Effective Calibration of Artificial Gene Regulatory Networks

D. Agostini[1], J. Costanza[1], V. Cutello[1], L. Zammataro[2], N. Krasnogor[3], M. Pavone[1], and G. Nicosia[1]
[1]Dept. of Mathematics and Computer Science, University of Catania
{costanza, vctl, mpavone, nicosia}@dmi.unict.it
[2]Dept. of Translational Medicine, University of Milan, IRCCS Istituto Clinico Humanitas
luca.zammataro@humanitasresearch.it
[3]School of Computer Science, University of Nottingham
Natalio.Krasnogor@Nottingham.ac.uk

Abstract

Knowing every single component of a given biological system is not enough to understand the complexity of the system but rather it becomes crucial to understand how these components interact with each others. It is not only important the knowledge of genes and proteins, but also knowing their structures and primarily the laws and parameters governing their dynamics, which is often unknown and impossible to measure directly. The *Gene Regulatory Networks* explain exactly how a genomic sequence encodes the regulation of expression of sets of genes, which progressively generate developmental patterns, and execute the construction of multiple states of differentiation. Their main aim is to represent the regulation rules underlying the gene expression. In this work we have designed the CMA-ES algorithm to infer the parameters in the S-system model of a gene regulatory network. This model is a well-known mathematical framework whose structure is rich enough to capture many relevant biological details, and it can model more complicated genetic network behaviour. CMA-ES has been compared against 7 state-of-the-art algorithms to evaluate its efficiency and its robustness. From a general point of view, it seems clear how CMA-ES is able to estimate in a better way the target parameters with respect to the state-of-the-art methods, either in terms of success rate or in terms of Euclidean distance. Finally, this research paper includes a study on the convergence of CMA-ES through Time-To-Target plots, which are a way to characterize the running time of stochastic algorithms; and a global sensitivity analysis method, the Morris algorithm.

Introduction

In the past few years studying how a system interacts with the environment, or how simple components effect the global behaviour of a given system, or even how parts of a system interact with each others has been the main and most challenging issue in many research areas. Many problems in science and engineering are often hard to solve mainly because of the difficulty in understanding their indirect causes and effects, which are not related in an obvious way. Assessing all the single parts of a structure, or knowing all single components of a given system is not enough to determine and understand the complexity of system, although we need to know how these objects interact. It is also well known that the information on the complex molecular features are contained in the genome of the organism, but is not clear what are the codes and mechanisms that translate the sequences into structures and functions. For example, from systems biology point of view, it is not only important a knowledge of genes and proteins, but it is of primary importance understanding their structures, dynamics, and how their parameters influence the global dynamics: such parameters are unknown and often impossible to measure directly. Moreover, studying dynamic properties of a biological system is not only very important to gain a deep understanding of biological processes, but also to develop efficient treatments against diseases. In systems biology *reverse engineering* the processes can be regarded as a central part of the discipline itself (Lee, 2005). Reverse engineering can be considered as a process from which is possible to infer structural and dynamics features of a given system from external observations and relevant knowledge. Thanks to that, today reverse engineering techniques play a central role in systems biology (Csete and Doyle, 2002; D'haeseleer et al., 2000). The main focus in reverse engineering field is the identification of genetic networks (Cho et al., 2007) in order to learn how transcription factors are connected to genes (the determination of the interactions between all genes and understanding of the regulatory networks are crucial to identify and develop novel drugs), and understand the gene expression profile that is a major issue in computational biology. In other words, reverse engineering can help us to answer questions as: (1) what are the functions of this gene? (2) which genes regulate this gene? (3) how several genes interact? (4) which genes are responsible for this disease? (5) which drugs will treat this disease? Of course, a method to interpret these answers is needed, in order to enhance our learning of living organisms. *Gene Regulatory Networks* (GRNs) explain exactly in which way genomic sequences encode the regulation of expression of sets of genes that progressively generate developmental patterns and execute the construction of multiple states of differentiation (Davidson and Levin, 2005). The main aim of GRN is to represent the regulation rules underlying the gene expression. Albeit the study of GRNs

nowadays is made easier thanks to the advances of new technologies however the solution to the problem is not trivial due to the enormously large space of the unknown parameters. In the last years several reverse engineering methodologies based on evolutionary algorithms have been presented (Kabir et al., 2010; Kikuchi et al., 2003; Noman and Iba, 2007), which are more suitable to effectively and efficiently reconstruct the networks in a given dynamic model. It is well known as evolutionary algorithms work better than standard methods when the problem to solve is nonlinear, and therefore or no solution is known a priori, or it is impossible to be analytically solved. The great advantage of the evolutionary approaches on these tasks is own their applicability to almost any models where mathematical analysis and reversing is unavailable or inefficient. A good comparative study among evolutionary algorithms in gene regulatory network can be found in (Schlitt and Brazma, 2007). In this research work, we present a new approach to infer parameters of a gene regulatory network from time-series gene expression data using *S-system* model (Irvine and Savageau, 1990). For this kind of task, one of the best population-based optimization algorithms has been used as learning paradigm: *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) (Hansen and Ostermeier, 2001).

