Automating Genetic Network Inference with Minimal Physical Experimentation Using Coevolution

Josh C. Bongard Hod Lipson

Computational Synthesis Laboratory
Sibley School of Mechanical and Aerospace Engineering
Cornell University, Ithaca, New York 14850
Email: [JB382|HL274]@cornell.edu

Keywords: Bioinformatics, System Identification, Evolutionary Robotics

Abstract. A major challenge in system biology is the automatic inference of gene regulation network topology—an instance of reverse engineering—based on limited local data whose collection is costly and slow. Reverse engineering implies the reconstruction of a hidden system based only on input and output data sets generated by the target system. Here we present a generalized evolutionary algorithm that can reverse engineer a hidden network based solely on input supplied to the network and the output obtained, using a minimal number of tests of the physical system. The algorithm has two stages: the first stage evolves a system hypothesis, and the second stage evolves a new experiment that should be carried out on the target system in order to extract the most information. We present the general algorithm, which we call the *estimation-exploration algorithm*, and demonstrate it both for the inference of gene regulatory networks without the need to perform expensive and disruptive knockout studies, and the inference of morphological properties of a robot without extensive physical testing.

1 Introduction

System biology is concerned with the synthesis of large amounts of biological detail in order to infer the structure of complex structures with many interacting parts. In this way, system biology can be viewed as an example of reverse engineering.

Figure 1 depicts the basic cycle underlying any attempt to reverse engineer a target system: obtain output from the target system using some input, update the system hypothesis, generate some new input, and repeat. The cycle continues until enough data has been obtained from the target system to ensure a sufficiently accurate reconstruction. This paper presents a new approach to the automation of reverse engineering using an evolutionary algorithm that is both independent of the applied problem domain and requires minimal testing of the physical system.

Evolutionary algorithms can be used in two different ways: to evolve a completely new system, or evolve a system that approximates some target system. Examples of the former approach involve the evolution of robot morphology/controller pairs [25] [19] [4] [10] and the use of genetic programming to evolve agent behaviors (eg. [17]). The latter approach involves the use of evolutionary algorithms for reverse engineering: the system hypothesis is evolved based on input/output data sets. Examples of the evolution of reverse engineering include symbolic regression (eg. [17] and [7]), evolution of artificial neural networks that reproduce specific input/output vectors (useful for learning) [28], and electronic [21] [22], metabolic [16] or genetic circuit [11] inference.

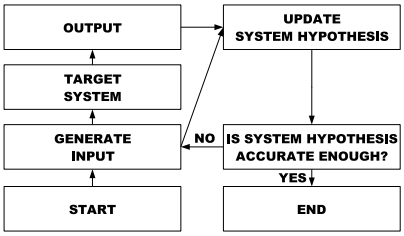

Fig. 1. Flow of a generalized reverse engineering algorithm.

The field of gene network inference is a rapidly burgeoning subfield in system biology [15], and is concerned with inferring genetic regulatory networks based on the results of a set of tests performed on the network in question. Many different models of the underlying genetic network have been used, usually classified based on the amount of biological detail inherent in the model (see [8] and [12] for an overview).

One of the most popular models is the Random Boolean Network (RBN) proposed by Kauffman [13], which is a discrete system in which a network of n genes, each of which is discretely regulated (turned on or off) by k other genes. The RBN's popularity as a model stems from its simplicity and general nature: it contains the minimum of biological detail. More detailed gene network models have been used (eg. [2]), as well as modeling the networks as differential equations [24] [11] or weight matrices [27]. In many of these models, the underlying gene network topology can be or is represented as a graph: nodes indicate genes, and directed labeled edges indicate gene regulation. In this paper we employ a graph-based method for representing gene networks.

In addition to the type of model, several methods have been used to infer genetic networks, including clustering algorithms (see [8] for an overview), correlation metrics [1], linear algebra [6], simulated annealing [23] and genetic algorithms [26] [11].

A number of input and output data pairs are required in order to obtain enough information about the target network to infer its structure correctly. As pointed out in [8], it is desirable to minimize the number of input/output pairs required, so that a minimum of experiments have to be conducted. Also, the *type* of experiment required should be as cheap as possible in terms of experimental difficulty and accuracy of acquired output data. Iba & Mimura [11] showed that by using a multi-population evolutionary algorithm to not only infer the hidden network, but also to propose additional experiments that would most help to refine the current best evolved network hypothesis. However their model requires the experimenter to perform costly knockout or lesion experiments in order to supply the algorithm with an actual subset of the regulatory network.

