Syst Synth Biol (2010) 4:55–84
DOI 10.1007/s11693-009-9050-7

RESEARCH ARTICLE

Evolving cell models for systems and synthetic biology

Hongqing Cao · Francisco J. Romero-Campero · Stephan Heeb · Miguel Cámara · Natalio Krasnogor

Received: 27 August 2009 / Revised: 30 October 2009 / Accepted: 17 December 2009 / Published online: 22 January 2010

Abstract This paper proposes a new methodology for the automated design of cell models for systems and synthetic biology. Our modelling framework is based on P systems, a discrete, stochastic and modular formal modelling language. The automated design of biological models comprising the optimization of the model structure and its stochastic kinetic constants is performed using an evolutionary algorithm. The evolutionary algorithm evolves model structures by combining different modules taken from a predefined module library and then it fine-tunes the associated stochastic kinetic constants. We investigate four alternative objective functions for the fitness calculation within the evolutionary algorithm: (1) equally weighted sum method, (2) normalization method, (3) randomly weighted sum method, and (4) equally weighted product method. The effectiveness of the methodology is tested on four case studies of increasing complexity including negative and positive autoregulation as well as two gene networks implementing a pulse generator and a bandwidth detector. We provide a systematic analysis of the evolutionary algorithm's results as well as of the resulting evolved cell models.

Keywords Systems biology · Synthetic biology · P systems · Evolutionary algorithms · Automated model design

S. Heeb · M. Cámara
School of Molecular Medical Sciences, University of Nottingham, Nottingham, UK

H. Cao · F. J. Romero-Campero · N. Krasnogor (✉)
School of Computer Science, University of Nottingham, Nottingham, UK
e-mail: Natalio.Krasnogor@nottingham.ac.uk

Introduction

Living cells are complex systems that arise from a rich array of interrelated biomolecular processes. In order to understand, manipulate and even coerce a cellular system into producing a target phenotype, the development of good models is a critical steppingstone (Szallasi et al. 2006). Thus sibling disciplines systems (Alon 2006; Klipp et al. 2005; Palsson 2006) and synthetic (Benner and Sismour 2005; Anderianantoandro et al. 2006; Basu et al. 2005) biology depend crucially on the availability of sophisticated and expressive modeling methodologies and tools.

Mathematical and computational modelling of cellular systems is a central methodology within systems biology and synthetic biology and it covers a wide spectrum of sophistication. At one end of the spectrum, modeling can be a very useful tool for clarifying the knowledge that is already available about a given biological entity because, through the process of model building, inconsistencies are detected and gaps in knowledge identified. If sufficient information is available the model might then be more than a formal description of available data and it can be tested against experimental data. Thus the model become an operational entity on its own right with which the biologist can interact in order to further clarify biological understanding. Moreover, the model might be sufficiently detailed as to allow the exploration of "what if" questions beyond the scope of the experimental data upon which the model was constructed. The ultimate goal, at the top end of the sophistication spectrum, for a mathematical or computational model will be to allow the in silico generation of novel biological hypothesis, new experimental routes and, ultimatly, optimised synthetic phenotypes. Klipp et al. 2005 identify the following key stages for model

development. One starts with *formulating a problem* the model is supposed to give answers to or insights about. Once the problem has been formulated the *verification of available data* ensues. All extant data about the biological system to be studied must be collected and curated. Ideally, data will be of a quantitative nature and will include interactomes' maps and details about the experimental data supporting high level descriptions. The next two steps involve the *selection of the modeling formalism* that will be used (e.g. macroscopic *vs.* microscopic, deterministic *vs.* stochastic, steady-state, temporal or spatio-temporal, etc.), a selection of the key model descriptors and the *prototyping of a draft model* with which to refine in an iterative manner the previous steps. Once a model candidate has been proposed, a *sensitivity analysis* should be carried out as to produce a control-map of the model and its (many) parameters. The goal is to identify which parameters the model is or is not robust to. The ultimate test for any model is its fit to reality, thus *experimental validation* , whenever possible, should be carried out. Unfortunately, this is not always possible and indeed, it is common to use models as "surrogates" in precisely those situations where experiments are infeasible (e.g. due to costs, lack of technology or ethical considerations). On the other hand, if experimental validation is indeed feasible, the step that follows is to clearly *state the agreements and disagreements* between model and reality and to *iteratively refine* the models thus obtained (Harel 2005; Cronin et al. 2006).

However promising and appealing modelling is for systems and synthetic biology, it is, indeed, a very difficult endevour that encompasses a variety of activities. Nowadays, model building is supported by a range of tools (e.g. Gilbert et al. 2006; Machne et al. 2006) and techniques. Regardless of the underlying modeling methodology, model building calls for the identification of the model's structure and the optimisation of its (many) parameters and these are, indeed, very difficult computational tasks. On the one hand, the space of all possible model topologies and kinetic parameters is vast and, on the other hand, there is no one-to-one mapping between physical reality and the space of models. That is, several models might equally well represent the knowledge that is available at any one time.

Mathematical modelling of cellular systems, in particular by means of ordinary differential equations (ODEs), is one of the most widely used techniques for modelling (Atkinson et al. 2003; de Hoon et al. 2003). Examples of the optimisation of ODEs' parameters include the optimisation of S-systems (Kikuchi et al. 2003; Morishita et al. 2003) capable of capturing non-linear dynamics. When a large number of parameters are involved within a system of ODEs, simplifying assumptions are made and linear weighted matrices models (Weaver et al. 1999; Yeung et al. 2002) are optimised instead. Most of the research in this area has focused on fine-tuning either the model structure or its parameters. For example, Mason et al. 2004, within the context of an evolutionary algorithm, used random local search as a mutation operator in order to evolve ODE models of interactions in genetic networks. Chickarmane et al. 2005 used a standard genetic algorithm (GA) to optimize the kinetic parameters of a population of ODE-based reaction networks in which the topology was fixed and the task was to match the model's behavior to a target phenotype such as switching, oscillation and chaotic dynamics. Spieth et al. 2004 proposed a memetic algorithm (Krasnogor and Smith 2000, 2005; Krasnogor and Gustafson 2002) to tackle the problem of finding gene regulatory networks from experimental DNA microarray data. In their work the structure of the network was optimized with a GA while, for a given topology, its parameters were optimized with an evolution strategy (Beyer and Schwefel2002). The two deterministic models they used were based on linear weight matrix and S-systems. Recent studies (Rodrigo et al. 2007a; Rodrigo and Jaramillo 2007) have used ODEs as modeling method and a Monte Carlo simulated annealing (SA) approach to perform optimization. In particular, they automatically design small transcriptional networks and kinetic parameters including well-known gene promoters.

(O)DEs models rely on two key assumptions, namely, continuity and determinism of cellular processes' time dynamics. These properties are difficult to justify in systems where low number of regulatory molecular species or slow interactions between them take center stage (Kaern et al. 2005). In such systems, the application of ODEs models is questionable and mesoscopic, discrete and stochastic approaches are more suitable (Gillespie 2007). *executable biology* (Fisher and Henzinger 2007) and (alternatively) *algorithmic systems biology* (Priami 2009) are gaining momentum as alternative ways of modeling large biological complex systems that overcome the above assumptions and provide some additional benefits. In executable biology, models are built not by specifying the so called transfer functions as it is done in traditional modeling with, e.g., differential equations, in which the rate of change of quantities is phenomenologically modeled, but rather mechanistically by defining algorithms (under a variety of possible formalisms) whose execution mimics the causal relation behind change and time/space dynamics in biological systems. Existent executable biology methodologies are rigorous and mathematically sound modeling techniques. Their allure for biological modeling is multifaceted. On the one hand, these modeling techniques are closer to the language of biology and thus they are not perceived by the biologist as complicated black boxes. That is, specifying a model (prototype) with, lets say, Petri Nets or P system is a transparent activity for the biologist and thus helps bridge the discipline gap inherent in any

multidisciplinary team. Thus executable biology models are deemed to be more expressive than other techniques. Moreover, executable biology models can capture the modularity behind many biological systems with reasonable ease. In turn, this permits an incremental approach to model specification, verification and testing. Furthermore, executable biology permits the detailed analysis of single simulation trajectories and of simulations ensembles. Both are required for developing a better intuition and understanding of how the biological system under study is likely to behave as the scientist is interested in both average behavior and extreme or outliers events. Integrative models based on the techniques mentioned above can not only be analysed through their easy of use for capturing biological knowledge or by simulations but, equally important, by a rigorous testing through model checking. Model checking allows one to introspectively analyse the various possible paths that the biological system might go through and obtain the likelihood of certain events taking place, thus unlike simulations, model checking techniques give guarantees about the properties and features being tested. Needless to say, these guarantees comes at a price in computational expense and hence the combination of simulations and model checking is the best compromise for analysing large complex biological systems. Executable biology models have been successfully used to model a variety of biological systems. For example Petri Nets (Heiner et al. 2008) where used to model the core of the ERK/MAPK pathway that mediates information transfer from the membrane to the nucleus in cell division and differentiation processes. Beta-binders, autoreactive lymphocyte recruitment and other bioprocesses have also been modeled with $\Pi-$calculus and, more generally, process algebras (Regev et al. 2001; Errampalli and Quaglia 2004). Both signaling pathways and gene regulatory networks were model checked with Prism in Calder et al. 2005; Romero-Campero et al. 2009, Romero-Campero and Krasnogor 2009 . P systems where the preferred modeling tool for a wide variety of biological phenomena (Romero-Campero et al. 2008a; Gheorghe et al. 2008). Live sequence charts and state charts were used to produce a multi-scaled model of *C. elegans* (Sadot et al. 2008). Executable biologys features, namely, high expressive power, the possibility to specify causal models in a biology-friendly language, facility for modular and incremental modeling, single/ensemble simulations and formal verification through model checking endows these formalisms with yet another important property: model structures and parameters are more easily discoverable. The process of integrative model building relies on a number of sources of information for specifying the model structure and parameters. In very many cases, one will not have all the parameters of the models (e.g. kinetic constants, diffusion constants, half-lives for molecules, affinity values, etc) and hence the modeler will need to perform one or more (iterative) stages of parameter fitting to experimental data. In other cases, the model being specified is only partially known and, for the current putative model structure, there does not exist any set of parameters that could fit observed experimental data and, at the same time, be general enough to capture future experimental data. Thus, in this case, discovery of model structure, rather than parameters, must be pursued. In this paper we employ a P systems based executable biology formalism that integrates stochastic and discrete modelling into a computational framework. P systems represent an unconventional computational paradigm (Păun 2002) that abstracts from the structure and functioning of the living cell. A P system consists of a cell-like membrane structure, with compartments containing multisets of objects representing molecules which evolve according to given rules that mimic molecular interactions. These rules are applied according to an adaptation of Gillespie's stochastic simulation algorithm (SSA; Gillespie 2007) to the multi-compartmental structure of P system models (Romero-Campero et al. 2009; Pérez-Jiménez and Romero-Campero 2006). We extend previous work (Romero-Campero et al. 2008b) by systematically studying several objective functions for guiding the search for cell biology models' structure and parameters. In this paper we focus on evolving models that can match a predefined target phenotype that is specified in terms of a collection of time series that the evolved models must match. From a systems biology perspective, these time series can be interpreted as experimental data that the model must explain, while from a synthetic biology viewpoint, these time series represent the functional requirements for a putative synthetic phenotype to be modeled. These time series could represent measurements such as cell's optical density, gene expression levels, etc. Figure 1 gives an example with two time series in which a number of issues are highlighted. First, not all time series range over the same scales, they could have vastly different functional forms with peaks and valleys at different places. Secondly, their asymptotic behavior (if it exists) can vary widely. In the figure, the maxima $M1$ and $M2$ occur at different points in time. Moreover, although the absolute error between target molecule 1 and model 1 output is larger than that for molecule 2 and its model, the relative error of the later is larger than the one for the former. Hence, using the root mean square deviation—as it is often done—as an objective function to compare target behavior versus model behavior might not be the best route for successful evolution. In this paper we study four alternative fitness methods to guide the search. In particular, we use an equally weighted sum method, a normalization method, a randomly weighted sum method and an equally weighted product

method to measure the fitness of the models evolved. The effectiveness of the methodology is tested on four case studies of increasing complexity including negative and positive autoregulation as well as two gene networks implementing a pulse generator and a bandwidth detector. We provide a systematic analysis of the evolutionary algorithm's results as well as of the resulting evolved cell models.

To sumarise, the key contributions of this paper are:

- The introduction of a "biologist-friendly" integrated pipeline that, at its core, contains a modeling framework based on P systems. We emphasize very recent developments in terms of the expression power of the framework as well as the facility for modular and incremental model building. The proposed pipeline is exemplified by drawing on some simple and well known regulatory motifs, e.g. positive/negative regulation, paradigmatic study cases such as the Lac operon promoter, as well as more complex state-of-the-art synthetic biology circuits such as a pulse generator and bandwidth detector. The paper demonstrates how a gradual increase in system complexity is accompanied, under our modeling framework, by a parsimonious increase in model complexity. This is so because the proposed framework is inherently suitable to abstraction, encapsulation and data hiding.
- The provision of a systematic study on the optimisation of systems and synthetic biology models' structures and parameters from a "white-box" perspective. Researchers unfamiliar with optimisation techniques are sometimes mislead to assume that off-the-shelf optimisation methods run with their "standard" parameters and objectives functions will magically output optimal solutions. This study highlights the potential sources of difficulties when applying optimisation methods to systems and synthetic biology stochastic models. We show how different target biological systems, which must be modeled, might call for different objective functions and we comment on the advantages and disadvantages of the various alternatives. The results indicate that care must be taken when automating the synthesis and optimisation of (partial) models and that the optimisation process cannot, in general, be done without knowledge of both the biological system being modeled and the details of the modeling formalism.
- We also show that as the proposed integrated pipeline couples a modeling framework that is incremental and modular with a sophisticated white-box optimisation method, one can obtain several circuit designs matching a required phenotype. The availability of alternative designs matching the requirements of a target phenotype might, in turn, open the doors to alternative experimental (i.e. wetlab) strategies. We further illustrate how other analysis techniques, namely model selection and sensitivity analysis, can be used to further refine the computational models thus obtained.

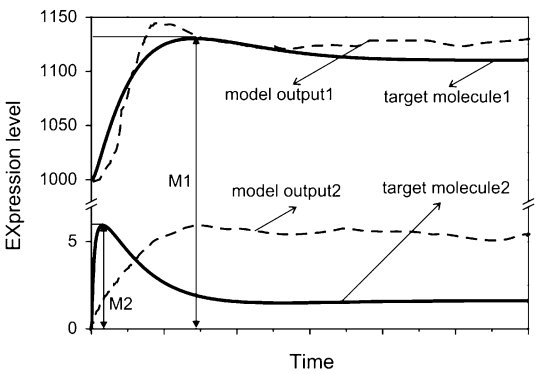

Fig. 1 An example of two target time series each with very specific profiles that must be matched by an evolved model's dynamics

The remainder of the paper is structured as follows. In the next section we describe our modelling methodology which includes the P systems modelling framework, the evolutionary algorithm used to evolve models an the four fitness methods used in this work. In "Experiments" section presents four case studies and the experimental design, with in "Results and discussions section". "Further experiments" section describes additional experiments and "Model selection" section analyses the evolved models. Finally, we end with some "Concluding remarks and future work" section.

Methodology

P systems modelling framework

In this paper we use a computational, modular and discrete-stochastic modelling approach based on P systems, an emergent branch of Natural Computing introduced by Gh. Păun (2002). More specifically, we use a variant called stochastic P systems developed for the specification and simulation of cellular systems (Pérez-Jiménez and Romero-Campero 2006).

A stochastic P system is a construct

$$\Pi = (O, L, \mu, M_{l_1}, M_{l_2}, \ldots, M_{l_n}, R_{l_1}, \ldots, R_{l_n})$$

where:

- O is a finite alphabet of objects representing molecules.
- $L = \{l_1, \ldots, l_n\}$ is a finite set of labels identifying compartment types.
- μ is a membrane structure containing $n \geq 1$ membranes defining compartments arranged in a hierarchical manner. Each membrane is identified in a one to one manner with labels in L which determines its type.

- M_{l_i} for each $1 \leq i \leq n$, is the initial configuration of membrane i consisting of a multiset of objects over O initially placed inside the compartment defined by membrane with label l_i.
- $R_{l_i} = \{r_1^{l_i}, \ldots, r_{k_{l_i}}^{l_i}\}$, for each $1 \leq i \leq n$, is a finite set of rewriting rules associated with the compartment with label $l_i \in L$ and of the following general form:

$$o_1[o_2]_l \xrightarrow{c} o_1'[o_2']_l \quad (1)$$

with o_1, o_2, o_1', o_2' multisets of objects over O (potentially empty) and $l \in L$ a label. These multiset rewriting rules affect both the inside and outside of membranes. An application of a rule of this form replaces simultaneously a multiset o_1 outside membrane l and a multiset o_2 inside membrane l by multisets o_1' and o_2', respectively. A stochastic constant c is associated specifically with each rule in order to compute its propensity according to Gillespie's theory of stochastic kinetics (Gillespie 2007). More specifically, rewriting rules are selected according to an extension of Gillespie's well known SSA (Gillespie 2007) to the multicompartmental structure of P system models (Pérez-Jiménez and Romero-Campero 2006).

Stochastic P systems have been successfully used in the specification and simulation of cellular systems, for instance signal transduction (Pérez-Jiménez and Romero-Campero 2006), prokaryotic gene regulation (Romero-Campero and Pérez-Jiménez 2008a) and bacterial colonies (Romero-Campero and Pérez-Jiménez 2008b).

