2009

Modeling, simulation, synthesis, and optimization of biochemical networks

Kent Allan Vander Velden
Iowa State University

Modeling, simulation, synthesis, and optimization of biochemical networks

by

Kent Allan Vander Velden

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Bioinformatics and Computational Biology

Program of Study Committee:
Peter J. Reilly, Co-Major Professor
Vasant Honavar, Co-Major Professor
Drena L. Dobbs
Irvin R. Hentzel
Robert L. Jernigan

Iowa State University
Ames, Iowa
2009

Table of contents

Chapter 1. General introduction

Introduction

The design and analysis of biomolecular networks is an ambitious goal of synthetic biology and genetic engineering[1]. Both of which may be able to advance by utilizing formal methods, such as modeling, developed and relied upon in more mature branches of engineering. Modeling of cellular processes is the application of mathematics to molecular biology that ideally provides a computational system that accurately describes the phenotype of the modeled system in response to changing environments, stimuli, and perturbations. Modeling allows initial experiments to be performed *in silico*[2] through simulation[3-7], and can be used to summarize knowledge or discover and fill in knowledge gaps[8]. Modeling will have a more significant role, verifying expectations and identifying potentially undesirable conditions[9], as more complicated constructs are considered.

Until recently, models presented in the literature relied on custom-crafted equations to approximate the actual process underlying an observed behavior of the modeled system. Regardless of the accuracies of the approximations, this approach creates models that are hard to validate or extend, stemming from the difficulty that researchers not trained in mathematics have understanding and applying the model[10]. What has helped modeling to be accepted as a routine approach in physical sciences and engineering is the identification of “building blocks” that have consistent properties regardless of their application. Progress has been made in this direction with the introduction and improvement of biochemical modeling environments. However, if an analogue to building blocks exists in molecular biology, in which some research suggests at least topologic building blocks exist[11], then the application of modeling to genetic engineering as a design tool could be made more powerful.

Background and significance

Synthetic biology: Molecular biology currently lacks the structure typical of engineering; however, several research groups are making progress to develop synthetic biology[4,12-14]. Theoretical approaches suggest that motifs are a common part of all networks[15,16], and with experimental evidence suggesting their presence in biological networks[11], prototype systems are being constructed that demonstrate the analogues of building blocks in genetic networks. Examples of engineered regulatory networks to date include various bi-stable switches[17-20], oscillatory systems mimicking circadian clocks[21], and Boolean functions[22,23], as well as more systematic and exhaustive efforts[24]. Lessons learned from these prototypes are helping to establish design principles[9] to enable construction of more elaborate systems. The most ambitious groups are already attempting to engineer viruses and a complete minimal organism[25,26].

Not all attention has been placed on engineered networks. Other groups have focused on the interface mechanisms for communication between cells and engineered and indigenous networks[27-29]. Further driving the field is the need to understand and manipulate the regulatory control of metabolic networks[30]. Resources that are advancing the field include gene interaction maps[31], protein–protein interaction maps[32,33], and large scale protein interactions[34-36] surveys across multiple species.

The difficulties of genetic engineering relate not only to the design of networks but also include the effects of the construct on the viability of the organism. This requires knowledge of the interactions between other networks in the organism and is presently outside the scope of current knowledge for all but the most studied organisms. For now, genetic engineers must remain aware of the possible consequences of incomplete knowledge of the target organism. Others have taken a network theoretical approach, exploring through simulation the required properties of a network to exhibit a particular phenotype[20,37-39]. For the purpose of this research the networks considered will be restricted to those with associated experimental data either based on laboratory experiments or generated through simulation.

Synthetic biology – The experiment of Guet: Published network diagrams make the modeling of regulatory networks look deceptively simple. In practice, most regulation mechanisms are simply not understood well enough to accurately model an arbitrary network, a consequence of having limited observations of complex interactions between arbitrary elements in an incompletely understood network. Guet sought to better understand regulation by removing several of these unknowns through constructing a library of artificial gene networks[24] containing all possible topologies between three of the best-understood transcription factors. This exhaustive genetic engineering approach provides several new observations, providing data for new insight into regulation and demonstrating the diversity of phenotypes possible with just a small number of regulatory elements. Beyond being an example of successful genetic engineering, the results of this set of experiments provide the data that form the basis of a validation set for this work.

Modeling formalisms: A modeling formalism is the language in which a network model is described. It consists of two main components: the presentation and the mechanics[12,40]. The formalism's presentation may consist of raw equations, textual descriptions, or graphical descriptions. The choice of presentation affects the ease in which a model can be described and the chance of ambiguities existing in the model. While presentation is limited by the underlying mechanics, there is no reason that a particular presentation should be any more constraining than the mechanics.

The formalism's mechanics concern how a model is actually simulated, such as Boolean, linear, non-linear, or agent-based[41] methods. The choice of mechanics can limit the explanatory power of the formalism. For instance, linear methods are easier to analyze and optimize, but they are limited in their abilities to capture certain dynamics or they violate physical laws such as mass conservation. Likewise, other methods such as stochastic ones may be more realistic[42-44], but they are prohibitively slow to use in general. However, it must be considered if such dynamics are required for the particular model. A balance must be formed between generality, performance, and mathematical convenience.

Modeling environments: The modeling environment is an implementation of a particular formalism that allows the researcher to capture the essence of how a network is believed to function. Through simulation and comparison to observations, it is possible to verify if the model is capturing the observed dynamics.

Modeling packages may be generic or specialized for modeling biochemical networks. Examples of generic packages that have been used include MATLAB, Mathematica, Mobius, and Excel[45]. Since a generic package is not directly designed to accommodate the simulation of network models, either an awkward interface must be used to describe the network or considerable custom code must be written, both of which can be error-prone.

Specialized packages have been released with greater frequency in the past few years to accommodate the expected needs of systems biology[46,47]. One of the original packages, Gepasi[48,49], is still one of the most popular. Others include E-Cell[50] for simulation of entire cells, Biospice, and Gene Network Analyzer, and COPASI, which is inspired by Gepasi. Perhaps the most elaborate system is the Systems Biology Workbench (SBW), which consists of a number of tools that operate together, including Jarnac as the computation engine and JDesigner as the graphical network design tool. Most specialized packages are standardizing on a common file exchange format called SBML[51], which helps to leverage the complementary features of each package. Although some packages have a rudimentary model-fitting mode, they are not suitable, in general, for complex networks[52].

Through improvement of the theoretical properties of regulatory networks and sophistication of the modeling environments, future tools may become the molecular biology analogue to the computer-aided design software packages currently used by engineers.

Model fitting: The process of model fitting is the assignment of kinetic parameters to the model so that the model is correctly able to predict the response to stimuli of the modeled system. If an evaluation function is available to measure the fit of the model to experimental observations, the process can be viewed as an optimization process. Although much research has been applied to optimization of metabolic networks[52-54], model fitting of regulatory networks is less refined[45,55,56]. One reason for this dichotomy is because metabolic networks are composed of largely static chemical reactions, while regulatory networks are more transitive by their nature. A problem that is constantly faced when analyzing data from regulatory networks is that the most easily available data, such as expression chips, can be misleading[57,58].

The choice of formalism has an effect on the ease of model fitting. A nonlinear formalism, such as one based on differential equations, will naturally be more difficult to optimize than one based on linear approximation such as s-systems. But ease of optimization alone is not enough to force a selection of modeling formalism.

Evolution of regulatory networks: Evolution produces apparently complex systems from supposedly random mutations guided by selection pressure. Computer scientists, inspired by evolution, created the field of evolutionary computation that uses the same principles thought to be at work in evolution. Nature has been very successful, more than engineers, in constructing robust systems in noisy environments through evolution. It is therefore only natural to apply the concepts of evolutionary computation to the problem of regulatory network modeling.

Evolution of regulatory networks differs from model fitting primarily in the degrees of freedom available. Model fitting primarily considers alterations of the model parameters alone with no impact on the model topology. In the evolutionary approach, the topology of the network is able to change as well as the parameters. Akin to network reverse engineering[59,60] in its goal, evolutionary methods do not rely on statistical inference.

Dissertation organization and accomplishments

This dissertation is divided into eight chapters representing progressive steps toward the goal of evolving models of biochemical networks that exhibit a phenotype of interest. The current chapter serves as introduction to the area of biochemical network modeling, with a brief literature review, and with in-depth review of relevant literature reserved to subsequent chapters.

Chapter 2 introduces the basis of the modeling formalism that will be used throughout this dissertation. This chapter was originally an invited paper[40] for PNPM 2003 (the 10th International Workshop on Petri Nets and Performance Models) held at the University of Illinois at Urbana-Champaign. The workshop audience included applied mathematicians and computer scientists specializing in modeling the performance of computer networks and architectures.

Having defined the mathematic formalism that will be used for our models, in Chapter 3 we develop a user-friendly modeling environment that serves as the environment in which our models are built, evaluated, and compared. We pursued this development instead of using available packages in order to maintain tight control over the environment, to ensure extensibility for our research, and to learn the details of the involved methods through implementation. Alternatives are either restricted to binary-only distributions, are based on unfamiliar languages, have unreasonable license agreements, or have inadequate performance. This software, referred to as PNE, has been released into the public domain under the GNU General Public License for anyone to use and modify.

Chapter 4 was originally published in the 2005 proceedings of the Pacific Symposium on Biocomputing and introduced a method of finding solutions to the equations of our models. Subsequently we present better methods, but Chapter 4 represents an important step.

Chapter 5 consists of a paper originally published in *Genetics*[61] in which our modeling environment is used to develop a model of the yeast galactose pathway. We subject this pathway to genetic selection similar to methods used in breeding simulations, enabling us to observe the dynamics of populations under selection. This is the first chapter in which we actually see the simulation of a real biochemical network.

In Chapter 6 a model-fitting environment is presented that attempts to fit a model to a set of experimental data by exploring values of free parameters that may include kinetic rates as well as the model topology. Ideally, measured values for all kinetic parameters would be available, but such availability is rare. Still rarer is complete understanding of the topology. A hybrid search strategy comprising a stochastic optimization method (genetic algorithm) and a local optimization method (simplex) is successfully used despite the presence of nonlinear dynamics. If given sufficient freedom, the system can be used to evolve new models as is demonstrated by identifying alternative genetic toggle-switch models. There is also discussion of how one can quickly estimate network similarity using graph theoretical algorithms to identify unique networks from the solution sets.

Chapter 7 presents the Guet network library. We first model these networks in PNE using our understanding of the regulatory elements involved. Then we apply the tool developed in Chapter 6 to these networks to try to find valid parameters. If building blocks can be identified that have consistent parameters in different networks, then it should be possible to construct, using these building blocks, models of new, yet unconsidered, regulatory networks. If such independence does not exist, it would suggest a nontrivial interaction within or between the regulatory motifs that is not currently part of the theory of regulation.

Finally, in the last chapter, Chapter 8, we conclude the main part of the dissertation by considering the preceding chapters retrospectively, summarizing what has been accomplished and suggesting directions of research that would advance this field.

Following the conclusions are two appendices. In Appendix A is an example of how the modeling environment that we have created can be applied to other domains. PNE is modified to target Petri Networks and several examples are described. There is considerable work supporting Petri Network theory and they have been applied to modeling regulatory networks.

Appendix B contains optimized models each of the Guet networks. Some match the experimental data while others do not. In some cases, comments have been added describing unusual properties of the particular network.

Literature cited

1. Bulter, T., Bernstein, J.R., and Liao, J.C. 2003. A perspective of metabolic engineering strategies: moving up the systems hierarchy. *Biotech. Bioeng.* **84**:815-21.
2. Francois, P. and Hakim, V. 2004. Design of genetic networks with specified functions by evolution in silico. *Proc. Natl. Acad. Sci. U.S.A.* **101**:580-5
3. Hasty, J., McMillen, D., *et al.* 2001. Computational studies of gene regulatory networks: in numero molecular biology. *Nat. Rev. Genet.* **2**:268-79.
4. Kaern, M., Blake, W.J., and Collins, J.J. 2003. The engineering of gene regulatory networks. *Annu. Rev. Biomed. Eng.* **5**:179-206.
5. Kaern, M. 2003. Regulatory Dynamics in Engineered Gene Networks. *Third Inter. Conf. Sys. Bio.* St. Louis, Ill. U.S.A.
6. Smolen, P., Baxter, D.A., and Byrne, J.H. 2000. Modeling transcriptional control in gene networks - methods, recent results, and future directions. *Bull. Math. Biol.* **62**:247-92.
7. Peccoud, J. and Vander Velden, K.A. Computer system for genotype to phenotype mapping using molecular network models. **No. 1660P**. 9-1-2003. U.S.A. patent application.
8. von Dassow, G., Meir, E., *et al.* 2000. The segment polarity network is a robust developmental module. *Nature* **406**:188-92.
9. Wall, M.E., Hlavacek, W.S., and Savageau, M.A. 2004. Design of gene circuits: lessons from bacteria. *Nat. Rev. Genet.* **5**:34-42.
10. May, R.M. 2004. Uses and abuses of mathematics in biology. *Science* **303**:790-3.

11. Ravasz, E., Somera, A.L., *et al*. 2002. Hierarchical organization of modularity in metabolic networks. *Science* **297**:1551-5.

12. de Jong, H. 2002. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comp. Biol.* **9**:67-103.

13. Alon, U. 2003. Biological networks: the tinkerer as an engineer. *Science* **301**:1866-7.

14. Ferber, D. 2004. Synthetic biology. Microbes made to order. *Science* **303**:158-61.

15. Milo, R., Shen-Orr, S., *et al*. 2002. Network motifs: simple building blocks of complex networks. *Science* **298**:824-7.

16. Milo, R., Itzkovitz, S., *et al*. 2004. Superfamilies of evolved and designed networks. *Science* **303**:1538-42.

17. Atkinson, M.R., Savageau, M.A., *et al*. 2003. Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in Escherichia coli. *Cell* **113**:597-607.

18. Cherry, J.L. and Adler, F.R. 2000. How to make a biological switch. *J. Theor. Biol.* **203**:117-33.

19. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in Escherichia coli. *Nature* **403**:339-42.

20. Thomas, R. 1998. Laws for the dynamics of regulatory networks. *Int. J. Dev. Biol.* **42**:479-85.

21. Elowitz, M.B. and Leibler, S. 2000. A synthetic oscillatory network of transcriptional regulators. *Nature* **403**:335-8.

22. Yokobayashi, Y., Weiss, R., and Arnold, F.H. 2002. Directed evolution of a genetic circuit. *Proc. Natl. Acad. Sci. U.S.A.* **99**:16587-91.

23. Setty, Y., Mayo, A.E., *et al*. 2003. Detailed map of a cis-regulatory input function. *Proc. Natl. Acad. Sci. U.S.A.* **100**:7702-7.

24. Guet, C.C., Elowitz, M.B., *et al*. 2002. Combinatorial synthesis of genetic networks. *Science* **296**:1466-70.

25. Smith, H.O., Hutchison, C.A., III, *et al*. 2003. Generating a synthetic genome by whole genome assembly: phiX174 bacteriophage from synthetic oligonucleotides. *Proc. Natl. Acad. Sci. U.S.A.* **100**:15440-5.

26. Rasmussen, S., Chen, L., *et al*. 2004. Evolution. Transitions from nonliving to living matter. *Science* **303**:963-5.

27. Gerchman, Y. and Weiss, R. 2004. Teaching bacteria a new language. *PNAS* **101**:2221-2.

28. Bulter, T., Lee, S.G., *et al*. 2004. Design of artificial cell-cell communication using gene and metabolic networks. *Proc. Natl. Acad. Sci. U.S.A.* **101**:2299-304.

29. Kobayashi, H., Karn, M., *et al*. 2004. Programmable cells: Interfacing natural and engineered gene networks. *Proc. Natl. Acad. Sci. U.S.A.* 0402940101.

30. Segre, D. 2004. The regulatory software of cellular metabolism. *Trends Biotechnol.* **22**:261-5.

31. Tong, A.H., Lesage, G., *et al*. 2004. Global mapping of the yeast genetic interaction network. *Science* **303**:808-13.

32. Giot, L., Bader, J.S., *et al*. 2003. A protein interaction map of *Drosophila melanogaster*. *Science* **302**:1727-36.

33. Yook, S.H., Oltvai, Z.N., and Barabasi, A.L. 2004. Functional and topological characterization of protein interaction networks. *Proteomics.* **4**:928-42.

34. Jeong, H., Tombor, B., *et al*. 2000. The large-scale organization of metabolic networks. *Nature* **407**:651-4.

35. Jeong, H., Mason, S.P., *et al*. 2001. Lethality and centrality in protein networks. *Nature* **411**:41-2.

36. Podani, J., Oltvai, Z.N., *et al*. 2001. Comparable system-level organization of Archaea and Eukaryotes. *Nat. Genet.* **29**:54-6.

