic circuit and its coding in a plasmid. In our opinion, the design in the *wet-lab* of bacteria programmed with neural networks by means of synthetic biology techniques will allow in the future the treatment of cancerous tumors, the design of probiotic foods or the application of intelligent bacteria oriented to the treatment of bioremediation problems.

3.1. PHB optimization problem

In the first simulation experiment, we programmed the bacteria with a perceptron neural network to solve a PHB production optimization problem. The results obtained show how a colony (Fig. 7) of synthetic bacteria successfully solved the optimization problem. Fig. 8 shows the response surfaces of the problem for different input values of molasses, urea and propionic acid. In general, the interdependence of the three substrates involved in the bioprocess are observed and their optimum values can be properly estimated. Thus, the optimum percentage of molasses occurs in the range 4.0–4.5%, the concentration of urea around 0.5 g/L and the concentration of propionic acid between 10 and 15 mM.

3.2. Programming bacterial behavior

In a second simulation experiment, we programmed bacteria with neural networks in order to evaluate their ability of responding with different behaviors to different environmental signals and the formation of microbial consortiums resulting from commensalistic relationship between bacteria. According to Fig. 9 we can conclude that synthetic bacteria are differentiated into three classes. Firstly, there are bacteria that emit green fluorescence (GFP) when receiving mainly arabinose, secondly there are bacteria that exhibit red fluorescence (RFP) when receiving mostly the AI-2 signal produced by other bacteria, and finally there are bacteria emitting cyan fluorescence (CRP) when receive mainly oxygen. We must also include those synthetic bacteria that do not receive any signal and/or are in their latent form of resistance, and which are shown in black in the experiments. An intermediate phenotype is also distinguished between the cyan and green bacteria, which are the bacteria receiving both oxygen and the carbon source. In addition, and although not observed in Fig. 9, during the simulation experiments there were bacteria that emitted yellow fluorescence when receiving galactose, i.e., the chemotactic signal. In addition, during the simulation, other intermediate phenotypes of bacteria were observed depending on the signals from the simulated environment.

In the conducted experiments, we observed two different patterns of microbial consortiums. The first pattern is a consequence of the dependence between the number of bacteria and the amount of available carbon source, i.e. arabinose. In this case we observed something that is obvious, and in particular how the size of the colony decreased as food became scarce. The second pattern is due to the presence of aerobic bacteria in the media when oxygen is present. In general, we can conclude that in a rich nutrient media synthetic bacteria interact with each other and with the simulated environment. The result of these interactions is that colonies form niches. For example, in an aerobic and rich nutrient environment some bacteria with cyan fluorescence occupy the region where oxygen is present, growing and clustering in that region. Likewise, other bacteria, e.g. those that emit green fluorescence, do not have to adapt to oxygen and only depend on arabinose, i.e. the carbon source, to survive. In the surrounding area a third population of red-fluorescing bacteria will grow, which are not directly reached by the carbon source. Interestingly, and because of the commensalism response, the red bacteria can survive in the neighborhood of the green-fluorescing bacteria. However, as the red-fluorescing bacteria feed on arabinose and move away from the green-fluorescing bacteria, they soon enter a latent state of resistance ending their growth.

On the other hand, in a rich nutrient and anaerobic medium it is observed how the colony of bacteria emitting green fluorescence, i.e. detecting arabinose, grows around the carbon source. In addition, we observe how in its surroundings the bacteria mostly acquire the red fluorescence phenotype of commensalism after receiving the AI-2 signal. We also observed the generation of a layer of latent (black bacteria) or resistant bacteria around the colony.

These findings suggest not only that synthetic bacteria respond adequately to perceptron programming, but also that such programming in a collective context leads to the coexistence of microbial patterns resembling those observed in real microbial colonies. According to Boetius et al. (2000) when bacteria establish commensalistic relationships then essential nutrient gradients are induced. The obtained results open the possibility of implementing trained neural networks within real bacterial colonies, resulting in the bacteria a mapping between environmental signals and observable behaviors. In fact, this has been suggested previously by establishing a parallelism between synaptic mechanisms between neurons and some forms of bacterial communication (Ram and Lo, 2018).

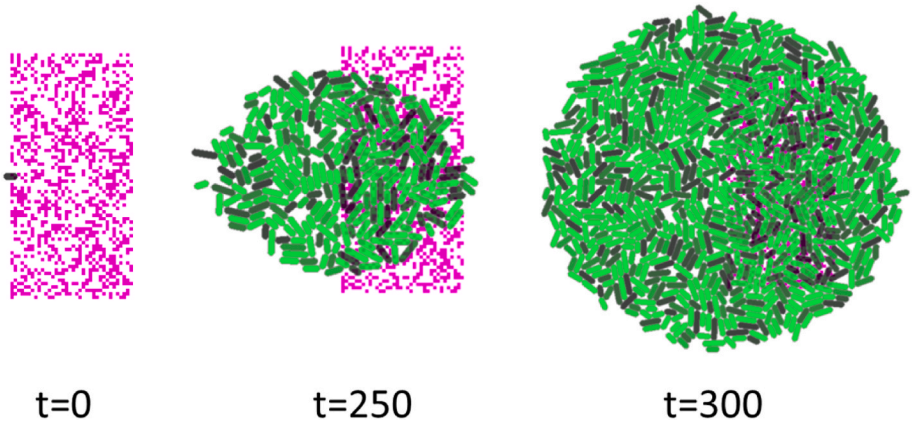

Fig. 7. PHB optimization problem. Exponential growth of the synthetic bacterial colony from initial time ($t = 0$), showing the microbial colony size at 250 and 300 min (simulated time). In the experiment each bacterium receives as input to its perceptron neural network different values of molasses, urea and propionic acid (not displayed in the simulation). Once the input is processed a certain concentration of PHB is obtained, emitting green fluorescence through the synthesis of the green fluorescent protein (GFP) reporter protein. The higher the fluorescence, the closer the PHB value estimated by the perceptron is to its optimal concentration. In red is shown the antibiotic present in the medium which selects those bacteria with molasses, urea and propionic acid values within the tolerable range of concentrations (for an explanation see Appendix). Although the search for the optimal PHB is conducted in parallel, there is no communication between bacteria, e.g. via conjugation, and therefore each cell is an autonomous agent independent of the other agents of the colony.