Modular modelling approach

Cellular functions are rarely performed by individual molecular interactions, instead cellular functions are the product of the orchestration of modules made up of many molecular species for which their interaction modality follows very specific patterns (Alon 2006). Biological modularity is thus one of the cornerstones of synthetic biology (Andreianantoandro et al. 2006). Modularity is a widely used approach in the design of complex systems. It was first applied to biological modelling in the PROMOT tool (Ginkel et al. 2003). Rodrigo et al. 2007a developed a new computational tool to produce model of biological systems by assembling models from biological parts. Recently Marbach et al. 2009 proposed a module extraction method to generate network structure where the extracted modules are biologically plausible as they preserve functional and structural properties of the original network. The importance of modularity has been recently emphasized by Mallavarapu et al. 2009. In this work we follow a modular modelling approach whereby models are incrementally and hierarchically built by combining modules stored in a predefined module library. This library comprises a set of elementary modules that specify basic gene regulatory mechanism as well as modules describing the regulation of specific gene promoters widely used in synthetic biology and systems biology (see below). A *module* is defined as a separable discrete entity that performs a specific biological (Hartwell et al. 1999) function. Recently, modularity in gene regulatory networks has been associated with the existence of non-random clusters of transcriptional regulatory factor binding sites in promoters that regulate the same gene or genes' operons (Davidson 2006). A *P system module* is defined as a set of rewriting rules, each of the form in (1), for which some of the objects, stochastic constants or the labels of the compartments involved might be *variables*. This facilitates reusability as large models can be built by integrating commonly found modules that are then further instantiated with experimentally specific values. In turn, this results on a particular set of rules representing a concrete cellular model. Formally, a P system module M is specified as $M(V, C, L)$ where V represents object variables, which can be instantiated using specific objects describing different molecular species, C are variables for the stochastic constants associated to the transformation rules, and L are variables for the labels of the compartments involved in the rules. For example, V might represent specific genes, proteins and other metabolites' names, C the kinetic constants pertinent to the rules defined for those genes, proteins and metabolites while L might represent different cell compartments, e.g., cytoplasm, lysosome, cellular membrane, etc., or –for multicellular systems– different cells altogether.

In what follows we present the P system modules in the library used in this work in order to illustrate the above definition.

1. *Constitutive or unregulated expression:* This module describes the case of a gene, gX, which is transcribed constitutively into its corresponding mRNA, rX, without the aid of any transcriptional regulatory factor. Translation of the mRNA, rX, into the corresponding protein pX is also specified. The mRNA and protein can be degraded by the cell machinery. These processes occur within compartment l and take place at rates determined by the stochastic constants $c_1, \ldots, c_4$.

$$UnReg(\{X\}, \{c_1, c_2, c_3, c_4\}, \{l\}) = \left\{ \begin{array}{l} r_1 : [gX]_l \xrightarrow{c_1} [gX + rX]_l \\ r_2 : [rX]_l \xrightarrow{c_2} [rX + pX]_l \\ r_3 : [rX]_l \xrightarrow{c_3} [\,]_l \\ r_4 : [pX]_l \xrightarrow{c_4} [\,]_l \end{array} \right\}$$

Note that X is a variable of this module that can be instantiated with a specific gene name to represent that such a gene is expressed constitutively. The variables for the stochastic constants can also be instantiated with particular values to represent different transcription, translation and degradation rates. In what follows we will refer to this circuit either as unregulated expression or constitutive expression.

2. *Positive regulated expression:* The positive regulation of a gene gX over another gene gY is represented in this module. In this case the corresponding protein pX acts as an activator binding reversibly to the gene gY yielding the complex $pX.gY$. This event turns on the production of the mRNA rY. Ultimately, the protein product pY is produced from the mRNA. The mRNA and the protein are also degraded in this case. These processes take place at rates determined by some stochastic constants $c_1, \ldots, c_6$.

$$PosReg(\{X, Y\}, \{c_1, c_2, c_3, c_4, c_5, c_6\}, \{l\}) = \left\{ \begin{array}{l} r_1 : [pX + gY]_l \xrightarrow{c_1} [pX.gY]_l \\ r_2 : [pX.gY]_l \xrightarrow{c_2} [pX + gY]_l \\ r_3 : [pX.gY]_l \xrightarrow{c_3} [pX.gY + rY]_l \\ r_4 : [rY]_l \xrightarrow{c_4} [rY + pY]_l \\ r_5 : [rY]_l \xrightarrow{c_5} [\,]_l \\ r_6 : [pY]_l \xrightarrow{c_6} [\,]_l \end{array} \right\}$$

By instantiating X and Y with specific gene names and $c_1, \ldots, c_6$ with particular values the positive regulation of a gene over another one with characteristic affinities and transcription, translation and degradation rates can be obtained.

3. *Negative regulated expression:* In contrast to the previous case the negative regulation of a gene gY by another gene gX is represented in the module by specifying pX as a repressor binding reversibly to the gene gY to produce the complex $pX.gY$. Under this situation transcription is completely inhibited. The binding and debinding of the repressor to the gene take place at rates determined by two stochastic constants c_1 and c_2.

$$NegReg(\{X, Y\}, \{c_1, c_2\}, \{l\}) = \left\{ \begin{array}{l} r_1 : [pX + gY]_l \xrightarrow{c_1} [pX.gY]_l \\ r_2 : [pX.gY]_l \xrightarrow{c_2} [pX + gY]_l \end{array} \right\}$$

The particular repression of a specific gene over another one with a characteristic affinity can be obtained from the previous module by instantiating X, Y, c_1 and c_2 accordingly.

Besides the above introduced modules, the library includes modules describing the regulation of some of the most widely used gene promoters in synthetic biology, namely, the lac operon promoter from *Escherichia coli*, the cro promoter from Phage lambda and the lux box site from *Vibrio fischeri*. In these modules the instantiation of a variable specifying an object with the name of a specific gene represents a construct where the corresponding gene is fused to the promoter modelled by the module.

4. *Lac operon promoter from* ***E.coli :*** The lactose operon was one of the first gene regulatory systems to be studied (Jacob and Monod 1961). It is negatively regulated by a repressor protein *LacI* (rules r_7 and r_8). In the absence of the repressor the genes regulated by the promoter are basally expressed according to rules $r_1, \ldots, r_4$. The repression can be removed by adding IPTG, a signal that binds to the repressor inactivating it (rules$r_5, \ldots, r_8$).

$$Plac(\{X\}, \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}, \{l\}) = \left\{ \begin{array}{l} r_1 : [\,Plac :: gX\,]_l \xrightarrow{c_1} [\,Plac :: gX + rX\,]_l \\ r_2 : [\,rX\,]_l \xrightarrow{c_2} [\,]_l \\ r_3 : [\,rX\,]_l \xrightarrow{c_3} [\,rX + pX\,]_l \\ r_4 : [\,pX\,]_l \xrightarrow{c_4} [\,]_l \\ r_5 : [\,pLacI + IPTG\,]_l \xrightarrow{c_5} [\,pLacI.IPTG\,]_l \\ r_6 : [\,pLacI.IPTG\,]_l \xrightarrow{c_6} [\,pLacI + IPTG\,]_l \\ r_7 : [\,pLacI + Plac :: gX\,]_l \xrightarrow{c_7} [\,pLacI.Plac :: gX\,]_l \\ r_8 : [\,pLacI.Plac :: gX\,]_l \xrightarrow{c_8} [\,pLacI + Plac :: gX\,]_l \end{array} \right\}$$

5. *The cro promoter from* ***PhageLambda :*** The genetic switch in the *Phage lambda* is another of the best studied gene regulatory systems (Ptashne 2004). This module describes in particular the regulation of the *PR* promoter of the *Cro* protein. This promoter is repressed by the direct and cooperative binding of a dimerised form of the *CI* protein (rules$r_5, \ldots, r_{10}$). The genes under the control of this promoter are constitutively expressed when the *CI* protein is not present (rules$r_1, \ldots, r_4$).

$$PR(\{X\},\{c_1,c_2,c_3,c_4,c_5,c_6,c_7,c_8,c_9,c_{10}\},\{l\})$$
$$= \left\{ \begin{array}{l} r_1 : [PR :: gX]_l \xrightarrow{c_1} [PR :: gX + rX]_l \\ r_2 : [rX]_l \xrightarrow{c_2} [\,]_l \\ r_3 : [rX]_l \xrightarrow{c_3} [rX + pX]_l \\ r_4 : [pX]_l \xrightarrow{c_4} [\,]_l \\ r_5 : [pCI + pCI]_l \xrightarrow{c_5} [pCI_2]_l \\ r_6 : [pCI_2]_l \xrightarrow{c_6} [pCI + pCI]_l \\ r_7 : [pCI_2 + PR :: gX]_l \xrightarrow{c_7} [pCI_2.PR :: gX]_l \\ r_8 : [pCI_2.PR :: gX]_l \xrightarrow{c_8} [pCI_2 + PR :: gX]_l \\ r_9 : [pCI_2 + pCI_2.PR :: gX]_l \xrightarrow{c_9} [pCI_4.PR :: gX]_l \\ r_{10} : [pCI_4.PR :: gX]_l \xrightarrow{c_{10}} [pCI_2 + pCI_2.PR :: gX]_l \end{array} \right\}$$

6. *The lux box from* ***Vibrio fischeri :*** The control of the *lux genes* by the *Plux* promoter in *Vibrio fischeri* constitutes the canonical example of the cell-cell communication system called *quorum sensing* (Diggle et al. 2007). This system relies in the sensing of a small diffusible signal $s3OC6$ (rules r_1 and r_2) by a protein *LuxR*. After sensing of $s3OC6$ the receptor protein dimerises (rules$r_3,\ldots,r_7$) and acts as an activator binding reversibly to a specific site called *lux box*. This event produces the expression of the genes under the control of the *Plux* promoter (rules$r_8,\ldots,r_{13}$). P systems were used in Bernardini et al. 2007 to capture a simplified form of quorum sensing.

$$PluxR(\{X\},\{c_1,c_2,c_3,c_4,c_5,c_6,c_7,c_8,c_9,c_{10},c_{11},c_{12},c_{13}\},\{l\})$$
$$= \left\{ \begin{array}{l} r_1 : [s3OC6ext]_l \xrightarrow{c_1} [s3OC6ext + s3OC6]_l \\ r_2 : [s3OC6]_l \xrightarrow{c_2} [\,]_l \\ r_3 : [s3OC6 + pLuxR]_l \xrightarrow{c_3} [pLuxR.s3OC6]_l \\ r_4 : [pLuxR.s3OC6]_l \xrightarrow{c_4} [s3OC6 + pLuxR]_l \\ r_5 : [pLuxR.s3OC6 + pLuxR.s3OC6]_l \xrightarrow{c_5} [pLuxR_2]_l \\ r_6 : [pLuxR_2]_l \xrightarrow{c_6} [pLuxR.s3OC6 + pLuxR.s3OC6]_l \\ r_7 : [pLuxR_2]_l \xrightarrow{c_7} [\,]_l \\ r_8 : [pLuxR_2 + Plux :: gX]_l \xrightarrow{c_8} [pLuxR_2.Plux :: gX]_l \\ r_9 : [pLuxR_2.Plux :: gX]_l \xrightarrow{c_9} [pLuxR_2 + Plux :: gX]_l \\ r_{10} : [pLuxR_2.Plux :: gX]_l \xrightarrow{c_{10}} [pLuxR_2.Plux :: gX + rX]_l \\ r_{11} : [rX]_l \xrightarrow{c_{11}} [\,]_l \\ r_{12} : [rX]_l \xrightarrow{c_{12}} [rX + pX]_l \\ r_{13} : [pX]_l \xrightarrow{c_{13}} [\,]_l \end{array} \right\}$$

Extensive experimental studies have helped determine the values for the different kinetic constants for the above model systems. However, in the modules library they appear as variables in order to allow, depending on the biological system to be modeled, either their instantiation with values derived from the literature or with new values capable of representing mutations on the underlying nucleotides sequences. In this way enhance or weakened interactions can be easily captured. The modules' library is encoded in XML files for easier electronic reuse by the evolutionary algorithm (see "Experiments").

Modularity affords two major advantages to the design of biological cellular models. Firstly, the use of modules assures model validity and plausibility. Modules are predefined as *building blocks* whose validity and plausibility are fundamented in specific biological knowledge, where each module can—and usually is—validated on its own terms. Secondly, the use of modules increases model diversity. Although the number of elementary modules in the library is limited, each of them can produce many instantiated modules depending on the specific values chosen for their different variables. These instantiated modules can then be combinatorially combined in many different ways thus producing a vast space of candidate models.

A nested evolutionary algorithm for evolving P system models

We propose a nested *evolutionary algorithm* (EA) to evolve P system models that could match a biological phenotype that is specified through a collection of time series representing molecular concentrations of various species. The EA's first layer searches for model structures using a GA; while the inner layer, also implemented as a GA, acts as a local search for the continuous parameters of the model. The pseudo code of both GAs are shown in Figs. 2 and 3 respectively. The details of the two GAs are described in what follows:

1. *Structure optimization of P system models*: In what follows we describe in details the problem representation, the fitness functions used and the genetic operators employed by the search algorithm.
   a. *Problem representation*: The modeling framework we employ as well as the evolutionary algorithm proposed, are prepared to deal with multi-compartment P systems. Multi-compartment models are needed when modeling, e.g., a cell's internal structures and organelles or when dealing with multi-cellular systems such as, e.g., bacteria biofilms, tissues such as plant root development

```
GA for model structure optimization
BEGIN
  t = 0;
  read the imported model library lib;
  randomly generate an initial population of
    valid P system models P(0)based on lib;
  calculate the fitness of the individuals
    in P(0);
  while(t < SOMAXGENO)
  {
    do the parameter optimization for each
      individual in P(t) by GA (see Fig. 3);
    recombine P(t) by tournament selection and
      elitism strategy;
    randomly choose two parents from P(t) to do
      genetic operations (crossover, mutation)
        based on fitness;
    calculate fitness of the individuals in P(t);
    keep the best individual to next generation;
    t = t + 1;
  }
  output the best individual in current population
    as the final model;
END
```

Fig. 2 GA for P system model structure optimization

```
GA for model parameter optimization
BEGIN
   GArate = 0.7*(1-best fitness/fitness(model p));
   ran = randf();
   if(ran < GArate)
   {
     randomly generate an initial population of
       parameter set P(0) for the model p;
     calculate the fitness of individuals in P(0);
     m = 0;
     while(m < POMAXGENO)
     {
       randomly choose M individuals from P(m) to
         do multi-parent crossover and generate
           an offspring pxo;
       calculate the fitness of pxo;
       if(the fitness of pxo is better than the
           worst one in P(m))
         replace the worst with pxo;
       P(m+1) = P(m);
       m = m + 1;
     }
     if(the fitness of the best individual in P(m)
         is better than fitness(model p))
       replace the original parameter set with
         this best individual;
   }
   else
   {
     do hill-climbing based on Gaussian mutation
        MAXHCSTEPS times for kinetic constants
          randomly chosen from the model p;
      output the parameter set at the last step as
        the final parameter set of the model p;
   }
END
```

Fig. 3 GA for P system model parameter optimization

(Twycross et al. 2009), etc. In this work, however, we aim at evolving models of bacterial systems, consequently, the membrane structure of all our models consists of a single membrane (alternatively called compartment). For a P system $\Pi = (O, \{l\}, [\,], M_l, R_l)$ with a single compartment, it is sufficient to specify only a vector whose components are the modules used to construct the rule set $R_l, \Pi = (m_1, \ldots, m_n)$.

As shown in Fig. 4, there are three levels in the data structure of a model representation. First, each rule is encoded using a structure which specifies the rule name, a flag indicating if the objects in the rule are all fixed (1) or some are variables (0), the list of objects on the left hand side (reactants) and on the right hand side (products) of the rule, a flag indicating if the associated stochastic constant is fixed (1) or is a variable (0), and the value of the stochastic constant. If the constant is variable, a lower bound, an upper bound and a precision must be specified as well. A P system module is then encoded using a structure which specifies the module name, a flag indicating if the module is fully instantiated (1) or not (0), the list of variables, the module size (the number of rules) and the set of rules included in the module. Finally, a P system model is encoded using a structure which specifies the membrane type or label, the model size, i.e. the number of modules, and the set of modules that it contains. When a model is constructed, the variables and the constants in each module must be instantiated with specific objects and constant values.