The S-system model for gene expression

Developing accurate computational and mathematical models is needed to study the response of the gene regulation and the gene sets with respect to their specific dynamics (many important cell functions are largely determined by dynamic processes of biochemical networks). Therefore, using mathematical models for the analysis of metabolic and regulatory pathways may contribute to a better understanding of the behaviour of metabolic processes. These models, once built, can be used to predict the behaviour of the organism under certain conditions (Sirbu et al., 2010), it has been also postulated that, once inferred the basic mechanisms of life, it should be (theoretically) possible to create synthetic organisms (Barrett et al., 2006). Of course, the choice of the model to use depends by how much information we try to capture: more information a model trying to learn - more parameters need to be inferred - more complex becomes the model. Nowadays, there exist several types of models in literature that describe a gene regulatory network, as: *Boolean networks* (D'haeseleer et al., 2000; Akutsu et al., 1999); *Bayesian networks* (Friedman et al., 2000), and *methods based on a steady-state description* (Tegner et al., 2003). Unfortunately, the main drawback of these models is that the gene expression is represented only in the two extreme levels, and therefore all genes are mapped only in a binary state: *on* (1) or *off* (0). This disadvantage makes limited use of such models since the real gene expression levels tend to be continuous rather than binary. An other drawback is also given from their not ability to capture the non-linear gene regulations, typical feature of the gene regulatory networks. To overcome this limitation, models based on *ordinary differential equations* (Chen et al., 1999) have been designed, which represent a very powerful and flexible model to describe complex relations among more components. One of the most popular and studied approaches, based on ODE, is the *S-system* model, whose structure is rich enough to capture many relevant biological dynamics, and it can models much more complicated GRN behaviour (Wessels et al., 2001). A comprehensive interesting comparative study on the three most used continuous systems based on ordinary differential equations has been made in (Swain et al., 2010) (S-system, *artificial neural network* (Vohradsky, 2001), and *general rate law of transcription* (Mendes et al., 2003)), where the advantages and disadvantages of each deterministic model used for modelling gene regulatory networks have been reported. In the last decades, inferring gene regulatory networks from time-series data has attracted a lot of attentions by many researchers in systems biology. It is then important to develop proper models that incorporate a suitable compromise among different requirements, as e.g. computational complexity, the ability to capture nonlinear gene regulations and the ability to handle noisy data. Moreover, it is also able to model much more complicated GRN behaviour (Wessels et al., 2001), and therefore it presents a good compromise between accuracy and mathematical flexibility. The S-system model is a type of power-law formalism used to model molecular networks, whose expression rates are described as the difference between the activation and degradation terms of a gene product. It is formally defined as a set of non-linear ordinary differential equations of the form:

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{n} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} X_j^{h_{ij}}, \qquad (1)$$

where n is the number of the genes; X_i is the expression level of the i–th gene; the exponential parameters g_{ij} e h_{ij} represent the effective interaction of X_j to X_i. In equation (1), the first term represents all influences that increase X_i, whilst the second term the ones that decrease X_i : if X_j has a positive exponent it means that it has a positive correlation on the aggregation process, whilst if it is negative then the genes are negatively correlated. Of course, if the exponent is zero then there not exist any influence on the aggregation process. From biochemical engineering point of view, the non-negative parameters $\alpha_i, \beta_i \in [R_l, R_u]$ are called *rate constants*, whereas the real value exponents $g_{ij}, h_{ij} \in [K_l, K_u]$ are referred to as *kinetic orders*. The aim on the S-system model is inferring the set of parameters $\Omega = \{\alpha_i, \beta_i, g_{ij}, h_{ij}\}$, such that the fitness function is minimized. Is easy to see how extracting the parameters Ω in a genetic network with n genes is not trivial task due to the high dimensionality of the problem: $2n(n+1)$ parameters indeed must be inferred. Obviously, the difficulty

of the problem increases by how much information need to be captured. To overcome this limitation and facilitate the regression task a *decoupled variant* of the model has been proposed (Kimura et al., 2005; Noman and Iba, 2006; Vilela et al., 2008a), which reduces the problem in n sub problems. Through this decomposition strategy, the original optimization problem is divided into n sub-problems where in any gene i the parameter values $(\alpha_i, \beta_i, g_{ij}, h_{ij})$ are individually estimated to attempt to capture the dynamics of the own gene. In this way, the original problem of $2n(n+1)$ dimensional is reduced in n sub problems each of $2(n+1)$ dimension. Thus, in the i–th sub-problem the expression level of the gene i is computed by the following ordinary differential equation:

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{N} Y_j^{g_{ij}} - \beta_i \prod_{j=1}^{N} Y_j^{h_{ij}}, \qquad (2)$$

where:

$$Y_j = \begin{cases} X_j, & \text{if} \quad i = j \\ \hat{X}_j, & \text{otherwise,} \end{cases}$$

with X_j computed solving the differential equation (2); and $\hat{X}_j$ estimated directly from experimentally observed time–series data using differential equation (1). Estimating the inferred set of the parameters is usually evaluated by the *Mean Squared Error* (MSE) between the experimentally observed expression levels, and the ones computed solving the system of equation (1). Therefore, the optimization task is inferring the parameters Ω in decoupled form in order to minimize MSE.

