Bio-mimetic Evolutionary Reverse Engineering of Genetic Regulatory Networks

Daniel Marbach, Claudio Mattiussi, and Dario Floreano

Ecole Polytechnique Fédérale de Lausanne (EPFL),
Laboratory of Intelligent Systems,
CH-1015 Lausanne, Switzerland
{Daniel.Marbach,Claudio.Mattiussi,Dario.Floreano}@epfl.ch
http://lis.epfl.ch

Abstract. The effective reverse engineering of biochemical networks is one of the great challenges of systems biology. The contribution of this paper is two-fold: 1) We introduce a new method for reverse engineering genetic regulatory networks from gene expression data; 2) We demonstrate how nonlinear gene networks can be inferred from steady-state data alone. The reverse engineering method is based on an evolutionary algorithm that employs a novel representation called Analog Genetic Encoding (AGE), which is inspired from the natural encoding of genetic regulatory networks. AGE can be used with biologically plausible, nonlinear gene models where analytical approaches or local gradient based optimisation methods often fail. Recently there has been increasing interest in reverse engineering *linear* gene networks from steady-state data. Here we demonstrate how more accurate *nonlinear dynamical models* can also be inferred from steady-state data alone.

Key words: Systems Biology, Gene Networks, Reverse Engineering, Steady-State Data, Genetic Algorithm, Analog Genetic Encoding (AGE)

1 Introduction

Genetic regulatory networks perform fundamental information processing and control mechanisms in the cell. Regulatory genes code for proteins that enhance or inhibit the expression of other regulatory and/or non-regulatory genes, thereby forming a complex web of interactions (Fig. 1a). Inference and simulation of gene networks may contribute substantially to our biological knowledge in the post-genomic era. Practical applications may have a strong impact on biotech and pharmaceutical industries, potentially setting the stage for rational redesign of living systems and predictive, model-based drug design [1]. Technologies to assay gene expression levels in terms of mRNA concentrations are advancing at a fast pace. Using oligonucleotide chips or quantitative PCR for instance, it is possible to probe a set of genes of interest that are part of an uncharacterized gene network (henceforth known as *target network*) under different conditions. The goal of reverse engineering is inferring the target gene regulatory network from this experimental data.

The choice of a suitable reverse engineering method depends on the type of model used to describe the target network. Here we focus on models that represent a genetic regulatory network as a dynamical system described by a system of ordinary differential equations. The linear model [1–5], which is based on a first-order approximation of gene expression dynamics, is by far the most widely used gene network model. Its main advantage is that reverse engineering can be tackled analytically using standard techniques of system identification [1–6]. However, gene regulation is known to be strongly nonlinear. Hence, the linearization is generally only valid in a small regime, i.e., close to a specific steady-state [1, 3, 5]. This implies that valuable data from perturbation experiments with a strong effect on the network (e.g., gene knockouts) cannot be used because they fall outside the valid regime of the first-order approximation [1, 5]. Furthermore, the inferred linear model is unlikely to correctly predict network response under strong perturbations [1, 5], as can be expected in disease for instance.

As both quantity and quality of experimental data improve, we can aim at a more biologically plausible, faithful reconstruction of the target network. This requires the conception of adequate inference methods that can handle complex, nonlinear gene models, where analytical approaches and local gradient based optimisation often fail. In this paper we propose a bio-mimetic approach based on artificial evolution [7] using Analog Genetic Encoding (AGE) [8, 9], an artificial genetic representation that has already proven its merits in benchmark problems in the fields of analog electronic circuits [8, 9] and artificial neural networks [10]. Unlike other reverse engineering algorithms based on global optimisation techniques such as simulated annealing [11, 12] or conventional evolutionary algorithms [4, 13–15], AGE allows simultaneous inference of model structure and numerical parameter values. Furthermore, AGE mimics the evolutionary process of incremental complexification that natural gene regulatory networks are subjected to.

In the past, most reverse engineering studies used time-series gene expression data. However, time-series data are more difficult to obtain experimentally than steady-state data and their information content is lower (samples in a time-series are not independent). Indeed, there has been a recent trend towards approaches based on steady-state perturbation data [1, 3, 16, 17]. These studies use analytical approaches based on first-order approximations. Here we demonstrate for the first time – to the best of our knowledge – how *nonlinear* models (network structure and parameters) can be inferred from steady-state data alone.

