Evolutionary Generation of Small Oscillating Genetic Networks

Matthijs van Dorp, Bruno Lannoo, and Enrico Carlon

Institute for Theoretical Physics, KULeuven, Celestijnenlaan 200D, 3001 Leuven, Belgium

Abstract. We discuss the implementation and results of an evolutionary algorithm designed to generate oscillating biological networks. In our algorithm we have used a type of fitness function which defines oscillations independent of amplitude and period, which improves results significantly when compared to a simple fitness function which only measures the distance to a predefined target function. We show that with our fitness function, we are able to conduct an analysis of minimal oscillating motifs. We find that there are several different examples of mechanisms that generate oscillations, which make use in various ways of transcriptional regulations, complex formation and catalytic degradation.

Keywords: Genetic algorithms, gene regulatory networks, protein interaction networks.

1 Introduction

Evolutionary algorithms have been used for about 10 years to investigate the structural properties of biological networks [1–6] by constructing them "ab-initio" using a predefined set of evolutionary rules. Here, we use an evolutionary algorithm to generate networks of interacting genes and proteins which have an oscillatory output. The aim is to systematically build up small networks (or "motifs") in order to gain some insights on the core mechanisms responsible for the oscillations. In the generated motifs, protein concentrations follow a stable self-sustained oscillating pattern. The motifs provide examples of oscillators that may be compared to known biological oscillators, such as circadian clocks, and they may improve understanding of the underlying mechanisms of oscillation in this type of networks.

Circadian clocks have long been studied using models that feature a negative feedback loop due to the regulation of gene expression, affecting the production rate of proteins. However, recent results [7] showed that circadian rhythms still persist even when such regulatory mechanisms are disabled. This proves that there are autonomous oscillations due to interactions between proteins, suggesting that the structure of feedback loops is probably more complex than initially expected. Efficient evolutionary algorithms may provide new insights in the modeling of circadian clocks, as many different network architectures could be generated and compared with each other. In addition, experimental techniques are

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 120–129, 2013.

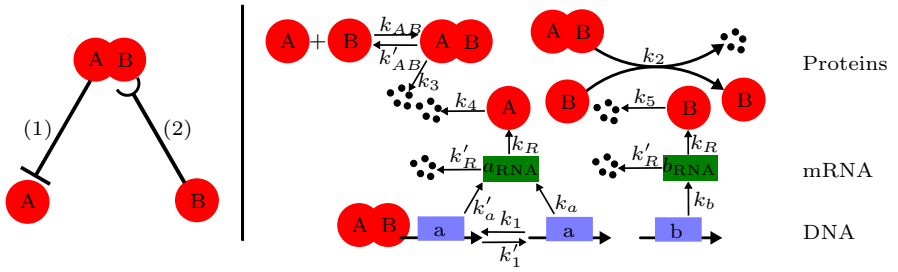


Fig. 1. An example of two representations of the same biological network, with a schematic representation (left) and a complete description with all kinetic rates (right). In the schematic representation, only the "core" interactions (those which define the topology) are given and only proteins are shown. The core interactions shown are (1) a repression of the synthesis of mRNA from gene A by the dimer AB (repression is indicated by a '$\dashv$' symbol, activation by a '$\rightarrow$' symbol), and (2) a catalytic degradation of the dimer AB by B, i.e. the reaction $B + AB \rightarrow B$ which is indicated as '—$\subset$'. The groups of small dots in the full representation indicate degradation.

rapidly improving, and in the near future high quality data is expected to allow discrimination between motifs based on their specific output.

In this paper we revisit an algorithm, originally proposed in [1], by improving its efficiency and extending it further. We show that such an algorithm is capable of rapidly generating oscillating motifs thanks to optimally selected score functions. We show how the different types of biomolecular reactions may be combined to produce an oscillating output. Quite interestingly, in view of the recent experimental results [7], the output also provides examples of purely post-transcriptional oscillators which do not rely on oscillations of mRNA concentrations.