The CMA-ES algorithm

To attempt to inferring the set of parameters of a genetic network using the S-system model, we have adopted the CMA-ES algorithm (Hansen and Ostermeier, 2001), one of the best population-based optimization algorithms that is very suitable primarily on non-linear and non-convex optimization task. Since CMA-ES algorithm is well known inside the evolutionary computation community, in this section we give a short description on its main features. It is a $(1+1)$ elitist evolutionary strategy that generates candidate solutions by adapting a covariance matrix C, such that steps promising large fitness progress are sampled more often. Conversely to other self-adaptive evolutionary algorithms, CMA-ES adapts the covariance matrix, at generation g, by additive updates of the form $C^{(g)} = \alpha C^{(g-1)} + \beta V^{(g-1)}$, where $V^{(g-1)} \in \mathbb{R}^{n \times n}$ is positive definite and $\alpha, \beta \in R_0^+$ are weighting factors. Let $v^{(g-1)} \in \mathbb{R}$ a promising mutation step, to increase the probability of sampling $v^{(g-1)}$ in the next generation, the rank-one update is performed in the equation: $C^{(g)} = \alpha C^{(g-1)} + \beta v^{(g-1)} {v^{(g-1)}}^T$. This update strategy shifts the mutation distribution towards the Gaussian with highest probability of generating $v^{(g-1)}$. The CMA-ES algorithm is based on three main procedures: (1) *main loop*, (2) *step size updating procedure*, and (3) *covariance matrix updating strategy*. The CMA-ES main loop follows the classical $(1+1)$ scheme, where the offspring $x_{offspring}$ replaces the parent x_{parent} if its fitness value is better. Successively, the algorithm updates the step size, which is based on the heuristic that increases it if the success rate is high, and reducing it otherwise. The procedure performs an update based on a binary variable (λ_{succ}), which is set to 1 if $f(x_{offspring}) \leq f(x_{parent})$, with learning parameter $c_p \in (0, 1]$ using a target success rate p_{succ}^{target}. If $p_{succ} > p_{succ}^{target}$ the argument is greater then zero and the step increased; if $p_{succ} < p_{succ}^{target}$, the argument is smaller than zero and the step size is decreased otherwise it remains unchanged. Finally, the update of the covariance matrix and the evolution path (p_c) takes place if $f(x_{offspring}) \leq f(x_{parent})$, and it depends on the values of p_{succ}; if p_{succ} is high the update of p_c is blocked in order to prevent a fast increase of the C axis when the step size is low, otherwise the update occurs by an exponential smoothing. The new covariance matrix is a weighted mean of the old matrix and the outer product $p_c p_c^T$. Major details on CMA-ES can be found in (Auger and Hansen, 2005; Hansen and Ostermeier, 2001; Cutello et al., 2010).

Results

For our experiments we have used the classical artificial genetic networks that include an overall of 5 different instances: 2 instances with 2 genes (Vilela et al., 2008a), where 12 parameters need to be inferred for each; 1 instance with 4 genes (Vilela et al., 2008a) and 40 parameters to be inferred; and finally 2 artificial networks with 5 genes (Vilela et al., 2008a; Kikuchi et al., 2003; Noman and Iba, 2006), where 60 parameters must to be inferred. Of course, thanks to these experiments we are also able to evaluate the performances and efficiency of CMA-ES on this new kind of complex optimization task. Due to a limit pages we show in this section the results on the networks with 5 genes. In all experiments, we have considered the ranges where compute parameters ($\alpha, \beta \in [R_l, R_u]$, and $g_{ij}, h_{ij} \in [K_l, K_u]$), as well as initial conditions, the same ones used in the relative papers from where each instance has been taken into account. About CMA-ES algorithm, instead, we have fixed $\mu = \lambda = 100$, and 100 sample points; as termination criterion has been used a maximum number of fitness function evaluation fixed to 10^8. Moreover, each experiment has been performed over 10 independent runs as proposed in (Vilela et al., 2008a). In the first experiments presented in this section we compare CMA-ES with the algorithm proposed in (Vilela et al., 2008a) (in the follows called *Voit*'s algorithm), which is based on eigenvector optimization of a matrix formed from multiple regression equations of the linearized decoupled S-system. In these experiments we have tested CMA-ES on artificial gene networks with 2 (two in-

Table 2: CMA-ES versus state-of-the-art optimization algorithms. The comparisons have been done on a genetic network with 5 genes considering the *Euclidean distance* (d_euc) as evaluation measure. In all algorithms, for each gene has been included the best computed parameters (the rate constants α_i, β_i and the kinetic orders $g_{i,j}$, $h_{i,j}$).