2 Evolutionary Reverse Engineering with AGE

The first step in the reverse engineering process generally consists in the choice of a gene network model type (e.g. the sigmoid model introduced in Sect. 3). AGE is not constrained to a specific model type, but can be used with a large class of nonlinear models termed *analog networks* [9]. An analog network is composed of a collection of devices connected by links of different strengths. Here, devices are genes and links correspond to regulatory interactions[1]. Without limiting ourselves to a specific model type, we assume that genes are characterized by a vector of internal parameters **p** (e.g. decay rate, maximum transcription rate, etc.) and regulatory links have a single parameter called weight w. Within this framework, reverse engineering requires the specification of the network structure (size, topology) and the specification of the numerical values of all gene parameter vectors **p** and connection weights w. Using AGE, we encode these elements in a bio-inspired artificial genome. The reverse engineering process then amounts to the artificial evolution of gene networks that best match the experimental gene expression data (see Sect. 3.2). Apart from the bio-inspired genotype and mutation operators, the evolutionary algorithm is similar to a standard genetic algorithm [7]. It acts on a population of gene networks, which are encoded as described below. At each generation, fitter individuals are selected with higher probability for reproduction. Offspring are produced from the selected genomes by applying crossover and mutation operators as described in Sect. 2.2.

2.1 Genetic Encoding

We stress that AGE is not supposed to be a detailed model of the workings of gene networks, but a bio-inspired genotype that abstracts key features distinguishing the biological encoding from traditional artificial encodings used in genetic algorithms. Nature has chosen a digital encoding of genomes based on sequences of characters. Similarly, the AGE genome is constituted by one or more chromosomes, which are sequences of characters drawn from a genetic alphabet. The genetic alphabet used here has 26 nucleotides, which we designate with letters ‘A’-‘Z’. In nature, the beginning and the end of genes are marked by signals encoded in the DNA (promoters and terminators). Analogously, we use special nucleotide patterns (‘GN’ and ‘TE’) termed ‘tokens’ to delimit genes in the artificial genome as illustrated in Fig. 1. Consequently, genes may be located anywhere in the genome. Sequences that are not part of a gene are non-coding.

In a cell, the potential regulatory interaction between two genes A and B is not encoded *explicitly* in the genome, but follows *implicitly* from a biochemical process which depends among other things on: i) The coding region of gene A, which encodes the characteristics of protein A; ii) The regulatory region of gene B, which contains the potential binding sites for the regulatory protein (Fig. 1a; Note that there are other mechanisms of gene regulation not discussed here). Thus, the strength of the interaction is implicitly encoded by the respective coding and regulatory sequence. One of the consequences of the implicit encoding is that a *single* mutation in a coding or regulatory sequence may affect zero, one or several regulatory interactions simultaneously. In contrast, in an explicit (direct) encoding a single mutation affects only one characteristic of the network.

Analogously, artificial genes in AGE have a regulatory and a coding region as shown in Fig. 1b. The regulatory influence w_{ij} of gene j on gene i is implicitly

[1] In this paper we consider the simplest case where all devices are of the same type, but AGE can also handle heterogeneous networks [9]. We plan to use several device types in the future for more complex models of gene-protein networks, for instance.

