Application of Evolutionary Algorithms for the Optimization of Genetic Regulatory Networks

Elise Rosati(✉), Morgan Madec, Abir Rezgui, Quentin Colman, Nicolas Toussaint, Christophe Lallement, and Pierre Collet

ICube Laboratory (Engineering Sciences, Computer Sciences and Imaging Laboratory, UMR 7357), University of Strasbourg/CNRS, 300 boulevard Sébastien Brandt, 67412 Illkirch Cedex 02, France
erosati@unistra.fr

Abstract. Synthetic biology aims at reinvesting theoretical knowledge from various do-mains (biology, engineering, microelectronics) for the development of new bio-logical functions. Concerning the design of such functions, the classical trial-error approach is expensive and time consuming. Computer-aided design is therefore of key interest in this field. As for other domains, such as microelectronics or robotics, evolutionary algo-rithms can be used to this end. This article is a first step in this direction: it describes the optimization of an existing artificial gene regulatory network using evolutionary algorithms. Evolutionary algorithms successfully find a good set of parameters (the simu-lated response of the system which fits at 99 % the expected response) in about 200 s (corresponding to 5000 generations) on a standard computer. This is the proof of concept of our approach. Moreover, results analysis allows the biologist not only to save time during the design process but also to study the specificity of a system.

Keywords: Synthetic biology · EASEA · Gene regulatory networks · Design automation · Biosystems modeling

1 Introduction

Synthetic biology is an emerging field which aims at reinvesting theoretical knowledge acquired during the past decades in biology and the know-how in the design of systems, such as large-scale integrated circuits, autonomous embedded systems (e.g. smartphone) or heterogeneous macro-systems (automotive, robotics, ...). As a consequence, this science is at the interface between biology, biotechnologies, microelectronics and computer science. Synthetic biology is coming of age as it now has many applications in several domains, such as the manufacturing of new low-cost drugs [1], the implementation of Boolean functions with biological material for cancer detection purpose [2], biological sensing [3] or the synthesis and the optimization of bio-fuels [4].

G. Squillero and P. Burelli (Eds.): EvoApplications 2016, Part I, LNCS 9597, pp. 184–200, 2016.
DOI: 10.1007/978-3-319-31204-0_13

Synthetic biology involves several aspects. One of the most interesting for us is not *what you can do* but *how you can do it*. More specifically, the question we try to answer is the following: *is it possible to switch from trial-error design process to virtual prototyping*. In this context, evolutionary algorithms can play an important role for design automation and system-level optimization.

In this paper, focus is put on a specific field of synthetic biology which consists in the design of artificial gene regulatory networks that can be integrated in a living cell (bacteria for instance) so that this network implements a new functionality. Design automation for gene regulatory networks has been widely investigated over the past decade [5,6] and remains a hot topic [7]. One of the most interesting features of such biological systems lies in the fact that they can be described by Boolean relationships at a high level of abstraction [8]. Thus, design methods and associated tools used for digital electronics can be directly applied to synthetic biology. GeNeDA (GEne NEtwork Design Automation) is an example of tools developed upon this principle [9,10]. GeNeDA is based on a digital synthesizer and a technological mapper, initially developed for the implementation of digital function in FPGAs (Field-Programmable Gate Array). These tools have been adapted to the biological context. In the same vein, several alternative tools have been developed [11,12].

Nevertheless, by opposition to microelectronics, there can be a huge gap between the Boolean abstraction of a gene regulatory network and its actual behavior. In particular, connexions between genes (when the protein synthesized by the expression of a gene #1 regulates the expression of another one, *e.g.* gene #2) are not so obvious. For instance, even if gene #1 is active, the amount of synthesized protein may be not sufficient in order to activate gene #2. In addition, several other mechanisms or functions can not be described by Boolean abstraction (*e.g.* the *amplification function* described in [2]). Another example is Basu's band detector [13] for which the output reporter protein (green fluorescent protein in this case) is synthesized only for an intermediate concentration of input protein (acyl-homoserin lactone). For such circuits, two alternatives exist. Firstly, the development of new tools based on mathematics that provide an intermediate level of abstraction. Secondly, the adaptation of *analog synthesis* methods from microelectronics, that is to say the design automation of analog circuits that are described by transfer function and Kirchhoff's networks.

On intermediate levels of abstraction, René Thomas' investigations deserve to be highlighted [14]. Indeed, he developed a formalism for the modeling of dynamical behaviour of biological regulatory network through multivalued logic variables, rules, graphs and graphical representation of the state space. A couple of years later, Gilles Bernot extended this approach to include temporal properties of gene regulatory networks [15]. An alternative has been recently investigated based on fuzzy logic which is used to describe the rules that govern the relations between protein concentration and gene states [16]. The main asset of fuzzy logic in comparison with standard multivalued logic is that the link between fuzzy value and actual concentration of protein is never lost. Fuzzy logic can be used to describe systems at an intermediate level of abstraction,

but also for design purposes, as it has been shown in [16]. In this case, an algorithm iteratively tests several combinations of rule matrices (picked up from a library) and finds the one that best meets the expected behaviour. Each rules matrix corresponds to a gene-protein interaction and its content provides the designer with important clues about the choice of the protein-gene interaction to implement.

On the other hand, research on *analog synthesis* started in the beginning of the 80's and this topic remains worth investigation in electronics. Several tools and methods have been demonstrated using specific formalisms and formal computation [17,18]. Nevertheless, these developments have not led to a generic tool which would have been widespread in analog designers community. The main reason is that they were too complex and too specific to be used for a large range of circuits. In addition, they require libraries and/or artificial learning methods or the translation of designer experience to formal rules, which is not straightforward. In a more general way, it was observed that the ratio between the implementation complexity of such algorithms and complexity of circuits that can be synthesized were poor in comparison to a hand-made design.