2 Algorithm

The allowed biological mechanisms and interactions used in this work are transcriptional regulation, formation and dissociation of protein complexes, and catalytic degradation. They are shown graphically in Figure 1(right), which provides an example of a small network in its full representation. In Fig. 1(left) we provide a schematic representation (as used throughout the paper) of the same network where only proteins are shown. The latter representation omits the genes, which produce mRNA, and the mRNAs which produce the gene-specific protein. In order to simulate the temporal behavior of a network, we use deterministic mass-action kinetics. The resulting set of nonlinear first-order ODEs are solved by the Runge-Kutta-Fehlberg Method (RKF45).

2.1 Evolutionary Fitness

An appropriate fitness function is highly important both for fast convergence of the algorithm and for the evolution of motifs that are as small as possible.

Additionally, the choice of the fitness function will cause certain evolutionary pathways to be much more accessible than others, and therefore the results will depend strongly on the fitness function that is chosen. Previous algorithms designed for the same purpose have varied the exact selection mechanism, such as elitist evolution or tournament selection [8, 9], but the notion of fitness has always been related to a specific concentration profile. We have abandoned this kind of scoring, as the associated requirements for amplitude and period are very restrictive. Instead, a fitness function should reward oscillatory behavior without being tied to any predefined shape, and not even distinguish between sawtooth profiles, pulses, and sine-like waves. The most straightforward way of defining oscillation independent of the profile is to analyse its peak-to-trough behavior. This leads to the fitness function

$$S = 20 - 2\sum_{i=1}^{10} \frac{|a_i - a_{i+1}|}{a_i + a_{i+1}} \min\left(1, |a_i - a_{i+1}|\right), \tag{1}$$

where the a_i are the first 11 extrema, ordered by time, of the concentration of a target protein $A(t)$. If $A(t)$ has less than 11 extrema, the sum only includes the available terms, and $S = 20$ for all monotonic functions. Equation (1) is the mathematical equivalent of the intuitive concept that oscillations are of a good quality if the peaks are relatively much higher than the troughs. The $\min\left(1, |a_i - a_{i+1}|\right)$ on the right is a technical correction which we added to prevent the algorithm from evolving very low-concentration oscillations. A significant advantage of this specific scoring function is that a strongly damped oscillation is recognised and rewarded, which is helpful for fast convergence. The convergence threshold is set to $S = 4$ since for such values stable oscillations are seen in any network, except for some rare cases which feature very weakly damped oscillations.

2.2 Topological Reduction

In spite of choosing a fitness function that favors fast convergence, the evolutionary procedure almost inevitably enlarges the size of the network by adding links and nodes which are not necessary for oscillatory behavior. The simplest topological reduction would consist of cutting away parts of the network as long as this does not destroy oscillations. However, it is likely that the network can be reduced further if the kinetic rates are varied to compensate for the removal of a component. Therefore, we have applied additional topological evolution, which computes a topological fitness score related to the topological components. We constrained the evolution by removing any networks that did not satisfy $S < 5$ in order to preserve oscillatory function, but other than this constraint the topological evolution could in principle evolve freely towards smaller networks. In order to improve convergence, we also incorporated the kinetic rates into the topological fitness score such that interactions could have their importance reduced gradually, increasing the likeliness that they could be removed entirely in a future generation.

2.3 Evolutionary Process

In order to illustrate the evolutionary algorithm, let us consider the pseudocode for the functions "evolve()" and "reduce_topological_size()" as displayed below.

```
function evolve()
  make_initial_network()
  for n = 0 to 300
    mutate_networks()        /* Create mutant for each network */
    score_networks()         /* Compute S for all networks */
    if S_min ≤ 4
      break                  /* Stop evolution once fitness threshold is reached */
    prune_networks(100)      /* Keep only 100 best scoring networks */
    if T > 15
      reduce_topological_size(S_min + 1)
  if S_min > 4
    return                   /* Failed to evolve oscillations */
  reduce_topological_size(5) /* Reduce T of oscillating network */
  for n = 0 to 20            /* Optimize network score */
    mutate_networks_kinetic()  /* Only evolve rate constants */
    score_networks()
    prune_networks(100)
  save_best_network()        /* Store network with lowest S */
```

```
function reduce_topological_size(int maxscore)
  for n = 0 to 500
    mutate_networks()        /* Create mutant for each network */
    check_fitness(maxscore)  /* Remove networks with bad fitness */
    score_topology()         /* Compute T for all networks */
    prune_networks(10)       /* Keep only 10 best scoring networks */
```

The first function, evolve(), shows the general course of evolution. Every generation consists of duplicating all networks and mutating each duplicate in a random way. These mutations include topological modifications as well as changes to kinetic rate constants. Subsequently, the score of all networks is computed and the worst performing networks are discarded. The second function, as described in Sec. 2.2, performs a topological reduction which aims to shrink the network size, and it is used by the first function when the networks have grown too large, or in order to reduce the size of an evolved network with oscillatory behavior.