Knockout studies in genetics (eg. [18]), lesion studies in neuroscience (eg. [29]) and ablation studies in embryology (eg. [9]) are a related set of time-honored tools in those fields, but they have three major drawbacks: such experiments are often difficult to perform, they are destructive to the object of study, and often provide misleading data about the relationships between parts of the system under study. For example, in higher model organisms such as mice, one or more generations must be raised in order to accurately measure the phenotypic effect of a disabled gene.

By carefully selecting the input to be processed by a target network, output data rich enough in information to infer topology can be obtained such that more costly knockout studies are not required. Here we present a coevolutionary algorithm that not

only evolves the hidden network, but also evolves these desirable input data sets. The next section describes the algorithm and the model of genetic networks we use; section 3 presents the results of our algorithm; section 4 provides discussion regarding the power and generality of this algorithm; and the final section offers some concluding remarks.

2 Methods

This paper presents an algorithm for reverse engineering networks. The networks can be interpreted as either biological or electronic networks, or can be interpreted as a representative of any coupled, non-linear system that takes inputs and produces outputs.

We first describe a graph-based model for representing genetic networks, and then describe the application of the estimation-exploration algorithm for inferring hidden instances of such networks based on sets of input and output gene product concentrations.

2.1 The Network

Many models of genetic regulatory networks employ a graph-based representation, in which nodes represent genes, and directed labeled edges from gene i to gene j indicate that gene i somehow influences the expression of gene j. For our purposes we have chosen to represent regulatory networks of n genes using an $n \times n$ matrix $\mathbf{R}$ with entries r_{ij}. If $r_{ij} > 0$, gene i contributes to the enhancement of gene j; if $r_{ij} < 0$, gene i contributes to the inhibition of gene j; if $r_{ij} = 0$, gene i does not directly regulate gene j. Now let an input data vector of gene product concentrations be represented by $\mathbf{g^{(t+1)}} = \{g_1^{(t)}, g_2^{(t)}, ...g_n^{(t)}\}$, in which $g_i^{(t)}$ indicates the gene product concentration at the beginning of some experiment. We can then calculate new gene product concentrations after some time period has elapsed using

$$g_j^{(t+1)} = \min(\ \max(\ 0\ ,\ g_j^{(t)} + \sum_{i=1}^{n} r_{ij} g_i^{(t)}\ )\ ,\ 1\ ) \tag{1}$$

Since the g_j variables indicate concentration, the *min* and *max* functions bound the value between 0 (for no concentration) and 1 (for concentration saturation). The vector $\mathbf{g}^{(t+1)} = \{g_1^{(t+1)}, g_2^{(t+1)}, ...g_n^{(t+1)}\} = [(\mathbf{R} + \mathbf{I})\mathbf{g}^{(t)}]_0^1$ then represents the bounded output data vector obtained from the hidden regulatory network $\mathbf{R}$, given input $\mathbf{g}^{(t)}$. All values of the input vector $\mathbf{g}^{(t)}$ and output vector $\mathbf{g}^{(t+1)}$ lie in the range $[0, 1]$. The values of $\mathbf{R}$ lie in the range $[-1, 1]$.

This discrete map approximates the differential equation model of regulation:

$$\frac{dg_i}{dt} = f_i(\mathbf{g}), \qquad 1 \leq i \leq n \tag{2}$$

in which the product concentration of gene i changes as a function of the product concentrations of the other genes (possibly including gene i). In our formulation, f_i is the thresholded multiplication of row i in $\mathbf{R}$ by the column of initial concentrations $\mathbf{g}^{(t)}$.

2.2 The Estimation-Exploration Algorithm

In this paper we present a general algorithm that allows for the reverse engineering of a hidden network based solely on input/output data: in this case, the hidden network is the connection matrix $\mathbf{R}$. Our algorithm has two stages: the estimation and

exploration phase, each of which has an associated genetic algorithm. The first stage evolves a plausible connection matrix based on input/output data sets (estimation), and the second stage evolves a new input vector that should produce an output vector rich in information when processed by the actual target connection matrix (exploration). In this way the algorithm performs two functions: it reverse engineers the hidden network, and it proposes *useful* experiments that will accelerate the inference process.

In the application of the estimation-exploration algorithm to genetic networks, first a random input vector is selected, and the corresponding output vector is computed using the actual target connection matrix. Then the resulting single input/output vector pair is passed into the algorithm.