Figure 5 illustrates our encoding by using a stochastic P system model which consists of two modules $UnReg(\{X = A\}, \{c_2 = 0.6, c_3 = 0.01, c_4 = 0.04\}, \{l = b\})$ and $NegReg(\{X = A, Y = A\}, \{c_6 = 0.015\}, \{l = b\})$ (c_1 and c_5 are non-fixed).

b. *Fitness evaluation*: Figure 6 shows the flowchart for the procedure used to evaluate a candidate model Π. Given the target time series, Π is run *MAXRUN* times using Gillespie's SSA

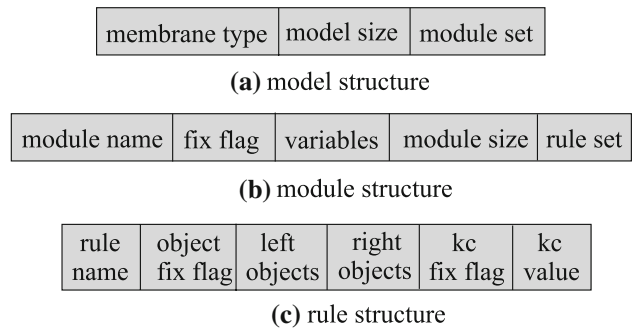

Fig. 4 A P system based model is represented through a three-level data structure

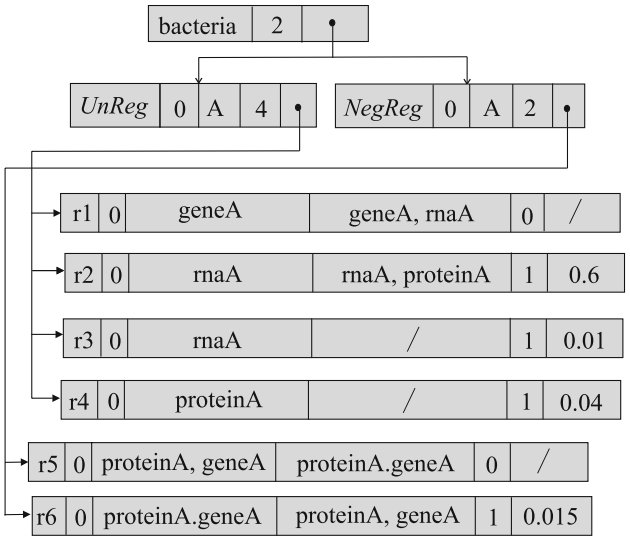

Fig. 5 An example of a P system based model for a bacterium that contains two modules and their rules

(Gillespie2007) and the output from these simulations compared against the target time series. The specific manner in which this comparison is done is at the core of this paper. We investigate four alternative fitness methods, namely, equally weighted sum method (F1), normalization method (F2), randomly weighted sum method (F3), and equally weighted product method (F4). The details of these four fitness methods are described in detail in "Four fitness method".

c. *Genetic operators*: In the GA used for the optimization of the modular structure we use crossover and mutation as the genetic operators.
Crossover can be done by exchanging single modules, *module-exchange crossover*, or by swapping multiple modules between two parents,*one-point crossover*.
Consider two parents $\Pi_1 = (m_1^1, \ldots, m_{n_1}^1)$ and $\Pi_2 = (m_1^2, \ldots, m_{n_2}^2)$ with n_1 and n_2 modules respectively. In the module-exchange crossover, two crossover points, i and j, are randomly selected within Π_1 and Π_2 and then the crossover is performed as follows:

if $(m_i^1 \cap m_j^2 = \emptyset)$
then swap m_i^1 *and* m_j^2;
else swap the kinetic constants of the common rules within m_i^1 *and* m_j^2;
calculate the fitness of both offspring;
choose the better one as the crossover offspring.

The one-point crossover is performed by randomly selecting one crossover position from Π_1 and Π_2 and swapping all the modules after the crossover points. To promote a parsimonious combinatorial search, a valid crossover offspring would be one in which the number of modules does not exceed a predefined maximal module set size, *MAXMSIZE*. If both offsprings are valid the one with the better fitness is chosen.

The structure mutation is performed by randomly selecting a module and making one of the three following variations: (1) randomly pick a rule with variable kinetic constant and change its values using Gaussian mutation; (2) keep the module type unchanged but change some objects in the module's rules; (3) randomly instantiate a module from those available in the library.

2. *Parameters optimization of P system models*: As the kinetic constants associated with each rule are used in Gillespie's SSA to compute the probability of applying each rule and the waiting time for the rule to be executed (Gillespie 2007), the stochastic constants of a P system model determine its behavior, and thus it is crucial to optimize them in order to obtain a desirable dynamics. Here we designed a GA (Yu et al. 2007) to optimize the constants of each candidate P system model for which their structures have been determined in the algorithm's previous stage.

The encoding of a parameter individual in the GA population is done as follows. Given a stochastic P system model generated in the previous stage with n modules $\Pi = (m_1, \ldots, m_n)$, first we calculate the total number of different rules, l, whose kinetic constants are variables in Π by applying set union over the set of rules of the modules $R_\Pi = \bigcup_{i=1}^{n} m_i = \{r_1, r_2, \ldots, r_l\}$. Then we represent each chromosome specifying the constants of Π in the parameter population using an l-dimensional row vector $C(\Pi) = (c_1, c_2, \ldots, c_l)$ where c_i is the constant associated with r_i for $i = 1, 2, \ldots, l$. Each constant is encoded as a floating number and generated randomly within the specific range and precision defined in the module library.

As shown in Figs. 2 and 3, we use a GA as the main optimization mechanism accompanied by a hill-climbing procedure based on Gaussian mutation. The rate for using the GA is determined adaptively based on the fitness of the model (Hinterding et al. 1997). The hill climbing is performed *MAXHCSTEPS* times by randomly choosing a module and a rule with a variable kinetic constant and doing Gaussian mutation on it. The new kinetic constant is kept only if the fitness is improved.

For the GA we use crossover as the sole genetic operator which is performed using a multi-parent crossover as follows. We randomly select $M > 2$ individuals $C_1, C_2, \ldots, C_M$ from the parameter population with $C_i = (c_1^i, c_2^i, \ldots, c_l^i)$ for $i = 1, 2, \ldots, M$. Then M coefficients α_i are randomly

generated satisfying: (1) $\alpha_i \in (a, b)$ where a and b are control parameters of our algorithm such that $a < 0$ and $b > 1$; (2) $\sum_{i=1}^{M} \alpha_i = 1$. Finally, a new vector of constants *pxo* is generated as a non-convex linear combination of C_i using the previous constants:

$$pxo = \sum_{i=1}^{M} \alpha_i C_i$$

If the fitness of *pxo* is better than that of the worst individual in the parameter population then replace it with *pxo*.

Four fitness methods

Suppose we have N target time series $(X_1, X_2, \ldots, X_N)$, each representing a specific protein, gene, rna, etc and where $X_j = (x_j^1, x_j^2, \ldots, x_j^M)^T$, that is, each time series has up to M data points. Each candidate stochastic model is run *MAXRUN* times (see Fig. 6) and an average model output obtained for each of the N time series: $(\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_N)$ where $\hat{X}_j = (\hat{x}_j^1, \hat{x}_j^2, \ldots, \hat{x}_j^M)^T, j = 1, \ldots, N$. These output time series are then used to calculate fitness as follows:

1. *Equally weighted sum method (F1)*: The fitness calculation formula for this method is:

$$\text{Fitness}(F1) = \sum_{j=1}^{N} \sum_{i=1}^{M} (|\hat{x}_j^i - x_j^i|)$$

This is the most commonly used method (Marler and Arora 2004) in which all the error items from different objects are considered to have the same significance. As we have showed in Fig. 1, using this method the fitness function can be dominated by the errors of the objects with large values, neglecting the errors of objects with small values. This can prevent the algorithm from finding a good compromise model for all the objects.

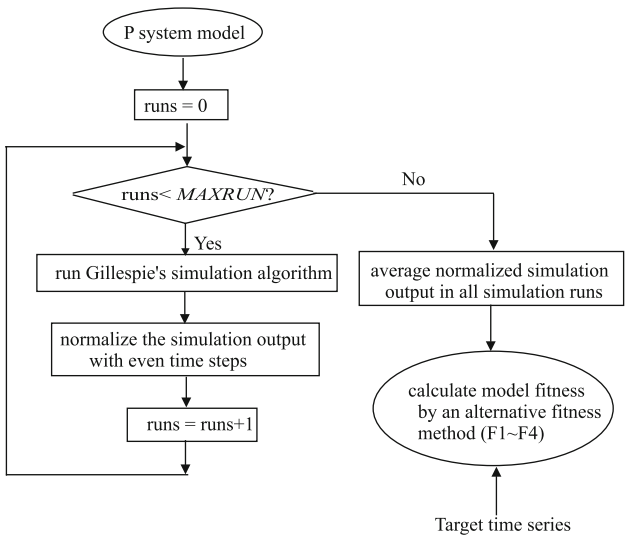

Fig. 6 Fitness evaluation procedure of a P system model

2. *Normalization Method (F2)*: Data normalization is an important data preprocessing technology for many applications. Sola and Sevilla 1997 systematically studied the importance of input data normalization for the application of neural networks to complex industrial problems by experimenting with five different data normalization procedures on the training data set. In essence, data normalization consists in the transformation of the original data into the range [0, 1] in order to make the data comparable at the same level. There are many such transformations. For example, the two formulas below:

$$\hat{f}_i(x) = \frac{f_i(x)}{max\{|f_i(x)|\}} \tag{2}$$

used in Leung and Wang 2000, Thompson et al. 2001 and

$$\hat{f}_i(x) = \frac{f_i(x) - min\{f_i(x)\}}{max\{f_i(x)\} - min\{f_i(x)\}} \tag{3}$$

used in Coello et al. 2002.

In this work we use formula (3) to normalize the absolute error for each data point. Hence the formula to calculate the fitness using the F2 method is as follows:

$$\text{Fitness}(F2) = \sum_{j=1}^{N} \sum_{i=1}^{M} \frac{(|\hat{x}_j^i - x_j^i|) - min(|\hat{x}_j^i - x_j^i|)}{max(|\hat{x}_j^i - x_j^i|) - min(|\hat{x}_j^i - x_j^i|)}$$

The above normalization method removes the saliency of large absolute errors and brings all the time series and their misfit values into an equal footing. On the other hand, by compressing all the data into a [0, 1] interval some of the time series subtleties might be lost.

3. *Randomly weighted sum method (F3)*: This method is similar to F1 but instead of assuming equal contribution from all the errors, here they are adjusted according to a normalized weight vector generated randomly.

A weight vector $(w_1, w_2, \ldots, w_N)$ is called normalized when it meets the following condition:

$$\forall j w_j \geq 0 \text{ and } \sum_{j=1}^{N} w_j = 1$$

This method was proposed by Ishibuchi and Murata 1998 to deal with the case of fitness functions that are

composed of a weighted sum of partial objectives. They argued that this method can provide multiple randomly generated search directions towards the Pareto frontier in multi-objective optimization problems. Jaszkiewicz 2002 adapted the method for solving the multiple-objective 0/1 knapsack problem. In Ishibuchi and Murata 1998, the normalized weight vectors were obtained by generating J random weights from [0,1] with uniform distribution and then by dividing each of the weights by their sum. As this approach does not assure uniform sampling of the normalized weight vectors, Jaszkiewicz proposed the following algorithm in Jaszkiewicz 2002 to ensure that the weights vectors are drawn with uniform probability distribution:

$$\begin{aligned}\lambda_1 &= 1 - \sqrt{[J-1]}rand()\\ &\cdots\\ \lambda_j &= \left(1 - \sum_{l=1}^{j-1}\lambda_l\right)\left(1 - \sqrt{[J-1-j]}rand()\right)\\ &\cdots\\ \lambda_J &= 1 - \sum_{l=1}^{J-1}\lambda_l\end{aligned}$$

where function *rand*() returns a random value within the range (0,1) with uniform probability distribution. In this paper, we use the algorithm above to randomly generate a normalized weight vector and to then obtain the weighted sum of all errors as the fitness value. We repeat this procedure K times and compute the average as the final fitness. Thus, the fitness calculation formula for this method is as follows:

$$\text{Fitness}(F3) = \frac{\sum_{n=1}^{K}\sum_{j=1}^{N}(w_j^n \sum_{i=1}^{M}(|\hat{x}_j^i - x_j^i|))}{K}$$

where w_j^n is the random weight for the jth target time series generated at the nth time.

4. *Equally weighted product method (F4)*: This fitness method is obtained by multiplying all the error items for each target time series and the fitness calculation formula is:

$$\text{Fitness}(F4) = \prod_{j=1}^{N}\sum_{i=1}^{M}(|\hat{x}_j^i - x_j^i|)$$

Bridgman 1922 was the first author to refer to this approach and later Gerasimov and Repko 1978 successfully applied this method to the multi-objective optimization of a truss. A related idea was pursued by Straffin 1993. Mazumdar et al. 1991 used this fitness function to solve problems of optimal network flow in complex telecommunications networks. Cheng and Li 1996 applied this method to a three-story steel shear frame with four objective functions. The main reason why we consider this approach as a potential fitness method is that with a product of terms, it is not necessary to ensure that errors of different target objects have similar magnitude. That is, even relatively small errors can have a significant effect on the final fitness value. A caveat, however, of any product-type fitness function is that it can introduce nonlinearities and numerical instabilities.

Experiments

Case studies definition

In order to benchmark our methodology, four test cases have been selected. These are gene regulatory networks of increased complexity that start with relatively simple negative and positive autoregulation cases and follows with gene networks that implement a pulse generator and a bandwidth detector. The target time series for all these case studies were generated in silico by simulating the target models and then using only the obtained time series to attempt to reverse engineer the circuits that gave rise to the various datasets. More specifically, and as a proof of concept, we start by studying networks consisting of a single gene regulating itself. Although autoregulation is a very simple mechanism, it has been shown to be a highly recurrent pattern in *E. coli* (Thieffry et al. 1998). It consists of a gene whose protein product regulates its own transcription either by repression, *negative autoregulation*, or enhancement, *positive autoregulation*. In this paper we study these two mechanisms first and check what kinds of P system models our algorithm can suggest. The third case study investigates regulatory networks consisting in three genes that are able to produce a pulse in the expression of a specific gene. This type of networks has been shown to be a recurrent pattern or motif in transcriptional regulation of cellular systems (Mangan and Alon 2003). A pulse generating synthetic network has also been designed and implemented in *E. coli* (Basu et al. 2004). The target time series used in this case were obtained by simulating this synthetic network. The last case study is the most complex one as it consists in the investigation of networks with five genes behaving as a bandwidth detector. More specifically, the network should be able to detect a signal within a specific range and produce as a response the expression of a specific gene. A specific such network has been synthetically designed and implemented in *E. coli* (Basu et al. 2005). As in the previous case study we simulated this

Table 1 Benchmark models generating the target time series

Test cases	Target models	Initial values
Test case 1	$\Pi = (m_1, m_2)$	Gene1 = 1
	$m_1 = UnReg(\{X = 1\}, \{\mathbf{c\_1=0.13,c\_2=0.04,c\_3=0.002,c\_4=0.000578}\})$	
	$m_2 = NegReg(\{X = 1, Y = 1\}, \{\mathbf{c\_1=0.056,c\_2=0.147}\})$	
	Simulation time: 6,000 s	
	Iinterval: 10 s	
Test case 2	$\Pi = (m_1, m_2)$	Gene1 = 1
	$m_1 = UnReg(\{X = 1\}, \{\mathbf{c\_1=0.0004,c\_2=0.016,c\_3=0.006,c\_4=0.0001}\})$	
	$m_2 = PosReg(\{X = 1, Y = 1\}, \{\mathbf{c\_1=0.04,c\_2=0.02,c\_3=0.014,c\_4=0.016,c\_5=0.006,c\_6=0.0001}\})$	
	simulation time: 30,000 s	
	Interval: 50 s	
Test case 3	$\Pi = (m_1, m_2, m_3, m_4)$	Gene1 = 1
	$m_1 = UnReg(\{X = 1\}, \{c_1 = 4.5, c_2 = 1, c_3 = 0.15, c_4 = 0.6\})$	Gene2 = 1
	$m_2 = PosReg(\{X = 1, Y = 2\}, \{c_1 = 1, \mathbf{c\_2=100,c\_3=5}, c_4 = 1, c_5 = 0.15, c_6 = 0.6\})$	Gene3 = 1
	$m_3 = PosReg(\{X = 1, Y = 3\}, \{c_1 = 1, \mathbf{c\_2=10,c\_3=8}, c_4 = 1, c_5 = 0.15, c_6 = 0.6\})$	
	$m_4 = NegReg(\{X = 2, Y = 3\}, \{c_1 = 1, \mathbf{c\_2=0.1}\})$	
	Simulation time: 118 min	
	Interval: 1 min	
Test case 4	$\Pi = (m_1, m_2, m_3, m_4, m_5)$	$Plac::gFP = 1$
	$m_1 = UnReg(\{X = LuxR\}, \{c_1 = 0.15, c_2 = 0.004, c_3 = 0.03, \mathbf{c\_4=0.001}\})$	$PR::gLacI = 1$
	$m_2 = PluxR(\{X = LacI\}, \{c_1 = 0.1, c_2 = 0.175, c_3 = 1, c_4 = 0.0063, c_5 = 1, c_6 = 0.0063, c_7 = 0.01875,$	$Plux::gCI = 1$
	$c_8 = 1, c_9 = 1, \mathbf{c\_\{10} = 0.001, c_{11} = 0.004, c_{12} = 0.03, \mathbf{c\_\{13} = 0.001\})$	$Plux::gLacI = 1$
	$m_3 = PluxR(\{X = CI\}, \{c_1 = 0.1, c_2 = 0.175, c_3 = 1, c_4 = 0.0063, c_5 = 1, c_6 = 0.0063, c_7 = 0.01875,$	$gLuxR = 1$
	$c_8 = 1, c_9 = 1, \mathbf{c\_\{10} = 0.1, c_{11} = 0.004, c_{12} = 0.03, \mathbf{c\_\{13} = 0.001\})$	$s3OC6ext = 5$
	$m_4 = PR(\{X = LacI\}, \{c_1 = 0.15, c_2 = 0.004, c_3 = 0.03, \mathbf{c\_4=0.001}, c_5 = 0.000166, c_6 = 0.002,$	
	$c_7 = 0.166, c_8 = 0.002, c_9 = 0.0083, c_{10} = 0.0002\})$	
	$m_5 = Plac(\{X = FP\}, \{c_1 = 0.15, c_2 = 0.004, c_3 = 0.03, \mathbf{c\_4=0.001}, c_5 = 0.000166, c_6 = 0.01,$	
	$c_7 = 11.6245, c_8 = 0.06\ \})$	
	Simulation time: 3,600 s	
	Interval: 36 s.	

The evolutionary algorithm must evolve, guided by one of the four alternative fitness functions, both the structure and parameters for each target time series. Constants highlighted in bold are to be evolved, the rest are input to the algorithm

synthetic network in order to obtained the target time series.