3 Results

The evolution results in a large pool of networks of various complexities. As the number of possible topologies is small when the complexity of the graph is restricted, we analyse the collection of about 15,000 evolved networks with a topological size $T \leq 5$ (where T is defined as the total number of genes,

protein complexes, regulations, and catalytic degradations). The total number of distinct topologies was less than 100. We shall discuss the differences and similarities between the topologies obtained.

3.1 Parameter Variation

The population of oscillating networks that results from repeatedly running an evolutionary algorithm depends not only on the fitness function, which is the same for all networks discussed here, but also on the choice of parameters and evolutionary limits. A very broad range of allowed kinetic rates decreases the performance of the algorithm, in particular because it increases the computational cost of numerically solving the system of ODEs for a network. We use a fixed time window within which the oscillations should take place, which is sufficiently large compared to allowed kinetic rates such that a wide range of periods is found (across more than one order of magnitude). Effectively, the minimum possible periods are dictated by the upper allowed limits of kinetic rates, while the maximum possible period is a consequence of the fitness computation which demands a minimum number of oscillations in the maximum time allowed for integration. We use several parameter settings in order to search different regions of parameter space. In the default setting, we assume kinetic rates to be around 1.0 such that all timescales in the system are similar. In a second case we assume that the formation of complexes is significantly faster than transcriptional regulation and degradation. In a third setting we have disabled the evolution of catalytic degradations, which lead to a larger proportion of repressing regulations, and in a final setting we loosen the limits on all mutable rates such that the search space is significantly larger.

3.2 Mechanisms of Oscillation

Oscillatory function is known to be connected to negative feedbacks that incorporate a time delay, and our simplest networks use no more than a few distinct patterns in spite of belonging to many different topologies. Examples of those patterns are given in Figures 2, 3, 4 and 5.

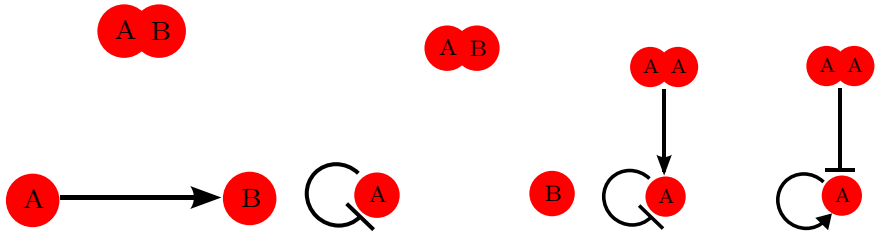
(i) Motifs without catalytic degradations. A few examples are shown in Fig.2 and 3. These networks fall into two subcategories, namely networks with two proteins and one regulation, and networks with one protein and two regulations. Both types feature multimers which help to generate time delays, and in most cases the complex formation is irreversible. Of note, there are two types of time delays due to complexation, namely competitive complexation, where two different ways of complex formation compete for the same protein (e.g. Fig. 3(a)), and chain-like complexation, where complexes of more than two components are formed (e.g. Fig. 3(b)). Oscillating networks with $T = 4$ (see Fig. 2) all belong to this group.

(ii) Motifs with catalytic degradations and regulations, such as the examples in Fig. 4. Again there exist two subcategories, because mechanisms can have

either two proteins and a single heterodimer formed by the two proteins; or only one protein, a dimer and either a trimer or a tetramer.

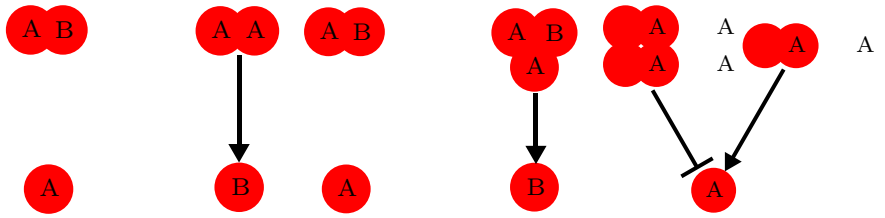
(iii) Motifs without regulations. They can also have one or two genes, and all feature reversible complexation as a method of generating a time delay. Examples are shown in Fig. 5.