The estimation genetic algorithm begins with a population of genomes, each of which is comprised of an $n \times n$ connection matrix $\mathbf{R}'$: the values are generated randomly over $[-1, 1]$ using a uniform distribution. Each genome is evaluated as follows. Each of the input vectors applied to the actual target network so far (which during the first iteration of the estimation phase is only one vector) is used to calculate a corresponding output vector based on the genome's connection matrix $\mathbf{R}'$. The *subjective error* associated with the genome is set to

$$\text{error}_{\text{subj}}(\mathbf{R}') = \frac{\sum_{k=1}^{x} \sum_{i=1}^{n} |g_{ik}^{(\text{tar})(t+1)} - g_{ik}^{(\text{guess})(t+1)}|}{xn} \tag{3}$$

where x is the total number of experiments performed on the target network so far, $g_{ik}^{(\text{tar})(t+1)}$ is the resulting concentration of gene i when experiment k is performed on the target network, and $g_{ik}^{(\text{guess})(t+1)}$ is the resulting concentration of gene i when experiment k is performed on $\mathbf{R}'$.

Subjective error is then an indirect measure of how well $\mathbf{R}'$ approximates the hidden network $\mathbf{R}$, based how well $\mathbf{R}'$ can reproduce the experimental results produced by $\mathbf{R}$. The *absolute error* of the network hypothesis $\mathbf{R}'$ is then defined as

$$\text{error}_{\text{abs}}(\mathbf{R}') = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |r_{ij} - r'_{ij}|}{n^2}, \tag{4}$$

which is a direct indication of how good the approximation is. Note that this error is not available to the algorithm, but can be used to measure the efficacy of the algorithm itself.

Once all of the genomes have been evaluated, pairs of genomes are selected, and the connection matrix of the genome with higher subjective error is replaced by the connection matrix of the genome with the lower subjective error. The copied matrix is then mutated: one randomly selected element of the matrix is replaced either by a new random value in $[-1, 1]$ (50% probability), or nudged up or down by 0.0001 (50% probability). Crossover is currently not used, but may be implemented in future improvements to the algorithm. For the work reported here, a population size of 1000 is used, and a total of 750 replacements are performed after each generation. Note that a given connection matrix may undergo more than one mutation if it is selected, copied, mutated and then selected again. Once a set of selections, replacements and mutations have occurred, all of the new genomes in the population are evaluated. This process is continued for 30 generations. When the estimation phase terminates at the end of the 30 generations, the $\mathbf{R}'$ with the least subjective error is passed on to the exploration phase.

The exploration phase also begins with a set of randomly-generated genomes, but the EA for this phase maintains genomes that encode input vectors instead of connection matrices. The vectors are initialized with random floating-point values in $[0, 1]$ chosen using a uniform distribution. Each genome is then evaluated as follows. The encoded input vector is applied to the connection matrix $\mathbf{R}'$ obtained from the estimation phase, and an output vector is obtained. The *information* associated with the genome is set to

$$i = 1.0 - \frac{n_0 + n_s}{n} \tag{5}$$

where n_0 is the number of genes in the output vector that have zero concentration, and n_s is the number of genes that have a saturation concentration of 1. The same method of selection and replacement is then applied as described for the estimation phase: the output vectors of genomes with higher information replace the output vectors of genomes with lower information. The new genomes (input vectors) are mutated as follows: a single gene is chosen at random, and replaced with a new concentration chosen from $[0, 1]$ with a uniform distribution. This method of selecting genomes based on their expected information content is an attempt to try to evolve input vectors that, when supplied to the target network, will produce output vectors that contain high information: gene concentrations in the output vector that have either zero or saturation concentration levels indicate less about the state of their regulating genes' concentrations than concentrations between these extrema.

The exploration phase is also executed for 30 generations. When it terminates, the best input vector is supplied to the target network $\mathbf{R}$. This input vector is used by $\mathbf{R}$ to compute a new output vector, which is then passed back into the next iteration of the estimation phase, along with the $j - 1$ previously-generated input/output vector pairs. When the estimation phase is run again, the initial random population is seeded with the best connection matrix found so far in order to accelerate evolution. The entire process—calculation of the output vector from $\mathbf{R}$, execution of the estimation and then the exploration phase, and experiment suggestion—iterates for 100 cycles, leading to 100 experiments performed on the target network.

This same algorithm can be applied for the non-destructive inference of other physical systems. For example we have applied it in the domain of evolutionary robotics. In that case the algorithm was contained within a robot simulator: the algorithm evolved controllers for an actual robot (exploration phase), and also evolved hypotheses regarding possible damage suffered by the actual robot (estimation phase). Based on the sensory feedback from the actual robot, the simulator could both refine its hypothesis regarding what damage had occurred, and modify the controller so that the actual robot could regain as much functionality as possible. Figure 2 compares these algorithms, as well as providing a general framework for how this co-evolutionary algorithm can be applied such that a bidirectional dialogue is maintained between the physical nonlinear system (such as a robot or biological network) and the algorithm.

The next section presents some results generated using this algorithm.