37. Tyson, J.J., Chen, K., and Novak, B. 2001. Network dynamics and cell physiology. *Nat. Rev. Mol. Cell Biol.* **2**:908-16.

38. Vilar, J.M., Kueh, H.Y., *et al*. 2002. Mechanisms of noise-resistance in genetic oscillators. *Proc. Natl. Acad. Sci. U.S.A.* **99**:5988-92.

39. Stewart, I. 2004. Networking opportunity. *Nature* **427**:601-4.

40. Vander Velden, K.A. and Peccoud, J. 2003. Modeling networks of molecular interactions in the living cell: structure, dynamics, and applications. *International Workshop on Petri Nets and Performance Models* **10**:2-10.

41. Burleigh, Ian, Suen, Garret, and Jacob, Christian. 2003. DNA in Action! A 3D Swarm-based Model of a Gene Regulatory System. *Proc. First Aust. Conf. A. Life*. Canberra, Australia.

42. Paulsson, J. 2004. Summing up the noise in gene networks. *Nature* **427**:415-8.

43. Blake, W.J., Kaern, M., *et al*. 2003. Noise in eukaryotic gene expression. *Nature* **422**:633-7.

44. Fedoroff, N. and Fontana, W. 2002. Genetic networks. Small numbers of big molecules. *Science* **297**:1129-31.

45. Welch, S.M., Roe, J.L., and Dong, Z.S. 2003. A genetic neural network model of flowering time control in *Arabidopsis thaliana*. *Agronomy Journal* **95**:71-81.

46. Ehrenberg, M., Elf, J., *et al*. 2003. Systems biology is taking off. *Genome Res.* **13**:2377-80.

47. Klapa, M.I. and Quackenbush, J. 2003. The quest for the mechanisms of life. *Biotechnol. Bioeng.* **84**:739-42.

48. Mendes, P. 1993. GEPASI: a software package for modeling the dynamics, steady states and control of biochemical and other systems. *Comput. Appl. Biosci.* **9**:563-71.

49. Mendes, P. 1997. Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3. *Trends Biochem. Sci.* **22**:361-3.

50. Tomita, M., Hashimoto, K., *et al*. 1999. E-CELL: software environment for whole-cell simulation. *Bioinformatics.* **15**:72-84.

51. Hucka, M., Finney, A., *et al*. 2003. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics.* **19**:524-31.

52. Mendes, P. and Kell, D. 1998. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics.* **14**:869-83.

53. Moles, C.G., Mendes, P., and Banga, J.R. 2003. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res.* **13**:2467-74.

54. Yen, J., Liao, J.C., *et al*. 1995. A hybrid approach to modeling metabolic systems using genetic algorithm and simplex method. *Proc. 11th IEEE Conf. A. Intel. App.* 277-83. Varna, Bulgaria.

55. Ronen, M., Rosenberg, R., *et al*. 2002. Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. *Proc. Natl. Acad. Sci. U.S.A.* **99**:10555-60.

56. Wong, P., Gladney, S., and Keasling, J.D. 1997. Mathematical model of the lac operon: inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose. *Biotechnol. Prog.* **13**:132-43.

57. Lee, P.S., Shaw, L.B., *et al*. 2003. Insights into the relation between mRNA and protein expression patterns: II. Experimental observations in Escherichia coli. *Biotechnol. Bioeng.* **84**:834-41.

58. Mehra, A., Lee, K.H., and Hatzimanikatis, V. 2003. Insights into the relation between mRNA and protein expression patterns: I. Theoretical considerations. *Biotechnol. Bioeng.* **84**:822-33.

59. Greller, L.D. and Somogyi, R. 2002. Reverse engineers map the molecular switching yards. *Trends Biotechnol.* **20**:445-7.

60. Friedman, N. 2004. Inferring cellular networks using probabilistic graphical models. *Science* **303**:799-805.

61. Peccoud, J., Vander Velden, K.A., *et al*. 2004. The selective values of alleles in a molecular network model are context dependent. *Genetics* **166**:1715-25.

Chapter 2. Modeling networks of molecular interactions in the living cell

A modified version of an invited article published in *The Proceedings of the 10th International Workshop on Petri Nets and Performance Models (PNPM 2003), University of Illinois, Urbana, Illinois, USA, September 2-5, 2003*

Kent A. Vander Velden[1] and Jean Peccoud[1]

[1]Pioneer Hi-Bred International, Inc., DuPont Agriculture & Nutrition, 7200 NW 62nd Avenue, Johnston, IA 50131, USA

Abstract

Interactions occurring in living cells between populations of macromolecules are now sufficiently understood to model them with some level of realism. Here, the structures and dynamics of these models are reviewed, and a number of open problems are discussed. Recent applications of such models indicate that there is a growing need for simulation environments specifically designed for the life sciences.

Introduction

Fifty years ago, biology became molecular with the publication of the crystallographic structure of the DNA molecule [1]. Since then, life scientists geared their efforts and resources to the characterization of the molecules involved in the biochemical processes supporting every aspects of the physiology of all sorts of living organisms spanning a wide range of organizational complexity. The molecular mechanisms of life turned out to be very similar in viruses, bacteria, plants, and animals, making their systematic dissection a very appealing proposition.

If the project is still far from completion, it is already possible to get a global perspective on the network of chemical reactions taking place in a number of model organisms. The recent development of databases with the ambition to record systematically and consistently all the reactions described in these organisms is probably the best indicator of the advancement of this scientific project [2-4]. Some of these databases are publicly available on the Internet, making it easy for a large and diverse community of scientists to access these data and keep abreast of their developments.

Networks of molecular interactions will be referred to as “molecular networks” in this document. They form the communication and control systems of living cells. Starting with individual networks that control fine grain components of the cell, such as uptake and conversion of molecules, exchanges between networks control more visible responses. Interactions of genes and proteins, through a variety of regulation mechanisms, comprise molecular networks and are the mechanisms for responses made to environmental signals and perturbations. Our research into the arena of molecular networks has touched on several areas necessary for realistic modeling, simulation, and analysis. Here we briefly present an overview of this research and the computational challenges it raises.

Structure

Biologists often support textual descriptions of interactions between molecules with pieces of artwork intended to illustrate the main features of the system dynamics. Unfortunately, the representation of molecular networks has not been standardized, making most figures found in the biological literature ambiguous and thus unsuitable for implementation in software. However, chemists have been using standard notations that can be adapted to meet the specific requirements of the life sciences.

Chemical equations

Molecular networks can be represented in a unambiguous way as sets of coupled chemical reactions using chemical equations.

In Equation (2.1) for instance, the first line represents the inactivation of the gene gal4g by glucose noted Glu. The second reaction represents the expression of the gene, i.e. the production of one protein molecule by the DNA molecule coding for this protein. The last reaction represents the spontaneous degradation of the protein.

$$\begin{aligned} &\text{gal4g} + \text{Glu} \underset{k_2}{\overset{k_1}{\rightleftarrows}} \text{gal4gX} \\ &\text{gal4g} \xrightarrow{k_3} \text{gal4g} + \text{Gal4p} \qquad (2.1) \\ &\text{Gal4p} \xrightarrow{k_4} \varnothing \end{aligned}$$

This type of notation naturally leads to a matrix representation of molecular networks. The general form of a chemical equation is:

$$\sum_{m=1}^{M} \alpha_{m,r} X_m \longrightarrow \sum_{m=1}^{M} \beta_{m,r} X_m \left(r = 1, 2, \ldots R\right) \qquad (2.2)$$

Equation (2.2) is completely determined by the two MxR matrices α and β called the reactant and product matrices respectively, and a vector X representing each of the M molecule species in the system. Data structures used to manipulate molecular networks in software are usually derived from these two matrices. The difference $\beta - \alpha$ is often referred to as the stoichiometric matrix.

There is a significant specificity in the way these equations are used in the context of biological molecular networks. Classically in chemistry, chemical equations need to preserve the mass and numbers of atoms. This constraint is known as the law of atomic balance. In biology, accounting for all atoms present in the system is not possible due to the size of the molecules involved. Chemical equations are thus used as a meta-language. Reactions expressing creation or removal of molecules from the system are permitted and necessary. Molecular networks define atom-free stoichiometries [5].

Diagrammatic representation

Vol'pert diagrams are graphical representations of sets of chemical equations (Figure 2.1). They offer a global perspective on the model that helps to understand the architecture of the network. It is often much easier to build a model of medium-sized systems using these diagrams.

Vol'pert diagrams are flat representations of molecular networks. This becomes limiting when the number of reactions in a network exceeds 100 or so. Beyond this limit, it becomes necessary to refine the graphical representation of the networks. Two approaches have been explored to address this problem:

- Hierarchies of diagrams can be defined so that a complex model can be broken down into several manageable sub-networks.
- A number of reaction mechanisms that are found in virtually all molecular networks have been characterized. They include mechanisms of enzyme-catalyzed reactions, mechanisms of gene activation or gene repression, etc. It is possible to simplify the diagrammatic representation of molecular networks by introducing new graphical objects corresponding to these canonic mechanisms.

It is naturally possible to combine both solutions. A more difficult problem is the problem raised by the high dimensions of the state-spaces generated by certain molecules. Many genes have multiple binding sites for proteins regulating their expression. Some proteins have several modification sites to which a phosphate group can be attached. For instance the tumor suppressor protein p53 has at least 12 different modification sites. It can thus exist in 2^{12} different states. Finding a way of representing all these state variables in a concise way remains an open problem.

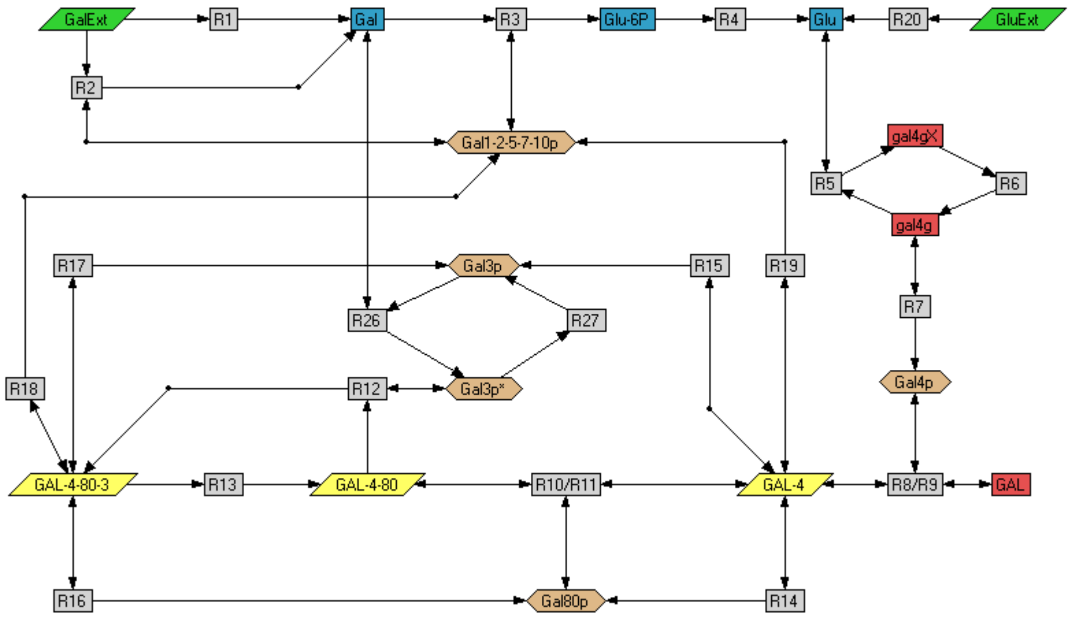


Figure 2.1. Vol'pert diagram of a molecular network

Dynamics

The analysis of the dynamics of chemical systems usually relies on differential equations. In the case of biological systems where the number of interacting molecules is small, it is more realistic to use stochastic models of molecular interactions. However, the computational cost of solving these models when their state space is large makes it necessary to develop approximation solutions.

Differential equations

The mass action rate law is commonly used to describe the kinetics of chemical reactions. The law states that the rate of a reaction is proportional to the concentration of its reactants. The generic form of a reaction rate is provided in Equation (2.3).

$$v_r = k_r \prod_{i=1}^{M} [X_i]^{\alpha_{i,r}} \tag{2.3}$$

The rate v of reaction r is the product of its reactant concentrations. If two molecules of the same type interact in a reaction, their concentration should appear twice. It is very convenient to use the reactant matrix as exponents to express these rates in a generic way. Last, the reaction rate depends on reaction specific kinetic rate constant. It is worth noting that the dimension of this constant depends on the order of the reaction, i.e. $\sum_{i=1}^{M} \alpha_{i,r}$.

From there it is possible to derive a set of ordinary differential equations (ODEs) describing the time-evolution of all the state variables. The net time evolution of a molecule concentration is the difference between the rates of all the reactions producing this molecule and the rates of all the reactions consuming the molecule. Since a single reaction event can consume or produce more than one copy of a molecule, the rates need to be adjusted by the stoichiometric coefficients.

$$\begin{aligned} \frac{dX_i}{dt} &= \sum_r \beta_{i,r} v_r - \sum_r \alpha_{i,r} v_r \\ &= \sum_r \gamma_{i,r} v_r \end{aligned} \tag{2.4}$$

ODEs provide a reasonable approximation of the dynamics of populations of molecules at the thermodynamic limit when the sizes of all populations of molecules are large.

Stochastic process

When modeling at the level of gene regulation, where genes are typically in single copy numbers, ODE solutions may represent a very poor approximation of the system dynamics. This observation was formulated in the early 1940s. Max Delbruck probably was the first author to address this question from a mathematical perspective [6]. Soon afterwards, Erwin Schrödinger commented on the expected fluctuations of the interactions between small populations of macromolecules confined to the small volume of the living cell [7].

This problem has been addressed from a theoretical perspective by physicists and chemists during the two following decades [8-11]. Their results specified the Markov process equivalent to the system of ODEs traditionally used in chemical kinetics. The intensity of the process is sum of the marginal intensities of the reactions. The marginal intensity of a reaction is the stochastic equivalent of the deterministic reaction rate. It specifies the average number of occurrences of a reaction by unit of time. Its structure is comparable to the deterministic reaction rate with a few modifications. As mentioned earlier, the dimension of the deterministic rate constant depends on the order of the reaction. Since the stochastic intensity is based on actual molecule numbers and not molecule concentrations, it is necessary to remove the volume from the kinetic constant. The modification of second term has to do with the probability of two molecules to interact. In any reaction where one molecule of each population interacts with molecules of another population, the term is analogous to the expression of the reaction rates corrected for the volume. If two molecules of the same population interact, then it is slightly different. A great deal of attention was brought to justify these terms in the early articles on stochastic models of the chemical reaction.

$$\begin{aligned} \lambda_r(X) &= \frac{k_r}{V^{\sum_{i=1}^{M}\alpha_{i,r}-1}} \prod_{i=1}^{M} \frac{X_i!}{(X_i-\alpha_{i,r})!} \\ \lambda(X) &= \sum_r \lambda_r(X) \end{aligned} \tag{2.5}$$

The dynamics can be represented by a pure jump process defined by two random variables. The instant of the next jump is exponentially distributed. The next reaction, destination of the next jump, is also randomly distributed. The probability of each reaction depends on the weight of its intensity relative to the sum of the intensities of all reactions.

$$\begin{aligned} &P_X(\tau > s) = e^{-\lambda(X)s} \\ &P\left(X_{t=\tau} = X_{t=0} + \gamma_r\right) = \frac{\lambda_r\left(X_{t=0}\right)}{\lambda\left(X_{t=0}\right)} \end{aligned} \tag{2.6}$$

Donald Gillespie derived from the mathematical definition of the process, a computer algorithm to perform exact simulations of this process [12,13]. A more effective version of this algorithm has been published recently [14]. These methods give excellent results when the mean time between jumps does not tend to infinitesimal values. When the system becomes stiff because the rate constant of one reaction is several orders of magnitude larger than the rate of the slowest reaction or because one population of molecules is several orders of magnitude larger than the smallest population, the computing cost of this approach becomes prohibitively expensive. This limitation is currently driving a very active field of research aiming at finding fast and dependable approximations of this stochastic dynamics.

Approximations

It may seem natural that the stable steady-states of the ODE be associated with the modes of the stationary distribution of the corresponding stochastic process. However, in general there is no one-to-one correspondence between equilibrium points and extrema of stationary distributions [5]. Similarly, the mean values of the variables of the stochastic process do not match the trajectories of the ODEs [15] even though the differences may be negligible. Hence a practical, but not very rigorous, approach is to use ODEs while building a model and switch to stochastic simulation after the dynamics appear correct. This allows one to focus on building a network initially and then explore the effects of stochastic noise later.