The details of the target models for the four test cases are shown in Table 1. If the initial value of the object is not listed in the table, it is set to the default value 0. Figure 7 illustrates the topologies of the four target models where the increasing complexity of the models from Test case 1 to Test Case 4 is evident. The parameters that were allowed to evolve in our study were chosen according to the possibility of performing directed evolution over them in the lab. For example in Test case 3, the parameters that were allowed to evolve were those representing the debinding of transcription factors from the promoters of the corresponding genes (c_2 from *PosReg* and *NegReg*) and the transcription initiation (c_3 from *PosReg*). These parameters can be easily altered in the lab by performing one point mutations in the promoter of the corresponding genes which weakens or stregthens the affinity of the corresponding transcription factors and the RNAP to the promoter. In test case 4, we also allowed to evolve parameters representing transcription initiation (c_{10} from *PluxR*) as described above. Here we also explored another possibility for directed evolution. We allowed to evolve parameters corresponding to the degradation rates of certain proteins (c_{13} from *PluxR* and c_4 from *UnReg*, *PR* and *Plac*). This can also be done in the lab by tagging the natural protein with specific sequences that are targeted for degradation by proteases.

Parameter settings and measures

Table 2 presents the parameter values for the experiments conducted. We run a total of 320 experiments: 20 independent runs for each of the four case studies under each of

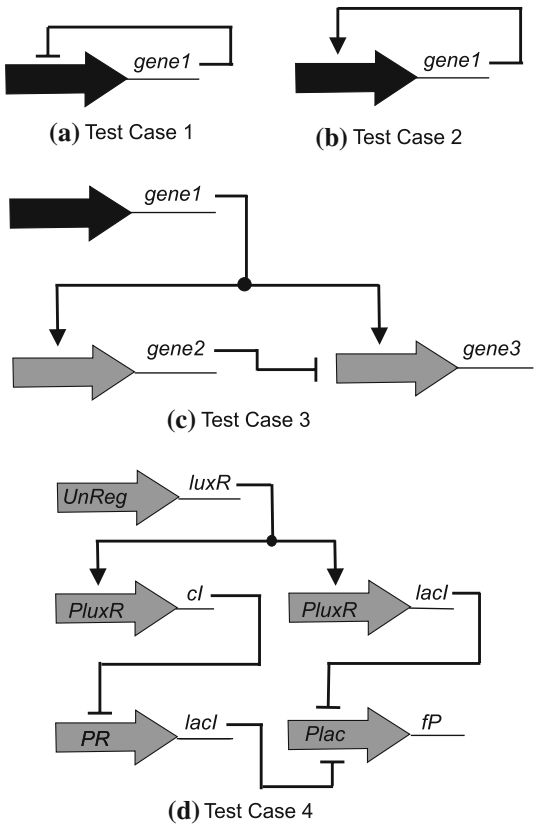

Fig. 7 Target models' topologies. In a), b), c), the black shaded big arrow represents constitutive expression of the gene. An arrow with (→) from a gene to another one represents positive regulation whereas an arrow with (⊣) represents negative regulation

the four fitness methods introduced in the previous section. All the experiments were performed on the Jupiter supercomputer of the University of Nottingham with a 1,024 CPU 2.2GHz gigabit cluster and the Linux operating system.

In addition, for each case study the constant range and precision associated with each rule in the module are listed in Table 3. Those values determine the search space for the GA used in the parameter optimization. If the constant is not listed in the table, it is fixed to the value in the target model. The constant range can be defined on a linear or logarithmic scale. In the linear scale the constant can take any value between the lower and the upper bound with the given precision. In the logarithmic scale, we are more interested in the order of magnitude of the corresponding constants. Therefore, the constant can take any power of ten whose exponent is between the specified lower and upper bound with the given precision.

In test case 4 almost all the kinetic constants are known since the modules here represent gene promoters that are widely used in synthetic biology. Although our focus in this case is structure optimization we also enable five constants to be tunable in order to allow our algorithm to explore mutations in the promoters to optimize the behavior of the system.

Results and discussions

In this section we present the results on the application of the evolutionary algorithm we propose to the four test cases discussed above.

Results for test case 1

The first row of Table 4 shows that for test case 1 all the best models obtained using the four different fitness methods have the same structure as the target model differing only in the stochastic kinetic constants.

Figure 8 shows the simulation results for the best models obtained using $F1, \ldots, F4$[1]. Note that the model obtained using F3 fits best the behavior of the target model with an RMSE value of 1.69. The model found by F4 is also very good whereas F1 and F2 produce slightly worst results.

As mentioned above these models share the same structure and differ only in the stochastic constants. In Table 5 we compare the stochastic constants in each model to the ones in the target model using the *relative error* computed following the formula below.

$$\frac{|\text{modelvalue} - \text{targetvalue}|}{\text{targetvalue}} \times 100\%$$

Observe that the fitness method F3 produces the best estimates for the constants in the target model with the exception of c_1 in the module *UnReg* which is in agreement with the results shown in Fig. 8.

Results for test case 2

Table 12 shows that for this test case the fitness methods F1, F3 and F4 found a model with the same structure as the target model in all the 20 independent runs whereas F2 only found a model with this structure in 15% of the runs. In most of the runs, method F2 found the following modular structure $\{UnReg, NegReg\}$ instead of the target $\{UnReg, PosReg\}$.

Figure 9 depicts the simulation results for the best models obtained using $F1, \ldots, F4$. The models found by F1, F3 and F4 only differ in the constants associated with the modules. When comparing only these three methods we can observe that F1 is the one that best matches the

[1] Note that if the model has the same structure as the target model, we mark it with a "*" on the right hand of the model graph.

Table 2 The parameter settings of the nested evolutionary algorithm

Two GAs	Parameters	Values	Meaning
GA for structure optimization	*POPSIZE*	50	Model population size
	SOMAXGENO	20	Maximal number of generations
	MAXMSIZE	6	Maximal number of modules in a model
	MAXRUN	50	Number of simulation runs to calculate the model fitness
	K	100	Number of times to produce the random weights for F3
GA for parameter optimization	*POPSIZE*	50	Parameter population size
	MAXHCSTEPS	50	Maximal number of steps to do hill climbing
	POMAXGENO	100	Maximal number of generations
	M	8	Number of selected parents to do the crossover
	a	−0.5	Lower bound of the random coefficients in crossover
	b	1.5	Upper bound of the random coefficients in crossover

Table 3 The range and the precision of the kinetic constants in the rule set of the modules for four test cases

Test cases	Module name	Constants	Scale	Range	Precision
Test case 1& Test case 2	*UnReg*	c_1	Linear	(0,0.2)	10^{-4}
		c_2	Linear	(0,0.05)	10^{-3}
		c_3	Linear	(0,0.01)	10^{-3}
		c_4	Linear	(0,0.001)	10^{-6}
	PosReg	c_1	Linear	(0,0.1)	10^{-3}
		c_2	Linear	(0,0.2)	10^{-3}
		c_3	Linear	(0,0.1)	10^{-3}
		c_4	Linear	(0,0.05)	10^{-3}
		c_5	Linear	(0,0.01)	10^{-3}
		c_6	Linear	(0,0.001)	10^{-6}
	NegReg	c_1	Linear	(0,0.1)	10^{-3}
		c_2	Linear	(0,0.2)	10^{-3}
Test case 3	*PosReg*	c_2	Linear	(0,200)	10^{-1}
		c_3	Linear	(0,10)	10^{-1}
	NegReg	c_2	Linear	(0,200)	10^{-1}
Test case 4	*UnReg*	c_4	Logarithmic	[−3,−1]	1
	PluxR	c_{10}	Logarithmic	(−3,1)	1
		c_{13}	Logarithmic	[−3, −1]	1
	PR	c_4	Logarithmic	[−3, −1]	1
	Plac	c_4	Logarithmic	[−3, −1]	1

evolution of *protein*1 whereas it produces the worst results for *rna*1. F4 is the one that performs best for the case of *rna*1 and F3 is the secondary.

The reason behind the assymetry in the performance of F1 for *protein*1 and *rna*1 is the big difference in the orders of magnitude between the two time series. The target for *protein*1 is within the range [0, 350] while the target for *rna*1 is within [0, 2]. Since the method F1 calculates the fitness value using an equally weighted sum of the errors for *protein*1 and *rna*1 it is very likely to find some models with a very small combined error which fits *protein*1 very well but not *rna*1. Actually, as shown in Fig. 9, the RMSE of the best F1 model is the smallest one with a value of 6.78 even though its simulation of *rna*1 is poor.

The method F4 calculates the fitness value as the product of the errors for *protein*1 and *rna*1 which makes both errors to contribute equally to the final fitness value despite their different scales. As expected, the simulation result of *rna*1 is improved significantly at the cost of slightly degrading the fitting accuracy of *protein*1 as can been seen for the best F4 model shown in Fig. 9. This model can be chosen as the best one for test case 2 as it presents a good compromise in the simulation of both *protein*1 and *rna*1.

The fitness method F3 also performs well for this test case as it can be observed in Fig. 9, which also presents a good compromise results for *protein*1 and *rna*1. It shows that this method has the potential to generate good

Table 4 The best evolved models (out of 20 runs) under different fitness methods for the four benchmarks

Test cases	Fitness methods	Best fitness model structure	As target(Y/N)
Test case 1	F1, $\cdots$, F4	$\Pi = (m_1, m_2)$	Y
		$m_1 = UnReg\{X = 1\}$	
		$m_2 = NegReg\{X = 1, Y = 1\}$	
Test case 2	F1, F3, F4	$\Pi = (m_1, m_2)$	Y
		$m_1 = UnReg\{X = 1\}$	
		$m_2 = PosReg\{X = 1, Y = 1\}$	
	F2	$\Pi = (m_1, m_2)$	N
		$m_1 = UnReg\{X = 1\}$	
		$m_2 = NegReg\{X = 1, Y = 1\}$	
Test case 3	F1, F4	$\Pi = (m_1, m_2, m_3, m_4)$	Y
		$m_1 = UnReg\{X = 1\}$	
		$m_2 = PosReg(\{X = 1, Y = 2\}$	
		$m_3 = PosReg\{X = 1, Y = 3\}$	
		$m_4 = NegReg\{X = 2, Y = 3\}$	
	F2	$\Pi = (m_1, m_2, m_3, m_4)$	N
		$m_1 = UnReg\{X = 1\}$	
		$m_2 = UnReg\{X = 2\}$	
		$m_3 = PosReg\{X = 2, Y = 3\}$	
		$m_4 = NegReg\{X = 1, Y = 2\}$	
	F3	$\Pi = (m_1, m_2, m_3, m_4)$	N
		$m_1 = UnReg\{X = 1\}$	
		$m_2 = UnReg\{X = 3\}$	
		$m_3 = PosReg\{X = 1, Y = 2\}$	
		$m_4 = NegReg\{X = 1, Y = 3\}$	
Test case 4	F1, F4	$\Pi = (m_1, m_2, m_3, m_4, m_5)$	Y
		$m_1 = UnReg\{X = LuxR\}$	
		$m_2 = PluxR\{X = LacI\}$	
		$m_3 = PluxR\{X = CI\}$	
		$m_4 = PR\{X = LacI\}$	
		$m_5 = Plac\{X = FP\}$	
	F2	$\Pi = (m_1, m_2, m_3, m_4, m_5)$	N
		$m_1 = PluxR\{X = LacI\}$	
		$m_2 = UnReg\{X = LuxR\}$	
		$m_3 = PR\{X = CI\}$	
		$m_4 = Plac\{X = FP\}$	
		$m_5 = PluxR\{X = CI\}$	
	F3	$\Pi = (m_1, m_2, m_3, m_4, m_5, m_6)$	N
		$m_1 = UnReg\{X = LuxR\}$	
		$m_2 = PluxR\{X = LacI\}$	
		$m_3 = PluxR\{X = CI\}$	
		$m_4 = PR\{X = LacI\}$	
		$m_5 = Plac\{X = FP\}$	
		$m_6 = PluxR\{X = LuxR\}$	

compromise solutions as it explores the search space in different directions by randomly generating the weights of the fitness function.

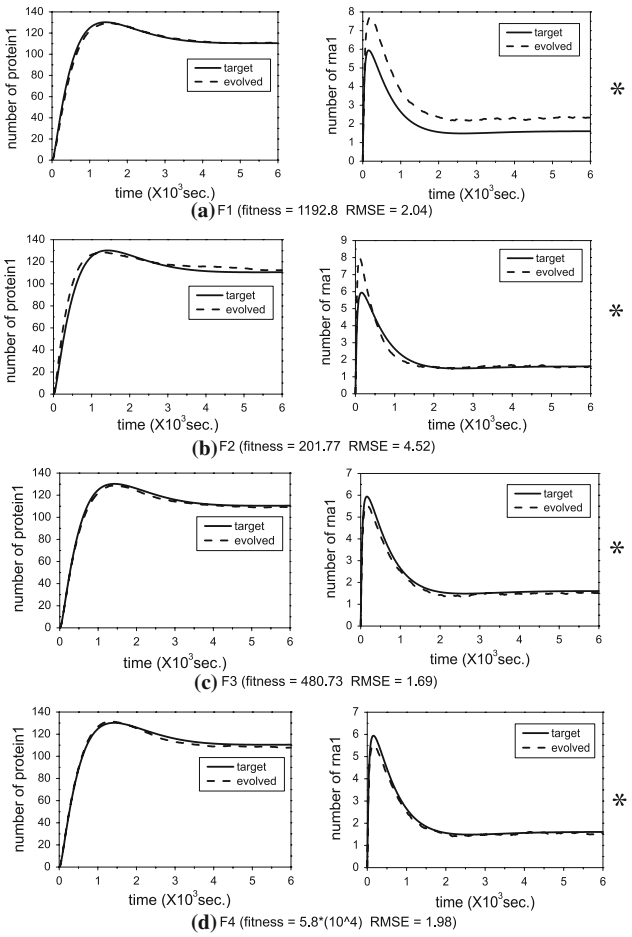

Fig. 8 Simulated results of the best fitness models for test case 1 obtained by four fitness methods $(F1, \ldots, F4)$

Finally, the simulations in Fig. 9 show that the alternative model found by F2 fails completely to reproduce the targeted behavior of both *protein*1 and *rna*1.

Since the best models found by the F1, F3 and F4 methods share the same structure as the target model we compare the relative error of the constants in these models to the ones in the target, see Table 6. In total there are ten adjustable parameters and the F4 method is the one that gets closer to them in terms of relative error. The differences in the simulation results for these three models sharing the same structure but with different constants further illustrate the importance of parameter optimization when modelling cellular systems.

Results for test case 3

As shown in Table 4 for this case study only the best models found by using F1 and F4 share the same structure with the target model. The F2 and F3 methods found models with an alternative structure which interestingly is

Table 5 Comparisons of the constants between the best fitness models obtained by F1, . . ., F4 and the target model for test case 1 (Best results are bolded)

Module set	Const. name	F1		F2		F3		F4		Target value
		Value	RE(%)	Value	RE(%)	Value	RE(%)	Value	RE(%)	
UnReg	c_1	0.1263	**2.85**	0.1737	33.62	0.1222	6	0.1251	3.77	0.13
	c_2	0.029	27.5	0.038	**5**	0.042	**5**	0.044	10	0.04
	c_3	0.002	**0**	0.003	50	0.002	**0**	0.002	**0**	0.002
	c_4	0.000612	5.88	0.000542	6.23	0.000581	**0.52**	0.000634	9.69	0.000578
NegReg	c_1	0.032	42.86	0.01	82.14	0.078	**39.29**	0.079	41.71	0.056
	c_2	0.131	10.88	0.031	78.91	0.2	**36.05**	0.2	**36.05**	0.147

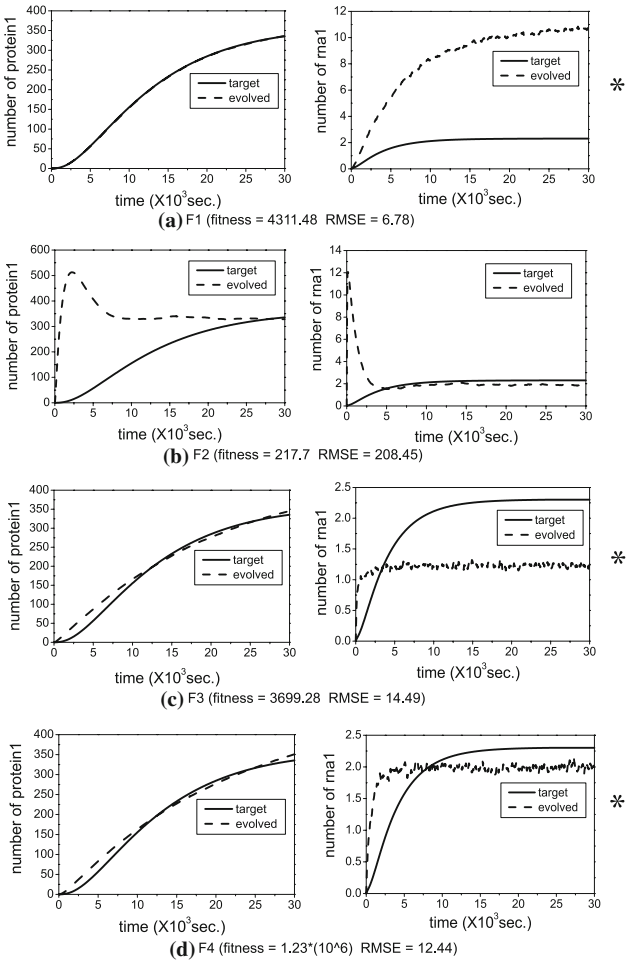

Fig. 9 Simulated results of the best fitness models for test case 2 obtained by four fitness methods (F1, . . ., F4)

the same for both methods differing in the instantiation of the variables that represent the objects.