3 Results

In order to test the algorithm, a set of 40 target networks were generated for various numbers of genes (n) and number of incoming regulatory connections (k). The first 10

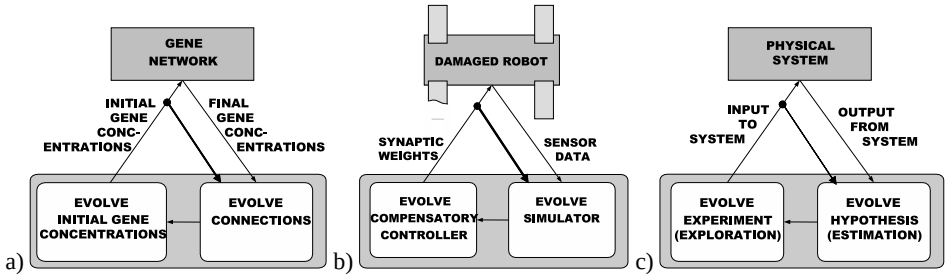

Fig. 2. Instantiations of the estimation-exploration algorithm. a: In order to reverse engineer a genetic network, the estimation phase evolves a connection matrix that describes the wiring of the network. Based on the best evolved connection matrix hypothesis, an input vector is evolved by the exploration phase that should return an information-rich output vector when applied to the hidden network. **b:** A neural network controller is evolved for an actual robot (exploration phase) that allows it to move. If it suffers damage, the estimation phase evolves a robot simulator that explains the damage. **c:** The general framework for applying the algorithm to a nonlinear system.

networks contained 5 genes ($n = 5$) and each gene was regulated by 2 genes ($k = 2$). The second 10 networks were generated using $n = 5$ and $k = 5$; the third 10 runs using $n = 10$ and $k = 2$; and the final 10 runs using $n = 10$ and $k = 10$.

The algorithm was applied to each of the 40 networks, and the cycle described in the above section was iterated 100 times; thus 100 experiments were performed on the hidden target network. Each pass through either the estimation or exploration phase involved the evolution of a population of 1000 genomes for 30 generations.

Two control algorithms were also applied to the same 40 networks as the proposed algorithm. The first control algorithm was random search: the algorithm performed as before, but instead of replacing genomes of low fitness with genomes of high fitness (for both phases), the genome with low fitness was replaced with a randomly generated genome. In the second control algorithm, the exploration phase is disabled: this phase simply returns a randomly generated input vector. Both control algorithms were executed for the same number of iterations (100), the same population size (1000), and the same number of generations (30) as the proposed algorithm.

Ten independent runs of the algorithm were conducted for each of the 40 hidden networks, and 10 independent runs of both control algorithms were also conducted for each network, leading to a total of $3 \times 10 \times 40 = 1200$ independent runs. Figure 3 shows the evolutionary progress of a typical run of both the proposed algorithm and the second control algorithm (in which the exploration phase is disabled) on a hidden network with $n = 10$ and $k = 10$. Figure 4 reports the resulting output vectors from the 100 experiments suggested by each of these two runs. Figure 5 shows the average performance for the proposed algorithm, compared to the two control algorithms, for the four different types of hidden target networks.

4 Discussion

As can been seen from Figure 3, the downward-sloping thick curves indicate that during each pass through the estimation phase, the subjective error of the best genome in the population decreases. However after a new experiment has been performed on the hidden target system, the subjective error of the best hypothesis so far (which is

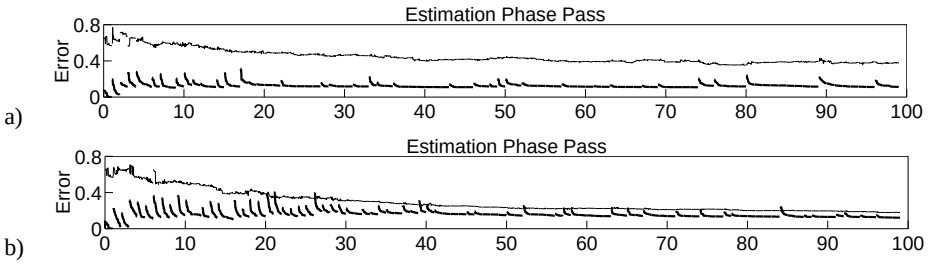

Fig. 3. Sample evolutionary progress for the second control and proposed algorithm. The thin line indicates the absolute error (see equation 4) between the actual connection matrix $\mathbf{R}$ and the best connection matrix evolved during that generation of the estimation phase, $\mathbf{R}'$. Note that this information is not available to the algorithm. The thick line indicates the subjective error of the best genome in the population at that generation according to equation 3. **a**: A sample run of the control algorithm with the exploration phase disabled on one of the hidden networks with $n = 10$ and $k = 10$. **b**: The progress of the proposed algorithm on the same hidden network. The evolutionary progress of the passes through the exploration phase for **b** are not shown.

used to seed the first generation of the next pass through the estimation phase) tends to increase: this is indicated by the successive curves seen during the first 500 generations of the second control algorithm (Figure 3**a**) and the first 1000 generations of the proposed algorithm (Figure 3**b**). This indicates that the new experiment exposed some previously hidden information about the target system that the best hypothesis so far did not account for.