gene	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}	d_{euc}
	CMA-ES												
X_1	5.0	0.0	0.0	1.0	0.0	−1.0	10.0	2.0	0.0	0.0	0.0	0.0	
X_2	10.0	2.0	0.0	0.0	0.0	0.0	10.0	0.0	2.0	0.0	0.0	0.0	
X_3	10.0	0.0	−1.0	0.0	0.0	0.0	10.0	0.0	−1.0	2.0	0.0	0.0	**0.0**
X_4	8.0	0.0	0.0	2.0	0.0	−1.0	10.0	0.0	0.0	0.0	2.0	0.0	
X_5	10.0	0.0	0.0	0.0	2.0	0.0	10.0	0.0	0.0	0.0	0.0	2.0	
	MO-HDE (Liu and Wang, 2008)												
X_1	4.95	0.0	0.0	1.007	0.0	−1.011	9.9	1.997	0.0	0.0	0.0	0.0	
X_2	9.95	1.992	0.0	0.0	0.0	0.0	9.96	0.0	1.999	0.0	0.0	0.0	
X_3	10.22	0.0	−0.968	0.0	−0.002	0.0	10.24	0.0	−0.966	1.998	0.0	0.0	0.05
X_4	7.93	0.0	0.0	2.009	0.0	−1.008	9.89	0.0	−0.004	0.0	1.993	0.0	
X_5	9.97	0.0	0.0	0.0	1.993	0.0	9.97	0.0	0.0	0.0	0.0	1.996	
	coop-CE (Kimura et al., 2005)												
X_1	4.917	−0.009	−0.003	1.019	−0.017	−1.014	9.922	2.021	−0.009	0.002	−0.009	−0.009	
X_2	10.03	1.995	0.002	−0.002	0.006	−0.001	10.026	0.002	1.995	−0.002	0.002	0.0	
X_3	9.851	−0.005	−0.991	−0.004	−0.003	0.002	9.835	−0.004	−0.993	2.036	−0.01	0.002	0.6178
X_4	8.02	−0.007	0.006	2.0	−0.002	−0.998	10.054	0.001	0.003	0.008	1.988	0.007	
X_5	9.875	−0.002	0.003	0.018	2.015	−0.02	9.892	0.004	0.002	0.008	−0.01	2.017	
	HDE (Tsai and Wang, 2005)												
X_1	5.0145	0.0	0.0	1.0128	0.0	−1.0031	10.01	1.9936	0.0	0.0	0.0	0.0	
X_2	9.9	1.99	0.0	0.0	0.0	0.0	9.871	0.0	1.99	0.0	0.0	0.0	
X_3	10.321	0.0	−0.963	0.0	0.0	0.0	10.344	0.0	−0.9594	1.9987	0.0	0.0	0.737
X_4	7.99	0.0	0.0	2.0157	0.0	−1.0026	9.981	0.0	0.0	0.0	2.0018	0.0	
X_5	9.966	0.0	0.0	0.0	1.985	0.0	9.967	0.0	0.0	0.0	0.0	1.997	
	***TDE*$_1$** (Noman and Iba, 2005)												
X_1	4.762	−0.021	−0.021	0.993	0.0	−1.013	9.607	1.916	0.0	0.0	0.0	0.0	
X_2	10.08	1.99	−0.001	0.035	0.0	0.0	9.817	0.0	1.938	0.0	0.012	0.0	
X_3	9.823	0.0	−1.00	−0.008	0.0	−0.001	9.835	0.0	−1.00	2.031	0.0	0.0	2.0597
X_4	7.182	0.0	−0.036	2.039	−0.052	−1.044	9.415	0.0	0.0	0.0	2.034	0.0	
X_5	10.103	0.0	0.005	0.05	1.997	−0.003	10.049	0.0	0.0	0.0	0.0	2.005	
	***TDE*$_2$** (Noman and Iba, 2006)												
X_1	4.99	0.0	−0.008	0.98	−0.004	−0.997	10.003	1.978	0.0	0.0	0.0	0.0	
X_2	10.051	1.995	0.004	0.009	0.002	−0.002	10.06	0.0	1.998	0.012	0.0	0.01	
X_3	9.936	0.004	−1.001	−0.001	0.0	0.0	9.937	−0.004	−1.001	2.007	0.0	0.001	2.2774
X_4	8.032	0.0	−0.011	1.949	0.0	−0.996	10.153	0.0	0.007	0.0	1.972	0.0	
X_5	10.011	0.0	0.003	0.023	2.002	−0.009	9.992	0.006	0.0	0.002	0.0	1.99	
	***PEACE*1** (Kikuchi et al., 2003)												
X_1	5.9	0.0	0.0	0.9	0.0	−0.9	10.6	1.7	0.0	0.0	0.0	0.0	
X_2	10.0	2.1	0.0	0.0	0.0	0.0	10.2	0.0	2.1	0.0	0.0	0.0	
X_3	9.6	0.0	−0.9	0.0	0.0	0.0	9.7	0.0	−0.9	2.3	0.0	0.0	74.0434
X_4	9.4	0.0	0.0	1.9	0.0	−0.9	11.5	0.0	0.0	0.0	1.8	0.0	
X_5	10.2	0.0	0.0	0.0	2.1	0.0	10.2	0.0	0.0	0.7	0.0	1.9	