One of the most outstanding breakthrough in the domain of *analog synthesis* has been made by Koza in 1997 [19] based on genetic programming algorithms. He demonstrates the potential of his method on a large set of electronic circuits (filters, amplifiers, controllers) for which genetic algorithms provided solutions (circuit topology and component dimensioning) very competitive in comparison with human intelligence [20]. At the time, the main shortcoming of evolutionary algorithms were that they required computing power that could only be provided by supercomputers. This is no longer true with current technologies: the exploitation of the parallelized computation over small networks and/or the exploitation of the performance of the Graphical Processor Units (GPU) [21] makes it possible to obtain such results with reasonably priced computing systems.

The topic of this paper is to introduce the basic principles of the use of genetic algorithms in genetic regulatory networks design automation. Designing a genetic regulatory network requires a first step of architecture design where abstract biological parts are assembled together. In a second step, these parts need to be actuated: modeling and optimization of the parameters of the system are required to choose which actual parts will be used (for example to choose between a strong or weak promoter). The next section of this paper describes the above-mentioned biological system that is used as case study to illustrate these principles. Then, focus is put on the settings used for the evolutionary algorithm. Results are given in Sect. 4 and discussed in Sect. 5. In this last section, the opportunity to use genetic programming, which is the next step toward gene regulatory network design automation, is also discussed.

2 Description of the Case Study

As a proof of concept, we choose to model and simulate a modified version of a biological band-pass system developed by Basu et al. [23] (cf. Fig. 1). This

system allows the detection of an intermediate concentration of acyl-homoserine lactone (AHL). It consists in two populations of cells: senders and receivers. Senders synthesize and emit isotropic AHL (the signal) inside a Petri dish. AHL is a small molecule able to diffuse in the gelose of the Petri dish and to enter cells. Receivers react to the concentration of AHL and produce GFP, a Green Fluorescent Protein (the output in this system) if the concentration in AHL ([AHL]) is comprised within a specific interval (around $5.10^{-2}\,\mu$M). In practice, one or many groups of senders are laid on specific spots on a Petri dish covered with receiver cells.

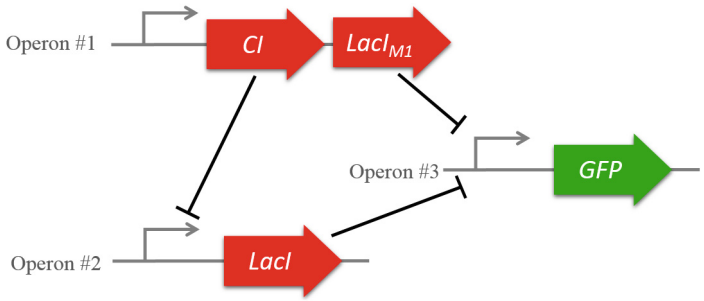

Fig. 1. Simplified Basu system

Our focus was put on the biological core that is computing the output, namely the receiver cells. They are composed of three "operons" (cf. Fig. 1). Operon #1 is positively regulated by AHL (via LuxR, a constitutively expressed protein) and expresses a modified LacI ($LacI_{M1}$). $LacI_{M1}$ inhibits the expression of operon #3 GFP. Operon #1 also produces CI, which in turn inhibits the expression of operon #2 LacI. LacI inhibits the epxression of GFP (operon #3). Each gene transcription into mRNA and subsequent translation into protein can be modeled by differential equations parametrized by the parameters given in Table 1. These 22 values (that constitute the genome of the individuals to be evolved) correspond to the different kind of regulators and promoters (regulated part of the operon). At the steady-state and for given parameters, GFP concentration only depends on [AHL], as described by Eqs. 1, 2, 3 and 4:

$$[LacI_{M1}] = \frac{k_{TL_1}}{d_{LacI_{M1}}} \cdot \frac{k_{TR_1}}{d_{mRNA_1}} \cdot \frac{1}{1 + \left(\frac{K_{A_1}}{[AHL]}\right)^{n_{A_1}}} \tag{1}$$

$$[CI] = \frac{k'_{TL_1}}{d_{CI}} \cdot \frac{k_{TR_1}}{d_{mRNA_1}} \cdot \frac{1}{1 + \left(\frac{K_{A_1}}{[AHL]}\right)^{n_{A_1}}} \tag{2}$$

Table 1. Parameters table. The boundaries correspond to the limits in which the parameters were constrained during initialization (as well as crossover and mutation if relevant). Dissociation constants are given in μM and transcription constants in $\mu M \cdot s^{-1}$.

algogene	Description	Boundaries	
		Min	Max
k_{TR_1}	Transcription constant of operon #1	10^{-3}	10^{3}
K_{A_1}	Dissociation constant of AHL	10^{-4}	10^{3}
n_{A_1}	Hill's number of AHL	1	4
d_{mRNA_1}	Degradation constant of mRNA of operon #1	10^{-3}	10^{-1}
k_{TL_1}	Translation constant of $LacI_{M1}$ (op. #1)	10^{-7}	10^{-5}
$d_{LacI_{M1}}$	Degradation constant of $LacI_{M1}$	10^{-4}	10^{-2}
k'_{TL_1}	Translation constant of CI (op. #1)	10^{-7}	10^{-5}
d_{CI}	Degradation constant of CI	10^{-4}	10^{-2}
k_{TR_2}	Transcription constant of operon #2	10^{-3}	10^{3}
K_{R_2}	Dissociation constant of CI	10^{-4}	10^{3}
n_{R_2}	Hill's number of CI	1	4
d_{mRNA_2}	Degradation constant of mRNA of operon #2	10^{-3}	10^{-1}
k_{TL_2}	Translation constant of operon #2	10^{-7}	10^{-5}
d_{LacI}	Degradation constant of LacI	10^{-4}	10^{-2}
k_{TR_3}	Transcription constant of operon #3	10^{-3}	10^{3}
K_{R_3}	Dissociation constant of $LacI_{M1}$	10^{-4}	10^{3}
n_{R_3}	Hill's number of $LacI_{M1}$	1	4
K'_{R_3}'	Dissociation constant of LacI	10^{-4}	10^{3}
n'_{R_3}	Hill's number of LacI	1	4
d_{mRNA_3}	Degradation constant of mRNA of operon #3	10^{-3}	10^{-1}
k_{TL_3}	Translation constant of operon #3	10^{-7}	10^{-5}
d_{GFP}	Degradation constant of GFP	10^{-4}	10^{-2}