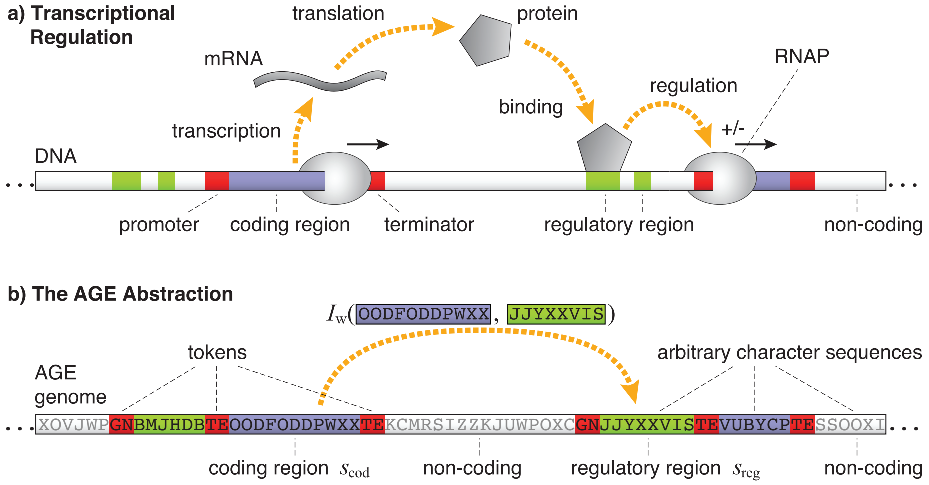
a) Transcriptional Regulation
translation
protein
mRNA
RNAP
regulation
binding
transcription
+/-
DNA
promoter
coding region
terminator
regulatory region
non-coding
b) The AGE Abstraction
I_w(OODFODDPWXX, JJYXXVIS)
tokens
arbitrary character sequences
AGE genome
XOVJWPGNBMJHDBTEOODFODDPWXXTEKCMRSIZZKJUWPOXCGNJJYXXVISTEVUBYCPTESSOOXI
coding region s_{cod}
non-coding
regulatory region s_{reg}
non-coding

Fig. 1. a) Simplified representation of transcriptional regulation. A gene is transcribed by RNA Polyermase (RNAP). Proteins are synthesized from mRNA (translation). Gene regulatory proteins bind to specific sites, enhancing or repressing the transcription rate of the following gene. Genetic regulatory networks are complex webs of such regulatory interactions. b) AGE chromosome encoding two genes. Analogous to the natural encoding, the beginning and the end of genes are marked by special nucleotide patterns in the artificial genome (tokens 'GN' and 'TE'). Genes have a coding region s_{cod} and a regulatory region s_{reg}, which may interact via an interaction map $I_{\text{w}}(s_{\text{cod}}\,, s_{\text{reg}})$ that abstracts the complex biochemical process illustrated in a)

encoded in the coding region $s_{\text{cod},j}$ and the regulatory region $s_{\text{reg},i}$ via an interaction map I_{w} that abstracts the complex biochemical process of transcriptional regulation: $w_{ij} = I_{\text{w}}(s_{\text{cod},j}\,, s_{\text{reg},i})$. The interaction map is based on local alignment of the two sequences [8, 9]. The closer the match between two subsequences of s_{cod} and s_{reg}, the stronger the interaction. Details are given in the Appendix.

In summary, decoding of the AGE genome involves the identification of valid genes (which must be correctly delimited by the corresponding tokens) and the subsequent application of the sequence interaction map to all pairs of coding and regulatory sequences. The interaction strength between two sequences may be zero, in which case there is no regulatory link between the two genes. Hence, the size of the decoded network is given by the number of genes in the genome and the topology and weights w follow from the computed interaction strengths.

For the gene parameters $\mathbf{p}$, it is desirable to use the same encoding as for the interactions [8]. Consider first the case where genes have a single parameter p. The value of p is decoded analogously to the weights by a sequence interaction map: $p = I_p(s_{\text{p},1}\,, s_{\text{p},2})$, where $s_{\text{p},1}$ and $s_{\text{p},2}$ are two additional sequences appended to the coding region of genes as shown in Fig. 2. Further gene parameters can be encoded by appending an additional pair of sequences for every additional parameter. Implementation details of I_p are also given in the Appendix.

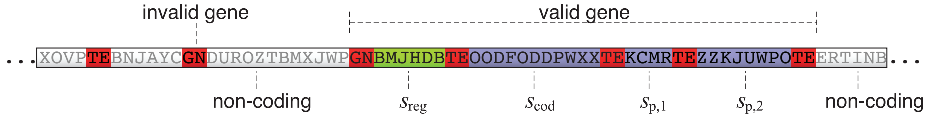

Fig. 2. In order to encode a gene parameter p, genes must have two additional sequences $s_{\text{p},1}$ and $s_{\text{p},2}$. Thus, a valid gene with one gene parameter has four sequences separated by tokens 'TE'. The token 'GN' to the left of the valid gene is not followed by the necessary four tokens 'TE', thus it is not coding

2.2 Genetic Operators

One of the key features of AGE is the possibility to apply a wide range of biologically inspired genetic operators that are believed to play important roles in the evolution and complexification of natural genetic regulatory networks [8]. From the point of view of the genetic operators, the tokens that delimit the genes have no special meaning – there is no distinction between tokens, coding and non-coding genome fragments. The operators described below are applied probabilistically to randomly chosen parts of the genome (see Sect. 3.2).