Importantly, the proposed algorithm tends to generate such information-rich experiments far beyond the 40th experiment, as compared to the control algorithm, in which the experiments lose their explanatory value (the curved subjective error trajectories become flat) before the 20th experiment. Although both algorithms eventually evolve a hypothesis that explains most of the input/output data pairs (indicated by the equally low subjective error curves at the end of the two runs), the proposed algorithm explains more *informative* input/output data pairs, thus leading to a better approximation of the hidden network (the absolute error (thin line) eventually becomes lower for the proposed algorithm, compared to the control algorithm.)

Figure 4 shows why the proposed algorithm is able to outperform the control algorithm. Between the 20th and 40th experiments, the proposed algorithm has enough informative input/output data pairs to evolve a good approximation of the hidden network. Using this approximation, it can evolve input vectors that produce a greater fraction of informative gene product concentrations: concentrations that fall between the two extremal concentrations of 0 and 1. This is indicated by the greater density of such concentrations (the black squares) in the output vectors seen in Figure 4**b**, compared to those in the output vectors obtained by the control experiment (Figure 4**a**). However given a random network, it is more difficult to obtain intermediate concentrations for some genes compared to others: both the proposed and control algorithm obtained only sporadic intermediate concentrations for gene 9 for this particular target network. Further improvements to the algorithm will entail changes in equation 5 in order to maximize intermediate concentrations for all genes.

Figure 5**d** indicates that the proposed algorithm consistently outperforms both the first control algorithm (random search) and the second control algorithm, in which ex-

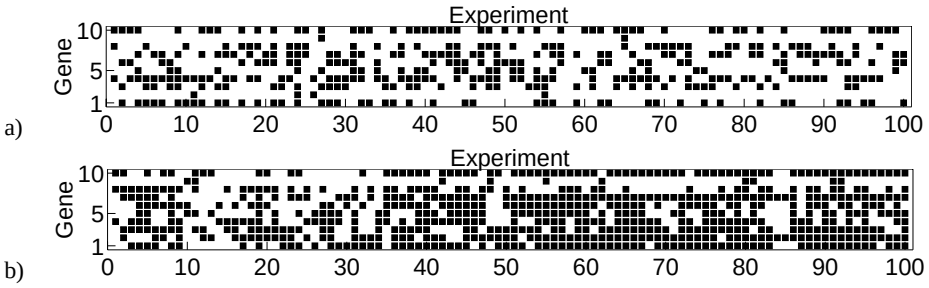

Fig. 4. Sample experimental results derived from the second control and proposed algorithm. Each column represents an output vector obtained when an input vector suggested by either the second control algorithm (**a**) or the proposed algorithm (**b**) was applied to the hidden target network. Blank squares indicate gene product concentrations that were either 0 or 1; dark squares indicate concentrations that fell between these extrema.

periments are suggested randomly. This indicates that evolving informative experiments in step with model hypotheses does improve the discovery of the hidden networks. Not surprisingly, both algorithms consistently outperform random search, for all four network types. However, interestingly, evolving informative tests is only beneficial for networks that are comprised of many genes. On further inspection this is not so surprising, because networks with many genes and high connectivity have a higher likelihood of reaching either of the extremal concentrations than smaller, less dense networks.

Figure 6 supports this claim: 1000 random networks were generated using values of n selected from $[2, 30]$ with a uniform distribution, and values of k selected from $[1, n' - 1]$, where n' is an already randomly selected value for n. For each of the 1000 random networks, 10 input vectors were randomly constructed, and the corresponding 10 output vectors were calculated using 1. The average fraction of output concentrations that were either 0 or 1 were computed for each network, and are plotted in Figure6**a** as a function of the number of genes in the network (n), and in Figure6**b** as a function of connectivity (k/n). Clearly, the fraction of non-informative output elements increases both with the number of genes, and with connectivity. Thus it becomes increasingly valuable not only to evolve hypotheses, but also to evolve informative experiments for the inference of larger and more complex gene networks.

Most importantly though, unlike the algorithm proposed by Iba [11], the experiments performed here on the target system do not require any internal or disruptive perturbation such as a knockout study: we simply supply a carefully evolved new set of initial conditions. Introducing gene product concentrations into a cell or subjecting the cell to external chemicals, rather than invasive knockout studies, may be a more attractive option for certain model organisms. However more study is required to determine how our input/output concentration data may be translated into actual biological experiments. A second advantage of our algorithm over Iba's algorithm is that it does not require direct comparison between proposed regulatory networks: both hypothesis and experiment quality is determined based on experimental output data, rather than on the internal topology of a given network hypothesis.