$$[LacI] = \frac{k_{TL_2}}{d_{LacI}} \cdot \frac{k_{TR_2}}{d_{mRNA_2}} \cdot \frac{1}{1 + \left(\frac{[cI]}{K_{R_2}}\right)^{n_{R_2}}} \tag{3}$$

$$[GFP] = \frac{k_{TL_3}}{d_{GFP}} \cdot \frac{k_{TR_3}}{d_{mRNA_3}} \cdot \frac{1}{1 + \left(\frac{[LacI_{M1}]}{K_{R_3}}\right)^{n_{R_3}} + \left(\frac{[LacI]}{K'_{R_3}}\right)^{n'_{R'_3}}} \tag{4}$$

3 Setup of the Evolutionary Algorithm

This section deals with the settings used for the evolutionary algorithm, *i.e.* population size, initialiser, crossover function, mutator function and evaluator.

In the following, to avoid confusion, *algogene* refers to the genes in the algorithm and *biogene* to the biological genes (composing the Basu system).

Population Size and Genetic Engine — 1000 individuals. At each generation, 1000 offspring were generated but to the difference of a generational engine, individuals for the next generation were selected *via* a binary tournament with weak elitism (the best of parents+offspring survives). Parents selection also uses a binary tournament.

Initialiser — Each parameter is initialized randomly within its respective range (see Table 1). For the four Hill's numbers, a random number is drawn between 1.0 and 4.0. Because we are in a biological system, for all the other parameters, a random value is drawn between the two logarithmic values of the interval (the log of min and max values of the "Boundaries" in Table 1); the parameter is then initialized to the power of this value. For example, for k_{TR_1} a random value x is drawn between -3 and 3 so that $k_{TR_1} = 10^x$.

Crossover Function — Different crossover functions were tested: a simple replacement (a tosscoin decides whether the child will be a copy of parent 1 or 2), a barycentric crossover, a BLX-α [22] crossover and an SBX [23] crossover (cf. Table 2). The crossover function is called for all children creation and involves 2 parents for the creation of 1 child.

Mutation Function — Different mutation functions were tested: a simple random draw in the previously defined range, relative mutation (addition of a gaussian noise) and auto-adaptive mutation (from Evolutionary Strategies, see [24]). The mutation operator is applied to each *algogene* with a probability of 0.05 %. If involved, the parameter sigma of the auto-adaptive mutation is mutated whenever its corresponding *algogene* is.

Evaluator — To evaluate the individuals, a first step was to select N absciss points (spread uniformly in the logarithmic scale) from 10^{-4} to 10^{1}. Then on each of these points the genome parameters are used to compare the value of the Basu function (Eqs. 1, 2, 3 and 4) to the target value. To obtain the target values, we approximated the band-pass system described by Basu et al. by a gaussian curve centered on $10^{-1.5}$, of maximal height 20 and of standard deviation 0.2 as follows:

$$f(x) = GFP_{max} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}} \tag{5}$$

The Mean Square Error (MSE) is calculated on all these points. The aim of the algorithm is to minimize the error. If the operators allowed the parameters to cross the boundaries used for initialization, an additional limiting step was used.

Table 2 shows the operators that were used in the results analysis: barycentric crossover operator, auto-adaptative mutation or gaussian noise, gaussian curve with 40 points for the evaluator. Stopping criteria was the number of generations. Analysis of the biological parameters is performed on the genome of the best individual after 5000 generations.

Table 2. Summary of the preliminary tests. When not indicated, mutator operator was applied with a probability of 0.05 %. Each score corresponds to the mean score of three to six runs after 1000 generations, with their standard deviation value. BLXα-x (respectively SBXν-x): x corresponds to the value of α (respectively ν). Evaluation was performed with 40 samples taken uniformly in the log domain.

Crossover		Mutator	Bound.	Average time (s)	Average score of the best
Operator	Domain				
Par.Replacement	lin	simple bounded	No	35.0	0.02 ± 0.007
Par.Replacement	lin	simple bounded 0.1 %	No	34.8	0.04 ± 0.013
Par.Replacement	lin	relative sig0.2	Yes	30.3	0.012 ± 0.005
Par.Replacement	lin	relative sig0.1	Yes	32.4	0.006 ± 0.001
Par.Replacement	lin	relative sig0.3	Yes	30.8	0.011 ± 0.003
Par.Replacement	lin	mut_autoad noise 1.0	Yes	32.5	0.05 ± 0.031
Par.Replacement	lin	mut_autoad noise sqrt(L)	Yes	28.5	0.02 ± 0.008
Barycentric	log	simple bounded 0.01 %	No	39.9	0.08 ± 0.004
Barycentric	log	simple bounded	No	40.1	0.07 ± 0.002
Barycentric	log	simple bounded 0.1 %	No	40.5	0.10 ± 0.009
Barycentric	log	simple bounded 0.2 %	No	40.6	1.27 ± 0.555
Barycentric	log	mut_autoad noise sqrt(L)	Yes	41.8	0.04 ± 0.002
BLXα-0.1	lin	simple bounded	No	36.4	12.23 ± 10.56
BLXα-0.1	log	simple bounded	No	40.2	0.05 ± 0.005
BLXα-0.1	log	simple bounded	Yes	40.3	0.05 ± 0.006
BLXα-0.2	log	simple bounded	No	40.3	0.03 ± 0.001
BLXα-0.2	log	simple bounded	Yes	40.0	0.04 ± 0.005
SBXν-1	log	simple bounded 0.005 %	Yes	35.2	4.15 ± 0.057
SBXν-1	log	simple bounded	Yes	35.3	2.66 ± 0.609
SBXν-2	log	simple bounded 0.005 %	Yes	38.7	1.00 ± 0.867
SBXν-2	log	Relative (sig = 0.1)	Yes	39.1	1.19 ± 1.309