- *Nucleotide deletion, insertion, and substitution*: A character is removed, inserted, or substituted in the genome. Random characters from the genetic alphabet are used for insertions and substitutions.
- *Chromosome fragment deletion, transposition, and duplication*: Two points are chosen in a chromosome and the intervening genome fragment is deleted, transferred or copied to another point of the genome.
- *Chromosome deletion/duplication*: A chromosome is deleted/duplicated.
- *Crossover*: Chromosomes of parents are recombined using homologous crossover, which is based on the search of a homologous crossover point [8].

The application of the genetic operators can invalidate genes (e.g., through invalidation of a token) and transform the corresponding fragments into non-coding genome, which may play the role of an evolutionarily useful repository of genetic fragments. On the other hand, new genes can be created, for example through the appearance of new tokens or the duplication of a genome fragment.

3 Experiments

When applying a novel reverse engineering technique directly to biological data, performance evaluation is difficult because the target network is unknown. Thus, we first test AGE using synthetic expression data generated in simulation from an *in silico* target gene network. Subsequently the inferred networks are compared with the target network in order to validate the results. This is a standard approach to assess the performance of reverse engineering methods [5].

3.1 The Test Case

Gene Network Model. We demonstrate the application of AGE using a standard sigmoid model [11–13, 18] defined by the system of state equations:

$$dx_i/dt \;=\; m_i \cdot \sigma\Big(\sum_{j \in R_i} w_{ij} x_j \Big) \;-\; \delta_i x_i \,, \tag{1}$$

where x_i is the expression level of gene i, m_i is the maximum transcription rate, and δ_i is the degradation rate. R_i is the set of regulators of gene i and w_{ij} represents the regulatory influence of gene j on gene i (positive for enhancers, negative for repressors). The activation function is a sigmoid $\sigma(z) = 1/(1+e^{-z})$.

In the experiments reported in this paper we use steady-state expression levels as input data for the inverse problem, though AGE could as well be applied to time-series data. At steady-state, the state equations become a set of algebraic equations:

$$0 = p_i \cdot \sigma\big(\sum_{j \in R_i} w_{ij} x_j \big) - x_i \,, \quad \text{with } p_i = m_i/\delta_i \,. \tag{2}$$

Synthetic Target Network and Expression Data. We employ the topology of a nine-gene subnetwork of the *E.coli* SOS pathway as described in [1] as test case (see Fig. 4). We refer to this topology as *SOS network*. There is no quantitative model of the SOS network available in the literature. Hence, numerical parameter values for the weights w and the parameter p of the steady-state equation introduced above are sampled randomly (see Appendix). The sign of the weights is set according to the SOS network topology (positive for enhancers and negative for repressors). The resulting *in silico* target gene networks are *random targets* with a realistic topology, which is a more biologically plausible approach than random generation of both topology and parameters [5, 13].

Synthetic gene expression data is obtained by applying a perturbation to the *in silico* target network and computing the steady-state expression levels of all genes[2]. This process is repeated for different perturbations to gather the necessary experimental data for reverse engineering. In our experiments we use two different types of perturbations that are commonly used for gene network inference: Gene knockouts, i.e. silencing of a particular gene, and gene over-expression, which consists in artificially boosting the transcription rate of a gene. A gene knockout can be simulated by setting the rate parameter m_i to zero. Consequently, the expression level x_i of this gene at steady-state will be zero. Over-expression is simulated by doubling the m_i value of the affected gene.

For the experiments reported below we generate expression data from the *in silico* SOS network for the wild type (unperturbed network) and for 9 knockout and 9 over-expression experiments (knockout and over-expression of each gene).