To address the need for stochastic simulation, but also the conflicting need for simulation speed, stochastic approximation methods are being developed. In particular, it has been demonstrated that under some conditions, the number of jumps occurring during a small time step can be approximated by a Poisson distribution [16].

Another approach would be to develop a hybrid model by partitioning the state-space into three categories of variables whose dynamics would be represented by a pure jump process, a diffusion process, and differential equations. This would lead to a generalized Markov process as it is defined by Gardiner [17]. This approach would require some *a priori* knowledge of the size of the populations of molecules corresponding to each dimension of the state space. Since this information is usually not available before the model is simulated, it is necessary to start by assuming that all variables jump between discrete values and find a way of approximating the time-evolution of each variable as we go. It is likely that different types of problems will require different types of approximations.

Software

Upon starting this project, a survey of the software tools available was conducted. Several applications have been developed to help biologists analyze the emerging properties of molecular networks. Some applications like Gepasi[18,19] or Scamp/Jarnac[20] rely on a textual specification of the models close to the notation used in chemical equations. This approach quickly becomes impractical when the number of reactions grows beyond 20 to 30 reactions.

A diagrammatic representation of the networks makes it much easier to capture the logic of larger networks. This observation was the rationale for the development of JDesigner, an add-on for Jarnac providing a network view of the models. Similarly, Pedro Mendes, who developed and maintains Gepasi, is currently working on the development of Copasi, which will also include a diagrammatic representation of the models.

The correspondence between molecular networks and stochastic Petri nets has been established [21]. This made it possible to use software originally designed to analyze the performability of computer architectures such as UltraSAN or Mobius [22], to solve the stochastic dynamics of molecular networks.

Simulation solutions are also already available from various vendors and more are expected in a near future. For instance, Princeton (NJ) based Physiome Sciences, Inc sells a software package named PathwayPrism™ with features somewhat similar to the Jarnac and JDesigner combination. A few prominent companies operating in the technical and scientific computing market are also working on the development of similar simulation platforms.

While several systems were available, none met all of our needs in terms of user interface, representations of models, and capability to switch between differential, stochastic, and hybrid dynamics. This lack of desired flexibility resulted in the decision to develop in house our proprietary modeling environment.

In this environment molecular networks are constructed using a graphical language and internally converted to a series of chemical reactions based on mass action reactions. Most simulation packages in the life sciences implement various types of rate laws commonly found in biological systems [23]. These specialized kinetics are approximations of the kinetics of common reaction mechanisms. They can thus to be modeled by the mass action rate law without the need for introducing specialized rate laws. This approach is entails a small computation penalty but is safer since it does not rely on any assumption ensuring the validity of the approximation. It also provides the freedom to explore alternative simulation and analysis techniques.

Applications

Building models of molecular networks is a way to distinguish aspects of a biological system that are well documented from those that need to be hypothesized. In some cases, the properties of models can be used to evaluate the biological realisms of the assumptions upon which the model was built. Recently, genetic constructs exhibiting complex dynamics have been engineered based on a prior analysis of molecular networks models.

Knowledge capture

While the ultimate goal of molecular network modeling might be a means to understand the dynamics of the network, there are intermediate rewards. Identification of weaknesses in one’s understanding of the network is one such reward. Questions are raised as soon as one starts to build a model, and become more complicated as one begins to fill in all the parameters needed for simulation. Frequently, the modeler needs to research or hypothesize stoichiometric coefficients, reaction and degradation rates, cooperative binding, formation of complexes etc. Modeling molecular networks is a way to capture the knowledge of biologists and to formulate working hypotheses.

Discovery

The modeling of molecular networks starts with the identification and placing into the modeling environment interactions between genes, proteins, and possible environmental factors. During the process of network construction, rate constants must be identified for each reaction. The rate constants describe the relative speed of each reaction and are equally important for simulation as the interactions. While interactions are often accessible from the literature, rate constants are rarely documented and are generally inaccessible through experiment. Fortunately, robustness seems to be a very common property of biological networks. Several networks are pretty insensitive to parameter values [24-27]. This property tends to be used as a criterion to assess the biological realism of a model. A model exhibiting a strong robustness indicates that the assumptions used to build it are to be favored. This indication can lead to the design of experiments aiming at the verification of these underlying hypotheses.

Stochastic modeling of molecular networks recently drove a series of experiments aiming at the experimental observation of molecular noise at the single cell level [28-30]. Our understanding of molecular interactions, in light of these living cells, is being revisited in the light shed by these new developments. Control mechanisms seem to be able to leverage molecular noise [31,32], and therefore the stability of molecular clocks to molecular noise is being investigated [33]. Cellular differentiation is analyzed as a first-exit problem [34,35].

Engineering

Models of molecular networks have driven the design of new genetic constructs exhibiting complex dynamics. In 2000, Elowitz designed the Repressilator, a construct consisting of three genes repressing each other and leading to the oscillating expression of a fluorescent protein [36]. The same year, Gardner designed a bi-stable construct called a toggle switch by combining two genes repressing each other [37]. Since then, a number of other constructs have been described (see [38,39] for recent reviews).

Even if practical applications of this new generation of constructs remain to be identified, they can already be regarded as a turning point in the history of biology. They clearly demonstrate that we already know enough about interactions between molecules in the living cell to model them with some level of realism. The minimal artificial networks that have been engineered so far can be compared to simple electrical circuits consisting of a few resistors, capacitors, and transistors, but it is likely that much more complicated constructs will be engineered in a near future.

Conclusion

This analogy between molecular networks and electrical circuits leads a number of scientists to believe that there will soon be a need for CAD applications to design genetic constructs. This trend is probably best illustrated by the University of California, Berkeley, which hosted the development of the SPICE circuit simulator and is now supporting the development of the BioSPICE project. This field of research is rich in opportunities for modelers, computer scientists, software engineers, and electrical engineers wishing to ride wave of a systems approach of molecular biology.

Acknowledgments

We are indebted to Lane Arthur, Mark Cooper, Roy Luedtke, Bob Merrill and many other Pioneer scientists for creating an environment very favorable to this research.

Literature cited

1. Watson, J.D. and Crick, F.H. 2003. A structure for deoxyribose nucleic acid. 1953. *Nature* **421**:397-8.
2. Kanehisa, M., Goto, S., *et al*. 2002. The KEGG databases at GenomeNet. *Nucleic Acids Res.* **30**:42-6.
3. Karp, P.D., Riley, M., *et al*. 2002. The MetaCyc Database. *Nucleic Acids Res.* **30**:59-61.
4. Karp, P.D., Riley, M., *et al*. 2002. The EcoCyc Database. *Nucleic Acids Res.* **30**:56-8.
5. Erdi, P. and Toth, J. 1989. Mathematical models of the chemical reaction. Manchester University Press, Manchester, United Kingdom.
6. Delbrück, M. 1940. Statistical fluctuations in autocatalytic reactions. *J. Chem. Phys.* **8**:120-4.
7. Schrödinger, E. 1944. What's lifer. Cambridge University Press, Cambridge, United Kingdom.

8. Bartholomay, A.F. 1958. Stochastic models for chemical reactions: I. Theory of the unimolecular reaction process. *The Bull. Math. Biophys.* **20**:175-90.

9. Bartholomay, A.F. 1959. Stochastic models for chemical reactions: II. The unimolecular rate constant. *The Bul. Math. Biophys.* **21**:363-73.

10. McQuarrie, D.A. 1967. Stochastic approach to chemical kinetics. *J. Appl. Probab.* **4**:413-8.

11. Oppenheim, I., Shuler, K.E., and Weiss, G.H. 1969. Stochastic and deterministic formulation of chemical rate equations. *J. Chem. Phys.* **50**:460-6.

12. Gillespie, D.T. 1976. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.* **22**:403-34.

13. Gillespie, D.T. 1977. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**:2340-61.

14. Gibson, M.A. and Bruck, J. 2000. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Chem. Phys. A* **104**:1876-89.

15. McQuarrie, D.A. 1964. Kinetics of small systems. II. *J. Chem. Phys.* **40**:2914-21.

16. Aparicio, J.P. and Solari, H.G. 2001. Population dynamics: Poisson approximation and its relation to the Langevin process. *Phys. Rev. Lett.* **86**:4183-6.

17. Gardiner, C.W. 1983. Handbook of stochastic methods for physics, chemistry, and the natural sciences. Springer-Verlag, Berlin, Heidelberg, and New York.

18. Mendes, P. 1997. Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3. *Trends Biochem. Sci.* **22**:361-3.

19. Mendes, P. 1993. GEPASI: a software package for modeling the dynamics, steady states and control of biochemical and other systems. *Comput. Appl. Biosci.* **9**:563-71.

20. Sauro, H.M. 1993. SCAMP: a general-purpose simulator and metabolic control analysis program. *Comput. Appl. Biosci.* **9**:441-50.

21. Goss, P.J.E. and Peccoud, J. 1998. Quantitative modeling of stochastic systems in molecular biology using stochastic Petri nets. *Proc. Nat. Acad. Sci. U.S.A.* **95**:6750-5.

22. Couvillon, J., Freire, R., *et al.* 1991. Performability Modeling with UltraSAN. *IEEE Soft.* **8**:69-80.

23. Heinrich, R. and Schuster, S. 1996. The regulation of cellular systems. Chapman & Hall, New York.

24. Alon, U., Surette, M.G., *et al*. 1999. Robustness in bacterial chemotaxis. *Nature* **397**:168-71.

25. Barkai, N. and Leibler, S. 1997. Robustness in simple biochemical networks. *Nature* **387**:913-7.

26. von Dassow, G., Meir, E., *et al*. 2000. The segment polarity network is a robust developmental module. *Nature* **406**:188-92.

27. Wagner, A. 2000. Robustness against mutations in genetic networks of yeast. *Nat. Genet.* **24**:355-61.

28. Blake, W.J., Kaern, M., *et al*. 2003. Noise in eukaryotic gene expression. *Nature* **422**:633-7.

29. Elowitz, M.B., Levine, A.J., *et al*. 2002. Stochastic gene expression in a single cell. *Science* **297**:1183-6.

30. Ozbudak, E.M., Thattai, M., *et al*. 2002. Regulation of noise in the expression of a single gene. *Nat. Genet.* **31**:69-73.

31. Hasty, J., Pradines, J., *et al*. 2000. Noise-based switches and amplifiers for gene expression. *Proc. Natl. Acad. Sci. U.S.A.* **97**:2075-80.

32. Rao, C.V., Wolf, D.M., and Arkin, A.P. 2002. Control, exploitation and tolerance of intracellular noise. *Nature* **420**:231-7.

33. Gonze, D., Halloy, J., and Goldbeter, A. 2002. Robustness of circadian rhythms with respect to molecular noise. *Pro. Nat. Acad. Sci. U.S.A.* **99**:673-8.

34. Aurell, E. and Sneppen, K. 2002. Epigenetics as a first exit problem. *Phys. Rev. Lett.* **88**:048101.

35. Aurell, E., Brown, S., *et al*. 2002. Stability puzzles in phage lambda. *Phys. Rev. E. Stat. Nonlin. Soft. Matter Phys.* **65**:051914.

36. Elowitz, M.B. and Leibler, S. 2000. A synthetic oscillatory network of transcriptional regulators. *Nature* **403**:335-8.

37. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**:339-42.

38. Hasty, J., McMillen, D., and Collins, J.J. 2002. Engineered gene circuits. *Nature* **420**:224-30.

39. Judd, E.M., Laub, M.T., and McAdams, H.H. 2000. Toggles and oscillators: new genetic circuit designs. *Bioessays* **22**:507-9.

Chapter 3. Biochemical network modeling environment

A modified version of a paper to be submitted to *Bioinformatics* under the title: GenoDYN: A modeling framework for molecular network analysis

Kent A. Vander Velden [1,2] and Jean Peccoud [3]

[1]Pioneer Hi-Bred International, Inc., DuPont Agriculture & Nutrition, 7200 NW 62nd Avenue, Johnston, IA 50131, USA

[2]Bioinformatics and Computational Biology Program, Iowa State University, Ames, IA 50011, USA

[3]Virginia Bioinformatics Institute, Washington Street, MC0477, Blacksburg, VA 24061, USA

Abstract

Motivation: The model-driven design of artificial gene networks requires a computational environment that adapts to this emerging domain concepts and methods used in other fields of engineering. Traditional gene network simulation software applications do not enable a hierarchical definition of models, the reuse of previously defined models in larger models, or the definition of generic performance metrics.

Results: GenoDYN provides an environment for analyzing artificial gene networks. GenoDYN supports the construction of network models using intuitive graphical representations of molecules, reactions, and network motifs, with additional controls for modeling external inputs. GenoDYN also supports analysis of the dynamics of the network model using continuous or stochastic simulation and built-in visualization tools such as line and phase plane plots, or time series of statistical distributions. GenoDYN includes a distributed computing framework for computationally demanding stochastic simulations. GenoDYN also provides built-in performance evaluation functions as well as user-specified evaluation functions written in a built-in scripting language.

Introduction

Model-driven design of artificial synthetic genetic systems meeting user-defined specifications is the ultimate vision of synthetic biology [1-9]. The development of the first artificial gene networks relied on a qualitative analysis of the dynamics of small systems of differential equations [10-12], but this method does not scale up beyond these few proof-of-concept results. Numerical simulations coupled with the automatic exploration of the design space [13-17] seems to be an interesting alternative to identifying robust designs capable of exhibiting desirable phenotypes, but this promising approach has been explored with software prototypes that lack mature modeling capabilities.

A modeling platform, used in engineering projects, should support the development of an abstraction hierarchy allowing users to analyze model properties at different levels of organization [18] by taking advantage of the modularity of artificial genetic systems [19, 20]. In addition, the modeling framework should be capable of expressing the artificial genetic system inputs as interactions with their physical or biological environment [21]. Being able to define arbitrary functions of the models' states to express the design performance is essential to evaluate designs. In addition to streamlining the modeling definition process, the modeling platform should be integrated with tools to explore the design space by optimizing parameter values or even the model structure. GenoDYN was developed with these requirements in mind. It supports a hierarchical definition of models that encourages reuse of previously defined models. It supports multiple simulation engines that are well integrated with sophisticated visualization capabilities to enable fast model development iterations. In situations where the computing cost of simulations exceeds the capability of the workstations used to run the client applications, users have the possibility to seamlessly execute their simulations on a dedicated cluster. Evaluating model performance is possible by using built-in generic functions or model-specific functions developed in a custom scripting system. Finally, the GenoDYN platform includes higher level applications that can be used for exploration of the design space.

GenoDYN joins a growing list of software environments used in systems biology for modeling biological networks [22-25]. Inevitably, some of the capabilities of GenoDYN overlap those of others. A recent review of available packages [26] can help prospective users identify the software solution most suitable for a particular research project.

This report focuses on GenoDYN's most specific features. Its organization reflects the modeling workflow by first describing model editing functions. Model simulation and evaluation are described in the following sections. Model evolution tools are briefly covered before discussing some of the limitations of this platform and its possible future developments.

Model definition

Basic editing

GenoDYN presents a canvas view for modeling biochemical networks. Directed edges between molecules and reactions model product and reactant relationships. Context-sensitive menus provide a palette of network entities such as a molecules and reactions. Different categories of molecules are available (e.g. DNA, RNA, proteins, metabolites, complexes) but the difference between them is limited to their graphical representation. Numerically all molecules are equivalent. Similarly, GenoDYN reactions are restricted to only mass-action kinetics. This choice simplifies the design of the simulation engine and makes the model interactions explicit and valid in all conditions, unlike specialized reaction kinetics that are valid only under certain assumptions [27].

Sections of a model can be selected, cut, and pasted. Because model entities must have unique names, when a new model section is introduced by a paste operation, molecule and reaction names of the new section in conflict with existing names will be suffixed with a random four-letter string.

The **`Edit`** menu includes additional features that help refine the visual representation of GenoDYN models such as randomized placement, snap to grid, and dynamic layout based on a ball-spring physics simulation.

Multiple concurrent canvases allow one to work on multiple models simultaneously and copy and paste sections between models. Models are stored as XML documents that can be converted to SBML [28] using a transformation language such as XSLT.

Environment specification

When modeling a biological system, it is often desirable to evaluate how it reacts to variations of the physical environment. Examples of such variations are circadian oscillations of light, temperature, nutrient availability, and changes of the growth medium such as the addition of gene expression inducers. The physical environment can be represented by variables that affect the dynamics of the model but are not affected themselves by the time evolution of the model's variables. In GenoDYN, control variables are model entities that can impose a boundary condition on the model dynamics.