Table 1: Success rate (SR) obtained by CMA-ES and *Voit*'s algorithm (Vilela et al., 2008a) on 10 independent runs. Both algorithms have been tested on an artificial network with 5 genes.

gene	CMA-ES	*Voit*'s alg. (Vilela et al., 2008a)
X_1	100%	100%
X_2	100%	100%
X_3	**100%**	0%
X_4	100%	100%
X_5	100%	100%

stances – normal and rescaled), 4 (rescaled) and 5 components. For each instance three different data sets have been used. All details about the instances can be found in the relative additional material (Vilela et al., 2008b). On the artificial genetic network with 2 genes both algorithms are comparable in term of success rate (SR), that is how many times the algorithm infers the parameters target. However, if we compare the results where both algorithms fail, it is possible to see as CMA-ES outperforms the compared algorithm in terms of *Euclidean distance* between the computed parameters and estimated parameters. This means that CMA-ES is able inferring the set of parameters closer to the estimated ones. The rescaled 2 genes network is, instead, equal to the normal one where however α and β are multiplied by a constant. In this experiment, using all three data sets, CMA-ES has been found the Ω target for each gene in all 10 runs with $SR = 100\%$, except for the gene X_2 of the 3rd data set where $SR = 70\%$. The compared algorithm, instead, presents $SR = 100\%$, excepts for the gene X_2 in the 2nd and 3rd data set, with respectively $SR = 80\%$ and $SR = 70\%$. In the overall, we can say that CMA-ES outperforms the *Voit*'s algorithm (Vilela et al., 2008a) in both artificial genetic networks with 2 components (in terms of

success rate and Euclidean distance). For the network instance with 4 genes, both algorithms are equivalent, since they have been able to find always the network target in all 10 runs ($SR = 100\%$). About the experiments on the artificial network with 5 genes, CMA-ES and the *Voit*'s algorithm have been compared on three different data sets. However, due to the limit pages we report in Table 1 only the results obtained on the 2nd data set. Inspecting the Table, is possible to see that albeit both algorithms reach a $SR = 100\%$ for the genes X_1, X_2, X_4, and X_5 CMA-ES is able also to inferring the estimated parameters for the gene X_3 on all 10 runs ($SR = 100\%$), where instead *Voit*'s algorithm fails with a zero success rate. Fig. 1 shows the gene expression

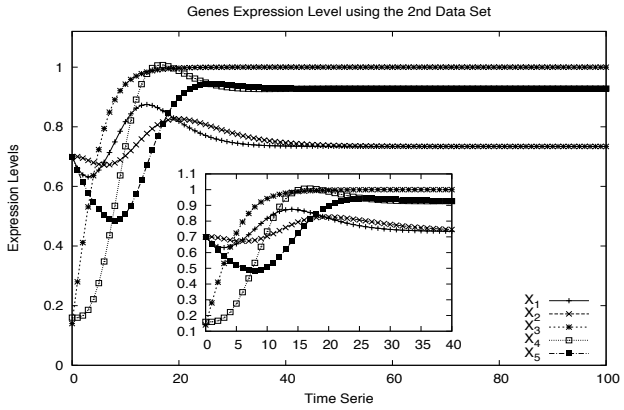

Figure 1: Gene expression levels computed by CMA-ES on the artificial network with 5 genes using 100 sample points.

levels computed by CMA-ES on the second data set. Looking the Table 1 is easy to understand how these curves represent exactly the gene expression levels of the target network. These plots have been obtained with a time-series based on 100 sample points. Instead, about the experiments in order to the other two data sets, we can say that from the obtained results is clear as both algorithms are able to estimate for all genes the target parameters, with $SR = 100\%$, about the first data set, unlike the third data set, where CMA-ES shows best performances (in the overall) than the *Voit*'s algorithm. In this last data set, CMA-ES outperforms *Voit*'s Algorithm on the genes X_1, X_4 and X_5 with a success rate of 100%, with respect to a success rate of 30% (X_1), and 0% (X_4 and X_5). Only on the gene X_3 CMA-ES fails over all 10 runs unlike the *Voit*'s algorithm that produces a $SR = 70\%$. However, although CMA-ES seems not comparable to the *Voit*'s algorithm for the gene X_3, if we take into account only the remaining 30% of the computed parameters by *Voit*'s algorithm, where $SR = 0\%$, and we compare them with all ones produced by CMA-ES is possible to note how the inferred parameters by our algorithm seem better in term of Euclidean distance from the expected target parameters. To better evaluate the robustness of our proposed algorithm on these kinds of complex optimization tasks we have compared CMA-ES with state-of-the-art algorithms on S-system models. For these new experiments the *Euclidean distance* from the estimated parameters has been chosen as evaluation measure. A new instance with 5 genes has been considered that is different from the previous one because different ranges have been used where compute Ω parameters. For this instance, moreover, it is important to point out that CMA-ES has been tested on 100 independent runs, producing an high success rate very closer to 100%. In Table 2 we report the comparisons of CMA-ES with the state-of-the-art, where only the best results for all algorithms have been included. The algorithms compared with CMA-ES are: (1) MO-HDE (Liu and Wang, 2008), a multi-objective optimization approach based on an hybrid differential evolution; (2) coop-CE (Kimura et al., 2005), a cooperative Coevolutionary algorithm; (3) HDE (Tsai and Wang, 2005), a hybrid differential evolution; (4) and (5) two different versions of trigonometric differential evolution (TDE_1 (Noman and Iba, 2005) and TDE_2 (Noman and Iba, 2006)); and finally (6) PEACE1 (Kikuchi et al., 2003) based on a Genetic algorithm. From the Table is clear as CMA-ES produces the best performances with zero Euclidean distance, whilst the best among the compared algorithms was able to reach 0.05 as Euclidean distance from the estimated parameters. The genetic algorithm is instead the one with worst performances. It is possible to claim that CMA-ES outperforms the current state-of-the-art optimization algorithms on S-system models.