[2] We compute the steady-states numerically using Powell's method of the GNU Scientific Library (GSL, http://www.gnu.org/software/gsl).

3.2 The Evolutionary Algorithm

The weights w and the single gene parameter p of the steady-state sigmoid model are encoded in the artificial genome as explained in Sect. 2. Details of the sequence interaction maps are given in the Appendix.

Since we do not yet model noise, the least squares optimisation criterion is suitable for the definition of the fitness. The goal of the evolutionary algorithm is to minimize the fitness function: $f(\hat{\mathbf{X}}) = \sum_{i=1}^{M} \sum_{j=1}^{N} (x_{ij} - \hat{x}_{ij})^2$, where $\mathbf{X}$ is the synthetic gene expression data generated from the target network (element x_{ij} corresponds to the expression level of gene j in experiment i) and $\hat{\mathbf{X}}$ are the corresponding expression levels in the inferred network. M denotes the number of different perturbation experiments and N is the number of genes. Thus, figuratively speaking, the reverse engineering process amounts to finding the gene network that best fits the target expression data.

We use the following parameters for the evolutionary algorithm. The population size is 200. We use elitism, i.e., the best individual is protected from replacement. At each generation, 40 parents are chosen using tournament selection [7]. From the 40 parents, 200 new individuals are created and genetic operators are applied with:

– Probability of homologous crossover (per individual)	0.1
– Prob. of nucleotide deletion, insertion and substitution (*per nucleotide*)	0.001
– Prob. of chromosome fragment deletion, transposition and duplication (per chromosome)	0.01
– Prob. of chromosome deletion and duplication (per chromosome)	0.001

The choice of the parameters listed above is not critical. They were chosen heuristically based on a series of test runs and the experiences reported in [8, 9]. We observe no significant difference in the quality of results obtained with different standard selection and replacement strategies. Note that the mutation rates were chosen such that the more disruptive whole chromosome and chromosome fragment mutations occur less frequently than single nucleotide mutations.

3.3 Results

The results of a batch of ten reverse engineering runs using synthetic data from a randomly initialized SOS gene network as explained above are shown in Fig. 3. For each run, we record the fitness $f(\hat{\mathbf{X}})$ of the best individual at every generation of the evolutionary algorithm. Of course, in addition to a good fit of the expression data, the structure of the inferred networks should match the target gene network. In a real biological application the structure of the target network is unknown, but in the synthetic test case employed here the accuracy of the inferred networks can be quantified. To this end we use the mean square error $E(\hat{\theta})$ of all parameters of the reverse engineered network $\hat{\theta}$ (including all weights w_{ij} and gene parameters p_i) compared to the true parameter values θ of the *in silico* target gene network: $E(\hat{\theta}) = 1/K \cdot \sum_{l=1}^{K} (\theta_l - \hat{\theta}_l)^2$, where θ_l denotes the l-th element of parameter vector θ and K is the total number of parameters.

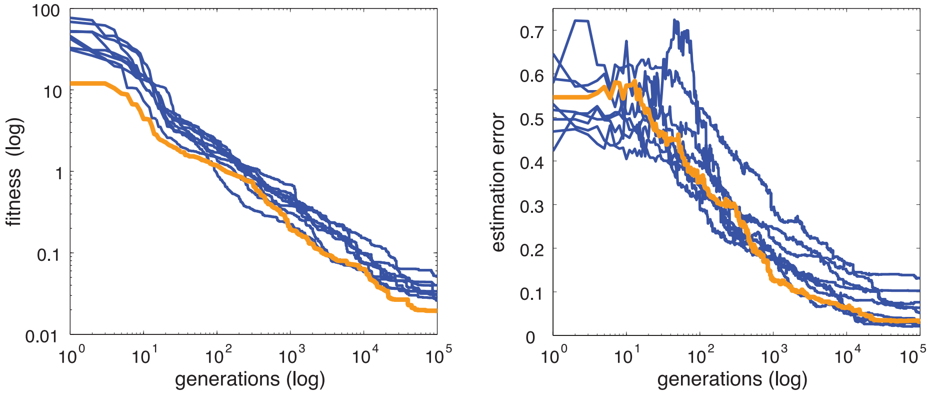

Fig. 3. Reverse engineering of the SOS network – ten runs. The fitness $f(\hat{\mathbf{X}})$ (left) and the estimation error $E(\hat{\theta})$ (right) are plotted for the best individual of each run. As the fitness is optimized (i.e., minimized), the inferred networks match the structure and parameters of the target network with increasing accuracy (the estimation error goes down). The run with the best final fitness is highlighted. For details, see main text

We refer to $E(\hat{\theta})$ as *estimation error* of the inferred network. In addition we also count the number of *false positives* and *false negatives*[3].