Human experimenters usually prefer to modify some initial set of conditions only slightly, and then measure the effect on the system in order to infer something about the internal structure of the system. However since the experiment is proposed automati-

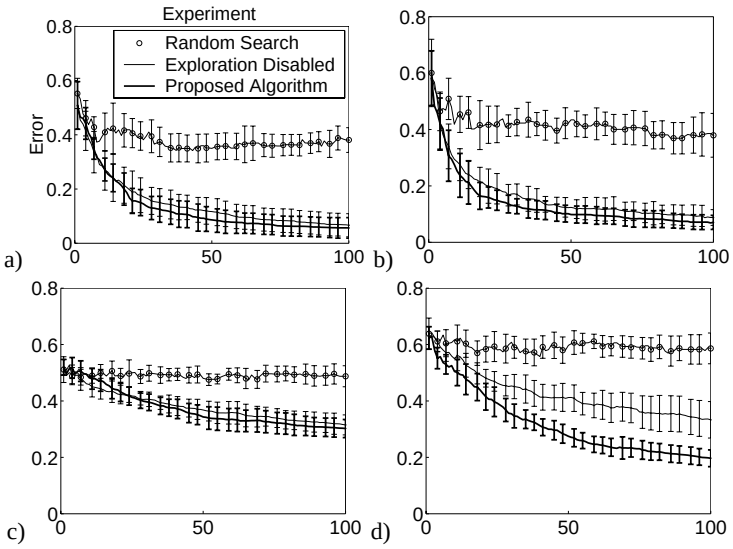

Fig. 5. Average performance of the three algorithms for the four different target network types. Comparative performance of the three algorithms, each averaged over 10 different hidden networks with $n = 5$ and $k = 2$ (**a**). Comparative performance for 10 different networks with $n = 5$ and $k = 5$ (**b**); for 10 networks with $n = 10$ and $k = 2$ (**c**); and for 10 networks with $n = 10$ and $k = 10$ (**d**).

cally by the exploration phase, and the results of the experiment are analyzed automatically by the estimation phase, the algorithm is free to suggest input vectors that are very different from those tested on the network before. It is believed that this will greatly speed the inference of the actual system, but remains to be tested rigorously.

4.1 Multiple and Diverse Applications

This mutual dialogue was found to hold in the application of the estimation-exploration algorithm to a completely different problem domain: evolutionary robotics. Instead of evolving networks (estimation) and experiments (exploration) as the algorithm does here, the evolutionary robotics application evolved neural network-based controllers for an actual robot (the exploration phase). When the actual robot suffered some unknown damage and returned sensory data, the estimation phase attempts to evolve a robot simulator that describes the damage. Using this hypothesis the exploration phase then tries to re-evolve a compensatory controller that allows the robot to regain functionality despite its handicap. Figure 7 shows a sample run. For more details, refer to [3].

Just like the gene network inference application shown here, data returned by the physical system (the robot) enhances the estimation phase's ability to predict the state of the system. The model of the system output by the estimation phase (for the robot application, a hypothesis about the damage suffered) aids the exploration phase by increasing its ability to evolve a controller for the damaged robot.

Also, the application of the estimation-exploration algorithm to robotics allows for recovery after minimal physical testing. Previously, evolutionary algorithms have been

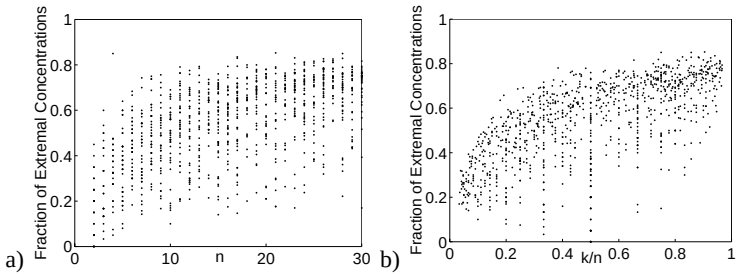

Fig. 6. Observability of various gene network types. 1000 gene networks with different values of n and k were generated randomly. For each network, 100 random input vectors were supplied, and the average fraction of resulting extremal output concentrations was calculated. **a:** The relationship between the number of genes (n) and the fraction of extremal values. **b:** The relationship between connection density (n/k) and the fraction of extremal values for the same 1000 networks.

used to repair (but not diagnose) damaged physical systems, but they rely on extensive hardware testing (on the order of thousands of evaluations), which is prohibitive for most physical systems [5] [20].