Two different types of control variables are available: square wave and interpolated. Square wave control variables have six parameters, making it possible to represent various transitions between two different states including periodic oscillations and impulse functions. Despite the flexibility of the square wave control variables, in some cases it is necessary to introduce control variables with a very specific dynamics. This would be the case when analyzing the network response to a set of experimental perturbations recorded in a reactor. In these cases it is possible to introduce in the model an interpolated control variable whose dynamics will be specified by importing a text file containing time series values.

GenoDYN calls "environment" a set of parameterizations of the model control variables. After the number of environments defined on the model has been specified in the **`Simulation>Environment>Edit menu`**, the dynamics of each control variable can be specified by successively selecting each environment in the drop list of the control variable definition dialogue.

Hierarchical modeling

GenoDYN supports a hierarchical approach to the definition of complex models by allowing the reuse of existing models as sub-networks of more complex models. Any molecule of a model can be exported by checking the corresponding box in the molecule definition dialogue. This exposes the molecule in the sub-network object, allowing it to be connected to other entities in the calling model (Figure 3.1). Subnetworks can be edited or defined directly into the larger models. Alternatively, a subnetwork can be imported directly from an existing model file. Modifications of the original subnetwork file are not propagated to the models using that file as subnetworks. This feature enables a structured approach to model development. Complex models can be broken down into more manageable components that can be analyzed individually before integration in larger models. The structure of large models becomes more apparent as the entire model can be represented by various subnetworks corresponding to pathways or components of the global network.

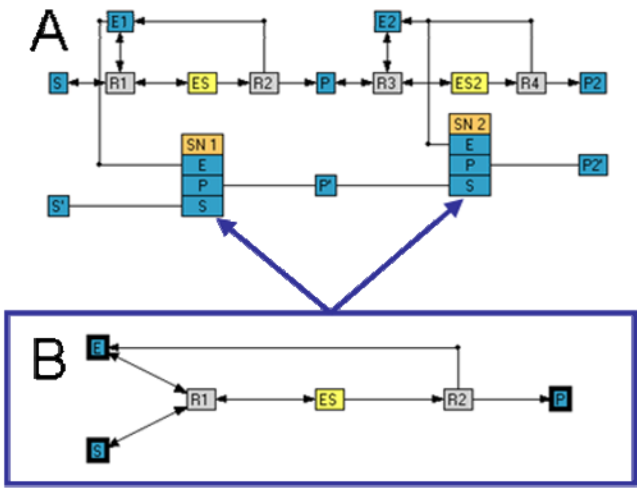


Figure 3.1. Hierarchical model using subnetworks. The model A contains two references to the sub-model B. Model A includes two parallel pathways from S to P2 and from S’ to P2’. Each pathway is composed of two Michaelis-Menten mechanisms. In the S to P2 pathway, all the molecular steps are visible whereas the use of the two subnetworks SN1 and SN2 makes it possible to represent the same molecular network in a more abstract and compact format in the S’ to P2’ pathway. The three variables E, S, and P of model B accessible to models at the next level in hierarchy are indicated by a thick black contour.

In addition, it is possible to build a library of subnetworks corresponding to common molecular mechanisms or network motifs. Models in a common shared directory can be directly inserted into a model using the library item of the context-sensitive menu.

Reporting

Models can be documented and exported using different methods. The model diagram can be copied and pasted into other applications (**`Edit>Copy to clipboard`**). Similarly, the diagram can be saved in SVG and PNG files. A comprehensive report can be generated and saved in a text file (**`Pathway>View Report`**). The model can be exported in different formats to be analyzed in other tools.

Simulation

ODE and stochastic simulation

Chemical kinetics has traditionally relied on Ordinary Differential Equations (ODEs) to describe the time evolution of molecule concentrations [29]. However, concentrations are the limit of the mean number of molecules per unit of volume when this number tends to infinity. When a model includes small populations of molecules, its dynamics is better described by a Markovian jump process with a discrete state space [30, 31]. This situation is frequently encountered in the modeling of gene regulatory networks where only a few copies of molecules like genes, mRNA, or even some transcription factors are present in living cells.

Hence, GenoDYN provides two modes for simulation: continuous, using CVODE [32] to solve ordinary differential equations (ODEs), and stochastic, using Gillespie's direct method [33]. Users can switch between the representations that are mathematically equivalent [34]. A typical modeling workflow starts by simulating the ODEs as a means to quickly view the system dynamics. The **`Simulation>Options`** dialogue box allows users to specify the simulation time frame and the sampling period used to collect data for visualization. The ODE solvers options allow users to let the solver find the steady state of the model and set integration parameters that can be used to fine tune the numerical integration.

In models with large numbers of molecules such as metabolic pathways, it may not be necessary to use another solver. However, for models involving small populations of molecules such as models including gene expression mechanisms, the model analysis will involve running the stochastic simulator. Stochastic simulation is valid for any number of molecules, but its computation time grows rapidly with the number of molecules and the rate of the fastest reactions. The parameters of the stochastic simulator are limited to the number of trajectories simulated and the update rate of the simulation visualization. For stiff systems having reactions occurring at very different rates, stochastic simulations may require significant computation time [35-37]. Simulations can always be aborted by pressing the **`Esc`** key.

Visualization

Simulation results are collected by sampling the concentration of each molecule at regular intervals. The sampling period affects the memory GenoDYN requires by controlling the number of data points recorded during the simulation. Oversampling simulations could result in a decrease of performance or excess memory consumption.

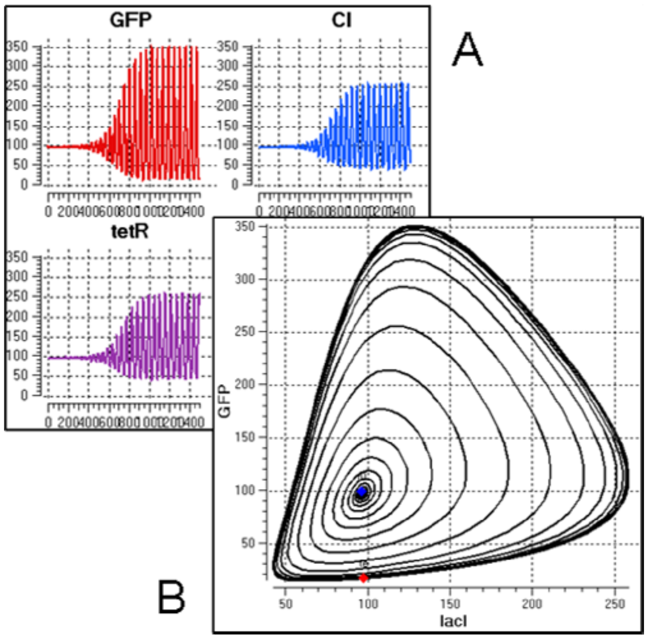


Figure 3.2. Visualization of the solution of an ODE model of a molecular network of the Repressilator. GenoDYN can either plot (A) the time evolution of individual variables or (B) the phase portrait representing the evolution of a variable as a function of another state variable.

For continuous simulations and stochastic simulations of individual trajectories, users can display the results as plots of concentration versus time, or as plots of one variable versus another in the phase plane view (Figure 3.2). Right clicking on any of the plots reveals a menu that allows users many options to customize the trajectory visualization.

Results from stochastic simulations consisting of ensembles of multiple trajectories are summarized using a series of concentration distributions over time (Figure 3.3). The resulting concentration distributions are represented as colored intensity plots representing the distribution histogram at each sampling time (Figure 3.3). The number of bins used to make the histograms can be manually set. In addition, it is possible to independently toggle the display of the distribution and its mean. These plots provide an avenue to quickly visualize the noise and stability of a network.

Finally, data from the plots can be exported to a text file for further analysis in a different environment.

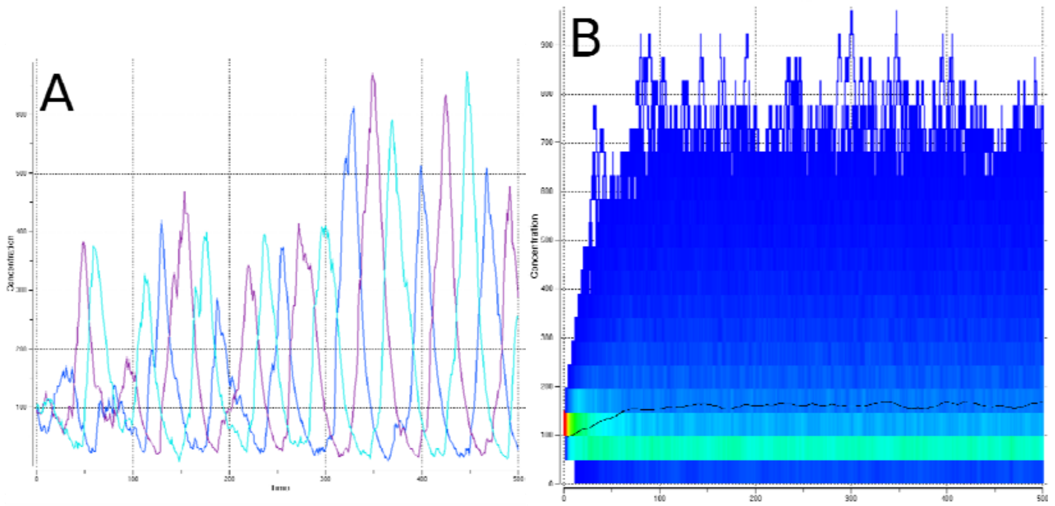


Figure 3.3. Visualization of the dynamics of stochastic models. (A) represents a single trajectory of the Repressilator by superimposing on a single plot the evolution of the number of proteins coded by the three genes in the network. Plot (B) represents the evolution of the statistical distribution of one of the protein levels estimated from the simulation of 1,000 trajectories. The solid line represents the mean value of these distributions. Because these trajectories are not synchronized, the distributions do not oscillate even though individual trajectories do.

Distributed simulation

Stochastic models give a more comprehensive insight into the dynamics of a molecular network but this benefit comes at a significant computational cost, since estimating the dynamics of the state variable statistical distributions requires the simulation of numerous trajectories. This can rapidly lead to significant computation times. Because of the trajectory independence of the Gillespie algorithm, it is possible to achieve a linear speedup by distributing the simulations over multiple processors.

Many computational biologists who could benefit from using a distributed computing environment for stochastic modeling of molecular networks do not have easy access to a computer cluster or do not have the skills to work in such environment. To eliminate this barrier to entry into distributed computing, GenoDYN running on a personal computer can be used as a front-end to a cluster or a grid. In the simulation options dialogue, one can specify the IP address of a remote computer where a calculation server runs in task supervisor mode. This calculation server receives the specification of a simulation, essentially a serialized form of the model file, from clients running on user desktops. Once set up, GenoDYN running on one's desktop computer sends a model to a centralized task supervisor, which in turn passes the model to idle calculation workers. The calculation supervisor collects results from the workers and returns them to the corresponding GenoDYN client. Multiple GenoDYN instances can be connected to a given calculation supervisor at once, with a dynamic pool of calculation workers residing within a cluster or an *ad hoc* distributed computing environment such as a pool of workstations (Figure 3.4).

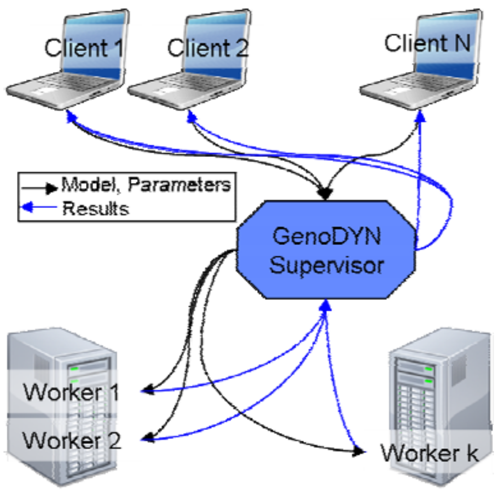


Figure 3.4. Distributed computing architecture

The calculation supervisor and the calculation worker use the same binary, which is independent of the GenoDYN client. The invocation of this binary determines which mode is used. The workers must be given the IP address of the supervisor and the GenoDYN client must be given the IP address of the supervisor. The supervisor does not need to be informed *a priori* of clients or workers. Both clients and workers are free to come and go and their IP addresses are discovered upon initial connection with the supervisor. As long as there is at least one worker, progress will always be made. One potential deployment strategy is to run workers with low priority on many desktop computers in a department or on a cluster. When there are available resources they can be utilized by those running GenoDYN. The supervisor and workers require no local access other than to write to an optional log, and can run without local permissions, thus improving security. Currently the supervisor and worker are only available on Linux, but this does not restrict the architecture of the GenoDYN client. Windows-based GenoDYN clients can connect to Linux-based supervisors.

To illustrate the potential benefit of this architecture we have dedicated a small cluster to GenoDYN. Any GenoDYN user can use the software remote simulation feature by pointing its client toward `xnode1.vbi.vt.edu` in the Remote CPU option of the **`Simulation>Options`** menu. This feature does not require a login account or any other special privileges. Note, however, that this experimental resource is currently limited to a small number of nodes and would not be able to support large simulation projects.

Model evaluation

Fitness function

GenoDYN supports the definition of fitness functions which are performance variables defined to evaluate a model. Two evaluation functions are built into GenoDYN: generic and scripted. The generic fitness function allows the definition of target values for any number of variables and different environments. The target values are entered using the **`Optimization>Fitness function>Options`** dialog box. Selecting the Generic fitness function activates the Generic Fitness Function Parameters table. A variable is first selected along with the target values for each environment defined for the model.

The fitness value of the generic fitness function is computed as the Euclidean distance between the target and simulated values of the selected variables at the last time point of the simulation and in each environment. The fitness value is shown in the status bar below the model canvas. The generic fitness function is a convenient way of testing the match of a model and a set of experimental measurements or desired behavior.

A more generic way of defining a performance function is to use a built-in scripting language to define arbitrary functions. The grammar of the scripting language has its basis in the C language with some extensions for conveniently accessing the model (Table 3.1). Within the scripting language one can change the rate constants, invoke simulations, and examine results. Essentially, the only part of the model that is not accessible from within the scripting language is the model topology. While the previously described fitness function could be implemented within this scripting language (Table 3.2), the dialog box may be more convenient for first time users.

Global variables holding fitness values (R/W):	`fitness objective`
Initial concentrations (R/W):	`initial`
Rate constant (R/W):	`kc`
Simulation results (RO):	`result`
Simulation invocation:	`simulate`
I/O:	`print`
Math functions:	`abs acos asin atan cos sin tan` `exp log sqrt sqr`
Variable types:	`float string int boolean void`
Constants:	`true false`
Control structures:	`break case continue default do` `else for if return switch while`
Comments:	`/*...*/ // ...`
Operators:	+ - * / = || && ^ ! < > % <= >= != == -- ++ <<

Table 3.1. Scripting language reserved keywords

When performing stochastic simulations fitness functions are currently applied to the first trajectory. In the future we plan to extend GenoDYN to allow the fitness functions to examine individual trajectories and statistical summaries of the ensemble of trajectories.

Optimization of model parameters

Once a performance or fitness function is defined, it is possible to examine the sensitivity of the network's performance to network parameters or to optimize the network parameters so as to maximize its performance (relative to the specified evaluation function). The built-in sensitivity analysis module of GenoDYN displays the values of the evaluation function as one or two parameters are varied over a range (Figure 3.5). Alternatively, network models constructed in GenoDYN can be exported to independent optimization packages. The authors have exported models to MATLAB, Mathematica, and Globsol [38, 39] as well as to C code referencing CVODE.