The ten runs shown in Fig. 3 were executed for $100'000$ generations of the evolutionary algorithm. Average computation time was roughly two days for one run on a standard desktop PC. All ten runs achieve a fitness below 0.1. Since fitness is a *sum* of square errors, the individual expression levels are fitted extremely accurately with a relative error in the order of 1% . As the reverse engineering algorithm optimizes the fitness, the estimation error of the inferred networks goes down. Four out of ten runs inferred the SOS network with high precision[4] (final estimation error between 0.02 and 0.03). In other words, these runs closely matched the structure, weights and gene parameters of the target network. The other runs converged at estimation errors of about 0.1 .

In a set of reverse engineering runs, one would like to choose the inferred network with the lowest estimation error $E(\hat{\theta})$. However, since $E(\hat{\theta})$ is unknown in a real application, this is not possible. Hence, we choose the inferred network with the best fitness as the most plausible reconstruction of the target network[5]. Here, the best run (see Fig. 3) achieves a fitness of 0.02 and the corresponding

[3] We count as *false positive* when a target weight $w_{ij} = 0$ and the absolute value of the inferred weight $|\hat{w}_{ij}| > 0.1$; A *false negative* occurs when $w_{ij} \neq 0$ and $|\hat{w}_{ij}| < 0.1$.

[4] Further analysis indicates that the accuracy achieved by the best runs corresponds to a lower bound given by the discretization of the search space due to the quantization of the parameters and weights.

[5] In a real application, one should not just consider the inferred network with the best fitness – which merely corresponds to the network with the highest *a posteriori* probability – but analyze all well-scoring (i.e., probable) inferred networks [13].

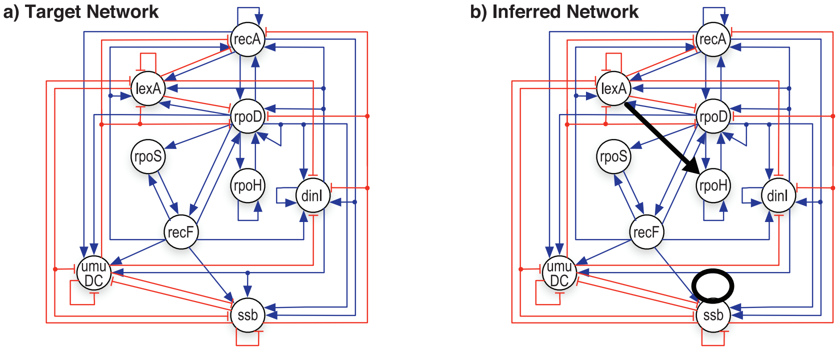

Fig. 4. a) Topology of the *E.coli* SOS network [1]. Arrows are enhancing, Tee ends denote inhibitory interactions. b) The topology inferred by the run with the best final fitness is identical except for one false positive (bold) and one false negative (encircled)

network has a very low estimation error of 0.03 with only one false positive and one false negative out of a total of 81 possible connections (see Fig. 4b).

In other experiments with target SOS networks with different random initializations of the parameter values we have obtained the same quality of results. AGE infers networks with very good fitness in every run. Roughly 40% of the runs also achieve very low estimation errors (i.e., they correctly infer the target, having only very few false positives and false negatives). Simpler networks, e.g., cascades of size six, are inferred correctly in every run. In addition, we have also tested gradient descent and observed that it is not successful at inferring the sigmoid model – even when restarted many times – because it prematurely converges to local optima with very bad fitness and high estimation error.