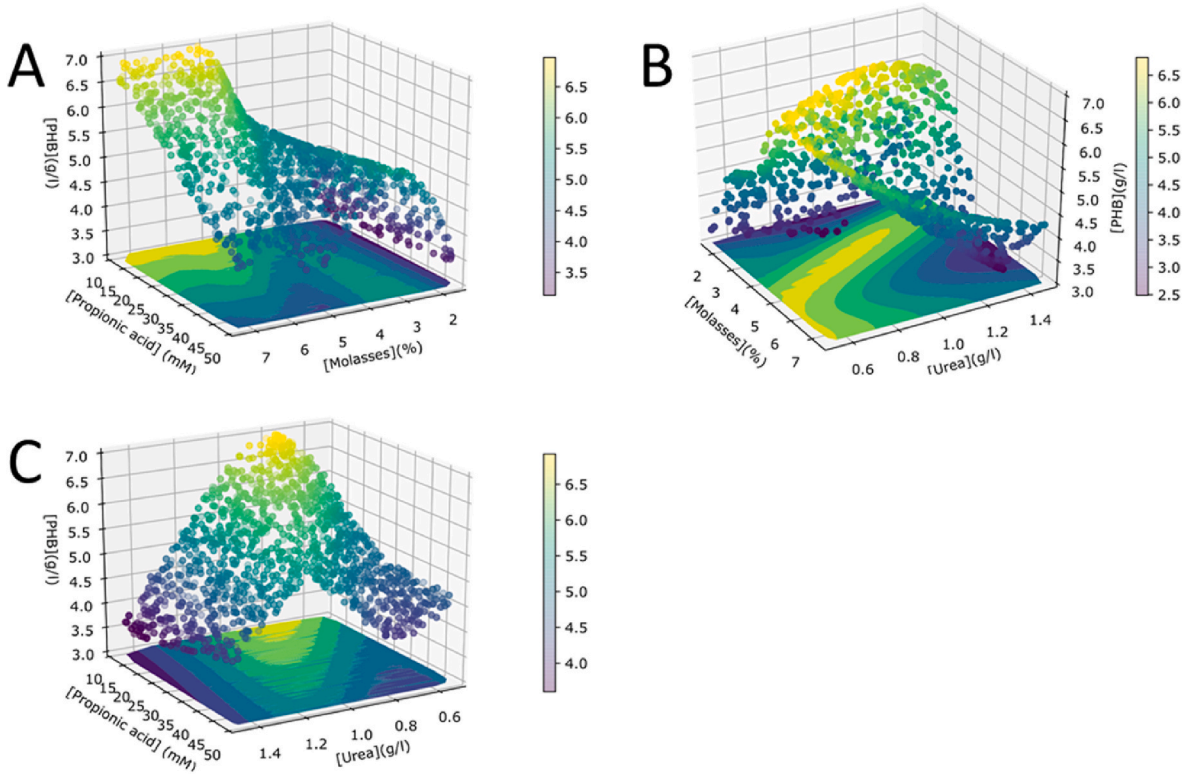

Fig. 8. Response surfaces of the PHB production model in three experiments with the variables urea (**A**), propionic acid (**B**) and molasses (**C**) maintained constant. The outputs of the neural network trained for the prediction and optimization of PHB production has been depicted on one of the axes of the figures.

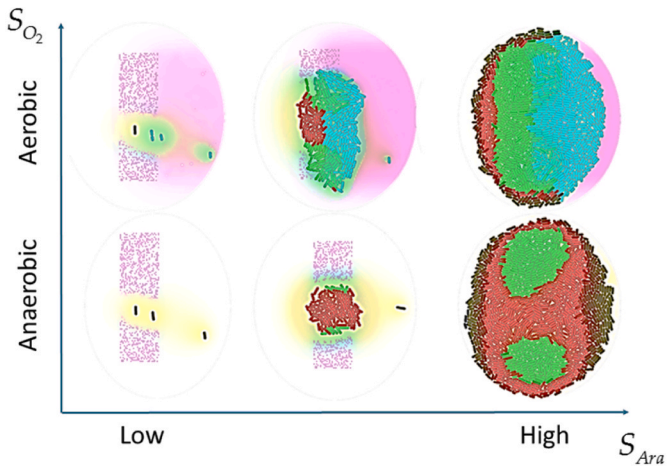

Fig. 9. Bacterial colony patterns. On the *x*-axis the richness of the carbon source increases from left to right while on the *y*-axis the oxygen levels increase from bottom (anaerobic) to top (aerobic).

3.3. Encoding the perceptron in a plasmid

Finally, we designed a plasmid (Fig. 10) coding the regulatory model of the perceptron neural network from the above *in silico* experiment. In the case of the plasmid (Lahoz-Beltra and Gargantilla Becerra, 2021) were inserted into *E. coli* presumably it would program the bacteria with the behaviors previously simulated in the synthetic bacterial colony. That is, due to the inserted plasmid the transformed bacteria would respond intelligently to environmental changes according to the commensal patterns described above (Fig. 9). In consequence, the fluorescence emitted by a bacterium would be given by the expression patterns depicted in Fig. 11.

It is important to note that the connection weights in the perceptron neural network reflect the Boolean gates of the logic circuit governing the genetic network (Fig. 12). In the perceptron, a positive weight promotes a bacterial response while a negative weight is equivalent to NOT gate.

According to Fig. 12 in the logic circuit of the genetic network the activation of the aerobic response only depends on the fadD sensor. On the other hand, if we consider the weights of the perceptron connections governing the aerobic response, we will conclude that the sum of the weights, i.e. 0.1 + 0.2+0.1, excluding the oxygen signal weight (0.6), is below the threshold of the output neuron (0.5). However, the weight of the oxygen signal connection (0.6) alone is enough to exceed the activation threshold. This reasoning allows us to reach an important conclusion, the Boolean logic circuit of the genetic network and the perceptron connections are functionally analogous.

Likewise, Fig. 12 shows how the commensalism response is regulated by the pLsr sensor through a NOT and NOR gates as well as by the pBAD sensor via a NOR gate. In this case, in the perceptron the only relevant weight in the motility response is realted to pLsr signal (0.6). Note that when the pBAD signal is received there is no response since the sum of the weights, i.e. 0.1–0.3 + 0.1, is below the threshold value (0.5).

An analysis of the stress response (Fig. 12) shows how the logic circuit is regulated by only one NOT gate related with the pBAD sensor signal, which means that as long as the gate is active, no stress response will be observed in the bacterium.