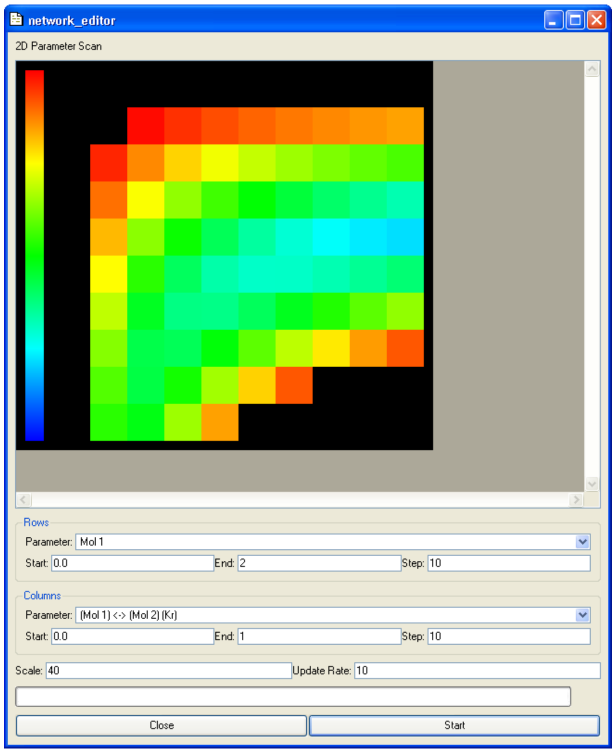
```
void main() {
  int i = 0;
  float res[5];
  float target[5];
  target[0] = .5;  target[1] = .5;
  target[2] = .5;  target[3] = .5;
  target[4] = .5;

  for(i=1; i<=5; i++) {
    kc("Rxn 1") = i/5.0;
    simulate();
    res[i-1] = result("Mol 1", 100);
  }

  fitness = 0;
  for(i=1; i<=n; i++) {
    fitness = fitness + sqr(target[i-1] - res[i-1]);
  }

  fitness = sqrt(fitness);
}
```

Table 3.2. Example of fitness function script

External to GenoDYN is an implementation of the hybrid genetic simplex algorithm (HGA), which allows specification of constraints on the allowed range of values for each parameter, or constraints that tie several model parameters to a single optimized parameter, thereby reducing the number of parameters to optimize. The HGA also extends the evaluation function to return both quantitative and qualitative scores, allowing one to identify networks that have the desired qualitative dynamics, but exhibit a poor fit with experimental data in quantitative terms and vice versa. As with all optimization routines, care must be taken to design an evaluation function that reflects the goals of optimization and helps direct the search for an optimal solution.

Examples

GenoDYN comes with an extensive library of examples that includes various models of the toggle switch [10], the Repressilator [11], and Guet's plasmid library [40]. We previously described an extensive analysis of the genetic properties of a model of the galactose switch pathway [41] which is in displayed in the screen shots of in Figure 3.6. The genetic analysis of this model illustrates well the flexibility of the GenoDYN modeling framework that allowed us to benefit from a sophisticated graphical user interface to develop the model. After the model had been defined, GenoDYN was integrated into an optimization environment specifically designed to conduct the genetic analysis described in this article.



Rows
Parameter: Mol 1
Start: 0.0 End: 2 Step: 10

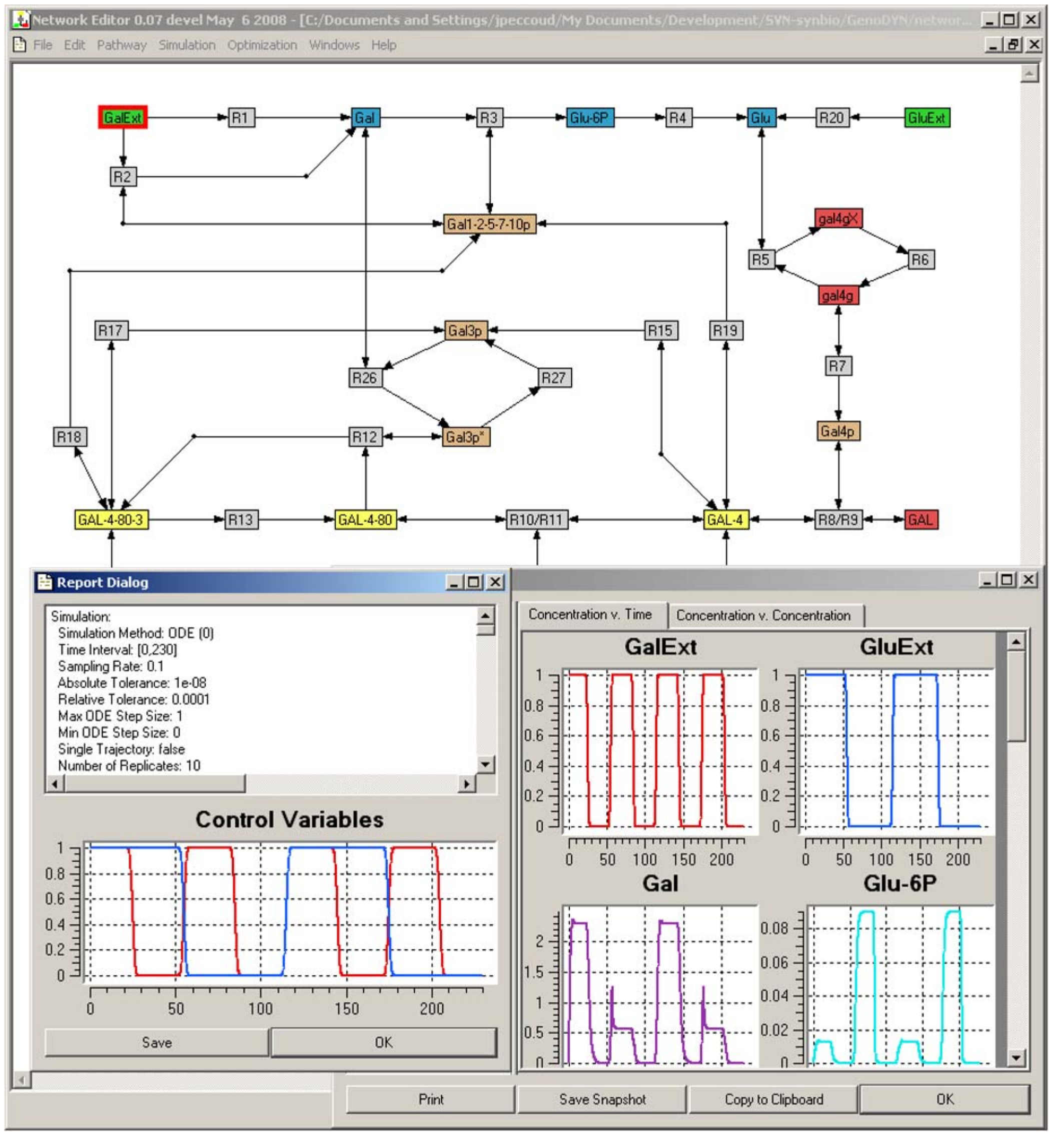
Columns
Parameter: (Mol 1) <-> (Mol 2) (Kr)
Start: 0.0 End: 1 Step: 10

Scale: 40 Update Rate: 10

Figure 3.5. Two-parameter sensitivity analysis

Conclusions

GenoDYN has been strongly inspired by our experience of using UltraSAN and later Möbius, two sophisticated modeling environments developed in the computer science community to analyze the performance of computer architectures [42, 43]. The notion of fitness function is a biological translation of the reward functions used in performance analysis. Similarly, the powerful hierarchical models used by Möbius led us to define functionally comparable model composition operators in GenoDYN.



Simulation:
Simulation Method: ODE (0)
Time Interval: [0,230]
Sampling Rate: 0.1
Absolute Tolerance: 1e-08
Relative Tolerance: 0.0001
Max ODE Step Size: 1
Min ODE Step Size: 0
Single Trajectory: false
Number of Replicates: 10

Figure 3.6. The main interface, along with the simulation results and report dialogs.

When analyzing the properties of artificial gene networks, the possibility of defining control variables and different physical environments greatly facilitates the evaluation of the network reaction to environmental inputs. We have also dedicated significant efforts to the implementation of advanced visualization functions that speed up the modeling cycle by allowing users to quickly understand the dynamics of the model they are building.

GenoDYN's distributed computing architecture may be one of its most innovative features. Its implementation is very portable and does not depend on specific middleware. Users are now one click away from a high performance computing environment allowing them to better analyze the stochastic dynamics of artificial gene networks. We are currently working to make GenoDYN available on the TeraGrid [44]. GenoDYN also provides a C++ framework for designing custom applications, many examples of which are included along with the GenoDYN source code. GenoDYN is therefore well positioned to be integrated in the design automation solution currently being developed for synthetic biology [45].

Acknowledgements

This work would not have been possible without the continued support of Peter Reilly and Vasant Honavar of Iowa State University and Lane Arthur and Chris Martin of Pioneer Hi-Bred. This work was partly funded by the Virginia Commonwealth Research Initiative.

Literature cited

1. Serrano, L., Benenson, Y., *et al*. 2005. Synthetic biology: Applying Engineering to biology. *Eur. Comm. NEST Rep.*
2. Endy, D. 2005. Foundations for engineering biology. *Nature* **438**:449-53.
3. Baker, D., Church, G., *et al*. 2006. Engineering life: building a fab for biology. *Sci. Am.* **294**:44-51.
4. de, L.V., Serrano, L., and Valencia, A. 2006. Synthetic Biology: challenges ahead. *Bioinformatics* **22**:127-8.
5. Arkin, A.P., and Fletcher, D.A. 2006. Fast, cheap and somewhat in control. *Genome Biol* **7**:114.
6. Chin, J.W. 2006. Programming and engineering biological networks. *Curr. Opin. Struct. Biol.* **16**:551-6.
7. Ball, P. 2007. Synthetic biology: Designs for life. *Nature*. **448**:32-33.
8. Sayut, D.J., Kambam, P.K., and Sun, L. 2007. Engineering and applications of genetic circuits. *Mol Biosyst* **3**:835-40.
9. Serrano, L. 2007. Synthetic biology: promises and challenges. *Mol. Syst. Biol.* **3**:158.
10. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in Escherichia coli. *Nature* **403**:339-42.
11. Elowitz, M.B., and Leibler, S. 2000. A synthetic oscillatory network of transcriptional regulators. *Nature* **403**:335-8.

12. Judd, E.M., Laub, M.T., and McAdams, H.H. 2000. Toggles and oscillators: new genetic circuit designs. *Bioessays* **22**:507-9.

13. Francois, P., and Hakim, V. 2004. Design of genetic networks with specified functions by evolution in silico. *Proc. Natl. Acad. Sci. U.S.A.* **101**:580-5.

14. Rodrigo, G., Carrera, J., and Jaramillo, A. 2007. Genetdes: automatic design of transcriptional networks. *Bioinformatics* **23**:1857-8.

15. Rodrigo, G., Carrera, J., and Jaramillo, A. 2008. Computational design and evolution of the oscillatory response under light-dark cycles. *Biochimie*. **90**:888-97.

16. Deckard, A., and Sauro, H.M. 2004. Preliminary studies on the in silico evolution of biochemical networks. *Chembiochem* **5**:1423-31.

17. Mason, J., Linsay, P.S., *et al*. 2004. Evolving complex dynamics in electronic models of genetic networks. *Chaos* **14**:707-15.

18. Goler, J.A., Bramlett, B.W., and Peccoud, J. 2008. Genetic design rising above the sequence. *Trends Biotechnol.* **26**:538-44.

19. Sauro, H.M. 2008. Modularity defined. *Mol. Syst. Biol.* **4**:166.

20. Hartwell, L.H., Hopfield, J.J., *et al*. 1999. From molecular to modular cell biology. *Nature* **402**:C47-C52.

21. Kobayashi, H., Kaern, M., *et al*. 2004. Programmable cells: interfacing natural and engineered gene networks. *Proc. Natl. Acad. Sci. U.S.A.***101**:8414-9.

22. Adalsteinsson, D., McMillen, D., and Elston, T.C. 2004. Biochemical Network Stochastic Simulator (BioNetS): software for stochastic modeling of biochemical networks. *BMC Bioinformatics* **5**:24.

23. Hoops, S., Sahle, S., *et al*. 2006. COPASI--a COmplex PAthway SImulator. *Bioinformatics* **22**:3067-74.

24. Sauro, H.M. 1993. SCAMP: a general-purpose simulator and metabolic control analysis program. *Comput. Appl. Biosci.* **9**:441-50.

25. Vallabhajosyula, R.R., and Sauro, H.M. 2007. Stochastic simulation GUI for biochemical networks. *Bioinformatics* **23**:1859-61.

26. Alves, R., Antunes, F., and Salvador, A. 2006. Tools for kinetic modeling of biochemical networks. *Nat. Biotechnol.* **24**:667-72.

27. Heinrich, R., and Schuster, S. 1996. *The regulation of cellular systems*. Chapman & Hall, New York.

28. Hucka, M., Finney, A., *et al*. 2003. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19**:524-31.

29. Erdi, P., and Toth, J. 1989. *Mathematical models of the chemical reaction*. Manchester University Press, Manchester, United Kingdom.

30. Gardiner, C.W. 1983. *Handbook of stochastic methods for physics, chemistry, and the natural sciences*. Springer-Verlag, Berlin Heidelberg New York.

31. Peccoud, J., and Ycart, B. 1995. Markovian modelling of gene products synthesis. *Theor. Popul. Biol.* **48**:222-34.

32. Cohen, S.D., and Hindmarsh, A.C. 1996. CVODE, a stiff/nonstiff ODE solver in C. *Comp. Phys.* **10**:138-43.

33. Gillespie, D.T. 1977. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**:2340-61.

34. Gillespie, D.T. 1977. Concerning the validity of the stochastic approach to chemical kinetics. *J. Stat. Phys.* **16**:311-8.

35. Griffith, M., Courtney, T., *et al*. 2006. Dynamic partitioning for hybrid simulation of the bistable HIV-1 transactivation network. *Bioinformatics* **22**:2782-9

36. Salis, H., and Kaznessis, Y. 2005. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *J. Chem. Phys.* **122**:54103.

37. Kaznessis, Y.N. 2006. Multi-scale models for gene network engineering. *Chem. Eng. Sci.* **61**:940-53.

38. Kearfott, R.B. 2008. GLOBSOL User Guide. *J. Global Optim.* **in press**.

39. Kearfott, R.B., Neher, M., *et al*. 2004. Libraries, tools, and interactive systems for verified computations four case studies. *Num. Soft. Res. Ver.* **2991**: 36-63.

40. Guet, C.C., Elowitz, M.B., *et al*. 2002. Combinatorial synthesis of genetic networks. *Science* **296**:1466-70.

41. Peccoud, J., Vander Velden, K.A., *et al*. 2004. The selective values of alleles in a molecular network model are context dependent. *Genetics* **166**:1715-25.

42. Peccoud, J., Courtney, T., and Sanders, W.H. 2007. Mobius: an integrated discrete-event modeling environment. *Bioinformatics* **23**:3412-4.

43. Goss, P.J.E., and Peccoud, J. 1998. Quantitative modeling of stochastic systems in molecular biology using stochastic Petrinets. *Proc. Natl. Acad. Sci. U.S.A.* **95**:6750-5.

44. Dong, S.C., Karniadakis, G.E., and Karonis, N.T. 2005. Cross-site computations on the TeraGrid. *Comp. Sci. Eng.* **7**:14-23.

45. Cai, Y., Hartnett, B., *et al*. 2007. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics* **23**:2760-7.

Chapter 4. Parameterization of a nonlinear genotype to phenotype map

A modified version of a paper published in *The Proceedings of the Pacific Symposium on Biocomputing, Kohala Coast, Hawaii, USA, January 4-8, 2005, pp.* 284-95

Jean Peccoud[1] and Kent A. Vander Velden[1]

[1]Pioneer Hi-Bred International, Inc., DuPont Agriculture & Nutrition, 7200 NW 62nd Avenue, Johnston, IA 50131, USA

Abstract

Mathematical models of networks of molecular interactions controlling the expression of traits could theoretically be used as genotype to phenotype (GP) maps. Such maps are nonlinear functions of the environment and the genotype. It is possible to use nonlinear least square minimization methods to fit a model to a set of phenotypic data, but the convergence of these methods is not automatic and may lead to a multiplicity of solutions. Both factors raise a number of questions with respect to using molecular networks as nonlinear maps. A method to fit a molecular network representing a bistable switch to various types of phenotypic data is introduced. This method relies on the identification of the model's stable steady states and the estimation of the proportion of cells in each of them. By using environmental perturbations, it is possible to collect time-series of phenotypic data resulting in a smooth objective function leading to a good estimate of the parameters used to generate the simulated phenotypes.

Introduction

Pharmacogenomics' ambition is to relate a phenotype, the effect of a drug, to the genotype of patients exposed to environmental conditions partly defined by the drugs they receive [1]. For a geneticist this project requires building a genotype to phenotype map (GP map) of drug effects. Mathematically, a GP map is a function f such as

$phenotype = \mathrm{f}(genotype, environment)$. It maps into a phenotypic space, the product of a genetic space generated by the genetic diversity in a population by the space of environmental conditions to which individuals of this population can be exposed [2]. The simplest GP map is the one upon which relies Mendelian genetics. The function is Boolean, indicating the presence or absence of a character. The environment is ignored and genes are considered independent of each other. Since most traits are quantitative and not binary, the genetics of quantitative traits relies on a more refined family of GP maps representing the phenotype as linear statistical models. In general multiple loci are assumed to contribute additively to the phenotype. In some cases terms representing digenic interactions are introduced. The effect of the environment on the phenotype is generally decomposed into an additive term and a genotype by environment term [3].