4 Results

The evolutionary algorithm has been implemented on the EASEA platform [25,26]. Computations have been performed on a standard computer without GPU acceleration. Results are summarized in Table 2 and discussed in the following subsections. The score reflects how far the function fed by the 22 parameters found by the algorithm is from the target function (precisions are given in the previous section).

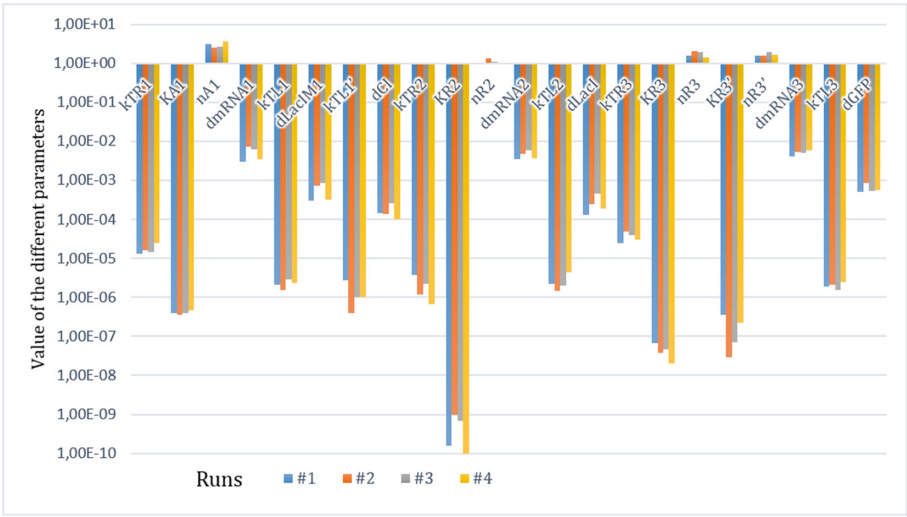

Fig. 2. Value of the *algogene*s for four runs after 5000 generations. The dissociation constants are given in M, the transcription constants in $M \cdot s^{-1}$, the translation and the degradation constants in s^{-1}.

In the following subsections, scores and computing time correspond to the mean of these values over 30 runs if not indicated otherwise. For clarity, only the first four runs are shown on Figs. 2 and 3.

4.1 Is Parental Replacement Relevant on a Biological Point of View?

When using parental replacement as crossover operator, the results are the best in terms of computing time (158 s) and score of the best individual ($9.91 \cdot 10^{-3}$ corresponding to a relative error (score of the best individual over the height h of the peak) of 0.05 %). Out of five runs, four parameters (k_{TR_1}, d_{CI}, K_{R_2}, n'_{R_3}) systematically reached their lower boundary and one (nA_1) reached its upper one. Decreasing the mutator's gaussian noise (variance is no longer the range of the interval for the parameter in the log domain but 1.0) did not alter significantly this tendency (out of four runs, the same parameters but k_{TR_1} reached systematically one of their boundary). A Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [27] algorithm could be tested.

4.2 Validity of the Results

As shown on Fig. 2, the algorithm solutions always requires for K_{R_3} (the repression constant of $LacI_{M1}$) to be lower than K_{R_2} by at least one order of magnitude. In the original work [13], this is also a requirement, which they solve by using CI as a strong repressor (to have a low K_{R_2} constant), and $LacI_{M1}$ as a

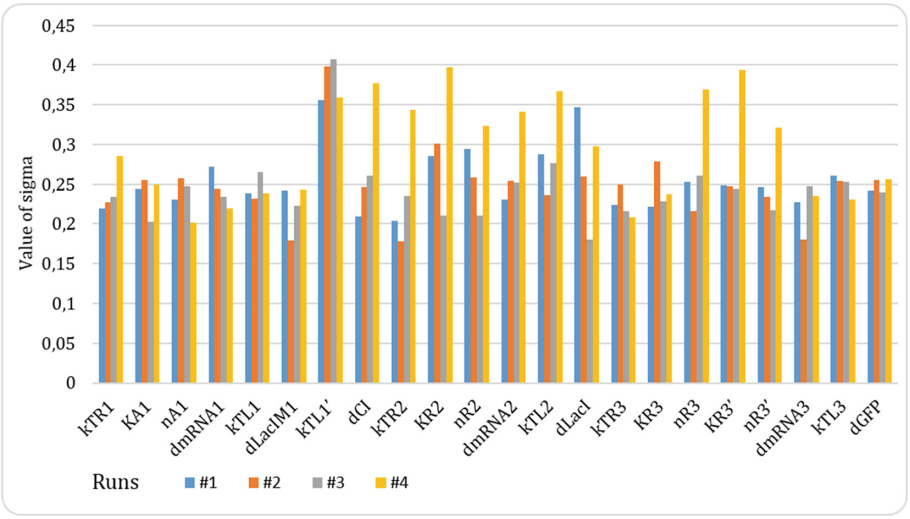

Fig. 3. Values of the auto-adaptive mutator sigma values for four runs after 5000 generations.

weaker repressor. Similarly, we observe that the constants related to LacI and $LacI_{M1}$ (K_{R_3} and $d_{LacI_{M1}}$ and respectively K'_{R_3} and d_{LacI}) are close for the two proteins. This is also the case in the original work. It is to be noted that the degradation constants of the proteins obtained by our algorithm are in accordance with the parameters used by Basu *et al.* to represent their actual system. As for the global synthesis constants (for a given protein, it corresponds to $\frac{k_{TR} \cdot k_{TL}}{d_{mRNA}}$), they are in the range of $0.1\,\mu\text{M} \cdot min^{-1}$ in our results and of $1\,\mu\text{M} \cdot min^{-1}$ in the original work.