(iv) Motifs without multimers. They all have two genes and are similar to certain motifs from (i) where the absence of the AB heterodimer is compensated for by catalytic degradation interactions. Because of the similarity, no examples are shown.



(a) The mixed feedback loop (MFL) (b) The mixed autorepression loop (MAL) (c) and (d) Two dimer-monomer (DMO) oscillators

Fig. 2. All four evolved motifs of size $T = 4$. The MFL and MAL motifs show a negative feedback by a regulation whose action is delayed due to the formation of the AB dimer, which acts as an exclusion mechanism. The MFL motif has been previously studied in [10]. The DMO motifs employ a repressing regulation as a negative feedback, while the time delay is due to TF binding/unbinding dynamics rather than dimerization.



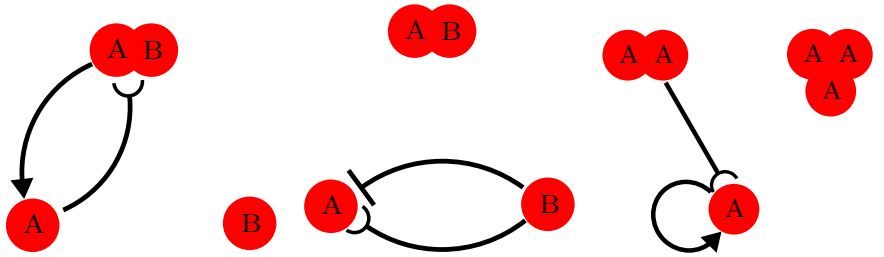
(a) Time delay due to competition (b) Time delay due to high-order complex (c) Single protein oscillator

Fig. 3. Three examples of pattern (i). Most motifs with this pattern are similar the MFL, MAL and MDO motifs presented in Fig. 2. Time delays are larger than in the original motifs due to usage of higher order multimers, such that oscillations are evolved more easily.

4 Algorithm Performance

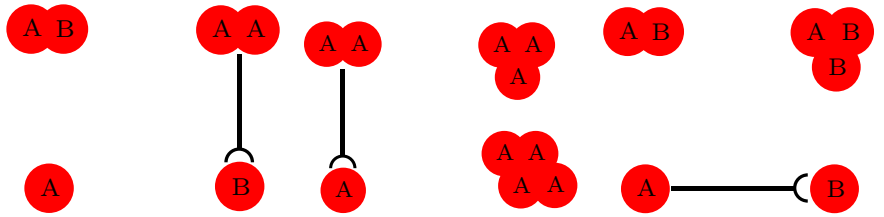
4.1 Efficiency

We have tried to optimize our algorithm towards generating networks at the fastest possible rate. We terminated our evolution after 300 generations, as the



(a) Negative feedback by degradation of activator (b) Negative feedback only due to reversible complexation (c) Delay of self-degradation by reversible trimer formation

Fig. 4. Examples of pattern (ii). Negative feedbacks due to catalytic degradations are often accompanied by reversible formation of a complex (denoted by gray dashed arrows), which results in a time delay.



(a) Time delay by competition and dimer (b) Time delay due to formation of high-order multimers (c) Time delay due to double chain formation

Fig. 5. Examples of pattern (iii). When no regulations are present, the catalytic degradation seems to need two separate time delay mechanisms in order to cause stable oscillations. This is seen as either two levels of reversible complexation as seen in (b) and (c), or as a reversible complexation and a dimer.

evolutionary process had likely become stuck in a local minimum if still no oscillations had been found. This was not a frequent problem, however, and depending on the parameter setting the success rate varied from 75% to 95%. The typical type required to complete one evolutionary process varied (depending on the parameter settings) from a few minutes to half an hour when running the program on three cores of a standard quad-core personal computer. In exceptional cases the computation time can be much longer. In addition, we enforced a maximal number of time integration steps since our RKF45 integration routine was very inefficient in some cases. If this number of steps was exceeded, the network was removed from the population regardless of its fitness performance.