The simulation results for the best models found using F1, . . ., F4 are shown in Fig. 11. Note that for brevity's sake we only present the dynamics of *protein*1, *protein*2 and *protein*3 since the dynamics of the corresponding *rna*'*s* is very similar to them differing only in the magnitude.

We start by comparing the best models found using F1 and F4 which have the same structure as the target but with different model parameters. The simulation results of *protein*1 and *protein*2 are quite similar for these two models and they almost coincide with the target time series. Nevertheless, the behavior of *protein*3 in both models is quite different. In the target time series the expression of *protein*3 presents a pulse that is only reproduced in the model obtained using the F1 model although with a 25% under-estimation, which results in a small RMSE with a value of 1.91. In contrast, *protein*3 follows saturating dynamics in the model obtained using F4 which produced a RMSE with a value of 6.56. Again, these results suggest the great effect of the constants contained in a model on the expected behavior of the simulated cellular system. In Table 7 we compare the different constants of the models sharing the same structure found using F1 and F4. This table shows that all the constant values of the best model found using F1 are much closer to the target values than those of the model found with F4 which explains why the former model performs better as illustrated in Fig. 11.

Interestingly, the algorithm found another two alternative model structures that are biologically plausible using the fitness methods F2 and F3. Fig. 10 depicts a graphical representation of their topologies. When comparing the behavior of these models to the target time series shown in Fig. 11, we observe that the model found using F3 is fairly good as it reproduces very well the dynamics of *protein*1 and *protein*2 and it also presents a small pulse in the expression of *protein*3. This is not the case of the alternative model found using F2 which fails to reproduce the dynamics of *protein*2 and *protein*3.

Results for test case 4

Table 4 shows that, as in test case 3, the methods F1 and F4 found a model with the same structure as the target whereas F2 and F3 discovered alternative model structures. The alternative model found using F2 differs from the target in

Table 6 Comparisons of the constants between the best fitness models obtained by F1, F3, F4 and the target model for test case 2 (Best results are bolded)

Module set	Const. name	F1		F3		F4		Target value
		Value	RE(%)	Value	RE(%)	Value	RE(%)	
UnReg { $X = 1$ }	c_1	0.0003	**25**	0.0076	1800	0.0055	1275	0.0004
	c_2	0.005	68.75	0.018	**12.5**	0.011	31.25	0.016
	c_3	0.005	**16.67**	0.008	33.33	0.007	**16.67**	0.006
	c_4	0.00016	60	0.000049	51	0.00005	**50**	0.0001
PosReg { $X = 1, Y = 2$ }	c_1	0.09	125	0.003	**25**	0.004	90	0.04
	c_2	0.048	140	0.043	115	0.013	**35**	0.02
	c_3	0.054	286	0.01	28.57	0.014	**0**	0.014
	c_4	0.005	68.75	0.018	**12.5**	0.011	31.25	0.016
	c_5	0.005	**16.67**	0.008	33.33	0.007	**16.67**	0.006
	c_6	0.00015	**50**	0.000049	51	0.000046	54	0.0001

Table 7 Comparisons of the constants between the best fitness models obtained by F1, F4 and the target model for test case 3 (Best results are bolded)

Module set	Const. name	F1		F4		Target value
		Value	RE(%)	Value	RE(%)	
PosReg { $X = 1, Y = 2$ }	c_2	132	**32**	42	58	100
	c_3	6	**20**	3	40	5
PosReg{ $X = 1, Y = 3$ }	c_2	23	**130**	139	1290	10
	c_3	9	**12.5**	1	87.5	8
NegReg { $X = 2, Y = 3$ }	c_2	0.2	**100**	123	122900	0.1
UnReg { $X = 1$ }	c_i	All are fixed				

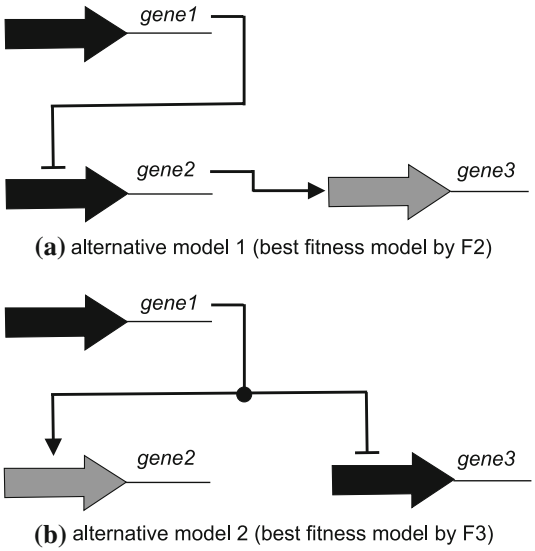

Fig. 10 Topologies of the two alternative models obtained by F2 and F3 for Test Case 3. The *black shaded big arrow* represents constitutive expression of the gene. An *arrow* with (→) from a gene to another one represents positive regulation whereas an *arrow* with (⊣) represents negative regulation

the module *PR* which is instantiated using *CI* instead of *LacI* and the model found using F3 includes an additional module *Plux* instantiated with *LuxR*.

The simulation results of the four best models found using F1, . . ., F4 are shown in Fig. 12. The models found using F1 and F4 perfectly match the four target time series. As they share the same model structure with the target we compare their model constants with the ones in the target, see Table 8. The constants that are different from the ones in the target model are highlighted and underlined. Note that the best model found using F1 has the same constants as the target whereas two of the seven tunable constants in the model found using F4 differ from the ones in the target. Since the latter model also perfectly reproduces the target time series we conclude that the target model is not sensitive to changes in the constant c_4 of module *UnReg* and the constant c_{13} of module *PluxR*.

As for the alternative model structure discoverd by F2, Fig. 12 shows that it fails to reproduce the dynamics of the target model. In contrast, the model found by F3 can be regarded as a good alternative model based on accurate match to all four target objects shown in Fig. 12.

It is worth mentioning that although the above three good models are obtained by different fitness methods (F1, F3 and F4), they all consistently achieve good simulation results and their RMSEs are very small, with values of 0.55, 1.05, and 1.03 respectively. This demonstrates the effectiveness of our algorithm in searching the global optimum from different directions.

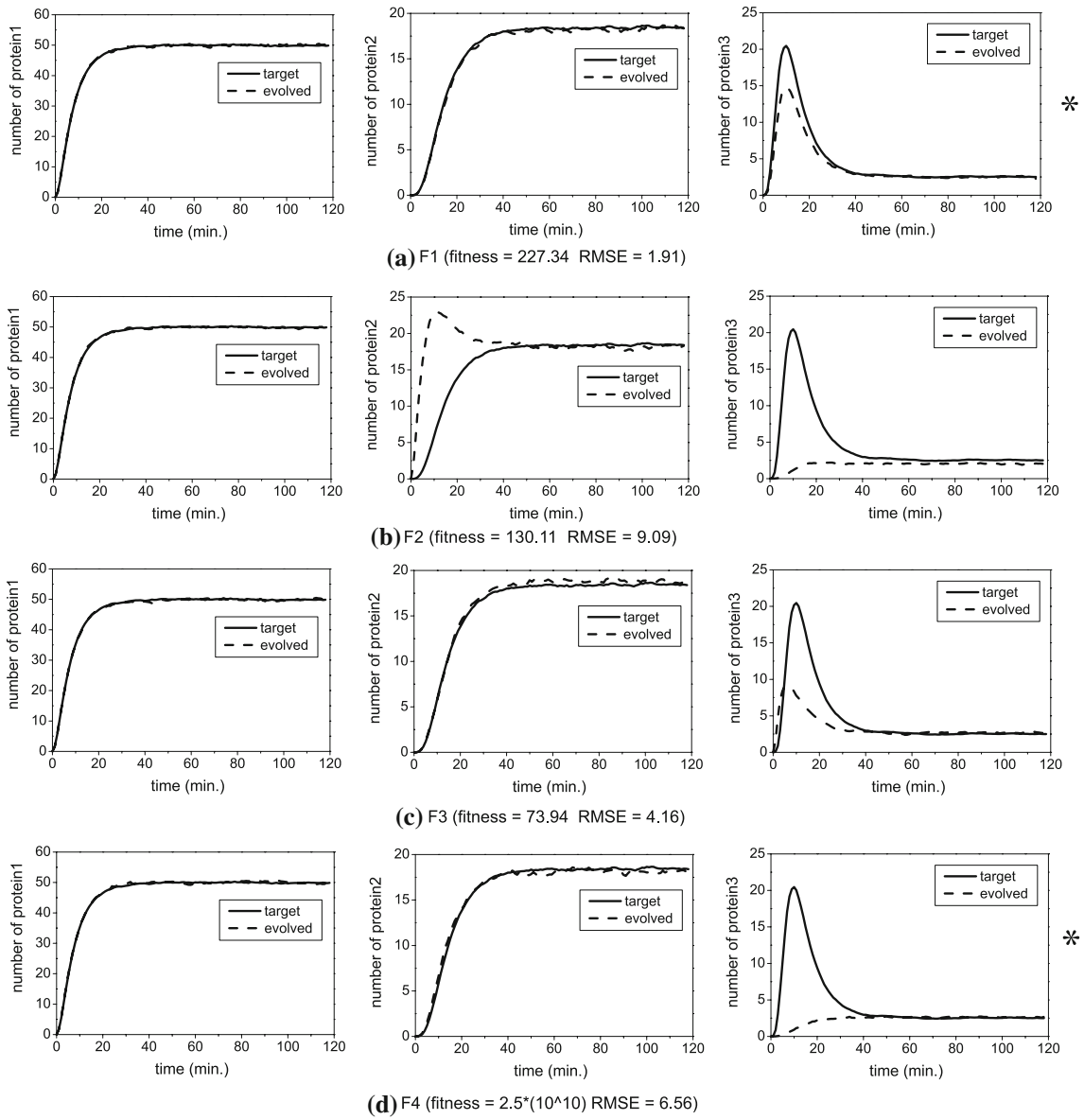

Fig. 11 Simulated results of proteins of the best fitness models for test case 3 obtained by four fitness methods $(F1, \ldots, F4)$

We are interested in examining how many alternative good models can be automatically found using our algorithm. By checking each model structure and their RMSE in 20 runs for each method, we obtainged another three alternative good models in terms of small RMSE besides the best model found by F3 as previously discussed. The details of these four alternative models are listed in Table 9 and their topologies are illustrated in Fig. 13. The common feature of these models is that all of them have an additional module besides the five modules in the target. As shown in Fig. 14, despite the variety of the additional modules, their simulation results are very similar and match the target accurately.

More interestingly, our algorithm frequently found another alternative model using F1, F3 and F4 as illustrated in Fig. 15, which has a simpler structure than the target excluding the module *PluxR* instantiated with *LacI*. Table 10 lists the statistical results for this model structure found by each fitness method. For each method, the average and standard deviation of the fitness and RMSE are calculated from the runs that found this model structure neglecting the model constants. This table shows that the RMSE values for F1, F3, and F4 are around six whereas the RMSE value for F2 is extremely large, 246.12. The simulation results for the best models with this alternative structure discovered using $F1, \ldots, F4$ are depicted in Fig. 16. As expected, the F2 model does not match the target time series, especially *pLacI* and *pFP*. However the simulation results for the F1, F3, and F4 models show how accurately this alternative structure can reproduce the target.

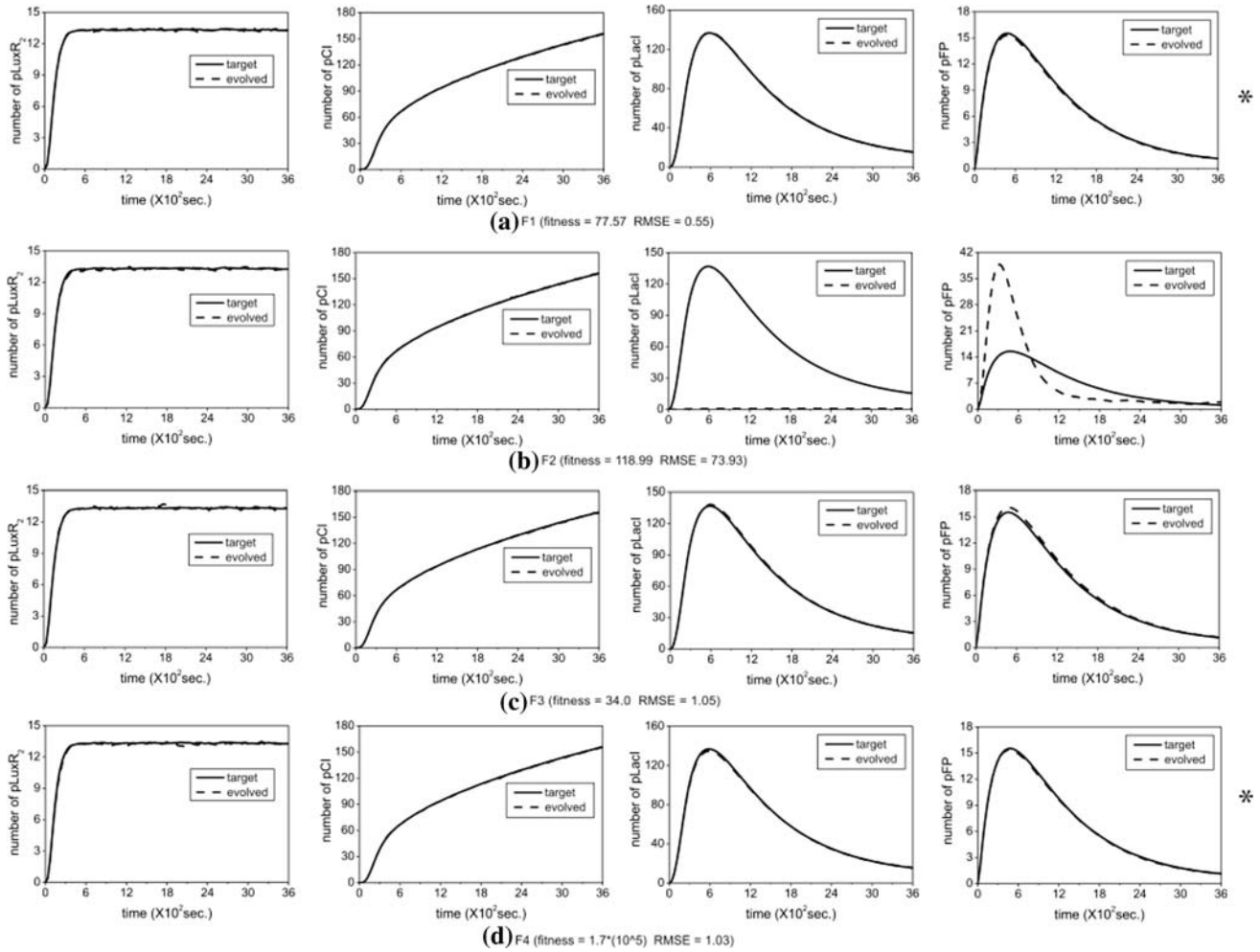

Fig. 12 Simulated results for the fittest models obtained by the four fitness methods $(\mathrm{F1}, \ldots, \mathrm{F4})$ in Test Case 4

Table 8 Comparisons of the constants between the best fitness models obtained by F1, F4 and the target model for test case 4 (Values different from the target are bolded and underlined)

Module set	Constants	F1	F4	Target
$UnReg\{X = LuxR\}$	c_4	0.001	**0.01**	0.001
$PluxR\{X = LacI\}$	c_{10}	0.001	0.001	0.001
	c_{13}	0.001	0.001	0.001
$PluxR\{X = CI\}$	c_{10}	0.1	0.1	0.1
	c_{13}	0.001	**0.01**	0.001
$PR\{X = LacI\}$	c_4	0.001	0.001	0.001
$Plac\{X = FP\}$	c_4	0.001	0.001	0.001

When comparing the stochastic constants, see Table 11, we noticed that the model found by F1 has the same constants as the target, whereas the F3 and F4 models have different c_4 in the module *UnReg*. Since the behavior of these models is extremely similar we conclude that this structure is robust for changes in this constant. In contrast, we observed that the constants found by F2 differ in the c_4 in the module *PR* which suggests that the structure is very sensitive to changes in this constant.