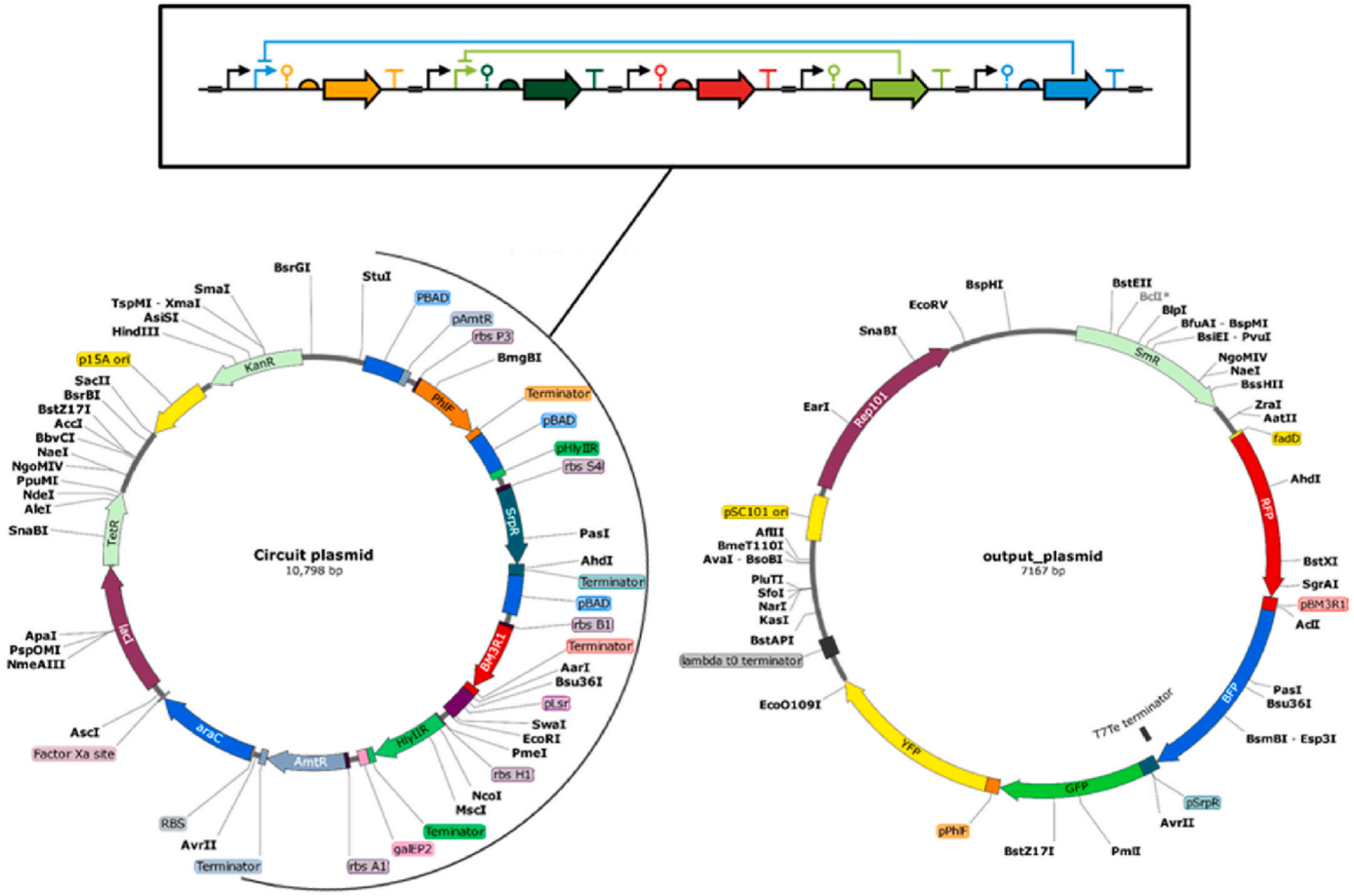

Fig. 10. Implementation of the perceptron neural network in two plasmids [22] obtained with Cello. (Top) Diagram of the gene regulation. (Bottom left) Circuit plasmid. Map of the regulatory plasmid showing the sequence involved in the regulation of the genetic circuit. Regions of the regulatory plasmid are sensitive to the molecules chosen as input signal. (Bottom right) Output plasmid. Plasmid map displaying the sequence of the fluorescent proteins or output. (NOTE: In the plasmids, the cyan CFP fluorescent protein was labeled as 'blue' BFP).

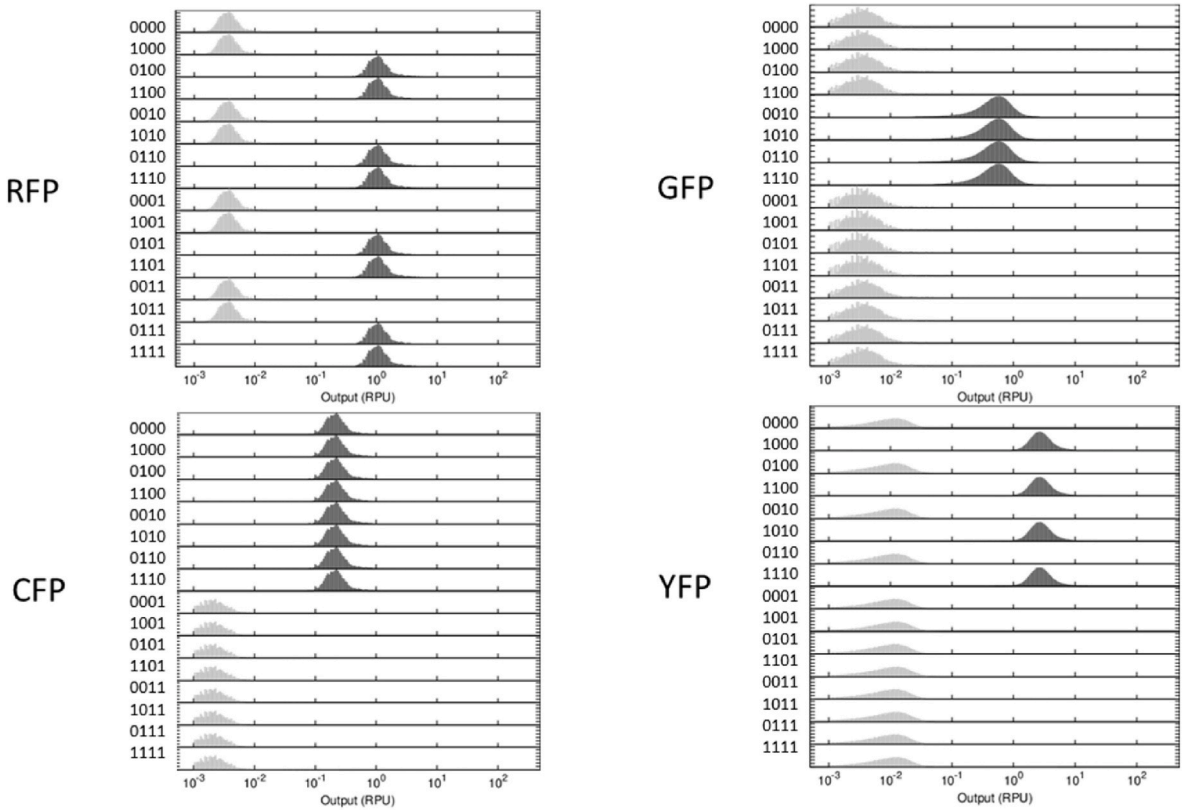

Fig. 11. Expression pattern of the fluorescent proteins RFP, GFP, CFP and YFP or perceptron output in response to a given input. The input is encoded by a binary string galEP2-fadD-pLsr-pBAD in which 0/1 indicates the absence/presence of the molecules representing the input signal.

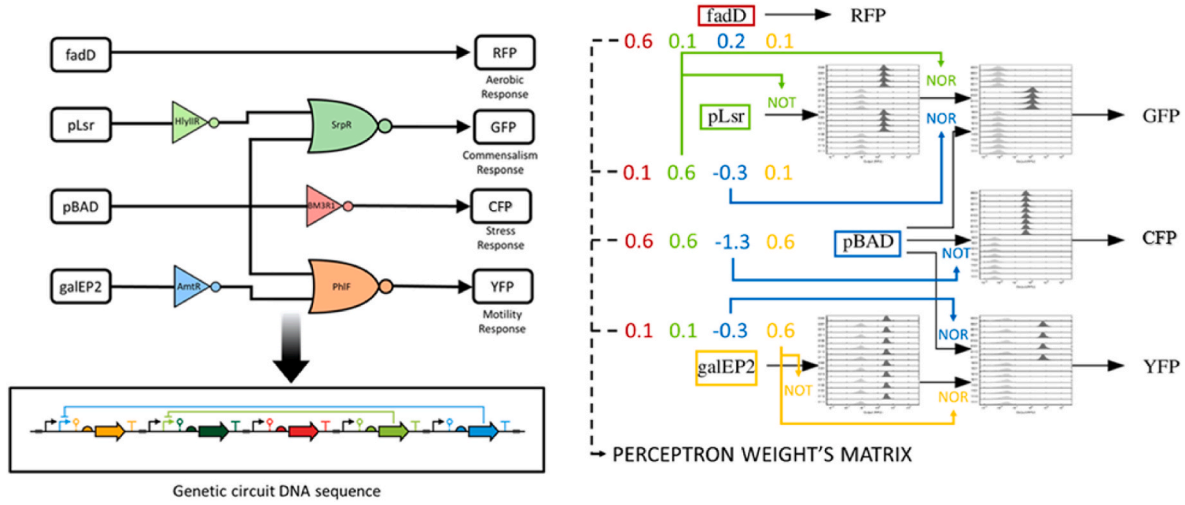

Fig. 12. Correspondence between the Boolean gates of the logic circuit and the connection weights of the perceptron neural network (for explanation see text).

Finally, an analysis of the response to motility (Fig. 12) suggests the following. The logic network is regulated by the signal coming from the galEP2 sensor, passing through a NOT and a NOR gate, and by the signal from the pBAD sensor, passing through a NOR gate. In this case, in the perceptron is observed that the only relevant connection weight in the motility response is the one corresponding to the galEP2 signal. However, if the pBAD signal is received at the same time, the motility response would not take place since the sum of the weights (0.1 + 0.1–0.3) is below the threshold value (0.5).