In Fig. 6 we have shown how the topological size T of the network depends on the number of evolutionary steps. In line with our expectations, networks typically end up smaller when evolution finishes early. More generations do not only increase the size of the networks by mutations, but they also integrate existing parts of the network better and better, such that it becomes harder to prune these parts once oscillations have been evolved.

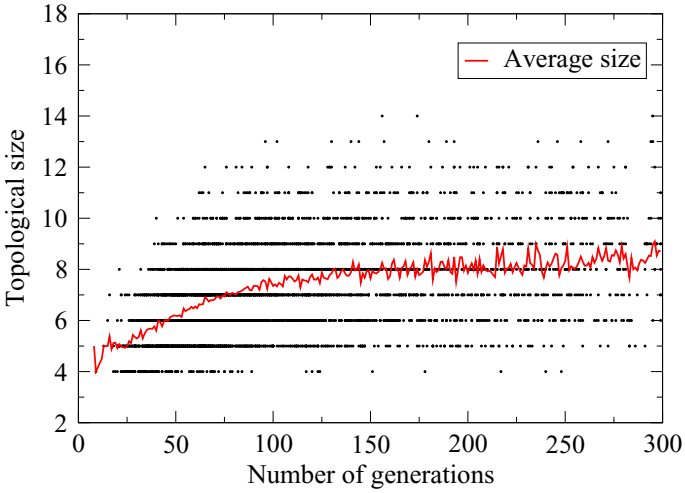


Fig. 6. The typical size of the final, reduced network grows with the number of generations it takes to evolve oscillations. The average was computed for all 39, 457 networks evolved in the large-range setting, and a random subset of 3917 networks are represented by dots. If no oscillations had evolved after 300 generations, the evolutionary process was abandoned.

4.2 Influence of Parameter Choices

As discussed in Sec. 3.1, we used four different parameter settings. The reason to do this was twofold: firstly, to avoid generating only networks in a very limited region of parameter space, and secondly, to allow for a comparison of results, such that we can estimate to what degree the results are a consequence of the parameters of the algorithm.

It turns out that some parameter choices exclude a significant portion of possible motifs. The default choice allows catalytic degradation, but except in rare cases, it failed to find small networks incorporating catalytic degradations with size $T \leq 5$. When the catalytic degradation was removed as an evolutionary option, the rate of finding networks with $T = 5$ increased from 3.3% to 5.2% and several mechanisms featuring repressing regulations became much more common. The setting with fast complex formation yielded several networks with $T = 4$, but the variety of networks was limited to only the MFL and MAL networks shown in Figures 2(a) and 2(b). The most successful setting was the default setting with very loose parameter limits, where the variable rates were allowed to vary across four orders of magnitude instead of just two. Nevertheless, this setting appeared to be inefficient for the evolution of networks with repressing regulations, such that some motifs found in other settings were not evolved, and it seems plausible that a parameter setting can be found which is much more conducive towards evolving motifs featuring repressing regulations. Additionally, this setting consumed roughly ten times more computation time per evolutionary

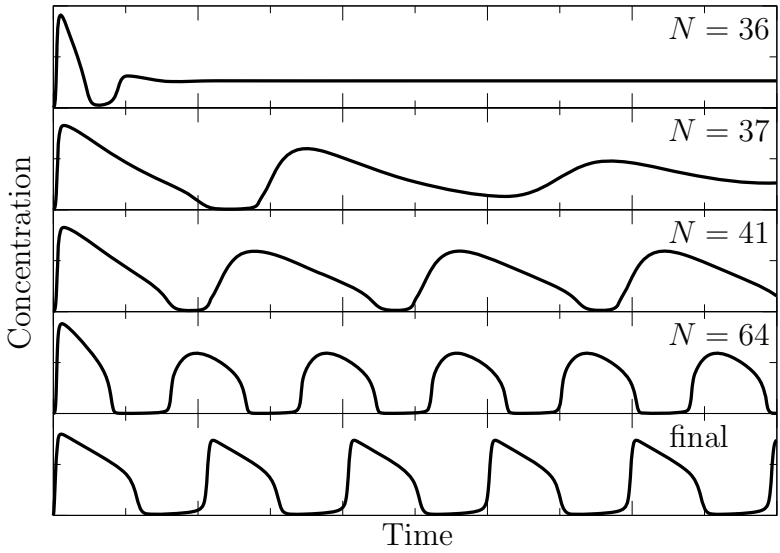


Fig. 7. An example of the evolution of sustained oscillations for a network with a topology as shown in 5(b). The amplitude, period and shape of the oscillations vary while the quality of the oscillation is gradually improving. Each graph shown represents the output for the best-scoring network in the N-th generation, where the values of N were chosen from a single run. The final graph shows the output of the final network after topological reduction.