4.3 The Assets of Auto-adaptive Mutation

Auto-adaptive mutation offers the possibility to retrieve information about key parameters of the system. After the algorithm reached its stopping criteria, we retrieved the sigma values of the best individual. Out of 30 runs, less than 5 % *algogene*s showed a sigma value lower than 0.2 whereas more than 17 % had sigma values greater than 0.3 (see Fig. 3). We observed that no *algogene* has a particularly low value of sigma. On the contrary, parameter k'_{TL1} has an average sigma value of 0.38, showing that it is less sensitive to variation.

4.4 Validation on Different Targeted Response

We tried to see whether the algorithm could solve similar problems with the same set-up by shifting the target function and increasing its peak (Fig. 4). Three other gaussian curves were tested: one with $\mu = 1$ and $GFP_{max} = 5\,\mu\text{M}$, another one with $\mu = 10^{-1}$ and $GFP_{max} = 10\,\mu\text{M}$ and a last one with $\mu = 10^{-3}$

and $GFP_{max} = 40\,\mu M$. In each case, relative error was under 0.3 %, in 200 s on average (see Fig. 4). For each peak, we took the average of each parameter out of three runs. It appears that most parameters are similar. Major differences can be observed for the dissociation constants of the regulators, namely activation constant K_{A_1} and repression constants K_{R_2}, K_{R_3} and K'_{R_3}. Apart for K'_{R_3} where the tendency is opposed, they decrease with the height of the peak. The same is observed for n_{R_3}. To observe whether this difference was due to the positional shift or the height difference, a new set of runs were performed with the same position for each peak ($10^{-1}\,\mu M$). The results showed that the tendency observed previously is absent. Moreover, the differences between the dissociation constants of each setup are negligible (they are non-existent or spread over less than one order of magnitude). No difference could be observed for the values of k_{TL_3}, the translation constant of GFP. Only minor differences could be observed (when observed).

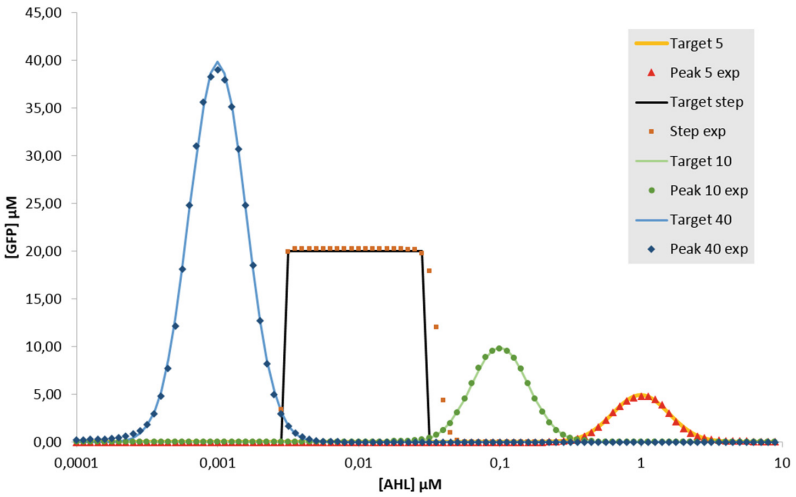

Fig. 4. Representation of the best individual's GFP response in function of [AHL] for different targets. The continuous lines show the targets and the points the corresponding results for three runs. "Target x" corresponds to a gaussian curve with height $h = x$ and "Peak x exp" the cognate results. For $h = 40$ (resp. 10, 5), the gaussian is centered on $\mu = 10^{-3}$ (resp. 10^{-1}, 10^{0}). "Target step" corresponds to a step function centered on 10^{-2} with a plateau value of $20\,\mu M$. The algorithm was run for 5000 generations with barycentric crossover and gaussian noise mutator.

We also wanted to see whether this biological function was able to return a step-shaped response. The target function was changed to a step function with a value of $20\,\mu M$ between $10^{-2.5}\,\mu M$ and $10^{-1.5}\,\mu M$ and $0\,\mu M$ everywhere else. The algorithm found decent solutions (average relative error inferior to 2 %) in 200 s on average (see Fig. 4).

4.5 Grouped Evolution of *algogene*s

In nature, some *biogene*s coevolve: they are constrained by the same rules during evolution [28,29]. We tried to reproduce this phenomenon by allowing the algorithm to divide the genome in n groups of *algogene*s (n varying from 1 to 22). In a group, the *algogene*s are crossed with the same parameters. From two to five groups, the scores are higher than the one group-algorithm. The algorithm is converging prematurely. Above five groups, compared to the one group-algorithm the scores are similar. However, the algorithm requires 1.15 more time when using groups.

We also tried to run the algorithm with autoadaptive groups. Each individual had a random number of groups, composed of *algogene*s randomly picked. Group setup of each individual was allowed to mutate. Crossover of two individual creates a child with a number of groups of one of his two parents. Results were not conclusive in terms of score, nor in terms of biological interest: it was not observed that specific *algogenes* had a higher tendency to form a group together.