These results show as the Boolean logic (Buchler et al., 2003; Moon et al., 2012) allows us to establish a correspondence between artificial neural networks and genetic networks. For this reason, it is possible to obtain by means of bioinformatics tools, e.g. Cello, the genetic sequence of the plasmid which is the functional counterpart of the perceptron

coded in the Gro script.

In the next section, we will further discuss what this correspondence between artificial neural networks and genetic networks means and entails.

4. Discussion

The present work is an example that illustrates on a practical level the ideas introduced by Liberman on a theoretical level. Moreover, disruptive notions such as 'a cell is a computer making computers' (Danchin, 2009) is shown in the simulation experiments. Engineered bacterial agents are able to perform intelligent tasks and to multiply resulting intelligent bacterial agents, propagating that intelligence generation after generation.The possibility of designing bacterial agents with genetic circuits that emulate neural networks through synthetic biology techniques opens up numerous practical possibilities in biotechnology. In this work, by means of simulation experiments we have explored two possible approaches displaying how to apply these 'smart' bacteria to solve practical problems. On the one hand, we have shown how a colony of bacteria with a trained perceptron can solve an optimization problem, in this case the production of PHB from different amounts of molasses, urea and propionic acid. Of course, this same method could be applied to the production of other biomaterials (Moradali and Rehm, 2020), such as a collagen (Wang et al., 2017; Avila Rodríguez et al., 2018) phosphoethanolamine cellulose (Thongsomboon et al., 2018; Hollenbeck et al., 2018), or synthetic spider silk (Edlund et al., 2018) for instance.

The question remains whether a readily trained network should be used for programming the cells, or if it would be perhaps more general to have the network train within the cells. The training of the network inside the bacterial cell would be a procedure equivalent to online learning in machine learning, i.e. the learning or adjustment of the weights of the neural network is incremental as new data are being incorporated. In this case the update of the network would take place as the bacterium evolves in a certain environment. In contrast, when, as we have done in this work, we provide the bacteria with a trained neural network, the approach is similar to offline or batch training in machine learning. That is, the neural network has been trained with a whole static data set. In our work the disadvantage of training the neural network by this procedure outside the bacterium is that once the classifier, i.e. the neural network, is incorporated inside the bacterial cell the learning is not continuously updated with new information from environment. The result is that the training to which the neural network was subjected may even become obsolete over time. However, the advantage of offline network training is that if we define a cost function with which we evaluate the learning, and therefore the adjustment of the weights in the network, once the network is trained the cost function will converge to a minimum. This fact ensures a correct programming of the synthetic bacteria, and therefore that the bacteria will behave properly according to how they have been programmed. Conversely, if we were to include the neural network in the untrained bacterium and assume that as the bacterium performs its tasks, online training takes place, the programming of the bacterium would occur naturally over time. Nevertheless, it could happen that the learning would never be completed properly and the bacterium would not be adequately programmed.

In the computer science scenario where artificial neural networks now are subject of extensive research, a good practice is to first use a large amount of resources for training, but then use less resources with a fixed, trained network (Li et al., 2020). Even hardware with different capacities is produced separately for training and inferring with these networks. Characterization of newtork resource use therefore becomes a key factor (Canziani et al., 2016). Thus, while a network for training and inferring has a more general purpose, it uses large amounts of resources (which is a major problem that needs to be solved in bacteria, as metabolical burden is a defining factor in the success of synthetic gene circuits). This is why our team chose to propose a mixed approach in which training is done at a computational level and later the already trained network is implemented into synthetic gene circuits for their siumlation. Another reason for this separation is that, we simulate bacteria with a trained perceptron in order to endow the bacteria with specific stereotyped behaviors, i.e. we have programmed them in order to govern the responses of the bacteria to different signals such as galactose, oxygen, the autoinducer AI-2 and arabinose in our example: defining a directed trained behavior produces more expectable and verifiable results to assess how well the proposed approach works.

Feed forward neural networks can approximate any differentiable function (Yayla et al., 2021; Cybenko, 1989). In this work, we have demonstrated how to map these neural networks to simulation specifications with the help of a cell programming language, Gro and a design automation software, Cello. Therefore, the power of this language pipeline should be briefly analyzed: first, the Cello language can represent any digital logic function as a set of genetic logic gates (which in fact, are genetic networks). This design is then translated to a Gro specification file in which the organization and logic are preserved. Therefore, the final form of the circuit is expressed as a composition of logic gates, implying that any logic function may be implemented in Gro. Since Cello uses NAND gates, that are functionally complete (Davidson, 1968), to implement genetic circuits, the whole spectrum of logic functions can be specified. Gro uses Boolean protein states (expressed/unexpressed) to implement corresponding genetic circuits. In the simulation, these states are stable and respond to inputs in order to change their values. Thus, they would account for storage capability. However, this capacity is not unlimited, as requested per a Turing Machine. In sum, Gro would be capable of simulating memory bounded Turing machines (Ben-Hur and Siegelmann, 2004). Within the context of biological computing, cell programming languages offer an open and general manner of representing computational algorithms. It does not add computing power over other classes of encoding such as DNA Computing (Adleman, 1998) or P-Systems (Paun, 2000). Nonetheless, it provides a more control-oriented and rational way of engineering biological systems due to repositories of characterized parts (McLaughlin et al., 2018; Wang et al., 2021) and standards for exchanging such designs among researchers (McLaughlin et al., 2020).

Simulation experiments demonstrate how bacterial responses lead to the formation of complex consortia or communities and commensalistic-like interactions between bacteria. At present, we know that these bacterial clusters underlie the formation of bacterial biofilms, which play a fundamental role in the pathogenesis of diseases (Vestby et al., 2020) caused by bacteria and in the resistance of bacteria to antibiotics. Furthermore, these 'smart' bacteria will act as suitable microorganisms for the design of microbial synthetic consortia for use in bioprocesses to produce medicines, biofuels or biomaterials (McCarty and Ledesma-Amaro, 2018), due to them being able to adapt to external changes either within a bioreactor or in a patient's organ or tissue (Nesbeth et al., 2016). It is important to note that if our work would have been conducted adopting a different approach, for example by not considering DNA as the molecule in which to encode the neural network, the final result would have been different. For example, Lyon (2015) considers that a bacterial cell contains itself all of the necessary elements for information processing. In this case, it is assumed that bacterial signaling networks are capable of integrating different features of information processing, such as sensory-motor, physiological,

chemosensory and communication functions, providing an adaptive response to an external stimulus. Thus, under this view, signal transduction in a bacterium would perform a function similar to that of a neural network. In fact, the idea that proteins constitute networks analogous to neural circuits is not new, and was proposed some time ago (Di Paola et al., 2004).

In sum, for the same reason that there are different approaches for designing microbial communities in biotechnology, there are also different strategies for implementing neural networks in bacteria. For example, there are proposals (Sarkar et al., 2020) for neural network architectures in which it is assumed that each bacterium would operate as an artificial neuron or 'bactoneuron'. In such work, the authors design a 2-to-4 decoder and a 2-to-1 demultiplexer based on the use of a network composed by bactoneurons. Other possibilities replace neural networks by genetic circuits regulated via quorum sensing systems (McCarty and Ledesma-Amaro, 2018). Therefore, the design of intelligent bacteria through genetic circuits that emulate at a logical level a neural network can be approached either through DNA computation or by plasmid coding techniques. In fact, the use of plasmids has been one of the classic approaches in bacterial computing models (Head et al., 2000; Wang et al., 2018).