Just like complex interactions between multiple genetic loci generate a diversity of phenotypes for pathologies that were considered monogenic [4], responses to drugs are generally considered multigenic traits [5,6]. Many of the genetic determinants controlling the response to drugs have been identified by a candidate-gene approach relying on the understanding of the molecular mechanisms of the drug action and metabolism. Integrating into a mathematical model the network of molecular interactions affecting the response to a drug is therefore an attractive avenue to build the GP map.

Using different approaches, a number of authors have recently demonstrated that it is possible to build mathematical models to predict the phenotype controlled by small artificial gene networks [7-11], larger natural networks [12,13], or even genome-wide metabolic pathways [14,15]. In order to use a mathematical model as a GP map it is necessary to bridge the molecular- and population-levels views of the genotype-phenotype relationship. When using mass-action models of molecular interactions, it has proved possible to analyze the genetic properties of a molecular network by associating genetic polymorphism with discrete kinetic values of the parameter of each interaction [16]. The possibility of determining the kinetic parameters of each interaction is key to using molecular networks as GP maps.

One way to estimate the GP map parameters is to find a set of parameters minimizing the difference between the phenotype predicted by the model and the observed phenotype. Since the phenotype is a nonlinear function of the parameters, this problem can be addressed by using a nonlinear least-square approach [17,18]. Nonlinear minimization methods are iterative algorithms that require a set of starting parameter values to converge to a local solution. Different starting values can result in different solutions with different quality of fit. This limitation has the potential to prevent a unique determination of the map parameters. The topology of the molecular network model and the experimental design both contribute to shape the objective function being minimized. The number and geometry of its local minima determines the possibility to find and identify solutions corresponding to the actual parameters' values that generated the set of observed phenotypes. Since for many real molecular networks, it is not possible to explore the entire parameter space, it is possible that no starting parameter values will converge toward the actual parameter set. It is also possible that many starting values will result in many solutions with similar fits, making it impossible to distinguish the solutions closest to the actual parameter set. Few authors used nonlinear least-square minimization to estimate GP map parameters [19,20] and it is likely that a number of people attempted this without success and never published their negative results.

This paper introduces an algorithm to estimate the parameters of a molecular network from time-series of molecular phenotypes collected after an environmental perturbation. The objective function used takes into consideration the possibility that phenotypic data collected at the cell population level result from a random distribution of the cells among multiple stable-steady states. The presence of a positive feed-back loop [21] creates the possibility of multistationarity. Multiple steady states have been observed in artificial gene networks [22-26] but also in natural regulatory networks [27], for which this possibility had not been considered even recently [28].

The algorithm considered in this article is automatic and can be applied to virtually any mass action model of molecular networks without requiring any manual mathematical derivation.

Methods

Model

The model used in this article is a mass action equivalent of a model of a bi-stable switch [29-31]. In the list of reactions below, G_i and GX_i refer to the active and inactive forms of the ith gene coding for the protein P_i, respectively, while L_i represents the ith ligand and PX_i the ith protein complexed with its ligand.

$$\left.\begin{array}{l} R_1{:}G_1 \xrightarrow{k_1} G_1 + P_1 \\ R_2{:}G_2 \xrightarrow{k_2} G_2 + P_2 \end{array}\right\}\text{Gene expression}$$
$$\left.\begin{array}{l} R_3{:}P_1 \xrightarrow{k_3} \varnothing \\ R_4{:}P_2 \xrightarrow{k_4} \varnothing \end{array}\right\}\text{Protein degradation}$$
$$\left.\begin{array}{l} R_5,R_6 : G_2 + 2P_1 \underset{k_6}{\overset{k_5}{\rightleftharpoons}} GX_2 \\ R_7,R_8 : G_1 + 2P_2 \underset{k_8}{\overset{k_7}{\rightleftharpoons}} GX_1 \end{array}\right\}\text{Repression} \qquad (4.1)$$
$$\left.\begin{array}{l} R_9,R_{10} : P_1 + L_1 \underset{k_{10}}{\overset{k_9}{\rightleftharpoons}} PX_1 \\ R_{11},R_{12} : P_2 + L_2 \underset{k_{12}}{\overset{k_{11}}{\rightleftharpoons}} PX_2 \end{array}\right\}\text{Repressor-ligand interaction}$$

The time-evolution of the model is represented by mass-action differential equations. The set of coupled differential equations can automatically be derived from the chemical equations Equation (4.1)[32].

Mass conservation relationships can be used to eliminate some variables from the model. Assuming that there is only one copy of each of the two genes in the system, the first mass-conservation relation makes it possible to eliminate the repressed forms of the genes. We also assume that the interaction between the small molecules representing the environment and the repressors are much faster than the other reactions. Using a quasi-steady state approximation, we eliminate R_9 to R_{12} from the model. This results in the list of reaction rates below where $\mathbf{X}$ is the vector representing the state of the system and r_i the rate of the reaction R_i:

$$\text{with } \mathbf{X} = \begin{pmatrix} G_1 \\ P_1 \\ P_2 \\ G_2 \end{pmatrix}, \left\{ \begin{array}{ll} r_1(\mathbf{X}) = k_1 G_1 & r_5(\mathbf{X}) = k_5 G_2 \left[\dfrac{1}{1+L_1} P_1\right]^2 \\ r_2(\mathbf{X}) = k_2 G_2 & r_6(\mathbf{X}) = k_6 (1 - G_2) \\ r_3(\mathbf{X}) = k_3 P_1 & r_7(\mathbf{X}) = k_7 G_1 \left[\dfrac{1}{1+L_2} P_2\right]^2 \\ r_4(\mathbf{X}) = k_4 P_2 & r_8(\mathbf{X}) = k_8 (1 - G_1) \end{array} \right. \qquad (4.2)$$

The differential equation representing the time evolution of the system is derived from this list of reaction rates.

Numerical identification of the steady states

The most generic way of finding steady states is to find the solutions of Equation (4.3) below. The notation below indicates that the reaction rates depend on the parameterization of the model, $\mathbf{K} = (k_1, \ldots, k_8)$, and the environment, $\mathbf{E} = (L_1, L_2)$:

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{dt}} = \mathbf{F}(\mathbf{X}, \mathbf{E}, \mathbf{K}) = \begin{pmatrix} r_8(\mathbf{X}) - r_7(\mathbf{X}) \\ r_1(\mathbf{X}) - r_3(\mathbf{X}) + 2\left(r_6(\mathbf{X}) - r_5(\mathbf{X})\right) \\ r_2(\mathbf{X}) - r_4(\mathbf{X}) + 2\left(r_8(\mathbf{X}) - r_7(\mathbf{X})\right) \\ r_6(\mathbf{X}) - r_5(\mathbf{X}) \end{pmatrix} = 0 \qquad (4.3)$$

Roots can be determined by minimizing $\|\mathbf{F}(\mathbf{X})\|$ starting from any point in the model state space. Since Equation (4.3) is nonlinear, it is not possible to analytically find its solutions. In order to alleviate this limitation, a grid of starting points is created in a region of the state space expected to include all the biologically relevant steady states of the model.

Variables corresponding to conserved molecules are bounded by the initial conditions. Assuming that each gene in the model has a single copy, then $0 \le G_i \le 1$ with $i = 1, 2$. The asymptotic values of the non-conserved molecules, i.e. proteins in this case, is somewhere between 0 and $k_{production} / k_{degradation}$, the asymptotic value corresponding to the maximum expression of the gene. Therefore, in the case of the model considered here, all the steady states are expected to be within $V = [0,1] \mathrm{x} [0, k_1/k_3] \mathrm{x} [0, k_2/k_4] \mathrm{x} [0,1]$.

It is therefore possible to regularly sample *V* with a user-specified resolution. By starting the minimization algorithm from each point in this grid, a numerical solution to Equation (4.3) will generally be found for each starting point. Numerical errors and differences of convergence toward the same limits will result in minor numerical differences between solutions reached from different starting conditions. If the distance between a solution and another previously found solution is less than some specified value, it is assumed that they are identical.

After the scan of *V* is complete, the stability of the steady states is analyzed by computing, at the steady state, the eigenvalues of the Jacobian matrix associated with Equation (4.3). If the real parts of all eigenvalues are negative, then the steady state is stable.

Fitting to asymptotic phenotypes

In the context of this article, "asymptotic phenotypes" refers to phenotypic data collected in the stationary regime [33,34] in different environments E_j with $j = 1,...,\nu$. Since in general all variables of the model cannot be observed, the number of data points collected in each environment μ is less than *M*, the total number of state-variables of the model. It is convenient to represent asymptotic phenotypes as a $\mu \mathrm{X} \nu$ matrix $\mathbf{P}$. Now that the experimental data set is structured, it is necessary to generate a predicted phenotype $\mathbf{Q}(\mathbf{K})$ corresponding to a given set of parameters **K**. Assuming that it is possible to compute $\mathbf{Q}(\mathbf{K})$, then the least-square distance that needs to be minimized to fit the model to the phenotypes, $d(\mathbf{K},\mathbf{P})$, is:

$$d(\mathbf{K},\mathbf{P}) = \sum_{i=1}^{\mu}\sum_{j=1}^{\nu}\left[Q_i\left(K,E_j\right) - P_i\left(E_j\right)\right]^2 \qquad (4.4)$$

Computing the predicted phenotype for a specified environment and set of parameters is immediate if they result in a single stable steady state **S**. In this case:

$$\mathbf{S}_i\left(\mathbf{K},\mathbf{E}_j\right) = \mathbf{Q}_i\left(\mathbf{K},\mathbf{E}_j\right) \;\; i = 1,...,\mu \;\; j = 1,...,\nu \,. \qquad (4.5)$$

In conditions where the model has two stable steady states **S** and **T**, then the observed phenotype **P** is likely to result from a distribution of cells in the two steady states. So, instead of having a direct correspondence between the predicted phenotype and the observed phenotype, the predicted phenotype is a weighted average of the two stable steady states. What is not known, though, is the proportion of cells in each of the steady states. This proportion needs to be estimated by solving a linear constrained least-square problem:

$$\mathbf{Q}(\mathbf{K},\mathbf{E}) = \min_{\alpha\in[0,1]}\left(\alpha\mathbf{S}(\mathbf{K},\mathbf{E}) + (1-\alpha)\mathbf{T}(\mathbf{K},\mathbf{E}) - \mathbf{P}(\mathbf{E})\right) \;. \qquad (4.6)$$

This approach can be generalized to more than two stable steady states.

Fitting to a time series of phenotypes

Observing the model state variables at different points in time is a natural way of collecting data characterizing the model dynamics [35,36]. Many experimental designs can lead to this type of data. Only a single simple experiment is considered in this paper but it demonstrates that system multi-stationarity needs to be considered to properly analyze the data.

A cell population is placed in a first environment E_1 until it reaches a stationary regime indicated by the stabilization of the phenotype. An instantaneous perturbation is applied to the environment, creating a new environmental condition E_2. Phenotypic data are recorded at different time points while the population stabilizes toward a new stationary regime. For instance, cells can be grown in absence of ligands. One of the ligands is added to the growth medium creating a new environment. Samples of cell culture are taken and phenotyped at different points in time after the ligand has been added. This design can be generalized to multiple environmental perturbations. $E_{1,j}$ and $E_{2,j}$ refer to first and second environments of the j$^{\text{th}}$ perturbation. The first phenotype of each time series is collected in the stationary regime before the perturbation is applied. All other phenotypes are collected in the second environment and are indexed by the instant of observation. Similarly, it is necessary to compute a series of predicted phenotypes corresponding to the series of experimental data. The distance between the predicted and the observed phenotypes is computed by summing the distance over all time-points:

$$d(\mathbf{K},\mathbf{P})=\sum_{k=1}^{\tau}\sum_{i=1}^{\mu}\sum_{j=1}^{\nu}\left[\mathbf{Q}_i\left(\mathbf{K},\mathbf{E}_{2,j},t_k\right)-\mathbf{P}_i\left(\mathbf{E}_{2,j},t_k\right)\right]^2 \tag{4.7}$$

Let $G(\mathbf{X}_0,\mathbf{K},\mathbf{E},t)$ be the solution of Equation (4.3) starting from $\mathbf{X}_0$. Computing the predicted phenotype for a specified environmental perturbation and set of parameters is immediate if the initial environment and parameter set result in a single stable steady state $\mathbf{S}(\mathbf{K},\mathbf{E}_1)$. In this case the predicted phenotypes are extracted from the solution of Equation (4.3) starting at $\mathbf{S}(\mathbf{K},\mathbf{E}_1)$: $G(\mathbf{S}(\mathbf{K},\mathbf{E}_1),\mathbf{K},\mathbf{E}_2,t_k)=\mathbf{Q}_i\left(\mathbf{K},\mathbf{E}_{2,j},t_k\right)$. If the parameter set leads to two steady states in the initial environment $\mathbf{S}(\mathbf{K},\mathbf{E}_1)$ and $\mathbf{T}(\mathbf{K},\mathbf{E}_1)$, then it is possible to estimate the proportion $\hat{\alpha}$ just as in Equation (4.6). The predicted phenotype would then be a weighted average of trajectories starting from the two initial conditions **S** and **T**.

Application

The number of variables observed in the phenotype and the number of environments where the phenotypes are observed are likely to have a significant impact on the possibility to match the model with phenotypic data. So, phenotypic data were simulated in different numbers of environments and by recording different numbers of observed variables.

Twelve series of phenotypic data were generated using the same set of parameters. The first six phenotypes were asymptotic phenotypes. The second group of six phenotypes were time series.

In both cases (asymptotic and time series), three of the phenotypes consisted in the observation of one protein, P_1. In the remaining three phenotypes the values of both proteins were recorded in the phenotype.

The asymptotic phenotypes were simulated in three different numbers of environments (three, five, and nine environments). Environments are represented by the concentrations of the two ligands, (L_1, L_2). The first three environments were: (0,0), (10^1, 0), and (0, 10^1). In the five-environments experiments, (1, 0) and (0, 1) were added to the first 3 environments. In the nine-environments experiments (10^{-1}, 0), (10^{-2}, 0), (0, 10^{-1}), and (0, 10^{-2}) were added to the five previous environments.

The times series phenotypes are transitions between two environments. In the first experiment, the transition from (10, 0) to (0, 10) was simulated. In the two-transition experiment, the transition from (5, 0) to (0, 5) was added. In the three-transition experiment, the transition from (1, 0) to (0, 1) was added to the two previous transitions.

The same set of 25 initial parameter values was used to fit the model to the asymptotic and time-series phenotypes, resulting in a series of 300 optimizations.

Results

Numerical identification of steady states

The method to find the steady states of the model works well on this model. By using only the eight “corners” of V, it seems that all the steady states of the system were found. Increasing the resolution of the grid did not result in a larger list of steady states. Depending on the environment and parameter values, two types of regimes were found: a single stable steady state or two stable steady states and one unstable steady state.

In the least-square minimization procedures, the specificity of this network made it possible to use only two initial conditions $(0.5, k_1/k_3, 0, 0.5)$ and $(0.5, 0, k_2/k_4, 0.5)$ to find the stable steady states of the system. This simplification speeds up the optimization process that often requires hundreds or even thousands of steady state determinations. These two initial conditions do not allow the identification of the unstable steady states of the system and this approach may not be applicable to other models.

A bifurcation diagram was generated by computing the steady states (stable and unstable) of the model over a range of L_1 concentrations in order to verify the steady state identification procedure while the concentration of the second ligand was kept at zero. The system is bi-stable for low concentrations of L_1 and beyond a critical concentration, the system becomes mono-stable. This result is consistent with the bifurcation diagram of a similar model [37] and also with our own bifurcation analysis run in XPP/AUT [38]. The positions of the stable steady states are not very much affected by the concentration of L_1, except in the vicinity of the critical concentration. This indicates the robustness of the phenotype to environmental perturbation.

Fitting to asymptotic phenotypes

An exploration of the neighborhood of the original set of parameters used to generate the phenotypes indicated that initial conditions very close to the original parameter set could not lead to a good fit (data not shown). This indicated that the objective function was rough and may be difficult to minimize. It turned out that convergence was much easier to achieve than initially anticipated. When the phenotype included the two protein concentrations a good fit was achieved for ⅓ of the initial conditions.