4.6 Results Obtained on an Alternative Biological System

To see whether this approach could be generalized to other types of genetic networks, we tried to optimize a bio-logic XOR gate. The complete description of the biological system can be found here [30]. A simplified illustration is given on Fig. 5. In the presence of both Phloretin (Ph) and Erythromycin (Er), mRNA

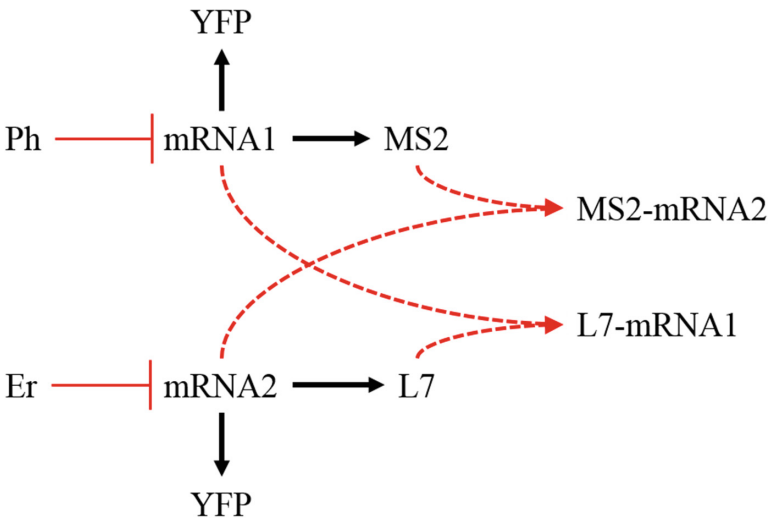

Fig. 5. Simplified XOR bio-logic gate. Barred red lines indicate a repression on the operon coding for the corresponding mRNA; red dashed arrows indicate a binding reaction; black heavy arrows indicate production. Ph: Phloretin. Er: Erythromycin. YFP: Yellow Fluorescent Protein (Color figure online).

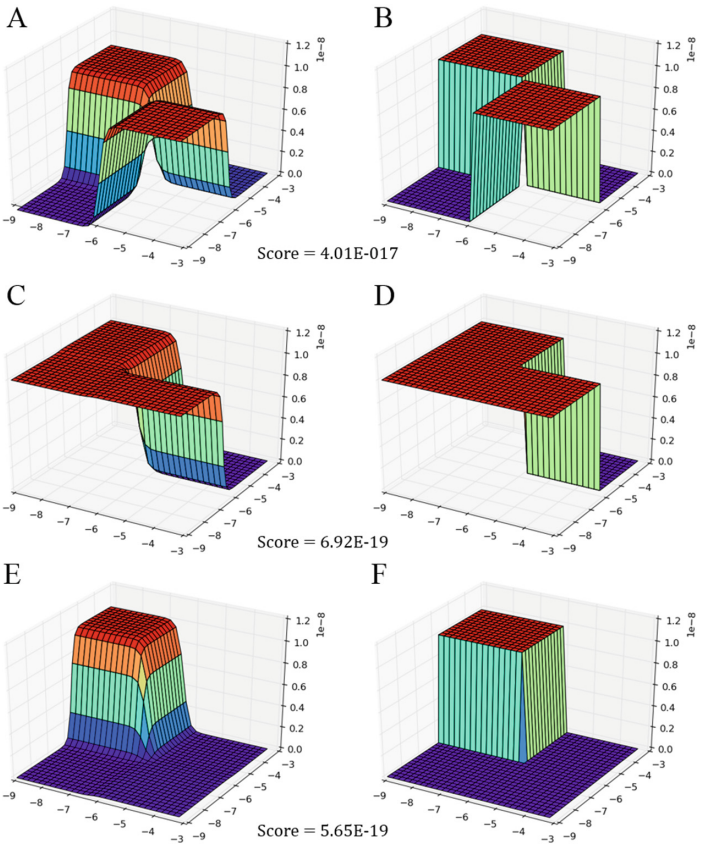

Fig. 6. [YFP] in function of [Er] and [Ph] for three different targets. B (resp. D and F) shows the target function for a XOR (resp. OR and INH) gate. A (resp. C and E) corresponds to the results for the target B (resp. D and F). Score corresponds to the fitness score of the best individual for each target.

are not synthesized. Thus Yellow Fluorescent Protein (YFP) is not produced. If there is no Ph nor Er, both mRNA are synthesized but inhibited because they are bound with L7 (for mRNA1) and MS2 (for mRNA2). Finally, when Ph (resp. Er) is present alone, mRNA2 (resp. mRNA1) is produced but not MS2 (resp. L7). As a consequence, mRNA2 or mRNA1 can be translated and YFP is synthesized.

The equation set that models this system is composed of 15 parameters (2 transcription rates, 1 translation rate, 2 dissociation constant and 2 Hill's numbers for Ph and Er repression, 2 dissociation constants for MS2-mRNA2 and L7-mRNA1 binding reaction and 3 translation rate and 4 decay rates).

Evaluation is performed by taking the MSE between the value of this function and the target function on 10 values of [Ph] times 10 values of [Er], spread uniformly on the log domain between 10^{-9} and 10^{-3} µM. The target function is

a classic binary XOR, with low plateaus being set at $10^{-12}\,\mu$M and high plateaus at $10^{-8}\,\mu$M. Threshold value is set at $10^{-6}\,\mu$M for both inputs (namely Er and Ph). The algorithm was able to produce a set of parameter with a score around 10^{-17} (Fig. 6). We also tried several variations of the XOR gate, namely an OR gate (no YFP when both Er and Ph are present) and an INH gate (YFP is produced when Er is present and pH is absent).

5 Discussion

5.1 Results Obtained with Evolutionary Algorithms

The obtained results show that the algorithm can find coherent solutions to a biological problem. Indeed, the original work emphasizes some specificity of their system (significant differences in the repressor constants) which the algorithm successfully shows. As the process contains modifying operators (crossover, mutation), the parameters could have virtually taken any value in their respective allowed range. Interestingly enough, all the values obtained were biologically relevant and consistent with the original paper.