Time-To-Target Analysis

Time-To-Target plots (Aiex et al., 2002) are a method to characterize the running time of stochastic algorithms to solve a given computational optimization problem. They display the probability that a given algorithm will find a solution as good as a target within a given running time. Nowadays they are standard graphical methodologies for data analysis to compare the empirical and theoretical distributions (Aiex et al., 2002, 2007). By Time-To-Target analysis two kinds of plots are produced: QQ-plot with superimposed variability information, and superimposed empirical and theoretical distributions.

We ran CMA-ES on the genetic network with 5 genes, and where the success rate in inferring the set of parameters of all genes is 100%. For this kind of experiments a different termination criterion has been properly tuned: until finding the parameters target for each gene ($SR = 100\%$). Because larger is the number of runs closer is the empirical distribution to the theoretical distribution, the plots presented in this section have been produced after 100 independent runs. The Fig. 2 shows the convergence process produced by CMA-ES using `tttplots.pl` on $n = 5$ network instance. In the top plot is showed the comparisons among empirical and theoretical distributions, whilst in bottom one is showed the

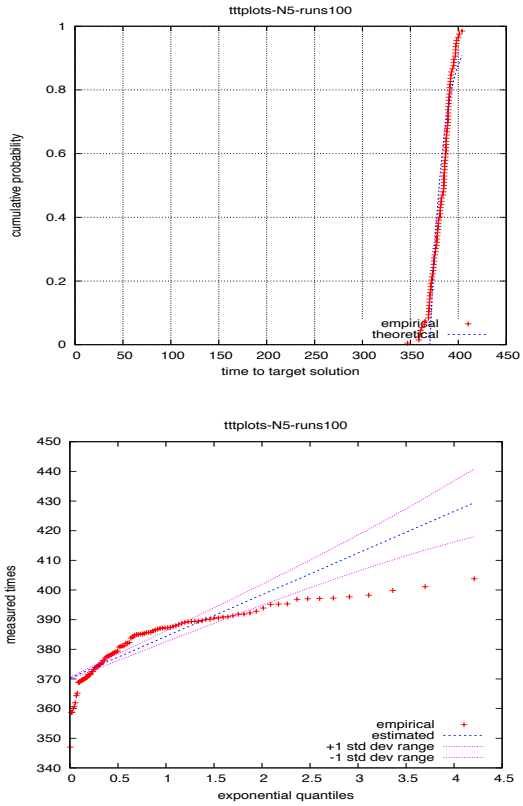

Figure 2: Time-To-Target plots for the network with 5 genes.

QQ-plot with variability information. Looking the top plot is possible to see as the curves of the empirical and theoretical distribution are equivalent. A different behaviour instead is showed on the bottom plot; this is likely due, since CMA-ES finds in quick way the parameters target.

Sensitivity Analysis – The Morris Method

The Morris method is one of the most popular models to evaluate the importance of any single parameter of a given system, showing own the main interactions between the parameters. In this method, any parameter assumes a discrete number of values chosen inside a range of variation; these values are called *levels*. Morris (Morris, 1991) has used a sensitivity analysis based on the *elementary effect* of the j–th parameter, defined as:

$$EE_j(p^*) = \frac{\left[f(p_1^*,\ldots,p_{j-1}^*,p_j^*+\Delta,p_{j+1}^*,\ldots,p_{N_p}^*)-f(p^*)\right]}{\Delta},$$

where Δ is a predetermined multiple of $1/(k-1)$ (k is the number of levels) . To understand what are the parameters, which influence on the output we have performed the Morris method for the S-Systems models, with $n = (2, 4, 5)$ genes. The obtained results are showed in Fig. 3. Two sensitivity measures, μ_j and σ_j, have been evaluated for any parameter