An interesting feature of the present work has been the availability of several languages with which to approach the programming of bacterial cells, a possibility that would have been unthinkable years ago. At a first stage, languages such as Gro allow us to get a first impression of how an artificial neural network might operate inside a bacterial cell. Using Gro we can write a script with the individual features of the neural network already trained as well as the main features of the synthetic bacteria cell. An interesting issue of the simulation is the parallel running of the neural network in the bacterial colony. Thus, the tryout stage of the neural network is executed simultaneously by all bacteria, noting the emergence of collective behaviors such as commensalism. In a second step, based on Verilog language—a language initially oriented to the programming of computer chips—we wrote a program in the Cello platform with which to translate the logic of the Gro script into logic gates and sensors, that will ultimately be encoded into a couple of plasmids to transform *E. coli* cells. This transformation step is important for two reasons: the first one is obvious and refers to the translation of the neural network logic into a genetic language. However, the second one stems from this translation: expressivity and some extent of explainability are two important features that are gained.

As mentioned above, Cello software maps the abstract design of a Gro simulation into a design made of digital logic gates. Hence, the very fact that it is possible to carry out the mapping of any simulation into logic gate designs determines that a high level of expressivity is achieved: an equivalent digital logic circuit is always found through Cello. This entails that any neural network represented as in this work can be translated and expressed in terms of genetic circuits.

An important problem in artificial neural networks is that they act as a black box, and it is difficult to study their internal structure (Oh et al., 2019; Castelvecchi, 2016). This approach therefore sheds new light on a possible way of simplifying and interpreting a trained network. In other words, the intricate mesh of weights obtained in the training phase of the neural network is interpreted as a set of logic gates (Snyder and Vishwanath, 2020; Choi et al., 2019), thus adding to how, from a global standpoint, the trained network can be rewritten in terms of logical operations. It should be stressed that this logic-oriented explanatory format represents the final trained network, not the process itself or the individual network weights. Considering the underlying logic gate representation, there is no need to understand the meaning of the network weights. Instead, the (logical) meaning of the whole trained neural network can be easily followed and interpreted.

The above described approach summarizes the procedure we have followed in this work, based on synthetic biology methods and programming in two languages oriented to this discipline, in particular Gro and Verilog. However, other languages have been proposed (Pedersen and Phillips, 2009; Umesh et al., 2010; Nowogrodzki, 2018; Spaccasassi et al., 2019) to approach cell programming. Consequently, it is reasonable to assume that some of these programming languages will lead to other experimental protocols different from the present one.

In the future, advances in synthetic biology and the use of heuristic algorithms from computer science (Gargantilla Becerra and Lahoz-Beltra, 2020; Gargantilla Becerra et al., 2021; Ortiz et al., 2021) will make it possible to endow bacteria with artificial intelligence. In this way it will be proved that Liberman's seminal idea that a cell is a computer goes beyond a simple metaphor or analogy. This fact will open the possibility of designing colonies with engineered bacteria with which it will be possible to solve a variety of problems from medicine to bioremediation, the production of biofuels as well as the most diverse applications in biotechnology.

Conflicts of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Acknowledgements

We thank to Pedro Marijuan for his helpful comments and suggestions.

Appendix

In the Gro simulator the bacteria are programmed by default to divide in a medium according to Malthusian or exponential growth kinetics. From a single bacterium begins the division and growth of the colony until it reaches the specified maximum size. Therefore, if no script is included in the initial bacterium, i.e. the bacterium is not programmed, all that will be observed is the growth of the colony without the bacteria running any program. Following we show the script written in Gro cellular programming language, implementing the neural network trained to solve the optimization PHB production problem.

In the script, (**a**) we bound the ranges of the percentage of molasses and the different concentrations of urea and propionic acid. It is important to note that the input values of urea and propionic acid in the neural network have been obtained randomly and therefore must be within the tolerable concentration ranges defined in (**a**). This condition is evaluated with an **evaluation** function (below **c**). Bacteria that do not meet the above condition are killed with Gro's command **die()** by the antibiotic present in the medium (Gargantilla Becerra and Lahoz-Beltra, 2020). Otherwise, those bacteria that meet the above condition acquire resistance to the antibiotic and the concentration of PHB is evaluated by means of the neural network. Therefore, the antibiotic selects the appropriate ranges of the perceptron input (Zafar et al., 2012), i. e. the molasses, urea and propionic acid values, while the perceptron is used to estimate the PHB value. (**b**) Settings of the *offline* adjusted weights and the values assigned to the bias. (**c**) Input values of molasses, urea and propionic acid are collected by genes gene[0], gene[1] and gene[2] respectively. (**d**) Input neurons. Note how with the **norm** function the values of gene[0], gene[1] and gene[2] are normalized before they are input to the neural network. (**e**) Intermediate or hidden neurons and (**f**) output neuron. Following we obtain from the value of the output neuron the concentration of PHB. The concentration of PHB is expressed by

the emission of green fluorescence in the bacteria, i.e. through the synthesis of the green fluorescent protein (GFP) reporter protein.

```
include gro

//EXPERIMENTAL SET UP

molasses_range:={1.50,7.50};
urea_range:={0.50,1.50};
propionic_acid_range:={10.00,50.00};
phb_range:={2.5,6.75};

// OFFLINE ADJUSTED WEIGHTS (w) AND BIAS (b)VALUES

w_i_1:={-0.8761,-0.1543,0.4184};
w_i_2:={-9.2423,2.1559,0.5502};
w_i_3:={-1.2167,-2.2256,0.3593};

w_h_1:={-1.1652,0.4563,0.6264};
w_h_2:={0.4272,-0.2029,-0.6274};
w_h_3:={2.3147,0.4147,2.5255};
w_h_4:={-3.0707,-0.3121,-0.2564};

w_o:={ -0.9643,-1.6782,-0.5076,0.5819};

b_i:={0.9966,0.4541,0.3147};
b_h:={1.2547,-1.6508,1.4789,-1.0983};
b_o:=-0.0358;


//ARTIFICIAL NEURAL NETWORK

program ANN() :={
  gfp:=0;
  t:=0;
  true:{t:= t+dt}
  inputw_i:={0.0,0.0,0.0};
  inputw_h:={0.0,0.0,0.0,0.0};
  y:=0;
  phb:=0;

  daughter : { gene[0]:= 4.29,
               gene[1]:= rand(150)/100.0,
               gene[2]:= rand(5000)/100.0,
               evaluation := comparative (gene[0]) (molasses_range) *
                             comparative (gene[1]) (urea_range) *
                              comparative (gene[2]) (propionic_acid_range)
             }

 daughter & evaluation = 0 & get_signal(antibiotic)>-0.01 :

    { die() }

 daughter & evaluation = 1 :

     { fprint (fp, gene[0]," ",gene[1]," ",gene[2]," "),
       gene[0] := norm (gene[0]) (molasses_range),
       gene[1] := norm (gene[1]) (urea_range),
       gene[2] := norm (gene[2]) (propionic_acid_range),
       inputw_i[0] := purelin((gene[0]*w_i_1[0]) + (gene[1]*w_i_1[1]) +
                      (gene[2]*w_i_1[2])+ b_i[0]),
       inputw_i[1] := purelin((gene[0]*w_i_2[0]) + (gene[1]*w_i_2[1]) +
                      (gene[2]*w_i_2[2]) + b_i[1]),
       inputw_i[2] := purelin((gene[0]*w_i_3[0]) + (gene[1]*w_i_3[1]) +
                      (gene[2]*w_i_3[2]) + b_i[2]),
       inputw_h[0] := tansig((w_h_1[0]*inputw_i[0]) + (w_h_1[1]*inputw_i[1]) +
                      (w_h_1[2]*inputw_i[2]) + b_h[0]),
       inputw_h[1] := tansig((w_h_2[0]*inputw_i[0]) + (w_h_2[1]*inputw_i[1]) +
                      (w_h_2[2]*inputw_i[2]) + b_h[1]),
       inputw_h[2] := tansig((w_h_3[0]*inputw_i[0]) + (w_h_3[1]*inputw_i[1]) +
                      (w_h_3[2]*inputw_i[2]) + b_h[2]),
       inputw_h[3] := tansig((w_h_4[0]*inputw_i[0]) + (w_h_4[1]*inputw_i[1]) +
                      (w_h_4[2]*inputw_i[2]) + b_h[3]),
       y := purelin((w_o[0]*inputw_h[0]) + (w_o[1]*inputw_h[1]) +
                    (w_o[2]*inputw_h[2]) + (w_o[3]*inputw_h[3])+b_o),
       phb := (((phb_range[1]-phb_range[0])*(y-0.1))/0.8)+phb_range[0],
       gfp:=phb*50,
       fprint (fp, phb, "\n" ),
       print(phb," ",gen)
     }


        foreach i in range 1200 do {
          set_signal(antibiotic,rand(180),(rand(360)-180),20)
        } end;
  };

ecoli ( [], program ANN() );
```