Auto-adaptive mutation was expected to be an additional hint towards which elements of the system are of key importance in the correct realization of the expected function. However, no *algogene* showed a particular constraint (low sigma value). The general tendency for sigma values to be rather above average than below is due to the way the sigma is mutated. Indeed, its value is multiplied by the exponential of a random number drawn from a Gaussian distribution centered on 0.0 with a variance of 1.0. The high value of k'_{TL_1} (translation of CI) sigma suggests that this parameter is allowed to vary from the returned optimized value. This would give the biologist more freedom regarding the promoter he needs to use.

Running the algorithm with other targets revealed key parameters to set the desired position of the peak. These parameters are dissociation constants of regulators, which makes sense since these parameters control the sensitivity of a promoter towards its regulator. Two groups can be distinguished. K_{A_1} and K_{R_2} allow to shift the peak, and K_{R_3} and $K(_{R_3}$ to sharpen it. Indeed, a decrease on K_{A_1} will lead to a higher sensitivity of operon #1 promoters towards AHL, so that a lower [AHL] will be required to initiate $LacI_{M1}$ and CI expression. This leads to an increase of GFP at lower [AHL], namely a shift of the rising edge of the peak to the lower concentrations. Similarly, a decrease on K_{R_2} will increase the sensitivity of operon #2 towards CI, resulting in a stronger repression of LacI by CI: the falling edge of the peak will shift to lower [AHL]. Finally, K_{R_3} and $K_{R_{3'}}$ control directly GFP expression and therefore act on the thinness of the peak. Indeed an increase on $K_{R_{3'}}$ decreases the sensitivity of GFP's promoter towards CI. The rising edge of the peak will therefore be shifted towards lower [AHL]. Seemingly, a decrease on K_{R_3} will result in a shift of the falling edge of the peak towards lower [AHL] as well. This can be verified by simulating GFP levels in function of [AHL] while varying the above mentionned constants (data not shown).

We can expect that a change in GFP peak height only would involve a similar change in GFP translating constant. As it is not the case, we investigated the product $\frac{k_{TL_3} \cdot k_{TR_3}}{d_{mRNA_3} \cdot d_{GFP}}$ (see Eq. 4), which indeed grows proportionally to the height of the peak. The algorithm is therefore capable of finding non trivial solutions regarding this biological problem.

With the possibility for the *algogene*s to evolve in groups, we expected to observe the appearance of groups of linked parameters. Indeed, as described above, the maximal [GFP] depends of the constants k_{TL_3} , k_{TR_3}, d_{GFP} and d_{mRNA_3}. When the algorithm found a good individual, if k_{TL_3} and k_{TR_3} (respectively d_{GFP} and d_{mRNA_3}) evolve in the same direction (or not at all), the individual should keep its good score. We therefore expected for such parameters to tend to gather in the same group. This tendency was not observed.

To further validate the strength of this approach, a comparison with other methods (such as particle swarm or simulated annealing) could be carried on.

5.2 From an Evolution Strategy to Genetic Programming

The algorithm used to obtain the presented results is close to an Evolution Strategy to optimize a highly dimensional biological system (22 parameters). What would be a following step is to create an algorithm able to find a relevant biological system to answer the biologists' needs. Indeed, a typical approach in synthetic biology begins with the specifications describing a given problem. The solution is often a biological system, that fulfills the specification requirements. The next step for the biologist is to design the said biological system, and finally optimize its components. The last step is realized by optimizing the constants, as shown above. A still missing link is to find, in a library (typically the Biobricks library [31]), the closest components to the returned parameters. However, the biologist still has to imagine the biological system's architecture before 'feeding' it to the algorithm. That is why an algorithm able to create a biological function fulfilling a biological set of requirements is needed.

Genetic Programming could be used for this. As a biological system can be abstracted in a biological function as seen above, we first tried to evolve a function with the same evaluation as previously. Preliminary results showed that the returned equations were too complicated in order to find the biological system associated to the obtained equation. In synthetic biology, systems are made of so called parts, DNA sequences coding for promoters, proteins, terminators, among others. A good idea would be to use the standard decomposition of systems into blocks to evolve a graph, composed of such elements. A new formalism must be imagined for mutation, crossover and evaluation. To what corresponds the nodes and edges is still unclear. Ideally, the algorithm would be able to build the most compact biological system which, given the right constants, would fulfill the requirements. This hints towards an evaluation process taking into account the size of the system (it is easier to produce a promoter sensitive towards one or two regulators than towards four or five). Because some systems' behavior depends heavily on their parameters (*e.g.* the system presented here) the graph should not

only produce an architecture (which product regulates the expression of which other product) but also include quantitative or semi-quantitative elements. Of course, as an optimization step is included afterwards, this step should be quite quick. Taking inspiration from fuzzy logic which introduces an intermediate level between binary and continuous elements, a further step would be to introduce parts with three different level of sensitivity, expression, etc.

6 Conclusion and Future Work

This work demonstrates the relevance of evolutionary algorithms in synthetic biology, in particular in the field of biological networks. Being able to optimize the parameters of a defined system before going to the bench is an important speedup in the process of biological networks design. Indeed, biologists typically have to test various sets of components before finding the most suitable. Such an algorithm gives hints on which parameters are of key importance and which kind of components should be used (*e.g.* a strong or weak repressor). Moreover, as this tool is capable of generating sets of parameters realizing a large variety of functions, it can also help to understand the specifics of a system, by varying the target function used in evaluation and analyzing the returned parameters. This modularity enlarges designers' horizon by giving them the possibility to preliminary test in silico any (crazy) idea they might have.

This optimizing step is to be preceded by a network conception step, also ideally automated. Genetic programming is a promising field in this regard. Formalism of the networks components, the operators to use and the evaluator is what comes next.