5 Conclusions

In this paper we have presented a co-evolutionary algorithm composed of two phases that automates the process of reverse engineering: the estimation phase evolves a model of the hidden system under study (in this case a genetic network), and the exploration phase evolves an experiment to be performed on the hidden system to gain more information about it. Moreover, as the estimation phase refines its description of the physical system, the exploration phase is able to propose more valuable experiments because its internal model of the physical system is more accurate. In this way the algorithm serves two functions: it infers the internal structure of some hidden system, and proposes increasingly useful experiments to be performed on it. Indeed the idea of automating scientific inquiry has received a lot of interest as of late [14].

Our model has three benefits for gene network inference. First, it does not require invasive, expensive, slow and disruptive experiments such as knockout or lesion studies. Rather, the exploration phase carefully evolves a low-cost experiment (a change in the initial gene product concentrations) that yields a large amount of information about the physical system. Second, the number of experiments performed on the hidden system is minimized, because each proposed experiment is carefully chosen. Finally, the careful selection of experiments becomes increasingly valuable as the hidden networks become larger and more densely interconnected, because large, dense networks often produce information-poor output data. These three points suggest that our algorithm may prove very useful for genetic network inference in particular, and for system biology research in general.

We have also shown how general this approach is by describing and showing how it can be applied to a completely different nonlinear physical system: a robot. Future prospects for this work will include improving the selection of experiments so that even more information is extracted from the system, generalizing the model still further by applying it to various other nonlinear physical systems, and formulating a rigorous set of guidelines for how to apply the algorithm to a large class of physical systems.

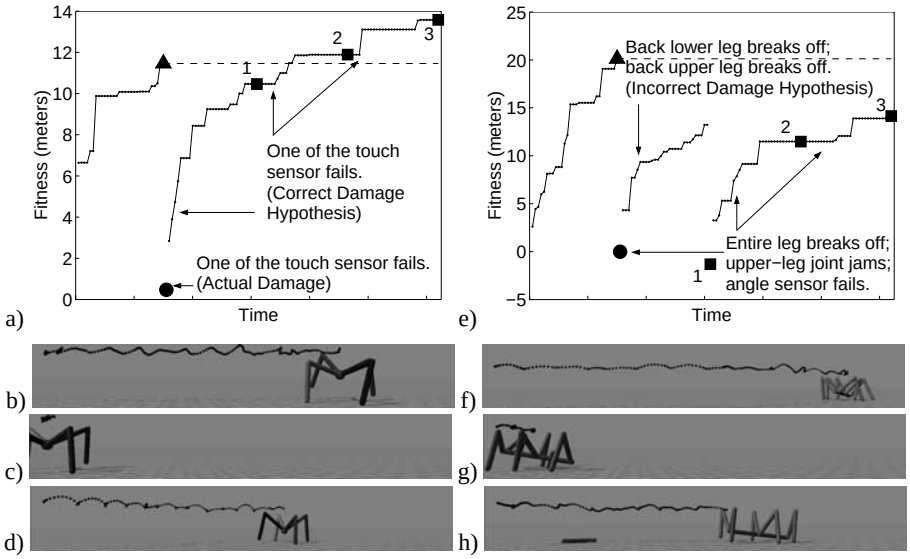

Fig. 7. Two typical robot damage recoveries. a: The evolutionary progress of four passes through the exploratory phase for a quadrupedal robot when it undergoes a failure of one of its touch sensors. The hypotheses generated by the three passes through the estimation phase (all of which are correct) are included. The small circles indicate the fitness (maximum forward locomotion) of the best controller after each generation of the estimation phase. The triangle shows the fitness of the first evolved controller on the physical robot (the behavior of the 'physical' robot (which is also simulated for now) with this controller is shown in **b**); the large circle shows the fitness of the robot after the failure occurs (the behavior is shown in **c**); the squares indicate the fitness of the physical robot for each of the three hardware trials (the behavior of the 'physical' robot during the third trial is shown in **d**). **e-h** The recovery of a hexapedal robot when it experiences severe, compound damage. The trajectories in **b-d** and **f-h** show the change in the robot's center of mass over time (the trajectories are displaced upwards for the sake of clarity).

Acknowledgment

This work was supported by the National Academies Keck Futures Grant for interdisciplinary research, number NAKFI/SIG07.