Table 9 Four alternative models found by F1, F3, F4 for test case 4

No.	Model structure	F	Fitness	RMSE
1	$\Pi = (m_1, m_2, m_3, m_4, m_5, m_6)$ $m_1, \ldots, m_5$: same as target $m_6 = Plac\{X = LuxR\}$	F1	85.84	0.57
2	$\Pi = (m_1, m_2, m_3, m_4, m_5, m_6)$ $m_1, \ldots, m_5$: same as target $m_6 = PR\{X = FP\}$	F1	99.61	0.69
3	$\Pi = (m_1, m_2, m_3, m_4, m_5, m_6)$ $m_1, \ldots, m_5$: same as target $m_6 = PluxR\{X = LuxR\}$	F3	34.0	1.05
4	$\Pi = (m_1, m_2, m_3, m_4, m_5, m_6)$ $m_1, \ldots, m_5$: same as target $m_6 = Plac\{X = CI\}$	F4	4.6×10^5	0.91

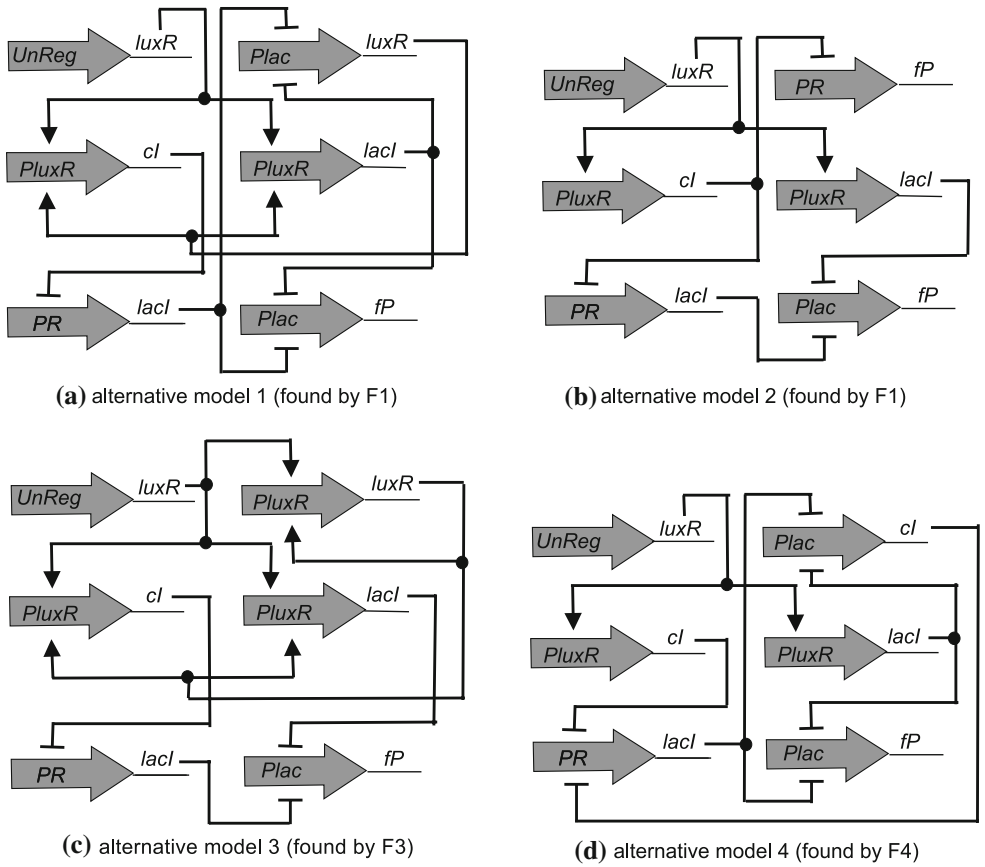

Fig. 13 Topologies of the four alternative models found by F1, F3, F4 for test case 4

Run times and diversity summary

Table 12 provides some statistics on the diversity of evolved models. We noted that the model diversity does not depend much on the fitness method used but only on the complexity of the test case. It is obvious that for more complex cases, the diversity of models increases. More specifically, for the two simple cases, test case 1 and 2, as expected, F1, F3 and F4 always found the target model structure in most runs. On the contrary, although F2 performed well for test case 1, in test case 2 it was able to find the target model only three times out of 20 runs. For the more complicated cases, namely test case 3 and 4, the results are quite different. All the fitness methods can find more than six alternative model structures for test case 3 and more than 14 for test case 4 which seldom have the same structure as the target. Noteworthy, the alternative models found by the evolutionary algorithm are comparable in quality to the target model in terms of RMSD.

Table 13 shows the average running time for the different benchmarks under the various objective functions used, (for reference only) the average fitness achieved is shown as well as the average RMSE for the best model evolved with a given objective function. Only the later value can be used to compare, within a test case, the merit of different objective functions as the fitnesses themselves are not comparable (e.g, as expected, F4 usually leads to very high values). Recall that for test case 1 and 2, the target models consist of only two modules with a total number of six and ten rules respectively. For these two simple cases, each run finished in less than one minute in average. In test case 3, the target model is composed of four modules containing a total number of 18 rules. One run in this case took on average two hours. Finally, in test case 4 the target model contains five modules with 48 rules and took approximately a week of runtime. Please note that these numbers are absolute upper bounds as each run involved a population of 50 individuals for the structural optimisation loop, each of which undergoes a 50 individuals GA run for its parameter optimisation. In turn, each time a model structure or parameter is changed 50 simulations with SSA were undertaken. Indeed, similar results to those reported above could be obtained with only five simulations each (experiments not reported). Hence, although expensive, the results provided can be substantially reduced if needed.

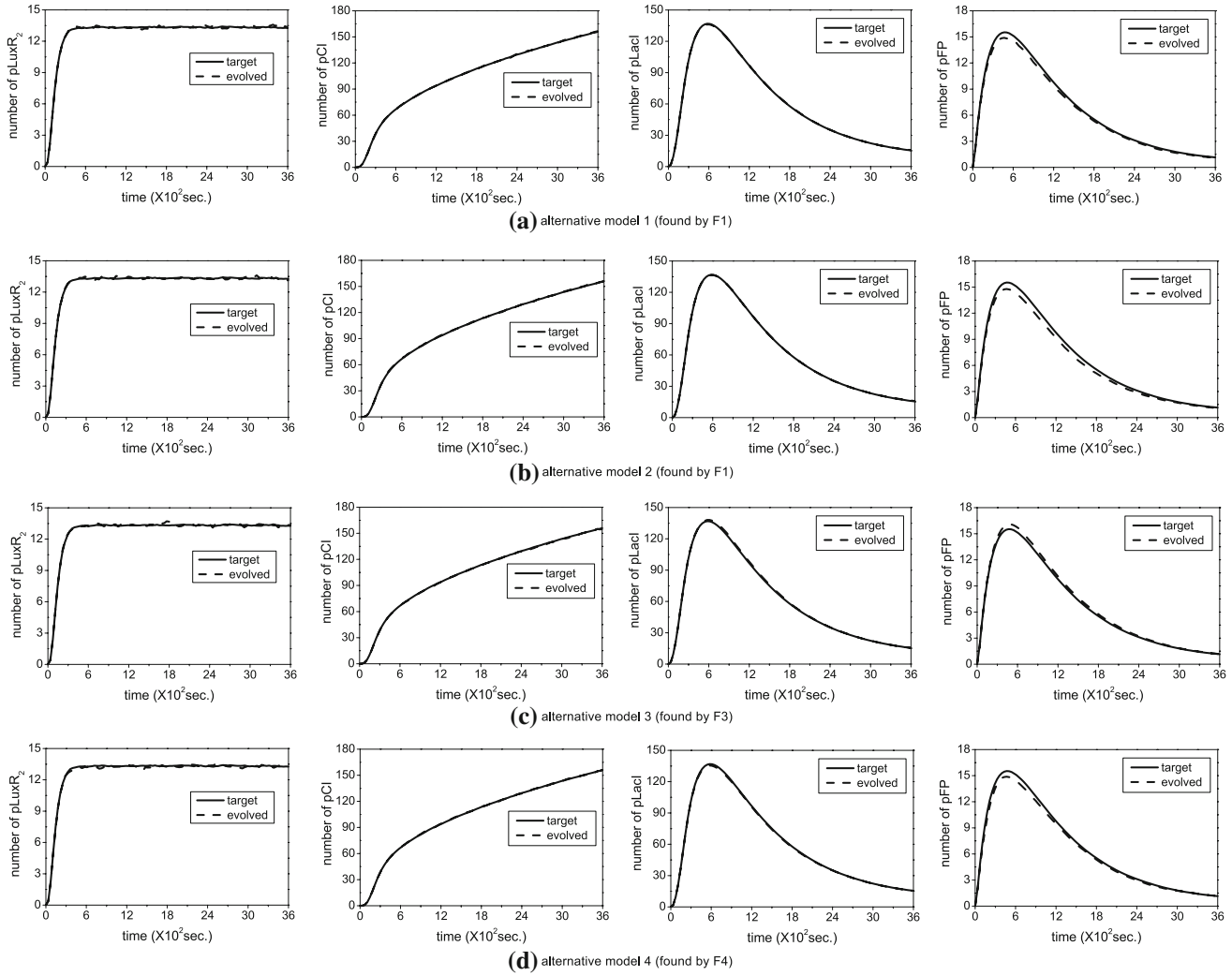

Fig. 14 Simulated results of four alternative models found by F1, F3, F4 for test case 4

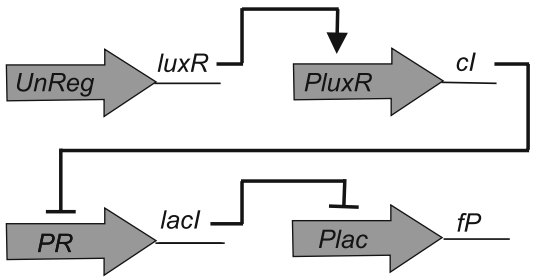

Fig. 15 Topology of the common model structure found by F1, . . . , F4 for Test Case 4

Evolutionary dynamics

Having analysed the quality of the best solutions, overall models' diversity and the run time, we turn next to the evolutionary dynamic of the proposed algorithm. We report results for benchmark four as it is the most challenging. Figures 17 and 18 show the average over 20 runs of the best fitness and average model diversity as a function of generations. As shown in Fig. 17, the evolving curve for F2 fitness is rather flat and the improvement over the first generations is small converging soon to a bad solution at Generation 6. This result supports our previous observations of F2 always performing the worst for all test cases. With respect to the F1 method we observe that the fitness value improves gradually converging to a good solution at Generation 10. It then becomes stable as indicated by the small error bars in the graph. The fitness method F4 produces a faster convergence than F1 becoming stable at Generation 8. Nevertheless, as shown by the bigger error bars, the best solution at the following generations is not as stable as when using F1. Error bars for F3 are also larger across all the generations which can be explained by the stochastic property of the random weighted sum fitness method. However, this method shows the fastest convergence at Generation 6.

Table 10 The statistical results of the common model structure most frequently found by F1, . . ., F4 for test case 4 (Best results are bolded)

Fitness method	Frequency	Fitness	RMSE
F1	4	745.05 ± 82.31	7.01 ± 0.87
F2	1	146.27 ± 0	246.12 ± 0
F3	6	177.21 ± 22.96	$\mathbf{6.23 \pm 0.41}$
F4	4	$(3.07 \pm 0.71) \times 10^6$	6.89 ± 0.45
Model structure	$\Pi = (m_1, m_2, m_3, m_4)$		
	$m_1 = UnReg\{X = LuxR\}$ $m_2 = PluxR\{X = CI\}$		
	$m_3 = PR\{X = LacI\}$ $m_4 = Plac\{X = FP\}$		

Figure 18 shows that unlike the evolution of the fitness, the average diversities are similar for all different fitness methods. The initial populations consists of 40 different model structures generated randomly but subsequently this number drops quickly and remains constant around 25 after generation 6. This result suggests that the different fitness methods tried have only a small effect on population diversity throughout the evolutionary process. The fact that half of the individuals in the population have different model structures, needless to say their various model parameters in all the generations, suggests that our algorithm succeeds on maintaining model diversity during evolution.

Finally, the results of the experiments are analyzed using a one-way ANOVA test combined with a post-hoc Tukey HSD test for multiple comparisons using a 95% confidence level as suggested by Lanzi et al. 2006. The data we use for statistical analysis are the RMSEs in 20 runs for each fitness method and each test case. The results of the statistical test are shown in Tables 14 and 15. The results for each test case are consistent and suggest the following dominance order: $F1 \rightarrow F2$, $F3 \rightarrow F2$, $F4 \rightarrow F2$. That is, F2 is a statistically significant inferior fitness method while the other three methods (F1, F3, F4) are

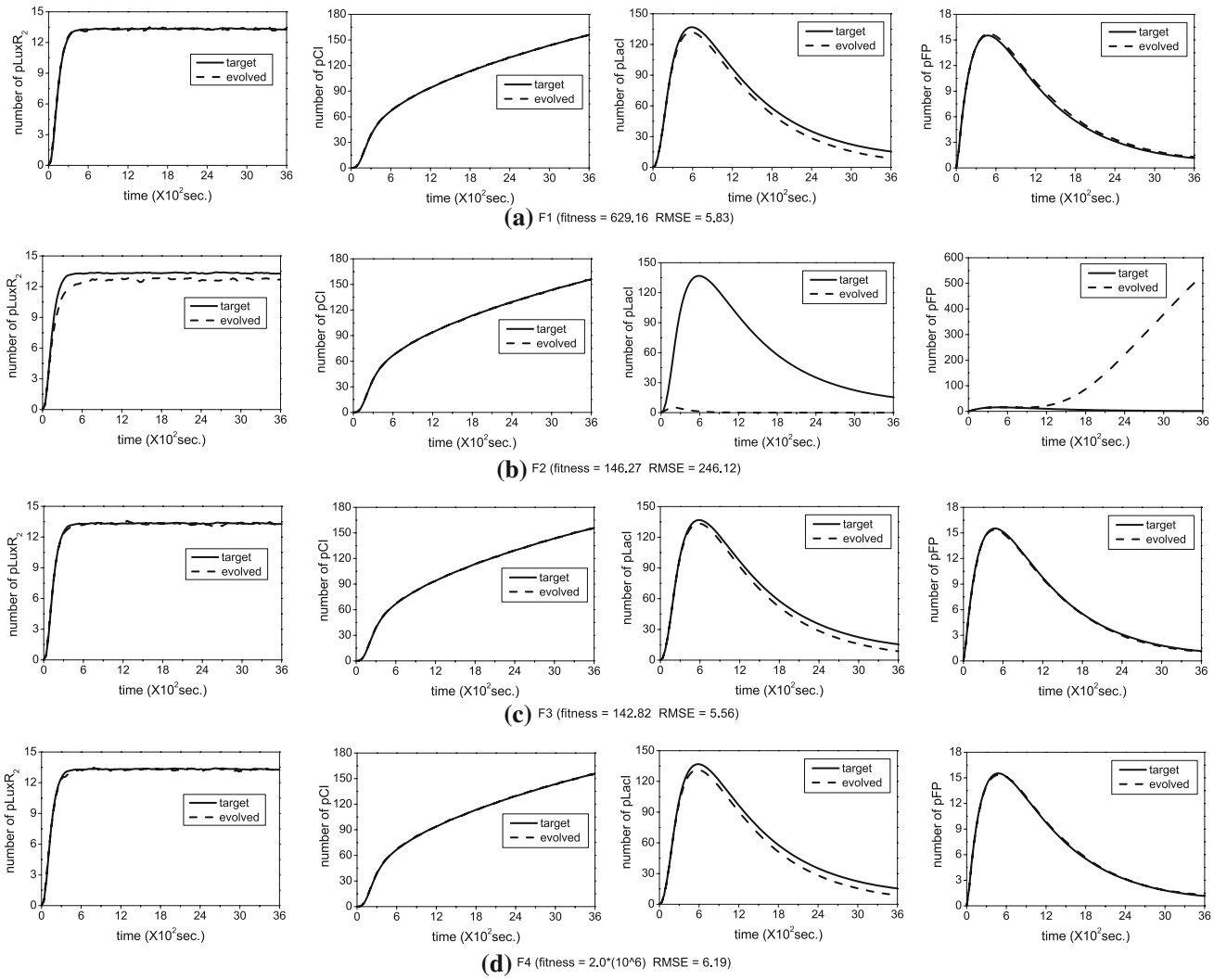

Fig. 16 Simulated results of the fittest models obtained by the four fitness methods (F1, . . ., F4) with the common model structure in test case 4

Table 11 Comparisons of the constants between the best fitness models with the common model structure by F1, . . ., F4 and the target model for test case 4 (Values different from the target are bolded and underlined)

Module set	Const.	F1	F2	F3	F4	Target
$UnReg\{X = LuxR\}$	c_4	0.001	**0.1**	**0.01**	**0.01**	0.001
$PluxR\{X = CI\}$	c_{10}	0.1	0.1	0.1	0.1	0.1
	c_{13}	0.001	0.001	0.001	0.001	0.001
$PR\{X = LacI\}$	c_4	0.001	**0.1**	0.001	0.001	0.001
$Plac\{X = FP\}$	c_4	0.001	0.001	0.001	0.001	0.001

Table 12 Summary of model diversity for different fitness methods on the four test cases across 20 runs. Best results appear in bold

Test cases	Fitness methods	Different methods	Models as target
Test case 1	F1	2	18
	F2	2	17
	F3	**2**	**19**
	F4	2	16
Test case 2	F1	**1**	**20**
	F2	2	3
	F3	**1**	**20**
	F4	**1**	**20**
Test case 3	F1	**6**	**4**
	F2	6	1
	F3	8	1
	F4	8	2
Test case 4	F1	**15**	**2**
	F2	15	0
	F3	14	1
	F4	**14**	**2**

comparable and their performance may vary with different case studies.

Further experiments

We conduct a suite of additional experiments on test case 4 as to ascertain the algorithm scalability. We keep the target model and all the parameter settings exactly the same as in previous experiments. However, rather than evolving only five variables within the elementary modules as shown in Table 3, we allow all the constants contained in each module to be evolved within predefined ranges and precisions (see Table 16) . As for the four elementary modules for Test Case 4, {*UnReg*, *Plac*, *PR*, *PluxR*}, the number of the constants to be evolved in each module are {4, 8, 10, 13} respectively. This leads to a 38-dimensional continuous optimisation problem.