4 Conclusion

We have introduced a new approach for reverse engineering genetic regulatory networks. The method is based on artificial evolution with a bio-inspired genetic encoding (AGE), which allows simultaneous inference of numerical parameter values and model structure (network size, topology and – in heterogeneous networks – the type of the nodes). AGE is not constrained to a specific gene network model type, but can be used with a large class of nonlinear models.

Using a standard sigmoid model as test case, we have successfully reverse engineered the *E.coli* SOS network from synthetic steady-state gene expression data. The SOS network arguably has a more complex and densely connected topology than typical target networks used for inference methods based on global optimisation techniques [11–15]. Thus, our results demonstrate the competitiveness of AGE for inference of complex nonlinear gene regulatory networks.

There has been a recent trend towards reverse engineering methods using steady-state perturbation data [1, 3, 16, 17]. However, those approaches are based on first-order approximations. Here we propose for the first time – to the best of our knowledge – the use of steady-state perturbation data for reverse engineering of *nonlinear* models. Considering the advantages of steady-state with respect to time-series data and based on the encouraging results of our test case, we believe this to be an extremely promising approach.

Experiments reported here were conducted with synthetic, noise-free expression data. We are currently working both on an application to real expression data and on a more realistic *in silico* test case based on a mechanistic model of a well-characterized gene-protein network. In addition, we also intend to take advantage of the flexibility of AGE in order to explore novel, more biologically plausible gene network models than the sigmoid model employed here.

Acknowledgments. We thank Peter Dürr, Sara Mitri, Tim Stirling and Fanny Riedo for discussions and comments on the manuscript. This work was supported by the Swiss National Science Foundation, grants no. 620-58049 and 200021-112060.

References

1. Gardner, T.S., di Bernardo, D., Lorenz, D., Collins, J.J.: Inferring genetic networks and identifying compound mode of action via expression profiling. Science **301**(5629) (2003) 102–5
2. D'Haeseleer, P., Wen, X., Fuhrman, S., Somogyi, R.: Linear modeling of mRNA expression levels during CNS development and injury. Pac Symp Biocomput (1999) 41–52
3. Brazhnik, P.: Inferring gene networks from steady-state response to single-gene perturbations. J Theor Biol **237**(4) (2005) 427–440
4. Corne, D., Pridgeon, C.: Investigating issues in the reconstructability of genetic regulatory networks. In: Congress on Evolutionary Computation. (2004)
5. Yeung, M.K.S., Tegnér, J., Collins, J.J.: Reverse engineering gene networks using singular value decomposition and robust regression. PNAS **99**(9) (2002) 6163–8
6. Ljung, L.: System identification: Theory for the user. Prentice Hall, Upper Saddle River, NJ (1999)
7. Bäck, T., Fogel, D., Michalewicz, Z., eds.: Evolutionary Computation 1: Basic Algorithms and Operators. Institute of Physics, Bristol (2000)
8. Mattiussi, C.: Evolutionary synthesis of analog networks. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, Lausanne (2005)
9. Mattiussi, C., Floreano, D.: Analog Genetic Encoding for the Evolution of Circuits and Networks. IEEE Transactions on Evolutionary Computation (2006, To appear)
10. Dürr, P., Mattiussi, C., Floreano, D.: Neuroevolution with Analog Genetic Encoding. In: Parallel Problem Solving from Nature - PPSN IX. Volume 4193 of LNCS. (2006) 671–680
11. Reinitz, J., Sharp, D.H.: Mechanism of eve stripe formation. Mech Dev **49**(1-2) (1995) 133–58
12. Jaeger, J., Surkova, S., Blagov, M., Janssens, H., Kosman, D., Kozlov, K.N., Manu, Myasnikova, E., Vanario-Alonso, C.E., Samsonova, M., Sharp, D.H., Reinitz, J.: Dynamic control of positional information in the early drosophila embryo. Nature **430**(6997) (Jul 2004) 368–371
13. Wahde, M., Hertz, J., Andersson, M.: Reverse engineering of sparsely connected genetic regulatory networks. In: Proceedings of the 2nd Workshop of Biochemical Pathways and Genetic Networks. (2001)