a b c d e f g

The Gro script shown below lists the fundamental sections of the code. First, (**h**) the connection weights are adjusted offline by means of the perceptron rule associating to each binary input string an output string (Table 1). Following the neurons of the input layer (**i**) and the neurons of the output layer (**j**) are displayed. Note that depending on the values of the neurons of the output layer (Table 1) the synthetic bacterium exhibits one or another behavior, implementing the different behaviors in section (**k**) of the script.

```
include gro

// Neural network parameters

w_1:={0.1,0.6,0.1,0.2};
w_2:={0.6,0.1,0.1,-0.3};
w_3:={0.1,0.1,0.6,-0.3};
w_4:={0.6,0.6,0.6,-1.3};
threshold:=0.5;

program ANN() :={
  s:=[e:=environment, score:=0.0, c:=behavior];
  gfp:=0;
  rfp:=0;
  yfp:=0;
  cfp:=0;

  t:=0;
  rep:=0;
  count:=0;
  m1:=0;
  m2:=0;
  Y:={};
  movility:= [ num := 0, mean := 0, std := 0 ];

  true:{ t:= t+dt,
         rep:= rep+dt,
       count:=count+dt,
       emit_signal(e,1),
       emit_signal(bac,get_signal(f)),
       set_signal(oxy,rand(400)-100,rand(500)-250, 0)
       }

  t > 0.25 : { t := 0, m1:=m2, m2:= get_signal(d)}

  rep >= 0.5 :{ s.e[0]:=detect(get_signal(d)),
                s.e[1]:=detect(get_signal(oxy)),
                s.e[2]:=detect(get_signal(bac)),
                s.e[3]:=detect(get_signal(f)),

  s.c[0]:= relu((s.e[0]*w_1[0]) + (s.e[1]*w_1[1]) + (s.e[2]*w_1[2]) +
  (s.e[3]*w_1[3])),
  s.c[1]:= relu((s.e[0]*w_2[0]) + (s.e[1]*w_2[1]) + (s.e[2]*w_2[2]) +
  (s.e[3]*w_2[3])),
  s.c[2]:= relu((s.e[0]*w_3[0]) + (s.e[1]*w_3[1]) + (s.e[2]*w_3[2]) +
  (s.e[3]*w_3[3])),
  s.c[3]:= relu((s.e[0]*w_4[0]) + (s.e[1]*w_4[1]) + (s.e[2]*w_4[2]) +
  (s.e[3]*w_4[3])),
  rep := 0
              }

    // Bacterial behavior table
    //

    s.e[0]=0 & s.e[1]=0 & s.e[2]=0 & s.e[3]=0 :{s.c[3]:=1}

    s.c[0]=0 & s.c[1]=1 & s.c[2]=1 & s.c[3]=1 : { colonization m1 m2,
                                                  comensalism (get_signal(bac)),
                                                  resistance (get_signal(f)),

    set("ecoli_growth_rate",get_signal(f_c)/5),
                                                  gfp:=0,
                                                  yfp:=m2*1000,
                                                  rfp:=get_signal(f_c)*1000,
                                                  cfp:=0
                                                       }

    s.c[0]=0 & s.c[1]=0 & s.c[2]=1 & s.c[3]=1 : {comensalism (get_signal(bac)),
                                                 resistance (get_signal(bac)),

    set("ecoli_growth_rate",get_signal(f_c)/5),
                                                  rfp:=get_signal(f_c)*1000,
                                                  yfp:=0,
                                                  gfp:=0,
                                                  cfp:=0
                                                 }

    s.c[0]=0 & s.c[1]=1 & s.c[2]=0 & s.c[3]=1 : {colonization m1 m2,
                                                 resistance (get_signal(f)),
                                                  yfp:=m2*1000,
                                                  rfp:=0,
                                                  cfp:=0,
                                                  gfp:=0
                                                 }
```

References

Abisado, R.G., Benomar, S., Klaus, J.R., Dandekar, A.A., Chandler, J.R., 2018. Bacterial quorum sensing and microbial community interactions. mBio 9 (3), e02331–17. https://doi.org/10.1128/mBio.02331-17.

Adleman, L.M., 1998. Computing with DNA. Sci. Am. 279 (2), 54–61.

Armitage, J.P., Holland, I.B., Jenal, U., Kenny, B., 2005. "Neural networks" in bacteria: making connections. J. Bacteriol. 187 (1), 26–36.

Avila Rodríguez, M.I., Rodriguez Barroso, L.G., Sánchez, M.L., 2018. Collagen: a review on its sources and potential cosmetic applications. J. Cosmet. Dermatol. 17 (1), 20–26.

Baluška, F., Levin, M., 2016. On having no head: cognition throughout biological systems. Front. Psychol. 7 (902) https://doi.org/10.3389/fpsyg.2016.00902.

Baluška, F., Lev-Yadun, S., Mancuso, S., 2010. Swarm intelligence in plant roots. Trends Ecol. Evol. 25 (12), 682–683.

Ben Jacob, E., Shapira, Y.I., Tauber, A., 2006. Seeking the foundations of cognition in bacteria: From Schrödinger's negative entropy to latent information. Physica A Stat. Mech. Appl. 359, 495–524.

Ben-Hur, A., Siegelmann, H.T., 2004. Computation in gene networks. Chaos: Interdiscipl. J. Nonlinear Sci. 14 (1), 145–151.

Boetius, A., Ravenschlag, K., Schubert, C.J., Rickert, D., Widdel, F., Gieseke, A., Amann, R., Jørgensen, B.B., Witte, U., Pfannkuche, O., 2000. A marine microbial consortium apparently mediating anaerobic oxidation of methane. Nature 407 (6804), 623–626.

Buchler, N.E., Gerland, U., Hwa, T., 2003. On schemes of combinatorial transcription logic, 9. In: Proceedings of the National Academy of Sciences, 100, pp. 5136–5141.

Calvo, P., Baluška, F., 2015. Conditions for minimal intelligence across eukaryota: a cognitive science perspective. Front. Psychol. 6 (1329) https://doi.org/10.3389/fpsyg.2015.01329.

Canziani, A., Paszke, A., Culurciello, E., 2016. An analysis of deep neural network models for practical applications. arXiv 1605, 7678.

Castelvecchi, D., 2016. Can we open the black box of AI? Nature News 538 (7623), 20.

Chilcott, G.S., Hughes, K.T., 2000. Coupling of flagellar gene expression to flagellar assembly in *Salmonella enterica Serovar Typhimurium* and *Escherichia coli*. Microbiol. Mol. Biol. Rev. 64 (4), 694–708.