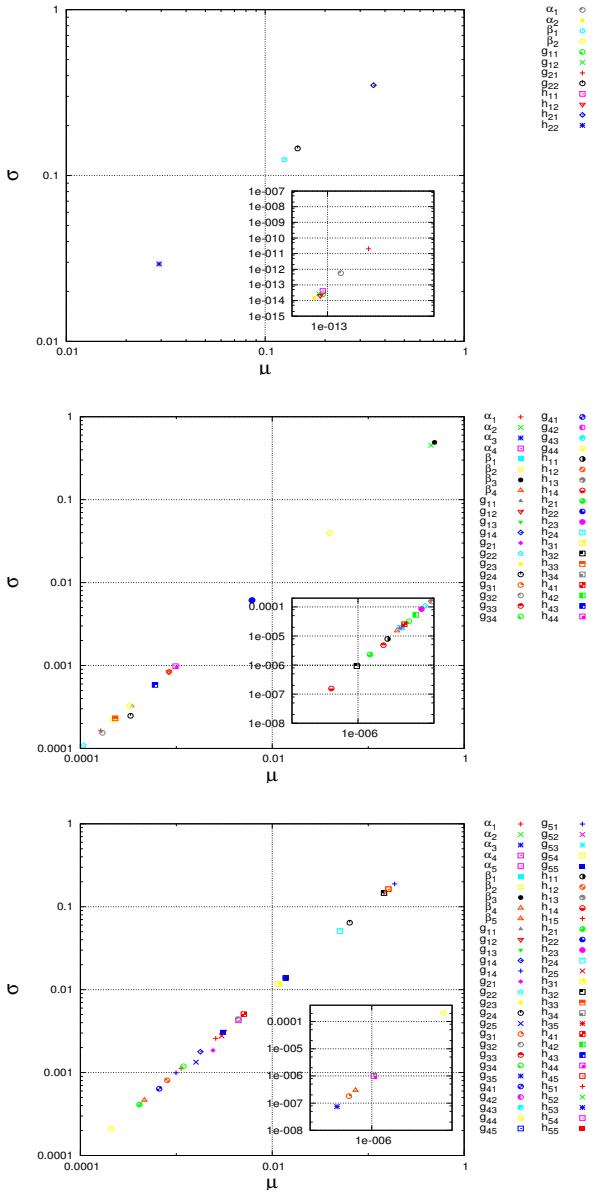

Figure 3: Sensitivity analysis by Morris method. A high value of μ indicates a parameter with an important overall influence on the output. A high value of σ indicates a parameter involved in interaction with other parameters or whose effect is nonlinear. The results show high μ and σ values for h_{21} and β_2 for the GRN with N=2 genes (top plot), for α_2 and β_3 for the GRN with N=4 (middle plot), for g_{24}, g_{44}, g_{51}, h_{15}, h_{24}, h_{32}, h_{45} for the GRN with N=5 (bottom plot).

j: the first represents an estimate of the mean of the distribution of the elementary effects, whilst the last indicates its standard deviation. High value of mean represents an important overall influence on the output by the given parameter; whereas high values of the standard deviation for the j–th parameter means that it is involved in interaction with other

parameters, or whose effect is nonlinear.

Conclusion

In this research paper we have presented a new approach for inferring the parameters in S-system models of gene regulatory networks. The CMA-ES algorithm has been used for this complex optimization task, and 5 different instances have been taken into account to evaluate its performances and its robustness: 2 instances with 2 genes - 12 parameters need to be inferred for each network; 1 instance with 4 genes - 40 parameters to be inferred; and 2 artificial networks with 5 genes - 60 parameters must to be inferred for each instance. The proposed algorithm has been compared with 7 state-of-the-art algorithms: (1) *Voit*'s algorithm, (2) MO-HDE; (3) coop-CE; (4) HDE; (5) and (6) two different versions of trigonometric differential evolution; and (7) PEACE1. The first experiments have been done on the instances with $n = (2, 4, 5)$ components taken from (Vilela et al., 2008a), comparing also CMA-ES with *Voit*'s algorithm. Due to the limit pages only the Table with the results obtained on a genetic network with $n = 5$ genes has been included in the paper. Analyzing the results obtained on all instances appear to be clear how CMA-ES is more able to estimating in a better way the parameters target, either in order to the success rate and in term of Euclidean distance. To have a better knowledge about the robustness of CMA-ES, we have compared it also with the current state-of-the-art algorithms, where the Euclidean distance has been used as evaluation metric. From these comparisons, CMA-ES is the only algorithm able to inferring the parameters effectively. Reviewing all experiments from an overall point of view is possible to claim that CMA-ES is an effective optimization algorithm for complex tasks, ranking as among one of the best reverse engineering methodologies on S-system models. Finally, in this research paper has been also included a study on the convergence process of CMA-ES through Time-To-Target plots, which are a way to characterize the running time of stochastic algorithms; and a global sensitivity analysis method, the Morris' algorithm.

References

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, 8:343–373.