process than the other settings. Broadly speaking, it seems likely that there is no single setting which is optimal, and a comprehensive set of oscillating motifs can only be generated by varying the parameter settings.

5 Conclusions

In this paper, we have presented an analysis of various examples of oscillatory biological networks generated by an evolutionary algorithm. The algorithm can efficiently produce oscillating networks of small size. We have discussed which choices in the implementation of the evolutionary algorithm were responsible for this improvement, and how our implementation in general influences what motifs are evolved.

The most significant improvement resulted from a fitness function that did not force evolution towards any particular type of oscillation. The freedom that resulted from a loose definition of the concept of oscillations allowed for the evolution of many small networks. Additionally, the algorithm was made more efficient by avoiding the evolution of very large networks.

Many different types of motifs appear to have an oscillatory output for certain rate constants. This is a positive feature of the algorithm since it suggests that it performs a wide search in the complex space of all possible topologies. In view

of the recent experiments on circadian clocks [7], it is also interesting that the algorithm generated several examples of post-transcriptional oscillators.

The abundance of motifs found by our algorithm might prove useful, serving as a set of candidates that might be found as core parts of networks found in systems biology. The small systems we have found are likely simple compared to biological mechanisms. However, as they hint at topological structures that are conducive towards oscillatory behavior, they may help in recognizing the core parts of biological examples of oscillating protein networks.

In this paper we have only evolved networks aimed at oscillatory output. Real biological evolution may have further selected through the available oscillating motifs using criteria as robustness to fluctuations and/or entrainability (i.e. the resetting of the phase of the oscillations). Further research aimed at analyzing the properties of the generated motifs against these types of criteria may suggest which motifs are most likely to be present in real biological networks.

References

1. François, P., Hakim, V.: Design of genetic networks with specified functions by evolution in silico. Proc. Nat. Acad. Sci. USA 101(2), 580–585 (2004)
2. Chu, D.: Evolving genetic regulatory networks for systems biology. In: Proc. IEEE Congress Evolutionary Computation, CEC 2007, pp. 875–882 (2007)
3. Sakamoto, E., Iba, H.: Inferring a system of differential equations for a gene regulatory network by using genetic programming. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 720–726 (2001)
4. Floares, A.: Automatic inferring drug gene regulatory networks with missing information using neural networks and genetic programming. In: IEEE International Joint Conference on Neural Networks, pp. 3078–3085 (2008)
5. Tagkopoulos, I., Liu, Y., Tavazoie, S.: Predictive behavior within microbial genetic networks. Science 320(5881), 1313 (2008)
6. Šter, B., Avbelj, M., Jerala, R., Dobnikar, A.: On the Origin and Features of an Evolved Boolean Model for Subcellular Signal Transduction Systems. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part II. LNCS, vol. 6594, pp. 383–392. Springer, Heidelberg (2011)
7. O'Neill, J.S., van Ooijen, G., Dixon, L.E., Troein, C., Corellou, F., Bouget, F.Y., Reddy, A.B., Millar, A.J.: Circadian rhythms persist without transcription in a eukaryote. Nature 469, 554–558 (2011)
8. Paladugu, S.R., Chickarmane, V., Deckard, A., Frumkin, J.P., McCormack, M., Sauro, H.M.: In silico evolution of functional modules in biochemical networks. IEE Proceedings Systems Biology 153(4), 223–235 (2006)
9. Jin, Y., Meng, Y., Sendhoff, B.: Influence of regulation logic on the easiness of evolving sustained oscillation for gene regulatory networks. In: Proc. IEEE Symp. Artificial Life ALife 2009, pp. 61–68 (2009)
10. François, P., Hakim, V.: Core genetic module: the mixed feedback loop. Phys. Rev. E 72, 031908 (2005)