Choi, A., Shi, W., Shih, A., Darwiche, A., 2019. Compiling neural networks into tractable Boolean circuits. In: Proc. AAAI Spring Symposium on Verification of Neural Networks (VNN).

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Mathematics of Control. Signals Syst. 2 (4), 303–314.

Danchin, A., 2009. Bacteria as computers making computers. FEMS Microbiol. Rev. 33, 3–26.

Davidson, E.S., 1968. An Algorithm for NAND Decomposition of Combinational Switching Functions. University of Illinois at Urbana-Champaign.

Di Paola, V., Marijuan, P.C., Lahoz-Beltra, R., 2004. Learning and evolution in bacterial taxis: an operational amplifier circuit modeling the computational dynamics of the prokaryotic 'two component system' protein network. Biosystems 74 (1–3), 29–49.

Edlund, A.M., Jones, J., Lewis, R., Quinn, J.C., 2018. Economic feasibility and environmental impact of synthetic spider silk production from *Escherichia coli*. New Biotechnol. 42, 12–18.

Eetemadi, A., Tagkopoulos, I., 2019. Genetic neural networks: an artificial neural network architecture for capturing gene expression relationships. Bioinformatics 35 (13), 2226–2234.

Feng, Y., Cronan, J.E., 2012. Crosstalk of *Escherichia coli* FadR with global regulators in expression of fatty acid transport genes. PLoS One 7 (9), e46275. https://doi.org/10.1371/journal.pone.0046275.

Gagliano, M., Renton, M., Depczynski, M., Mancuso, S., 2014. Experience teaches plants to learn faster and forget slower in environments where it matters. Oecologia 175, 63–72.

Gargantilla Becerra, A., Lahoz-Beltra, R., 2020. A microbial screening in silico method for the fitness step evaluation in evolutionary algorithms. Appl. Sci. 10 (3936) https://doi.org/10.3390/app10113936.

Gargantilla Becerra, A., Gutiérrez, M., Lahoz-Beltra, R., 2021. A synthetic biology approach for the design of genetic algorithms with bacterial agents. Int. J. Parallel Emergent Distrib. Off. Syst. 37, 1–18. https://doi.org/10.1080/17445760.2021.1879072.

Guiziou, S., Mayonove, P., Bonnet, J., 2019. Hierarchical composition of reliable recombinase logic devices. Nat. Commun. 10 (456) https://doi.org/10.1038/s41467-019-08391-y.

Gutiérrez, M., Gregorio-Godoy, P., Pérez Del Pulgar, G., Muñoz, L., Sáez, S., Rodriguez-Patón, A., 2017. A new improved and extended version of the multicell bacterial simulator gro. ACS Synth. Biol. 6, 1496–1508.

Hauk, P., Stephens, K., Mckay, R., Virgile, C.R., Ueda, H., Ostermeier, M., Ryu, K.S., Sintim, H.O., Bentley, W.E., 2016. Insightful directed evolution of *Escherichia coli* quorum sensing promoter region of the lsrACDBFG operon: a tool for synthetic biology systems and protein expression. Nucleic Acids Res. 44 (21), 10515–10525.

Head, T., Rozenberg, G., Bladergroen, R.S., Breek, C.K.D., Lommerse, P.H.M., Spaink, H. P., 2000. Computing with DNA by operating on plasmids. Biosystems 57, 87–93.

Hollenbeck, E.C., Antonoplis, A., Chai, C., Thongsomboon, W., Fuller, G.G., Cegelski, L., 2018. Phosphoethanolamine cellulose enhances curli-mediated adhesion of uropathogenic *Escherichia coli* to bladder epithelial cells. Proc. Natl. Acad. Sci. Unit. States Am. 115 (40), 10106–10111.

Jang, S.S., Oishi, K.T., Egbert, R.G., Klavins, E., 2012. Specification and simulation of synthetic multicelled behaviors. ACS Synth. Biol. 1, 365–374.

Kim, J., Hopfield, J.J., Winfree, E., 2004. Neural network computation by *in vitro* transcriptional circuits. In: NIPS'04 Proceedings of the 17th International Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, 2004. MIT Press, Cambridge, MA, USA, pp. 681–688.

Kong, W., Celik, V., Liao, C., Hua, Q., Lu, T., 2014. Programming the group behaviors of bacterial communities with synthetic cellular communication. Bioresour. Bioprocess. 1, 24. https://doi.org/10.1186/s40643-014-0024-6.

Lahoz-Beltra, R., 2004. Bioinformática: Simulación, Vida Artificial e Inteligencia Artificial. In: Transl.:Spanish., Ediciones Díaz de Santos. Spain, Madrid, pp. 327–342.

Lahoz-Beltra, R., Gargantilla Becerra, A., 2021. Programming of bacterial behavior with a plasmid encoding a perceptron neural network. figshare. Software. https://doi.org/10.6084/m9.figshare.14399141.v3.

Lahoz-Beltra, R., Navarro, J., Marijuan, P., 2014. Bacterial computing: a form of natural computing and its applications. Front. Microbiol. 5 (101) https://doi.org/10.3389/fmicb.2014.00101.

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., Gonzalez, J., 2020. Train big, then compress: rethinking model size for efficient training and inference of transformers. In: International Conference on Machine Learning, PMLR, pp. 5958–5968.

Li, X., Rizik, L., Kravchik, V., Khoury, M., Korin, N., Danie, R., 2021. Synthetic neural-like computing in microbial consortia for pattern recognition. Nat. Commun. 12, 3139.

Liberman, E.A., The cell as a molecular computer (m.c.)-I, 1972. General ideas and hypotheses. Biofizika 17 (5), 932–943.

Liberman, E.A., 1979. Analog-digital molecular cell computer. Biosystems 11, 111–124.

Liberman, E.A., Minina, S.V., 1996. Cell molecular computers and biological information as the foundation of nature's laws. Biosystems 38, 173–177.

Lissek, T., 2017. Interfacing neural network components and nucleic acids. Front. Bioeng. Biotechnol. 5 (53) https://doi.org/10.3389/fbioe.2017.00053.

Lyon, P., 2015. The cognitive cell: bacterial behavior reconsidered. Front. Microbiol. 6 (264) https://doi.org/10.3389/fmicb.2015.00264.

Macia, J., Solé, R., 2014. How to make a synthetic multicellular computer. PLoS One 9 (2), e81248. https://doi.org/10.1371/journal.pone.0081248.

Macia, J., Manzoni, R., Conde, N., Urrios, A., de Nadal, E., Solé, R., Posas, F., 2016. Implementation of complex biological logic circuits using spatially distributed multicellular consortia. PLoS Comput. Biol. 12 (2), e1004685. https://doi.org/10.1371/journal.pcbi.1004685.

Majumdar, S., Pal, S., 2017. Bacterial intelligence: imitation games, time-sharing, and long-range quantum coherence. J. Cell Commun. Signal. 11 (3), 281–284.

McCarty, N.S., Ledesma-Amaro, R., 2018. Synthetic biology tools to engineer microbial communities for biotechnology. Trends Biotechnol. 37 (2), 181–197. https://doi.org/10.1016/j.tibtech.2018.11.002.

McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. 5, 115–133.

McLaughlin, J.A., Myers, C.J., Zundel, Z., Mısırlı, G., Zhang, M., Ofiteru, I.D., Goñi-Moreno, A., Wipat, A., 2018. SynBioHub: a standards-enabled design repository for synthetic biology. ACS Synth. Biol. 7 (2), 682–688.