References

1. Arkin, A., Shen, P., Ross, J.: A test case for correlation metric construction of a reaction pathway from measurements. In: *Science* **277**: 1275–1279 (1997)
2. Arkin, A., Ross, J., McAdams, H.H.: Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells. In: *Genetics* **149**: 1633–1648 (1998)
3. Bongard, J.C., Lipson, H.: Automated damage diagnosis and recovery for remote robotics. To appear in: *Proceedings of the 2004 International Conference on Robotics and Automation (ICRA)*, New Orleans, USA (2004)
4. Bongard, J.C., Pfeifer, R.: Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. In: Spector, L., Goodman, E.D. (eds.): *Proceedings of The Genetic and Evolutionary Computation Conference*: 829–836 (2001)

5. Bradley, D.W. and Tyrrell, A.M.: Immunotronics: novel finite-state-machine architectures with built-in self-test Using self-nonself differentiation. In *IEEE Transactions on Evolutionary Computation*, 6(3): 227–38 (2002)
6. Chen, T., He, H.L., Church, G.M.: Modeling gene expression with differential equations. In: *Pacific Symposium on Biocomputing* **4**: 29–40 (1999)
7. Davidson, J.W., Savic, D.A., Walters, G.A.: Symbolic and numerical regression: Experiments and applications. In: *Information Sciences* **150**(1-2): 95–117 (2003)
8. D'haeseleer, P., Liang, S., Somogyi, R.: Genetic network inference: From co-expression clustering to reverse engineering. In: *Bioinformatics* **16**(8): 707–726 (2000)
9. Hill, R.J., Sternberg, P.W.: Cell fate patterning during *C. elegans* vulval development. In: *Development* **suppl.** 9–18 (1993)
10. Hornby, G.S., Pollack, J.B.: Creating high-level components with a generative representation for body-brain evolution. In: *Artificial Life* **8**(3): 223–246 (2002)
11. Iba, H., Mimura, A.: Inference of a gene regulatory network by means of interactive evolutionary computing. In: Information Sciences **145**: 225–236 (2002)
12. de Jong, H.: Modeling and simulation of genetic regulatory systems: A literature review. In: *J. Comput. Biol.* **9**(1): 69–105 (2002)
13. Kauffman, S.A.: *The Origins of Order*, Oxford University Press. Oxford, UK. (1993)
14. King, R. D., Whelan, K. E., Jones, F. M., Reiser, P. G. K., Bryant, C. H., Muggleton, S. H., Kell, D. B., Oliver, S. G.: Functional genomic hypothesis generation and experimentation by a robot scientist. In: *Nature* **427**: 247–252 (2004)
15. Kitano, H.: *Foundations of Systems Biology*, MIT Press. Cambridge, MA. (2001)
16. Koza, J.R., Mydlowec, W., Lanza, G., Yu, J., Keane, M.A.: Reverse engineering of metabolic pathways from observed data using genetic programming. In: Altman, R. B. et al (eds.): *Pacific Symposium on Biocomputing*: 434–445 (2001)
17. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Natural Selection*, MIT Press. Cambridge, MA. (1992)
18. Lewis, E.B.: Clusters of master control genes regulate the development of higher organisms. In: *Journal of the American Medical Association* **267**: 1524–1531 (1992)
19. Lipson, H. and Pollack, J.B.: Automatic design and manufacture of artificial lifeforms. In *Nature*, 406: 974–978 (2000)
20. Mahdavi, S.H. and Bentley, P.J.: An evolutionary approach to damage recovery of robot motion with muscles. In *Seventh European Conference on Artificial Life (ECAL03)*: 248-255 (2003)
21. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the evolutionary design of digital circuits–Part I. In: *Journal of Genetic Programming and Evolvable Machines* **1**(1): 8–35 (2000)
22. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the evolutionary design of digital circuits–Part II. In: *Journal of Genetic Programming and Evolvable Machines* **3**(2): 259–288 (2000)
23. Mjolsness, E., Sharp, D.H., Reinitz, J.: A connectionist model of development. In: *J. Theor. Biol.* **152**: 429–454 (1991)
24. Sakamoto, E., Iba, H.: Identifying gene regulatory network as differential equation by genetic programming. In: *Genome Informatics*: 281–283 (2000)
25. Sims, K.: Evolving 3D morphology and behaviour by competition. In: *Artificial Life IV*: 28–39 (1994)
26. Tominaga, D., Okamoto, M., Kami, Y., Watanabe, S., Eguchi, Y.: Nonlinear numerical optimization technique based on a genetic algorithm. http://www.bioinfo.de/isb/gcb99 /talks/tominaga
27. Weaver, D. C.: Modeling regulatory networks with weight matrices. In: *Proc. Pacific Symp. Bioinformatics* **5**: 251–258 (2000)
28. Yao, X.: Evolving artificial neural networks. In: *Proceedings of the IEEE* **87**(9): 1423–1447 (1999)
29. Young, R.M.: *Mind, Brain and Adaptation in the Nineteenth Century. Cerebral Localization and its Biological Context from Gall to Ferrier*, Clarendon Press. Oxford, UK. (1970)