References

1. Ro, D.K., Paradise, E.M., Ouellet, M., Fisher, K.J., Newman, K.L., Ndungu, J.M., Ho, K.A., Eachus, R.A., Ham, T.S., Kirby, J., et al.: Production of the antimalarial drug precursor artemisinic acid in engineered yeast. Nature **440**(7086), 940–943 (2006)
2. Xie, Z., Wroblewska, L., Prochazka, L., Weiss, R., Benenson, Y.: Multi-input RNAi-based logic circuit for identification of specific cancer cells. Science **333**(6047), 1307–1311 (2011)
3. Levskaya, A., Chevalier, A.A., Tabor, J.J., Simpson, Z.B., Lavery, L.A., Levy, M., Davidson, E.A., Scouras, A., Ellington, A.D., Marcotte, E.M., et al.: Synthetic biology: engineering escherichia coli to see light. Nature **438**(7067), 441–442 (2005)
4. Peralta-Yahya, P.P., Zhang, F., Del Cardayre, S.B., Keasling, J.D.: Microbial engineering for the production of advanced biofuels. Nature **488**(7411), 320–328 (2012)
5. Beal, J., Weiss, R., Densmore, D., Adler, A., Appleton, E., Babb, J., Bhatia, S., Davidsohn, N., Haddock, T., Loyall, J., et al.: An end-to-end workflow for engineering of biological networks from high-level specifications. ACS Synth. Biol. **1**(8), 317–331 (2012)
6. Marchisio, M.A. (ed.): Computational Methods in Synthetic Biology, Methods in Molecular Biology, vol. 1244. Springer, New York (2015)

7. Myers, C.J.: Computational synthetic biology: progress and the road ahead. IEEE Trans. Multi-scale Comput. Syst. **1**(1), 19–32 (2015)
8. Bhatia, S., Roehner, N., Silva, R., Voigt, C.A., Densmore, D.: A framework for genetic logic synthesis **103**(11) (2015)
9. Icube laboratory - genetic netowrk design automation (2015). http://geneda.fr
10. Madec, M., Pecheux, F., Gendrault, Y., Bauer, L., Haiech, J., Lallement, C.: EDA inspired open-source framework for synthetic biology. In: 2013 IEEE Biomedical Circuits and Systems Conference (BioCAS), pp. 374–377. IEEE (2013)
11. Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J.C., Densmore, D.: Eugene-a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. PloS ONE **6**(4), e18882 (2011)
12. Yaman, F., Bhatia, S., Adler, A., Densmore, D., Beal, J.: Automated selection of synthetic biology parts for genetic regulatory networks. ACS Synth. Biol. **1**(8), 332–344 (2012)
13. Basu, S., Gerchman, Y., Collins, C.H., Arnold, F.H., Weiss, R.: A synthetic multicellular system for programmed pattern formation. Nature **434**(7037), 1130–1134 (2005)
14. Thomas, R., Thieffry, D., Kaufman, M.: Dynamical behaviour of biological regulatory networks—i. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull. Math. Biol. **57**(2), 247–276 (1995)
15. Bernot, G., Comet, J.P., Richard, A., Guespin, J.: Application of formal methods to biological regulatory networks: extending thomas' asynchronous logical approach with temporal logic. J. Theor. Biol. **229**(3), 339–347 (2004)
16. Gendrault, Y., Madec, M., Lemaire, M., Lallement, C., Haiech, J.: Automated design of artificial biological functions based on fuzzy logic. In: 2014 IEEE Biomedical Circuits and Systems Conference (BioCAS), pp. 85–88. IEEE (2014)
17. Doboli, A., Vemuri, R.: Exploration-based high-level synthesis of linear analog systems operating at low/medium frequencies. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **22**(11), 1556–1568 (2003)
18. Lohn, J.D., Colombano, S.P.: Automated analog circuit synthesis using a linear representation. In: Sipper, M., Mange, D., Pérez-Uribe, A. (eds.) ICES 1998. LNCS, vol. 1478, pp. 125–133. Springer, Heidelberg (1998)
19. Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A., Dunlap, F.: Automated synthesis of analog electrical circuits by means of genetic programming. IEEE Trans. Evol. Comput. **1**(2), 109–128 (1997)
20. Koza, J.R.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, Norwell (2003)
21. Maitre, O., Baumes, L.A., Lachiche, N., Corma, A., Collet, P.: Coarse grain parallelization of evolutionary algorithms on GPGPU cards with EASEA. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1403–1410. ACM (2009)
22. Eshelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. In: Whitley, L.D. (ed.) FOGA, pp. 187–202. Morgan Kaufmann, San Mateo (1992)
23. Deb, K., Agrawal, R.W.: Simulated binary crossover for continuous search space. Complex Syst. **9**, 115–148 (1995)
24. Schwefel, H.P.: Adaptive Mechanismen in der biologischen Evolution und ihr Einfluß auf die Evolutionsgeschwindigkeit

25. Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Take it EASEA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 891–901. Springer, Heidelberg (2000)
26. Collet, P., Krüger, F., Maitre, O.: Automatic parallelization of EC on GPGPUs and clusters of GPGPU machines with EASEA and EASEA-CLOUD. In: Tsutsui, S., Collet, P. (eds.) Massively Parallel Evolutionary Computation on GPGPUs, pp. 15–34. Springer, Heidelberg (2013)
27. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
28. Ehrlich, P., Raven, P.: Butterflies and plants: a study in coevolution. Evolution **18**(4), 586–608 (1964)
29. Goh, C.S., Bogan, A.A., Joachimiak, M., Walther, D., Cohen, F.E.: Co-evolution of proteins with their interaction partners. J. Mol. Biol. **299**(2), 283–293 (2000)
30. Ausländer, S., Ausländer, D., Müller, M., Wieland, M., Fussenegger, M.: Programmable single-cell mammalian biocomputers. Nature **487**(7405), 123–127 (2012)
31. Registry of standard biological parts. http://parts.igem.org/Main_Page