McLaughlin, J.A., Beal, J., Mısırlı, G., Grünberg, R., Bartley, B.A., Scott-Brown, J., Vaidyanathan, P., Fontanarrosa, P., Oberortner, E., Wipat, A., Gorochowski, T., Myers, C.J., 2020. The Synthetic Biology Open Language (SBOL) version 3: simplified data exchange for bioengineering. Front. Bioeng. Biotechnol. 8, 1009.

Mitchell, A., Romano, G.H., Groisman, B., Yona, A., Dekel, E., Kupiec, M., Dahan, O., Pilpel, Y., 2009. Adaptive prediction of environmental changes by microorganisms. Nature 460 (7252), 220–224.

Moon, T.S., Lou, C., Tamsir, A., Stanton, B.C., Voigt, C.A., 2012. Genetic programs constructed from layered logic gates in single cells. Nature 491 (7423), 249.

Moradali, M.F., Rehm, B.H.A., 2020. Bacterial biopolymers: from pathogenesis to advanced materials. Nat. Rev. Microbiol. 195–210. https://doi.org/10.1038/s41579-019-0313-3.

Nagaki, T., Yamada, H., Toth, A., 2000. Maze-solving by an amoeboid organism. Nature 407, 470.

Nesbeth, D.N., Zaikin, A., Saka, Y., Romano, M.C., Giuraniuc, C.V., Kanakov, O., Laptyeva, T., 2016. Synthetic biology routes to bio-artificial intelligence. Essays Biochem. 60, 381–391.

Nielsen, A.A.K., Der, B.S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E.A., Ross, D., Densmore, D., Voigt, C.A., 2016. Genetic circuit design automation. Science 352 (6281), aac7341. https://doi.org/10.1126/science.aac7341.

Nowogrodzki, A., 2018. Gene design goes automatic. Nature 564, 291–292.

Oh, S.J., Schiele, B., Fritz, M., 2019. Towards reverse-engineering black-box neural networks. In: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer, Cham, pp. 121–144.

Oishi, K., Klavins, E., 2014. A framework for implementing finite state machines in gene regulatory networks. ACS Synth. Biol. 3, 652–665.

Ortiz, Y., Carrión, J., Lahoz-Beltra, R., Gutiérrez, M., 2021. A framework for implementing metaheuristic algorithms using intercellular communication. Front. Bioeng. Biotechnol. 9 https://doi.org/10.3389/fbioe.2021.660148.

Păun, G., 2000. Computing with membranes. J. Comput. Syst. Sci. 61 (1), 108–143.

Pedersen, M., Phillips, A., 2009. Towards programming languages for genetic engineering of living cells. J. R. Soc. Interface 6, S437–S450.

Qian, L., Winfree, E., Bruck, J., 2011. Neural network computation with DNA strand displacement cascades. Nature 475, 368–372.

Ram, A., Lo, A.W., 2018. Is smaller better? A proposal to use bacteria for neuroscientific modeling. Front. Comput. Neurosci. 12 (7) https://doi.org/10.3389/fncom.2018.00007.

Roggo, C., Carraro, N., van der Meer, J.R., 2019. Probing chemotaxis activity in *Escherichia coli* using fluorescent protein fusions. Sci. Rep. 9 (1), 3845. https://doi.org/10.1038/s41598-019-40655-x.

Salek, M.M., Carrara, F., Fernandez, V., et al., 2019. Bacterial chemotaxis in a microfluidic T-maze reveals strong phenotypic heterogeneity in chemotactic sensitivity. Nat. Commun. 10, 1877.

Sarkar, K., Bonnerjee, D., Bagh, S., 2020. A single layer artificial neural network with engineered bacteria. arXiv 2001, 00792v1. https://arxiv.org/abs/2001.00792v1.

Shong, J., Jimenez Diaz, M.R., Collins, C.H., 2012. Towards synthetic microbial consortia for bioprocessing. Curr. Opin. Biotechnol. 23 (5), 798–802.

Snyder, C., Vishwanath, S., 2020. Deep networks as logical circuits: generalization and interpretation. arXiv preprint arXiv 2003, 11619.

Spaccasassi, C., Lakin, M.R., Phillips, A., 2019. A logic programming language for computational nucleic acid devices. ACS Synth. Biol. 8, 1530–1547.

Tamsir, A., Tabor, J.J., Voigt, C.A., 2011. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. Nature 469 (7329), 212–215.

Thongsomboon, W., Serra, D.O., Possling, A., Hadjineophytou, C., Hengge, R., Cegelski, L., 2018. Phosphoethanolamine cellulose: a naturally produced chemically modified cellulose. Science 359 (6373), 334–338.

Trewavas, A., 2016. Intelligence, cognition, and language of green plants. Front. Psychol. 7 (588) https://doi.org/10.3389/fpsyg.2016.00588.

Umesh, P., Naveen, F., Rao, C.U.M., Nair, A.S., 2010. Programming languages for synthetic biology. Syst. Synth. Biol. 4, 265–269.

Vestby, L.-K., Grønseth, T., Simm, R., Nesse, L.L., 2020. Bacterial biofilm and its role in the pathogenesis of disease. Antibiotics (Basel) 9 (2), 59. https://doi.org/10.3390/antibiotics9020059.

Wang, T., Lew, J., Premkumar, J., Poh, C.L., Naing, M.W., 2017. Production of recombinant collagen: state of the art and challenges. Eng. Biol. 1 (1), 18–23.

Wang, X., Zheng, P., Ma, T., Song, T., 2018. Small universal bacteria and plasmid computing systems. Molecules 23(6) 1307. https://doi.org/10.3390/molecules23061307.

Wang, B., Yang, H., Sun, J., Dou, C., Huang, J., Guo, F.B., 2021. BioMaster: an integrated database and analytic platform to provide comprehensive information about BioBrick parts. Front. Microbiol. 12, 4.

Weickert, M.J., Adhya, S., 1993. The galactose regulon of *Escherichia coli*. Mol. Microbiol. 10 (2), 245–251.

Westerhoff Hans, V., Brooks Aaron, N., Simeonidis, Evangelos, García-Contreras, Rodolfo, He, Fei, Boogerd Fred, C., Jackson Victoria, J., Goncharuk, Valeri, Kolodkin, Alexey, 2014. Macromolecular networks and intelligence in microorganisms. Front. Microbiol. 5 (379) https://doi.org/10.3389/fmicb.2014.00379.

Whiting, J.G.H., Jones, J., Bull, L., Levin, M., Adamatzky, A., 2016. Towards a *Physarium* learning chip. Sci. Rep. 6, 19948. https://doi.org/10.1038/srep19948.

Xue, T., Zhao, L., Sun, H., Zhou, X., Sun, B., 2009. LsrR-binding site recognition and regulatory characteristics in *Escherichia coli* AI-2 quorum sensing. Cell Res. 19, 1258–1268.

Yang, C.Y., Bialecka-Fornal, M., Weatherwax, C., Larkin, J.W., Prindle, A., Liu, J., Garcia-Ojalvo, J., Süel, G.M., 2020. Encoding membrane-potential-based memory within a microbial community. Cell Syst. 10 (5), 417–423. https://doi.org/10.1016/j.cels.2020.04.002 e3.

Yayla, M., Günzel, M., Ramosaj, B., Chen, J.J., 2021. Universal approximation theorems of fully connected binarized neural networks. arXiv preprint arXiv 2102, 2631.

Zafar, M., Kumar, S., Kumar, S., Dhiman, A.K., 2012. Artificial intelligence based modeling and optimization of poly(3-hydroxybutyrateco-3-hydroxyvalerate) production process by using Azohydromonas lata MTCC 2311 from cane molasses supplemented with volatile fatty acids: a genetic algorithm paradigm. Bioresour. Technol. 104, 631–641.