This can be explained by observing that an infinite number of parameterizations have the same steady states. Solutions of Equation (4.3) verify Equation (4.8). The minimization problem defined by asymptotic phenotypes is unidentifiable. It is not possible to estimate the eight kinetic parameters but only the four equilibrium constants.

$$
\begin{aligned}
r_8(\mathbf{X}) - r_7(\mathbf{X}) &= k_8(1-G_1) - k_7 G_1 \left[\frac{1}{1+L_2} P_2\right]^2 = 0 \\
r_1(\mathbf{X}) - r_3(\mathbf{X}) &= k_1 G_1 - k_3 P_1 = 0 \\
r_2(\mathbf{X}) - r_4(\mathbf{X}) &= k_2 G_2 - k_4 P = 0 \\
r_6(\mathbf{X}) - r_5(\mathbf{X}) &= k_6(1-G_2) - k_5 G_2 \left[\frac{1}{1+L_1} P_1\right]^2 = 0
\end{aligned}
\tag{4.8}
$$

Fitting to time series of phenotypes

The convergence criteria used in this case was a root mean square of residuals less than 10^{-1}. Using this criterion, 14 convergences were observed (9% of the 150 optimizations using time series phenotypes) that can be broken down into 13% of convergence when only one protein is observed and 5% when both proteins are recorded. These rates of convergence need to be confirmed by analyzing a larger number of initial conditions using a faster implementation of this algorithm. However, they are surprisingly high and indicative of a relatively smooth performance function.

All optimization solutions were indexed (not shown) for further analysis. In some cases very similar solutions were found. For instance solution 13 is very close to solution 14 and solution 11 is very close to solution 12. It is worth observing that if solutions 11, 13, and 14 all originated from the same initial condition, solution 12 was found using a different initial condition. Also solutions 11 and 12 are not very far in the parameter space from solutions 13 and 14. Solution six is also located in the same area. Interestingly, these five solutions are all very close to the original set of parameters used to generate the phenotype. The solutions were verified by plotting the time course of the two protein concentrations and the profiles are consistent with the objective function used to generate the solutions. Protein concentrations corresponding to solution 11 were plotted over a wide range of initial conditions. Visually they are indistinguishable from the plots generated by the original set of parameters (Figure 4.1).

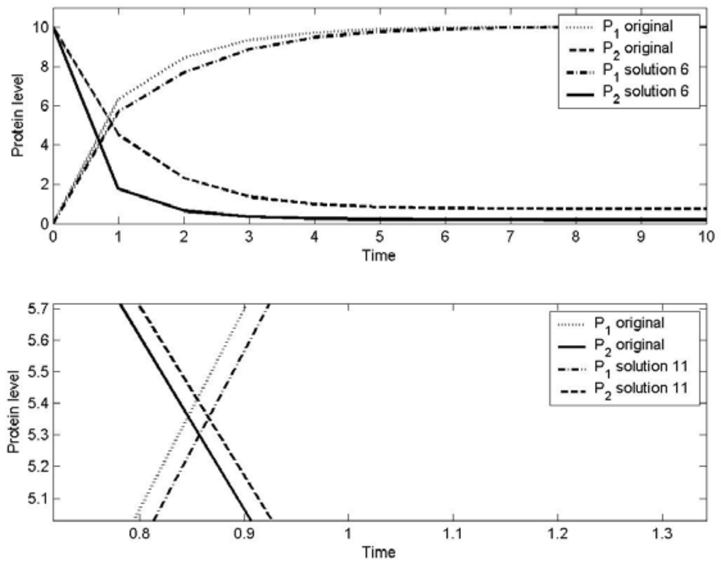


Figure 4.1: In order to visually assess the quality of the fit, the ODE was integrated using two solutions of the time-series optimization experiment and the original set of parameters used to generate the simulated phenotypes. The initial condition for the integration was set to (1, 0, 10, 0) and the environment to (0, 10). Solution six (top) was found when only one protein level was used in the phenotype. It is interesting to see that the fit for P_1 is better than the fit for P_2. The RMS computed using the two protein concentrations at the 11 time points is 0.83. Solution 11 (bottom) gives a very good fit of both of the protein expression profiles leading to a RMS of 0.06. It is necessary to zoom in on specific region of the plot to be able to visually distinguish the trajectories generated by the original parameter set and the trajectories generated by the parameters of Solution 11.

Discussion

Results

Even though this work focuses on a single molecular network model, results presented here are likely to be relevant to other models.

- The specific structure of molecular networks makes it possible to search for steady states in a limited volume of the model state space.
- The possibility of multi-stability should always be considered. In a population of cells observed in a stationary regime, cells can be randomly distributed between multiple steady states. Therefore, the measurement of a gene activity at the cell population level is a weighted average of the molecule concentrations

corresponding to the different stable steady states of the model. For a given set of parameter values, different repartitions of the cells in the different steady states lead to different qualities of fit between the model parameterization and the observed phenotypes. In the context of this paper, a linear minimization step was introduced to find the repartition minimizing the distance between the model and the experimental data.

- Asymptotic phenotypic data can only lead to the determination of the equilibrium constants but not the kinetic constants.
- Environmental perturbations can be used to collect time-series of phenotypic data. The relaxation profile observed is a weighted average of trajectories originating from the different stable steady states in the first environment.

Necessary improvements of the algorithm

In order for this method to be used for routine analysis it will be necessary to address a few issues.

- The steady state finding algorithm needs to be systematically validated. In some cases very stiff parameter sets hampered the convergence of the steady state identification procedure. The reasons for this behavior need to be understood. Since the steady state identification algorithm is the bottleneck of the whole optimization process it is worth trying to improve it.
- Determining the stability of the steady states is also an important step of the algorithm. Numerical errors prevent an accurate determination of the stability in the vicinity of critical points. It is not clear what is the impact of this issue on the outcome of the minimization process. Limit cycles are not considered in this algorithm.
- The local optimization method described in this paper needs to be coupled to a global search strategy to explore the parameter space more systematically.
- In cases where the time of sampling cannot be controlled, it could be necessary to take the actual sampling time into consideration when fitting the model to the data.

- A random term representing the measurement error needs to be added to the phenotypic data. The effect of this term on the convergence of the least-square minimization should be characterized. The addition of an error term would transform the least-square minimization problem into a nonlinear regression problem that could lead to computing confidence intervals for the parameter estimates.

Research directions

We are working on a generalization of this algorithm to handle phenotypic data collected on a multiplicity of genotypes just like several environmental conditions have been considered in this paper. Along the same line, the current model assumes only one copy of each gene. Introducing a diploid genome with two homologous copies of each gene would require predicting the phenotype of heterozygous individuals, which requires developing a model of dominance at the parameter level. If only homozygous individuals are considered or a total dominance is assumed, the model would remain unchanged.

Geneticists have been building models of the genotype to phenotype relationship for traits of other organisms for more than a century. By deciphering networks of molecular interactions, they hope to be able to build nonlinear GP maps inspired by the mechanisms controlling the expression of complex traits. It is expected that these maps would capture epistatic interactions between the genetic determinants contributing to these traits. Such a map would help a breeder to define more effective breeding strategies using molecular markers to manipulate alleles of genes contributing to trait variations or using transgene to introduce new sources of genetic variation, help a human geneticist to better understand how multiple genes can contribute to the development of a pathology, and help pharmacogeneticists to customize a medication to the genotype of their patients. Mathematical methods, such as those described here, are needed to analyze molecular data. The next challenge may be to find ways of associating macroscopic phenotypes such as a patient response to a treatment, with the molecular data we collect and analyze.

Acknowledgements

We are grateful to Mark Cooper, Chris Winkler, Dean Podlich, David Bickel, Bard Ermentrout and four anonymous reviewers for valuable comments and suggestions. This work would not have been possible without the support of Lane Arthur and Bob Merrill.

Literature cited

1. Evans, W.E. and Relling, M.V. 2004. Moving towards individualized medicine with pharmacogenomics. *Nature* **429**:464-8.
2. Cooper, M., Chapman, S.C., *et al*. 2002. The GP Problem: quantifying gene to phenotype relationships. *In Silico Biol.* **2**:151-64.
3. Falconer, D.S. and MacKay, T.F.C. 1996. Quantitative Genetics. Longman Group Ltd., Harlow, United Kingdom.
4. Weatherall, D.J. 2001. Phenotype-genotype relationships in monogenic disease: lessons from the thalassaemias. *Nat. Rev. Genet.* **2**:245-55.
5. Evans, W.E. and Relling, M.V. 2004. Moving towards individualized medicine with pharmacogenomics. *Nature* **429**:464-8.
6. Evans, W.E. and McLeod, H.L. 2003. Pharmacogenomics--drug disposition, drug targets, and side effects. *N. Engl. Med.* **348**:538-49.
7. Hasty, J., McMillen, D., and Collins, J.J. 2002. Engineered gene circuits. *Nature* **420**:224-30.
8. Kaern, M., Blake, W.J., and Collins, J.J. 2003. The engineering of gene regulatory networks. *Annu. Rev. Biomed. Eng.* **5**:179-206.
9. Atkinson, M.R., Savageau, M.A., *et al*. 2003. Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in *Escherichia coli*. *Cell* **113**:597-607.
10. Kramer, B.P., Viretta, A.U., *et al*. 2004. An engineered epigenetic transgene switch in mammalian cells. *Nat. Biotechnol.* **22**:867-70.
11. Isaacs, F.J., Hasty, J., *et al*. 2003. Prediction and measurement of an autoregulatory genetic module. *Proc. Natl. Acad. Sci. U.S.A.* **100**:7714-9.
12. Ozbudak, E.M., Thattai, M., *et al*. 2004. Multistability in the lactose utilization network of *Escherichia coli*. *Nature* **427**:737-40.

13. Setty, Y., Mayo, A.E., *et al*. 2003. Detailed map of a cis-regulatory input function. *Proc. Natl. Acad. Sci. U.S.A.*. **100**:7702-7.

14. Covert, M.W., Knight, E.M., *et al*. 2004. Integrating high-throughput and computational data elucidates bacterial networks. *Nature* **429**:92-6.

15. Patil, K.R., Akesson, M., and Nielsen, J. 2004. Use of genome-scale microbial models for metabolic engineering. *Curr. Opin. Biotechnol.* **15**:64-9.

16. Peccoud, J., Vander Velden, K.A., *et al*. 2004. The Selective Values of Alleles in a Molecular Network Model Are Context Dependent. *Genetics* **166**:1715-25.

17. Bates, D.M. and Watts, D.G. 1988. Nonlinear regression analysis and its applications. John Wiley & Sons, New York.

18. Dennis, J.E. and Schnabel, R.B. 1983. Numerical methods for unconstrained optimization and nonlinear equations. Prentice Hall, Inc., Englewood Cliffs, NJ.

19. Setty, Y., Mayo, A.E., *et al*. 2003. Detailed map of a cis-regulatory input function. *Proc. Natl. Acad. Sci. U.S.A.* **100**:7702-7.

20. Ronen, M., Rosenberg, R., *et al*. 2002. Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. *Proc. Natl. Acad. Sci. U.S.A.* **99**:10555-60.

21. Thieffry, D. and Thomas, R. 1998. Qualitative analysis of gene networks. *Pac. Symp. Biocomput.* 77-88.

22. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**:339-42.

23. Kramer, B.P., Viretta, A.U., *et al*. 2004. An engineered epigenetic transgene switch in mammalian cells. *Nat. Biotechnol.* **22**:867-70.

24. Atkinson, M.R., Savageau, M.A., *et al*. 2003. Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in *Escherichia coli*. *Cell* **113**:597-607.

25. Tchuraev, R.N., Stupak, I.V., *et al*. 2000. Epigenes: design and construction of new hereditary units. *FEBS Lett.* **486**:200-2.

26. Isaacs, F.J., Hasty, J., *et al*. 2003. Prediction and measurement of an autoregulatory genetic module. *Proc. Natl. Acad. Sci. U.S.A.* **100**:7714-9.

27. Ozbudak, E.M., Thattai, M., *et al*. 2004. Multistability in the lactose utilization network of *Escherichia coli*. *Nature* **427**:737-40.

28. Setty, Y., Mayo, A.E., *et al*. 2003. Detailed map of a cis-regulatory input function. *Proc. Natl. Acad. Sci. U.S.A.* **100**:7702-7.

29. Kramer, B.P., Viretta, A.U., *et al*. 2004. An engineered epigenetic transgene switch in mammalian cells. *Nat. Biotechnol.* **22**:867-70

30. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**:339-42.

31. Tchuraev, R.N., Stupak, I.V., *et al*. 2000. Epigenes: design and construction of new hereditary units. *FEBS Lett.* **486**:200-2.

32. Peccoud, J., Velden Velden, K.A., *et al*. 2004. The selective values of alleles in a molecular network model are context dependent. *Genetics* **166**:1715-25.

33. Setty, Y., Mayo, A.E., *et al*. 2003. Detailed map of a cis-regulatory input function. *Proc. Natl. Acad. Sci. U.S.A.* **100**:7702-7.

34. Guet, C.C., Elowitz, M.B., *et al*. 2002. Combinatorial synthesis of genetic networks. *Science* **296**:1466-70.

35. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**:339-42.

36. Kramer, B.P., Viretta, A.U., *et al*. 2004. An engineered epigenetic transgene switch in mammalian cells. *Nat. Biotechnol.* **22**:867-70

37. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**:339-42.

38. Ermentrout, B.G. 2002. Simulating, Analyzing, and Animating Dynamical Systems: A Guide to Xppaut for Researchers and Students. Society for Industrial Mathematics. Philadelphia.

Chapter 5. Values of alleles in a molecular network model are context dependent

A modified version of a paper published in the journal *Genetics*, Vol. 166, 1715-1725, April 2004

Jean Peccoud[1], Kent A. Vander Velden[1], Dean Podlich[1], Chris Winkler[1], Lane Arthur[1], and Mark Cooper[1]

[1]Pioneer Hi-Bred International, Inc., DuPont Agriculture & Nutrition, 7200 NW 62nd Avenue, Johnston, IA 50131, USA

Abstract

Classical quantitative genetics has applied linear modeling to the problem of mapping genotypic to phenotypic variation. Much of this theory was developed prior to availability of molecular biology. The current understanding of the mechanisms of gene expression indicates the importance of non-linear effects resulting from gene interactions. We provide a bridge between genetics and gene network theories by relating key concepts from quantitative genetics to the parameters, variables, and performance functions of genetic networks. We illustrate this methodology by simulating the genetic switch controlling the galactose metabolism in yeast and its response to selection for a population of individuals. Results indicate that genes have heterogeneous contributions to phenotypes and that additive and non-additive effects are context dependent. Early cycles of selection suggest strong additive effects attributed to some genes. Later cycles suggest the presence of strong context dependent non-additive effects that are conditional on the outcomes of earlier selection cycles. A single favorable allele cannot be consistently identified for most loci. These results highlight the complications that can arise with the presence of non-linear effects associated with genes acting in networks when selection is conducted on a population of individuals segregating for the genes contributing to the network.

Introduction

Recently there has been interest in interpreting the quantitative genetic properties of gene networks at the population level [1,2]. This is warranted on at least three grounds: (i) much of the molecular genetic evidence points to the roles of genes in non-linear networks in the determination of gene-to-phenotype relationships, (ii) we have a growing body of data on the structural and functional properties of the genomes of organisms and as this pool of data continues to expand it is becoming more feasible to construct models of gene networks, and (iii) for many aspects of basic and applied genetics it is necessary to study the properties of allelic variation for genes at the level of phenotypic effects and variation within populations. Bridging the molecular and population level views of gene-to-phenotype relationships is a challenging area of research for quantitative genetics. At present there is no agreed upon quantitative framework but a number of approaches are being investigated. We constructed a model of the gene network controlling the galactose metabolism pathway in yeast using differential equations. This model has been used as a genotype to phenotype map with which to evaluate the performance of individuals in simulations of a mass selection process. Combining these two approaches makes it possible to analyze the epistatic interactions between the genes controlling this pathway and their impact on the selection process.

Fundamental to genetics is the relationship between the genotype of an individual, the environment where it lives, and its resulting phenotype. This relationship is often referred to as genotype to phenotype (GP) mapping. Since the true mechanisms of gene expression have historically been poorly understood, geneticists have derived such mappings from the joint distributions of genotypic and phenotypic data. The simplest mapping, Mendelian genetics, considers traits that are completely determined by individual genes. Many traits, however, are more complex than that; they are quantitative in nature and are influenced by contributions from alleles at multiple loci. These multiple gene cases have been studied using linear statistical models that allow both additive and non-additive (dominance and epistasis) effects [3]. Complex traits are also often dependent on the environment in which a genotype is expressed. In addition to the direct effect of the environment, genotype by environment (GxE) interactions can have important effects on complex traits. Traditionally, genotype to phenotype mappings have predominantly been linear combinations of terms representing dominance, epistasis, GxE interactions, and genotype by genotype (GxG) interactions [4].