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. (2007). Ttplots: A perl program to create time-to-target plots. *Optimization Letters*, 1:355–366.

Akutsu, T., Miyano, S., and Kuhara, S. (1999). Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Proceedings of Pacific Symposium on Biocomputing*, pages 17–28.

Auger, A. and Hansen, N. (2005). A restart cma evolution strategy with increasing population size. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1769–1776. IEEE Press.

Barrett, C. L., Kim, T. Y., Kim, H. U., Palsson, B., and Lee, S. Y. (2006). Systems biology as a foundation for genome-scale synthetic biology. *Current Opinions in Biotechnology*, 17(5):488–492.

Chen, T., He, H. L., and Church, G. M. (1999). Modeling gene expression with differential equations. In *Proceedings of Pacific Symposium on Biocomputing*, pages 29–40.

Cho, K. H., Choo, S. M., Jung, S. H., Kim, J. R., Choi, H. S., and Kim, J. (2007). Reverse engineering of gene regulatory networks. *IET System Biology*, 1:149–163.

Csete, M. and Doyle, J. (2002). Reverse engineering of biological complexity. *Science*, 295:1664–1669.

Cutello, V., Nicosia, G., Pavone, M., and Stracquadanio, G. (2010). Entropic divergence for population based optimization algorithms. In *Proceedings of IEEE World Congress on Computational Intelligence*, pages 1–8. IEEE Press.

Davidson, E. and Levin, M. (2005). Gene regulatory networks. In *Proceedings of the National Academy of Sciences of the United States of America - PNAS*, volume 102(14), page 4935.

D'haeseleer, P., Liang, S., and Somogyi, R. (2000). Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 18(8):707–726.

Friedman, N., Linial, M., Nachman, I., and Peer, D. (2000). Using bayesian networks to analyze expression data. In *Proceedings of Research in Computational Molecular Biology - RECOMB*, pages 127–135.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Irvine, D. H. and Savageau, M. A. (1990). Efficient solution of nonlinear ordinary differential equations expressed in s-systems canonical form. *SIAM journal of Numerical Analysis*, 27(3):704–735.

Kabir, M., Noman, N., and Iba, H. (2010). Reverse engineering gene regulatory network from microarray data using linear time-variant model. *BMC Bioinfomatics*, 11 (suppl. 1) S56.

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

Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S., and Konagaya, A. (2005). Inference of s-system models of genetic networks using a cooperative algorithm. *Bioinformatics*, 21(7):1154–1163.

Lee, D. (2005). Component-based software architecture for biosystem reverse engineering. *Biotechnology and Bioprocess Engineering*, 10:400–407.

Liu, P. K. and Wang, F. S. (2008). Inference of biochemical network models in s-system using multiobjective optimization approach. *Bioinformatics*, 24(8):1085–1092.

Mendes, P., Sha, W., and Ye, K. (2003). Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19:122–129.

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33 (2):161–174.

Noman, N. and Iba, H. (2005). Reverse engineering genetic networks using evolutionary computation. *Genome Informatics*, 16 (2):205–214.

Noman, N. and Iba, H. (2006). Inference of genetic networks using s-system information criteria for model selection. In *Proceedings of ACM Genetic and Evolutionary Computation Conference*, pages 263–270. ACM.

Noman, N. and Iba, H. (2007). Inferring gene regulatory networks using differential evolution with local search heuristics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4 (4):634–647.

Schlitt, T. and Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8.

Sirbu, A., Ruskin, H. J., and Crane, M. (2010). Comparison of evolutionary algorithms in gene regulatory network model inference. *BMC Bioinformatics*, 11 (58).

Swain, M. T., Mandel, J. J., and Dubitzky, W. (2010). Comparative study of three commonly used continuous deterministic methods for modeling gene regulation networks. *BMC Bioinformatics*, 11 (459).

Tegner, J., Yeung, M. K. S., Hasty, J., and Collins, J. J. (2003). Reverse engineering gene networks: Integrating genetic perturbations with dynamical modeling. In *Proceedings of the National Academy of Sciences of the United States of America - PNAS*, volume 100 (10), pages 5944–5949.

Tsai, K. Y. and Wang, F. S. (2005). Evolutionary optimization with data collocation for reverse engineering of biological networks. *Bioinformatics*, 21 (7):1180–1188.

Vilela, M., Chou, C., Vinga, S., Vasconcelos, A., Voit, E. O., and Almeida, J. (2008a). Parameter optimization in s-system models. *BMC Systems Biology*, 2 (35).

Vilela, M., Chou, C., Vinga, S., Vasconcelos, A., Voit, E. O., and Almeida, J. (2008b). Supplementary material to *parameter optimization in s-system models*.

Vohradsky, J. (2001). Neural network model of gene expression. *The FASEB journal: official publication of the Federation of American Societies for Experimental Biology*, 15 (3):846–854.

Wessels, L. F. A., Van Someren, E. P., and Reinders, M. J. T. (2001). A comparison of genetic network models. In *Pacific Symposium on Biocomputing*, volume 6, pages 508–519.