14. Noman, N., Iba, H.: Inference of gene regulatory networks using S-system and differential evolution. In: GECCO'05. (2005)
15. Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S., Konagaya, A.: Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. Bioinformatics **21**(7) (2005) 1154–63
16. Kholodenko, B.N., Kiyatkin, A., Bruggeman, F.J., Sontag, E., Westerhoff, H.V., Hoek, J.B.: Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. PNAS **99**(20) (2002) 12841–12846
17. de la Fuente, A., Brazhnik, P., Mendes, P.: Linking the genes: inferring quantitative gene networks from microarray data. Trends Genet **18**(8) (Aug 2002) 395–398
18. Weaver, D.: Modeling regulatory networks with weight matrices. In: Pacific Symposium on Biocomputing. (1999)

Appendix

We give here only a short description of the sequence interaction maps I_w and I_k introduced in Sect. 2.1 due to space limitation. For details, refer to Refs. [8, 9].

The sequence interaction map $I_\mathrm{w}(s_\mathrm{cod}\,, s_\mathrm{reg})$ that decodes the weights w_{ij} from the respective coding and regulatory regions of genes i and j is defined as a composed map $N_\mathrm{w}(L(s_\mathrm{cod}\,, s_\mathrm{reg})$, which is formed by a generic interaction map $L(s_\mathrm{cod}\,, s_\mathrm{reg})$ and a network-specific map $N_\mathrm{w}(i)$. The generic interaction map $L(s_\mathrm{cod}\,, s_\mathrm{reg})$ is defined as the local alignment score of the two sequences [8], using the same alignment parameters as Ref. [9]. Figuratively speaking, the closer the match between two subsequences of s_cod and s_reg, the stronger the interaction. Simpler techniques of sequence comparison such as exact matching or Hamming distance would compromise evolvability [8]. The network-specific map $N_\mathrm{w} : [i_\mathrm{min}, i_\mathrm{max}] \mapsto [0, w_\mathrm{max}]$ transforms integer local alignment scores i into floating-point weights. Alignment scores smaller than the threshold i_min are mapped to zero (no interaction), scores greater than i_max are truncated to w_max, and scores in between are mapped linearly onto the positive interval $[0, w_\mathrm{max}]$. In the experiments reported in this paper, we used $i_\mathrm{min} = 11$, $i_\mathrm{max} = 31$ and $w_\mathrm{max} = 2$.

The alignment score given by the interaction map is always positive. In order to represent negative weights, genes actually have two sequences $s_\mathrm{cod+}$ and $s_\mathrm{cod-}$ corresponding to enhancing and repressing regulatory activity (for clarity, only one sequence s_cod was mentioned in Sect. 2.1). The weight is defined by the stronger interaction: $w = +I_\mathrm{w}(s_\mathrm{cod+}, s_\mathrm{reg})$ if $I_\mathrm{w}(s_\mathrm{cod+}, s_\mathrm{reg}) \geq I_\mathrm{w}(s_\mathrm{cod-}, s_\mathrm{reg})$ and $w = -I_\mathrm{w}(s_\mathrm{cod-}, s_\mathrm{reg})$ otherwise.

Analogously to I_w, the sequence interaction map $I_p(s_\mathrm{p,1}\,, s_\mathrm{p,2})$ used for decoding gene parameter p from the respective sequences $s_\mathrm{p,1}$ and $s_\mathrm{p,2}$ is implemented as a composed map $N_p(L(s_\mathrm{p,1}\,, s_\mathrm{p,2}))$, using the same generic interaction map L defined above. $N_p : [i_\mathrm{min}\,, i_\mathrm{max}] \mapsto [p_\mathrm{min}\,, p_\mathrm{max}]$ maps the integer local alignment scores onto the interval of parameter p analogously to N_w described above.

Finally, we briefly discuss the random sampling of numerical parameter values for the *in silico* target network (Sect. 3.1). Weights w_{ij} were initialized uniformly in the range $[0.15, 1.5]$ and parameters p_i in the range $[1/2, 2]$. These ranges were selected empirically with the goal to obtain rich nonlinear dynamics in the target networks, i.e., so that on average the total regulatory input for the sigmoid activation functions was neither completely saturated nor constrained to a very small, almost linear regime.