Table 17 collects the statistics for the 20 runs using the four fitness methods. As expected, in terms of RMSD and average fitness, the results obtained degrade slightly with

Table 13 The average fitness and running time for different fitness methods on the four test cases

Test cases	F	Fitness	RMSE	Run-time
Test case 1	F1	2000 ± 671.8	3.54 ± 1.24	$50 \pm 1(s)$
	F2	272.55 ± 34.04	13.23 ± 6.82	$49 \pm 1(s)$
	F3	961.7 ± 467.11	$\mathbf{3.46 \pm 1.65}$	$56 \pm 3(s)$
	F4	$(2.81 \pm 1.93) \times 10^5$	5.0 ± 1.99	$54 \pm 2(s)$
Test case 2	F1	10226 ± 2727	$\mathbf{19.04 \pm 6.2}$	$49 \pm 2(s)$
	F2	246.39 ± 14.24	134.33 ± 47.61	$46 \pm 2(s)$
	F3	5740 ± 1738	21.89 ± 5.51	$49 \pm 2(s)$
	F4	$(3.46 \pm 1.51) \times 10^6$	36.9 ± 19.78	$43 \pm 2(s)$
Test case 3	F1	518.75 ± 156.93	4.86 ± 1.9	$122 \pm 22(m)$
	F2	152 ± 20.73	8.55 ± 1.08	$116 \pm 28(m)$
	F3	89.54 ± 11.42	$\mathbf{4.63 \pm 1.45}$	$121 \pm 22(m)$
	F4	$(2.6 \pm 6.1) \times 10^{11}$	4.96 ± 2.08	$107 \pm 13(m)$
Test case 4	F1	638.89 ± 329.68	$\mathbf{5.65 \pm 2.83}$	$149 \pm 30(h)$
	F2	138.47 ± 16.38	75.72 ± 60.47	$178 \pm 42(h)$
	F3	350.03 ± 393.27	12.32 ± 14.46	$149 \pm 42(h)$
	F4	$(1.53 \pm 5.24) \times 10^7$	5.89 ± 2.52	$136 \pm 30(h)$

Averages are taken over 20 runs. The RMSD for the best model evolved under each objective function is also averaged for the 20 runs and reported. The best RMSD appears in bold

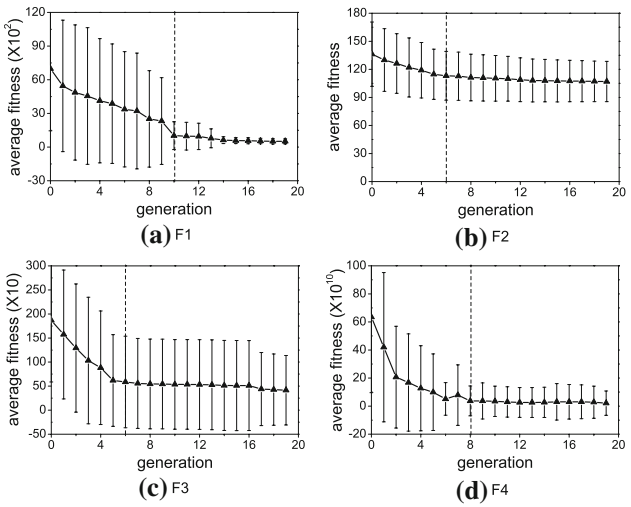

Fig. 17 Average over 20 runs of the best fitness under four different fitness methods (F1, . . . , F4) for test case 4

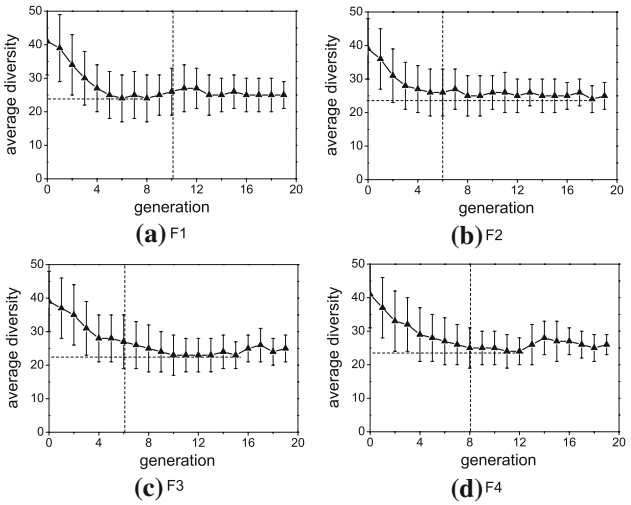

Fig. 18 Average over 20 runs of population diversity under four fitness methods (F1, . . . , F4) for test case 4

Table 14 One-way ANOVA test for different fitness methods and four test cases

Test cases	F-value	Significant? Y/N
Test case 1	31.82	**Y**
Test case 2	114.6	**Y**
Test case 3	25.0	**Y**
Test case 4	14.61	**Y**

the increased dimensionality (compared with the results in Table 13). Interestingly, the algorithm still manages to reverse engineer the target model even in this extended 38-dimensional problem when using fitness method 3 and 4. Furthermore, the simulation results for the best fitness

Table 15 TUKEY's HSD POST HOC test for the cases that get a significant *F*-value in Table 14

Pairs	(*F*1,*F*2)		(*F*1,*F*3)		(*F*1,*F*4)		(*F*2,*F*3)		(*F*2,*F*4)		(*F*3,*F*4)	
Test cases	*Q*-value	Signi? Y/N	*Q*-value	Signi? Y/N	*Q*- value	Signi? Y/N	*Q*-value	Signi? Y/N	*Q*-value	Signi? Y/N	*Q*-value	Signi?Y/N
Test case 1	11.7	**Y**	0.09	N	1.77	N	11.8	**Y**	9.94	**Y**	1.86	N
	F1→F2		–		–		**F3→F2**		**F4→F2**		–	
Test case 2	22.5	**Y**	0.55	N	3.45	N	21.9	**Y**	19.1	**Y**	2.90	N
	F1→F2		–		–		**F3→F2**		**F4→F2**		–	
Test case 3	9.44	**Y**	1.55	N	0.25	N	11.0	**Y**	9.19	**Y**	1.80	N
	F1→F2		–		–		**F3→F2**		**F4→F2**		–	
Test case 4	8.44	**Y**	1.46	N	1.28	N	6.99	**Y**	7.16	**Y**	0.18	N
	F1→F2		–		–		**F3→F2**		**F4→F2**		–	

Table 16 The range and the precision of the kinetic constants in the rule set of the modules for further experiments on test case 4

Module name	Constants	Scale	Range	Precision
UnReg	c_1	Linear	(0.1, 0.3)	10^{-2}
	c_2	Linear	(0.001, 0.01)	10^{-3}
	c_3	Linear	(0.01, 0.05)	10^{-2}
	c_4	Logarithmic	[−3, −1]	1
PluxR	c_1	Linear	(0, 0.2)	10^{-2}
	c_2	Linear	(0, 0.2)	10^{-2}
	c_3	Linear	(0, 2)	10^{-1}
	c_4	Linear	(0, 0.01)	10^{-3}
	c_5	Linear	(0, 2)	10^{-1}
	c_6	Linear	(0, 0.01)	10^{-3}
	c_7	Linear	(0, 0.02)	10^{-3}
	c_8	Linear	(0, 2)	10^{-1}
	c_9	Linear	(0, 2)	10^{-1}
	c_{10}	Logarithmic	[− 3,1]	1
	c_{11}	Linear	(0.001, 0.006)	10^{-3}
	c_{12}	Linear	(0.01, 0.05)	10^{-2}
	c_{13}	Logarithmic	[−3, −1]	1
PR	c_1	Linear	(0.1, 0.3)	10^{-2}
	c_2	Linear	(0.001, 0.01)	10^{-3}
	c_3	Linear	(0.01, 0.05)	10^{-2}
	c_4	Logarithmic	[−3, −1]	1
	c_5	Linear	(0.0001, 0.0003)	10^{-5}
	c_6	Linear	(0,0.005)	10^{-3}
	c_7	Linear	(0.1, 0.3)	10^{-3}
	c_8	Linear	(0, 0.005)	10^{-3}
	c_9	Linear	(0.001, 0.01)	10^{-3}
	c_{10}	Linear	(0, 0.0005)	10^{-4}
Plac	c_1	Linear	(0.1, 0.3)	10^{-2}
	c_2	Linear	(0.001, 0.01)	10^{-3}
	c_3	Linear	(0.01, 0.05)	10^{-2}
	c_4	Logarithmic	[−3, −1]	1
	c_5	Linear	(0.0001, 0.0003)	10^{-5}
	c_6	Linear	(0, 0.02)	10^{-3}
	c_7	Linear	(9, 12)	10^{-2}
	c_8	Linear	(0.01, 0.08)	10^{-2}

models found by F1, F3, and F4 are quite good. Table 18 shows the model structure, the fitness and RMSE of the best models obtained by F1, ..., F4. Figure 19 illustrates their simulation results. Again, as indicated by its large RMSE value (70.44), the simulation results of the best F2 model are mediocre. As for fitness methods F1 and F3, each finds an alternative model structure with small RMSE values of 7.4 and 6.94 respectively. As shown in the figure, the simulation results of these two models are very similar and both are quite good. As for the best F4 model, our algorithm can find the same model structure as the target and its simulation results are slightly worse than previous models but still good and acceptable.

Table 17 Statistical results for test case 4 with all parameters to be evolved under four fitness methods in 20 runs

Fitness methods	Average fitness	Average RMSE	Different models	Model as target
F1	3198 ± 3087	26.47 ± 28.37	13	0
F2	122.35 ± 18.25	76.57 ± 39.66	16	0
F3	1011 ± 1299	31.22 ± 41.52	17	1
F4	$(6 \pm 12) \times 10^{10}$	49.47 ± 41.17	15	2

Table 18 The best fitness models for test case 4 with all parameters to be evolved under four fitness methods in 20 runs

Fitness methods	Best fitness model structure	As target (Y/N)	Fitness	RMSE
F1	$\Pi = (m_1, m_2, m_3, m_4, m_5)$	N	1126.58	7.4
	$m_1 = UnReg\{X = LuxR\}$			
	$m_2 = Plac\{X = FP\}$			
	$m_3 = PluxR\{X = CI\}$			
	$m_4 = UnReg\{X = CI\}$			
	$m_5 = PR\{X = LacI\}$			
F2	$\Pi = (m_1, m_2, m_3, m_4, m_5)$	N	92.3	70.44
	$m_1 = UnReg\{X = LuxR\}$			
	$m_2 = Plac\{X = FP\}$			
	$m_3 = PluxR\{X = CI\}$			
	$m_4 = PluxR\{X = LacI\}$			
	$m_5 = PR\{X = FP\}$			
F3	$\Pi = (m_1, m_2, m_3, m_4, m_5)$	N	223.91	6.94
	$m_1 = UnReg\{X = LuxR\}$			
	$m_2 = PR\{X = FP\}$			
	$m_3 = PluxR\{X = CI\}$			
	$m_4 = PR\{X = LacI\}$			
	$m_5 = Plac\{X = FP\}$			
F4	$\Pi = (m_1, m_2, m_3, m_4, m_5)$	Y	8.79×10^8	16.75
	$m_1 = UnReg\{X = LuxR\}$			
	$m_2 = PluxR\{X = LacI\}$			
	$m_3 = PluxR\{X = CI\}$			
	$m_4 = PR\{X = LacI\}$			
	$m_5 = Plac\{X = FP\}$			

Model selection

In this paper we have presented four different fitness calculation methods that guide the search of models reproducing a prefixed behaviour. All our models are biologically plausible as a result of the methodology that we follow to construct them. More specifically, our modules act as biologically plausible building blocks and the operators used to

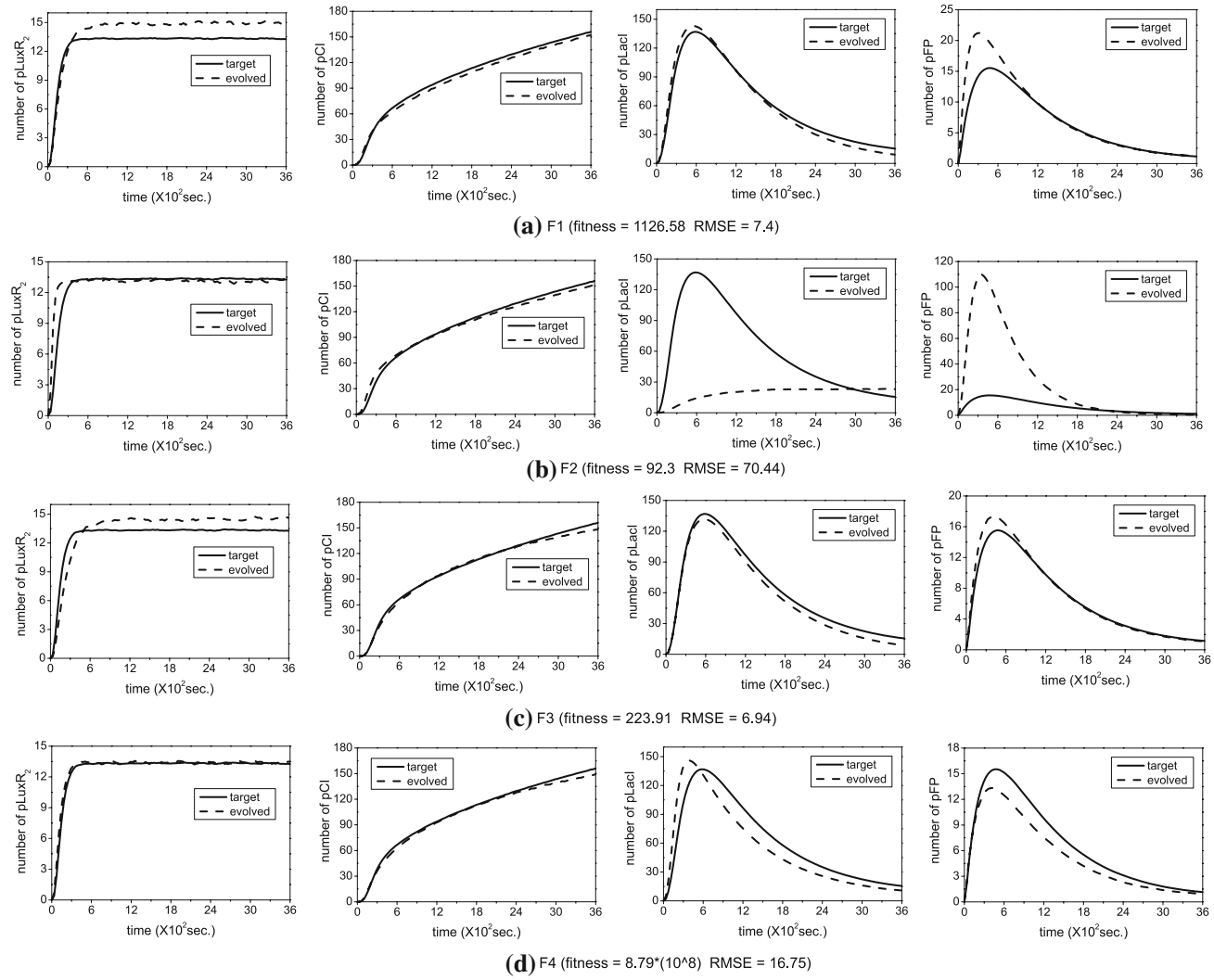

Fig. 19 Simulated results of the fittest models obtained by the four fitness methods $(\mathrm{F1}, \ldots, \mathrm{F4})$ in test case 4 with all parameters to be evolved

combine and vary them, crossover and mutation, preserve biological plausibility. Consequently our methodology produces a set of candidate biologically plausible models that comparably match a prefixed behaviour which makes difficult to decide which model is the best one.

In this section we apply *model selection theory* (Burnham and Anderson 2002) in order to compare and rank the five alternative models proposed by our algorithm for the more complex test case 4, see Figs. 13 and 15. In particular we associate two scores to each model using *small-sample corrected Akaike's Information criterion* (AIC; Burnham and Anderson 2002) and *minimun description length criterion* (MDL; Grumwald 2002) according to the formulas in Eqs. (4) and (5) where $\mathrm{RSS} = \sum_{j=1}^{N} \sum_{i=1}^{M} (\hat{x}_j^i - x_j^i)^2$ is the residual sum of square errors, $D = N \times M$ is the number of data points and K is the number of rules of each model. Both scores are approximations of theoretical measures that are not computable in general. Akaike's information criterion is an approximation of Kullback-Leibler divergence which measures the amount of information lost when building a model. Whereas the minimum description length criterion aims at capturing the general idea of choosing the model that provides the shortest description of the data.

$$\mathrm{AIC} \approx D\log(\mathrm{RSS}) + 2K\left(\frac{D}{D-(K+1)}\right) \tag{4}$$

$$\mathrm{MDL} \approx D\log(\mathrm{RSS}) + K\log(D) \tag{5}$$

These two scores share the same first part, $D\ log(\mathrm{RSS})$, which evaluates how good the model replicates the prefixed behaviour. However they differ in the second part that penalizes complex models with more parameters over simpler ones. This second part aims at preventing the overfitting of the sample data when using complex models. In this respect, the penalty in Akaike's information criterion penalizes more strongly models with a number

Table 19 Akaike's information criterion (AIC) and minimun description length criterion (MDL) scores associated with the alternative models found for Test Case 4

Model	RSS	K	AIC	MDL
Alternative model 1	33.75	39	704	719.08
Alternative model 2	48.85	41	773.83	789.17
Alternative model 3	112.34	39	914.99	930.07
Alternative model 4	82.99	42	860.3	884.76
Common model	3,132.02	33	1,484.38	1,498.82

Table 20 Sensitivity of *FP* expression in the long run with respect to 1% change in the rate of increase of 3*OC*6, parameter c_1 in the module *PluxR* associated with the alternative models found for Test Case 4 and the target model

Model	Sensitivity
Alternative model 1	987
Alternative model 2	408.13
Alternative model 3	534
Alternative model 4	637
Common model	17
Target model	208

of parameters approaching the number of data points than the minimum description length criterion.

The scores associated which each model are presented in Table 19 and they determine the following rank of models; alternative model 1, 2, 4, 3 and common model. Note that although the common model is the simpler one consisting of 33 rules it is ranked the last one due to its high fitting error.

Finally, we apply another criterion to rank our alternative models based on *sensitivity analysis* (Saltelli et al. 2000). These networks respond to an extracellular signal, 3*OC*6, producing as output the expression of a fluorescence protein (*FP*). A desirable property for them is robustness in the number of *FP* molecules in the long run with respect to small changes in the rate of increase of the extracellular signal 3*OC*6, parameter c_1 in the module *PluxR*. In order to quantify this we computed the local sensitivity coefficient of the number of *FP* molecules at 3,600 min associated with the parameter c_1. This was performed using the *indirect method* or *finite-difference approximation* according to formula (6) and a perturbation of 1% in c_1. The output associated with *FP* was computed by averaging 1000 simulations for the original and perturbed models respectively.

$$S_{c_1} = \frac{\partial FP}{\partial c_1} \approx \frac{FP(c_1 + \Delta c_1) - FP(c_1)}{\Delta c_1} \quad (6)$$

The sensitivity coefficients for c_1 associated with each alternative model and the target model are presented in Table 20. Observe that surprisingly the simplest model, the common model, has the lowest sensitivity coefficient even lower than the target model. In this respect, the common model found by all the different fitness calculation methods is a very good candidate that replicates the target robustly with respect to small changes in the rate of increase of signals.

These results suggest the necessity of including terms that penalize complex and sensitive models in the fitness calculation methods. This will be taken into account in future enhancements of our methodology.

Conclusions and future work

This paper proposes a new methodology in cell systems biology modelling. It mainly includes the following four main features:

1. A computational, stochastic and discrete modelling approach based on P systems.
2. Modular modelling approach using modules of rules as building blocks for our models.
3. A nested EA designed to perform structural and parameter optimization using a two-layer GA.
4. Four alternative fitness calculation methods applied to cope with different cases.

The effectiveness of the methodology is tested on four case studies predesigned with increasing complexity, namely, negative and positive autoregulation and two gene networks implementing a pulse generator and a bandwidth detector. The four different fitness methods are applied to each test case and their results are compared and analyzed.

Based on the experimental results, we draw some conclusions as follows:

1. When using the fitness method F2, our algorithm is able to find the target model for simple cases, but it fails for more complicated cases. Even when good model structures are found it fails to obtain good estimates for the stochastic constants which produces a behavior that deviates considerably from the target.
2. When using the methods F1, F3, and F4, our algorithm always finds good models that can accurately reproduce the dynamical behavior of the target cellular system. Specifically, for simple cases all these methods consistently find a single model structure, *i.e.* the target one, nevertheless the diversity of the models found by our algorithm using the methods increases significantly with the complexity of the system. For example, for the relatively complex cellular system in test case 4 our algorithm was able to propose a variety of alternative model structures which reproduce the target

behavior. More interestingly, some of these models are simpler than the target one. This result is very encouraging as it could help biologists to design new experiments to discriminate among competing hypothesis (models) and then only engineered in the lab the one that has been proven as the best. This is a potentially very useful feature to help close the loop between modeling and experimentation in both synthetic and systems biology.

3. The statistical analysis of the experimental results suggests that when comparing different fitness methods, F2 always performs the worst whereas the other three methods (F1, F3, F4) are comparable and their performance varies with different case studies. Generally speaking, if some target output objects of the predesigned cellular system have very different orders of magnitude in their time series, F3 and F4 work better than F1 when trying to obtain a good compromise solution.
4. Many results agree with the fact that a minor discrepancy in the stochastic constants between two models with the same structure will produce completely different dynamical behaviors. This shows the great importance of parameter optimization for the kinetic constants in the model. Fortunately, more than one experiment demonstrate that our parameter optimization algorithm implemented as a GA works well for both continuous and discrete parameters.
5. With regard to the evolution of the average model diversity, we conclude that the fitness method used has little effect on the model diversity during the evolution. This is essentially determined by the nested EA itself. Nevertheless, it shows that the fitness method has some influence on the convergence and stability of the algorithm. As mentioned previously, F2 performs the worst. The improvement of its fitness is small and the algorithm converges soon to a bad solution. As for the other three methods, the algorithm can always find good solutions after reasonable number of generations. Summing up, the order of the convergence is: F3 $>$ F4 $>$ F1 and the order of the stability is: F1 $>$ F4 $>$ F3.

There are several conceivable extensions to our work, including:

1. We notice that the biggest drawback for our algorithm is its time cost, specially for modelling relatively complex cellular systems. We are aware that in order to obtain a solution in acceptable time, some key control parameters in the algorithm need to to be set to smaller values, like the maximal number of generation, the population size, the number of simulations to calculate the fitness of an individual *etc*. In order to study more complicated regulatory transcriptional networks we plan to explore the following possible solutions to this problem:
   - As most running time is spent in the fitness calculation which is based on the multiple simulations by Gillespie's SSA, in the future we will use a GPGPU based parallel implementation of the SSA algorithm.
   - Computationally expensive fitness functions can sometimes be approximated through local or global models and other surrogate techniques. This is under investigation.
   - To systematically chart the "control map" of the algorithm as to ascertain its sensitivity to population sizes and number of simulations as to try to reduced them.
2. Since all our experimental results have clearly shown that the stochastic constants have profound impact on the dynamic behavior of a cellular system, it is very important to adopt an efficient algorithm for parameter optimization. We intend to investigate other advanced optimization algorithms such as *Estimation of Distribution Algorithms* (EDA), *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES), *Differential Evolution* (DE) *etc*.
3. By improving and extending our algorithm, we aim to apply it to the automatic design of more complex and challenging regulatory transcriptional networks as well as the eukaryotic cellular systems with relevant compartmentalized structure.

Acknowledgments This work was supported by the Engineering and Physical Sciences Research Council (EPSRC Grant no: EP/E017215/1) and the Biotechnology and Biological Sciences Research Council (BBSRC Grant no: BB/F01855X/1). We thank James Smaldon and Jamie Twycross for aiding in setting the algorithm to run in the University of Nottingham supercomputer cluster.

References

Alon U (2006) An introduction to systems biology (mathmatical and computational biology beries). Chapman & Hall/Crc, London

Andreianantoandro E, Basu S, Karig DK, Weiss R (2006) Synthetic biology: new engineering rules for an emerging discipline. Mol Syst Biol 2:2006.0028

Atkinson MR, Savageau MA, Myers JT, Ninfa AJ (2003) Development of genetic circuitry exhibiting toggle switch or ocillatory behavior in *Escherichia coli*. Cell 113:597–607

Basu S, Mehreja R, Thiberge S, Chen M, Weiss R (2004) Spatiotemporal control of gene expression with pulse-generating networks. Proc Natl Acad Sci USA 101(17):6355–6360

Basu S, Gerchman Y, Collins CH, Arnold FH, Weiss R (2005) A synthetic multicellular system for programmed pattern formation. Nature 434:1130–1134

Benner SA, Sismour AM (2005) Synthetic biology. Nat Rev Genet 6:533–543

Bernardini F, Gheorghe M, Krasnogor N (2007) Quorum sensing p systems. Theor Comput Sci 371(1-2):20–33

Beyer H, Schwefel H (2002) Evolution strategies—a comprehensive introduction. Nat Comput 1:3–52

Bridgman PW (1922) Dimensional analysis. Yale University Press, New Haven

Burnham KP, Anderson DR (2002) Model selection and multimodel inference—a practical information-theoretic approach, 2nd ed. Springer, Berlin

Calder M, Vyshemirsky V, Gilbert D, Orton R (2005) "Analysis of signalling pathways using the PRISM model checker." In proceedings computational methods in systems biology (CMSB'05), pp 179–190

Cheng FY, Li D (1996) Multiobjective optimization of structures with and without control. J Guid Control Dyn 19:392–397

Chickarmane V, Paladugu SR, Bergmann F, Sauro HM (2005) Bifurcation discovery tool. Bioinformatics 21(18):3688–3690

Coello CAC, Vanveldhuizen DA, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer Academic Publishers, Dordrecht

Cronin L, Krasnogor N, Davis BG, Alexander C, Robertson N, Steinke J, Schroeder S, Khlobystov A, Cooper G, Gardner P, Siepmann P, Whitaker B (2006) The imitation gamea computational chemical approach to recognizing life. Nat Biotechnol 24:1203–1206

Davidson EH (2006) The regulatory genome. Gene regulatory networks in development and evolution. Academic Press, Elsevier

de Hoon MJL, Imoto S, Kobayashi K, Ogasawara N, Miyano S (2003) Inferring gene regulatory networks from time-ordered gene expression data of *Bacillus Subtilis* using differential equations. In proceedings of the Pacific symposium on biocomputing vol 8, pp 17–28

Diggle SP, Crusz SA, Camara M (2007) Quorum sensing. Curr Biol 17(21):R907–R910

Errampalli CD, Quaglia P (2004) A formal language for computational systems biology. OMICS J Integr Biol 8:370–380

Fisher J, Henzinger T (2007) Executable cell biology. Nat Biotechnol 25:1239–1249

Gerasimov EN, Repko VN (1978) Multicriterial optimization. Sov Appl Mech 14:1179–1184

Gheorghe M, Krasnogor N, Camara M (2008) P systems applications to systems biology. Biosystems 91:435–437

Gilbert D, Fuss H, Gu X, Orton R, and Robinson S (2006) Computational methodologies for modelling, analysis and simulation of signalling networks. Brief Bioinform 7(4):339–353

Gillespie DT (2007) Stochastic simulation of chemical kinetics. Annu Rev Phys Chem 58:35–55

Ginkel A, Kremling A, Nutsch T, Rehner R, Gilles ED (2003) Modular modelling of cellular systems with ProMot/D iva. Bioinformatics 19:1169–1176

Grumwald P (2000) Model selection based on minimum description length. J Math Psychol 44:133–152

Harel D (2005) A turing-like test for biological modeling. Nat Biotechnol 23:495–496

Hartwell LH, Hopfield JJ, Leibler S, Murray AW (1999) From molecular to modular cell biology. Nat Impacts 402:47–52

Heiner M, Gilbert D, Donaldson R (2008) Petri nets for systems and synthetic biology. Formal methods for computational systems biology. Lecture notes in computer science 5016/2008, pp 215–264

Hinterding R, Michalewicz Z, Eiben A (1997) "Adaptation in evolutionary computation: a survey." In proceedings 4th international conference on evolutionary computation. IEEE Press, New york, pp 65–69

Ishibuchi H, Murata T (1998) Multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans Syst Man Cybern C 28:392–403

Jacob F, Monod J (1961) Genetic regulatory mechanisms in the synthesis of proteins. J Mol Biol 3:318–356

Jaszkiewicz A (2002) On the performance of multiple-objective genetic local search on the 0/1 kanpsack problem-a comparative experiment. IEEE Trans Evol Comput EC-6:402–412

Kaern M, Elston TC, Blake WJ, Collins JJ (2005) Stochasticity in gene expression: from theories to phenotypes. Nat Rev Genet 6:451–464

Kikuchi S, Tominaga D, Arita M, Takahashi K, Tomita M (2003) Dynamic modeling of genetic networks using genetic algorithm and S-system. Bioinformatics 19(5):643–650

Klipp E, Herwig R, Kowald A, Wierling C, Lehrach H (2005) Systems biology in practice: concepts, implementation and application. Wiley-VCH, Weinheim

Krasnogor N, Smith J (2000) "Mafra: a java memetic algorithms framework." In workshops proceedings of the 2000 international genetic and evolutionary computation conference (GECCO2000), A. Wu, Ed., 2000. Online. Available: http://www.cs.nott.ac.uk/nxk/PAPERS/womaMafra.pdf

Krasnogor N, Gustafson S (2002) "Toward truly "memetic" memetic algorithms: discussion and proofs of concept." In advances in nature-inspired computation: The PPSN VII workshops. PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading

Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: model, taxonomy and design issues. IEEE Trans Evol Comput EC-9:474–488

Lanzi PL, Loiacono D, Wilson SW, Goldberg DE (2006) "Prediction update algorithms for XCSF:RLS, kalman filter, and gain adaptation." In GECCO'06: proceedings 8th annual conference on genetic and evolutionary computation. ACM Press, New York, NY, USA, pp 1505–1512

Leung Y, Wang Y (2000) Multiobjective programming using uniform design and genetic algorithm. IEEE Trans Syst Man Cybern C Appl Rev 30(3):293–303

Machne R, Finney A, Muller S, Lu J, Widder S, Flamm C (2006) The sbml ode solver library: a native api for symbolic and fast numerical analysis of reaction networks. Bioinformatics 22(11): 1406–1407

Mallavarapu A, Thomson M, Ullian B, Gunawardena J (2009) Programming with models: modularity and abstraction provide powerful capabilities for system biology. JR Soc Interface 6: 257–270

Mangan S, Alon U (2003) Structure and function of the feed-forward loop network motif. Proc Natl Acad Sci USA 100(21):11,980–11,985

Marbach D, Schaffter T, Mattiussi C, Floreano D (2009) Generating realistic in silico gene networks for performance assessment of reverse engineering methods. J Comput Biol 16(2):229–239

Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Mutidisc Optim 26:369–395

Mason J, Linsay PS, Collins JJ, Glass L (2004) Evolving complex dynamics in electronic models of genetic networks. Chaos 14(3): 707–715

Mazumdar R, Mason LG, Douligeris C (1991) Fairness in network optimal flow control: optimality of product forms. IEEE Trans Commun 39:775–782

Morishita R, Imade H, Ono NOI, Okamoto M (2003) Finding multiple solutions based on an evolutionary algorithm for

inference of genetic networks by s-system. In proceedings of the IEEE congress on evolutionary computation, pp 603–612
Palsson BO (2006) Systems biology: properties of reconstructed networks. Cambridge University Press, Cambridge
Păun G (2002) Membrane computing: an introduction. Springer, Berlin
Pérez-Jiménez MJ, Romero-Campero FJ (2006) P systems, a new computational modelling tool for systems biology. Trans Comput Syst Biol VI:176–197
Priami C (2009) Algorithmic systems biology. Commun ACM 52(5):80–88
Ptashne M (2004) A genetic switch. Cold Spring Harbor Laboratory Press, New York
Regev A, Silverman W, Shapiro E (2001) "Representation and simulation of biochemical processes using the pi-calculus process algebra." Proceedings of the Pacific symposium on biocomputation, pp 459–470 [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/11262964
Rodrigo G, Jaramillo A (2007) Computational design of digital and memory biological devices. Syst Synth Biol 1:183–195
Rodrigo G, Carrera J, Jaramillo A (2007a) Asmparts: assembly of biological model parts. Syst Synth Biol 1:167–170
Rodrigo G, Carrera J, Jaramillo A (2007b) Genetdes: automatic design of transcriptional networks. Bioinformatics 23(14):1857–1858
Romero-Campero FJ, Pérez-Jiménez MJ (2008a) Modelling gene expression control using P systems: the lac operon, a case study. BioSystems 91(3):438–457
Romero-Campero FJ, Pérez-Jiménez MJ (2008b) A model of the quorum sensing system in vibrio fischeri using P systems. Artif Life 14(1):95–109
Romero-Campero F, Twycross J, Bennett M, Camara M, Krasnogor N (2008a) "Modular assembly of cell systems biology models using p systems." In proceedings of the Prague international workshop on membrane computing, series. Lecture notes in computer science, vol (to appear). Springer
Romero-Campero FJ, Cao H, Camara M, Krasnogor N (2008b) "Structure and parameter estimation for cell systems biology models." In proceedings 2008 genetic and evolutionary computation conference (GECCO'2008). ACM Inc., pp 331–338
Romero-Camero F, Krasnogor N (2009) "An approach to biomodel engineering based on p systems." In proceedings of computation in Europe (CIE 2009), vol (to appear). [Online]. Available: http://www.cs.nott.ac.uk/ nxk/PAPERS/CiE.pdf
Romero-Campero FJ, Twycross J, Camara M, Bennett M, Gheorghe M, Krasnogor N (2009) Modular assembly of cell systems biology models using p systems. Int J Found Comput Sci 20:427–442
Sadot A, Fisher J, Barak D, Admanit Y, Stern M, Harel D (2008) Towards verified biological models. IEEE/ACM transactions on computational biology and bioinformatics 5(2):223–234
Saltelli A, Chan K, Scott EM (ed) (2000) Sensitivity analysis. Wiley, London
Sola J, Sevilla J (1997) Importance of input data normalization for the applicatiob of neural networks to complex industrial problems. IEEE Trans Nuclear Sci 44(3):1464–1468
Spieth C, Streichert F, Speer N, Zell A (2004) "A memetic inference method for gene regulatory networks based on s-system." In proceedings of the IEEE congress on evolutionary computation, pp 152–157
Strffin PD (1993) Game theory and strategy. The Mathematical Assiciation of America, Washington, DC
Szallasi Z, Stelling J, Periwal V (2006) System modeling in cellular biology. MIT press, Cambridge
Thieffry D, Huerta A, Perez-Rueda E, Collado-Vides J (1998) From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in escherichia coli. BioEssays 20(5):433–440
Thompson JD, Plewniak F, Ripp R, Thierry J, Poch O (2001) Towards a reliable objective function for multiple sequence alignments. J Mol Biol 314:937–951
Twycross J, Band L, Bennett M, King J, Krasnogor N (2009) Stochastic and deterministic multiscale models for systems biology: an auxin-transport case study. BMC Bioinformatics (under review)
Weaver DC, Workman CT, Stormo GD (1999) "Modeling regulatory networks with weight matrices." In proceedings of the Pacific symposium on biocomputing 4, 112–123
Yeung MKS, Tegner J, Collins JJ (2002) "Reverse engineering gene networks using singular value decomposition and robust regression." In proceedings of the national academy of science 99:6163–6168
Yu J, Cao H, He Y (2007) A new tree structure code for equivalent circuit and evolutionary estimation of parameters. Chemometrics Intell Lab Syst 85:27–39