Even though these linear statistical relationships allowed geneticists to represent the phenotypic variability of a large number of simple traits, working beyond the limitations of linear mappings is one of the main challenges faced by genetics today. Interactions between genes contribute to complex phenotypes in plants [5-8], mice [9], and microorganisms [10,11]. Genetic factors that contribute to many pathologies do not have any direct effect on the phenotypes that are essentially determined by GxE and GxG interactions [12,13]. These observations are interpreted as non-linear effects of gene interactions and are usually referred to collectively as epistatic effects [14].

De Jong has recently reviewed various families of models that have been used to represent genetic networks [15]. Considering the small copy number of the molecules involved in gene expression mechanisms, Markovian models [16,17] based on a stochastic version of the mass action law are an appealing representation of gene network dynamics. However, the cost of computing Monte-Carlo simulations limits their use to only those pathways having a well-documented stochastic outcome at the cellular level [18,19]. Approximating the network dynamics by a system of differential equations provides a useful compromise between a realistic representation, speed of simulation, and a wealth of theoretical properties and analysis techniques that can complement numerical simulations.

Materials and methods

Modeling the galactose genetic switch: The galactose pathway is an attractive system for dynamic modeling since it integrates a gene network, a metabolic pathway, and a response to environmental perturbations. In a first approximation, it is possible to associate the phenotype to the activity of the metabolic pathway and the genotype to the genes in this pathway. Our model of the GAL system (Figure 5.1 and Table 5.1) is a simplistic representation of the complex mechanisms of gene expression. It is representative of the common understanding of the molecular mechanisms responsible for the response of yeast to the presence of galactose and glucose in its environment.

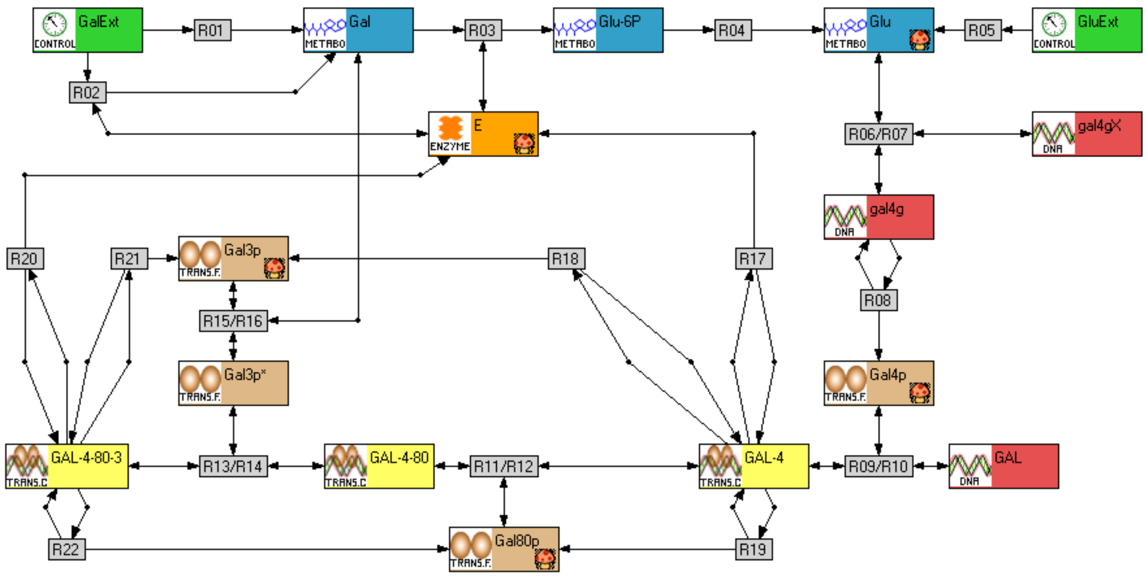


Figure 5.1: Diagrammatic representation of the Galactose switch

Recent overviews of the GAL switch have been provided by Ideker [46] and Ostergaard [47]. GalExt and GluExt are the two environmental variables of the system. Galactose is transported into the cell primarily by Gal2p using an ATP-dependent mechanism. It is necessary to take into consideration a small passive diffusion of galactose into the cell to trigger the induction of the GAL genes by galactose. Although there are a number of well-characterized metabolites between galactose and glucose 6-phosphate, we represent the whole pathway by a single step catalyzed by a hypothetical enzyme labeled E. Since the glucose-6-phosphatase catalyzing the transformation of Glu-6P into glucose is not part of the GAL network, it was omitted from the model. The gene coding for Gal4p, gal4g, can be in a repressed form gal4gX when complexed by Mig1 in the presence of glucose [48,49]. For simplicity we considered a single enzyme in the pathway coded by a single gene noted GAL. The expression of GAL is induced by Gal4p. When in the induced state GAL-4, it expresses the E enzyme along with the Gal3p and Gal80p transcription factors. Gal80p represses this expression by binding to the GAL-4 complex. Gal3p is the galactose sensor of the GAL system. Galactose binds Gal3p through an ATP-dependent mechanism. The resulting complex Gal3p* binds to the GAL-4-80 complex and induces the expression of the GAL genes[50].

	Reaction Equation	**A1A1**	**A1A2**	**A2A2**
D01	Gal80p $\rightarrow$ 0	**30**	50	70
D02	Gal4p $\rightarrow$ 0	16	36	**56**
D03	Gal3p $\rightarrow$ 0	22	40	**58**
D04	*Glu $\rightarrow$ 0*	*50*	*50*	***50***
D05	E $\rightarrow$ 0	66	82	**98**
R01	*GalExt $\rightarrow$ Gal*	*1*	***1***	*1*
R02	E + GalExt $\rightarrow$ Gal + E	5	6	**7**
R03	E + Gal $\rightarrow$ Glu-6P + E	**5**	12	19
R04	*Glu-6P $\rightarrow$ Glu*	***100***	*100*	*100*
R05	*GluExt $\rightarrow$ Glu*	***10***	*10*	*10*
R06	Glu + gal4g $\rightarrow$ gal4gX	7	10	**13**
R07	Glu + gal4g $\leftarrow$ gal4gX	**1**	2	3
R08	gal4g $\rightarrow$ gal4g + Gal4p	4	23	**42**
R09	GAL + Gal4p $\rightarrow$ GAL-4	**3**	7	11
R10	GAL + Gal4p $\leftarrow$ GAL-4	8	9	**10**
R11	GAL-4 + Gal80p $\rightarrow$ GAL-4-80	2	3.5	**5**
R12	GAL-4 + Gal80p $\leftarrow$ GAL-4-80	**3**	5	7
R13	Gal3p* + GAL-4-80 $\rightarrow$ GAL-4-80-3	6	8	**10**
R14	Gal3p* + GAL-4-80 $\leftarrow$ GAL-4-80-3	1	10	**19**
R15	Gal + Gal3p $\rightarrow$ Gal3p*	1194	**1320**	1446
R16	Gal + Gal3p $\leftarrow$ Gal3p*	700	809	**918**
R17	GAL-4 $\rightarrow$ GAL-4 + E	**10**	19	28
R18	GAL-4 $\rightarrow$ GAL-4 + Gal3p	1	2	**3**
R19	GAL-4 $\rightarrow$ GAL-4 + Gal80p	15	**101**	187
R20	GAL-4-80-3 $\rightarrow$ GAL-4-80-3 + E	**330**	336	342
R21	GAL-4-80-3 $\rightarrow$ GAL-4-80-3 + Gal3p	178	309	**440**
R22	GAL-4-80-3 $\rightarrow$ GAL-4-80-3 + Gal80p	294	**338**	382

Table 5.1: Chemical equations and parameters

Reactions are labeled in the first column. The chemical equation of the reaction is given in column 2. Each parameter has two allelic values A1 and A2. The columns A1A1, A1A2, and A2A2 indicate the parameter values used when genotypes are homozygous (A1A1 and A2A2) or heterozygous (A1A2). Parameters in bold characters indicate the genotype of the individual with the highest performance that was generated at the 35th generation of the 34th run. Parameters highlighted by a gray background correspond to the favorable alleles that were consistently fixed in more than 95% of the 1,000 runs. Lines (Reactions) in italic are non-segregating in Experiment 1 and Experiment 2 because they correspond to interactions outside of the GAL system. Parameter values highlighted in gray were made non-segregating in Experiment 2.

Biology: To date, the effect of the environment has often been ignored in models of gene networks. Alternatively, it is possible to consider the environment as a set of external parameters, where simulation runs with various parameter values can be compared to evaluate the impact of the environment on the model dynamics [20]. For many situations it seems that this approach is able to capture the biological logic of the network. In the case of the galactose pathway of yeast, the environment can change the state of the genetic switch by inducing or repressing the expression of the GAL genes. However, the relationship between the network and its environment is not one-way. The induction of the GAL genes by galactose results in the transformation of galactose into glucose. This transformation introduces a feedback loop by which the induced state of the GAL system leads to a modification of the environmental conditions that lead to this induction. In an effort to capture this behavior, we introduced in the model GalExt and GluExt, which can be regarded as external pools of molecules not affected by the dynamics of intracellular reactions. Passive diffusion or active transport of these molecules into the cell can be represented by chemical equations transforming these molecules into their intracellular counterparts, Gal and Glu, respectively (reactions R01, R02, and R05 in Table 5.1). Gal and Glu can be regarded as the variables indicative of the intracellular environment. The value of the two control variables GalExt and GluExt indicate the presence of sugars in the environment. Absence and presence were indicated by 0 and 10, respectively. The combination of GalExt and GluExt values defines an environment.

Dynamics: The time-evolution of the model is represented by mass-action differential equations. The set of coupled differential equations can automatically be derived from the chemical equations of Table 5.1 [21]. Specifically, the matrices of stoichiometric coefficients for the reactants $\alpha_{i,r}$ and products $\beta_{i,r}$ of the reactions can be used to represent the generic form of a chemical equation:

$$R_r = \sum_{i=1}^{M} \alpha_{i,r} X_i \xrightarrow{k_r} \sum_{i=1}^{M} \beta_{i,r} X_i \tag{4.1}$$

The rate v_r of each reaction depends on the concentration of its reactants:

$$v_r = k_r \prod_{i=1}^{M} [X_i]^{\alpha_{i,r}} \tag{4.2}$$

The time-evolution of a molecule concentration is ruled by the balance between the rates of the reactions producing this molecule and the ones using it as a reactant:

$$\frac{d[X_i]}{dt} = \sum_r \beta_{i,r} v_r - \sum_r \alpha_{i,r} v_r \tag{4.3}$$

The complete set of differential equations is given in the Appendix in MATLAB format.

Genotypes, phenotypes, and traits: In order to analyze the response of a gene network to selective pressure, it is necessary to establish a correspondence between the basic properties of genetics at a population level and the characteristics of genetic networks. Our analysis relies on the following:

Segregating loci as model parameters: The reaction rates are genetically determined. It is well established that directed mutations of promoters or protein domains can affect the rates of protein-DNA interactions, protein-protein interactions, gene expression, or even affect the catalytic properties of an enzyme. Hence, each parameter is determined by a number of segregating loci. The precise mapping of the genetic space onto the parameter space depends on the number of genes involved (*N*) and the extent of genetic polymorphism. In the case of a bimolecular reaction like R09 (Table 5.1, Figure 5.1), the rate of the binding of the Gal4p protein on the GAL promoter can be determined by the sequence coding for Gal4p and by the regulatory sequence of GAL. Potentially, two loci could determine the rate of this reaction but if only one of them is polymorphic, it is not necessary to consider in the model the locus corresponding to the conserved sequence. In the context of this paper, a single locus was associated with each parameter (*i.e.* *N*=27).

Alleles as discrete parameter values: The association between loci and parameters makes it natural to associate allelic polymorphism with variation of specific parameter values. Each polymorphic locus is assumed to have two alleles in this paper (larger numbers can be considered). A null allele translates into a zero value of the corresponding parameter. Alleles having a less dramatic effect result in parameters having an x-fold higher or smaller value than the wild-type. The within locus parameter values are assumed to be additive so that the heterozygous genotype is given the average parameter value of the homozygous genotypes for the two alleles. Different levels of dominance at the individual parameter level can be allowed but are not considered here.

In the context of this article, we do not consider the possibility of introducing mutations. The genetic space is thus finite and discrete. Its 3^N genotypes are the 3^N parameter combinations resulting from the selection of one of the three possible parameter values (columns) in each of the 27 lines of Table 5.1.

Phenotypes as vectors of traits: Traditionally the phenotype of an individual is defined by the value of the biometric data that can be measured at some point in time (e.g. grain yield of crops, the number of bristles on a segment of *Drosophila spp*). These biometric data rarely translate directly into molecular variables but they are indicative of the performance of the individual. In order to relate a model to experimental observations, it is necessary to derive trait values from the model itself.

Traits as functions of a model: The biometric data collected to score a trait are static, time-independent observations. Even though life is a dynamic process that develops in time, phenotypes are observed in standard conditions that remove time from the observation. Even traits tightly associated with the timing of development are considered static in genetics. The transition from vegetative growth to reproduction or flowering time provides a good illustration of this point. The whole developmental process is reduced to a single datum, the time of the transition to flowering. The genetic analysis of this trait relies primarily on this single observation of individuals in a population. Traits are a means to score the various characteristics of genotypes. In the case of the GAL system, the most obvious trait is the capability to process galactose when it is the only source of carbon available. How does this translate in the context of our model of the galactose switch? There are several possible interpretations of this trait. The variable representing the enzymes or the variables representing metabolites can be used as indicators. In this case we elected to use Glu-6P as an indicator of the state of the galactose pathway. In order to quantify the trait, we assigned target values for Glu-6P in the 3 environments (we ignore the trivial case where no sugar is present Gal-Glu-). Arbitrarily, we decided that Glu-6P should be 0 in the two environments where the pathway should not work (Gal-Glu+, Gal+Glu+) and 2 in Gal+Glu-. The system of differential equations was integrated between t=0 and t=10^4 where it is assumed to reach steady-state. By noting $X_{-+}\left(10^4\right)$ the value of Glu-6P at time 10^4 in the Glu+Gal- environment, this first trait is:

$$T_1\left(k_1,\ldots,k_{27}\right)=\sqrt{\left(X_{+-}\left(10^4\right)-2\right)^2+\left(X_{++}\left(10^4\right)-0\right)^2+\left(X_{-+}\left(10^4\right)-0\right)^2} \qquad (4.4)$$

A second trait was also defined for this model. Comparable levels of external galactose and glucose are expected to lead to comparable levels of internal glucose. By noting $Y_{-+}\left(10^4\right)$ the value of Glu at time 10^4 in the Gal-Glu+ environment, this second trait is:

$$T_2\left(k_1,\ldots,k_{27}\right)=\sqrt{\left(Y_{+-}\left(10^4\right)-2\right)^2+\left(Y_{++}\left(10^4\right)-2\right)^2+\left(Y_{-+}\left(10^4\right)-2\right)^2} \qquad (4.5)$$

A trait value can be computed for each of the genotypes of the genetic spaced considered in this article. So, for instance, $T_1\left(30,36,22,50,66,1,7,5,100,10,7,1,4,3,8,2,3,6,1,1194,700,10,1,15,330,178,294\right)$ is the trait value of the genotype where all loci are A1A1 expect D02 (A1A2) and R02 (A2A2)

Performance as a function of traits: A numerical performance function is computed for selection purpose. This summarizes results from a number of elementary traits that determine how well an individual performs in a given environment. There are multiple ways of combining several trait values in a performance function. In the context of this work, we considered:

$$\Phi\left(k_1,\ldots,k_{27}\right)=T_1\left(k_1,\ldots,k_{27}\right).T_2\left(k_1,\ldots,k_{27}\right) \qquad (4.6)$$

Simulation of selection: To simulate effects of selection operating on the model of the galactose pathway, we developed a simple genetic algorithm application that was interfaced with a gene network simulator utilizing CVODE [22]. For this article we have limited ourselves to a mass selection strategy where the phenotype of an individual is the only criterion used to evaluate the performance of a genotype.

The initial population (500 individuals) contained equal numbers of each allele at all segregating loci in the galactose pathway model. A constant selection pressure of 20% was applied to all cycles of selection across all simulations. We simulated a case where there was sustained directional selection for smaller values of the performance function over 100 cycles of selection. One thousand replicates of the simulation were conducted.

Results

Model: The model of the molecular network described in this paper has two specific features not commonly found in the literature on gene networks: (1) control variables are used to represent the dynamic interaction of the model with the environment; (2) trait and performance functions are defined to evaluate the performance of a model parameterization.

Control variables: For the sake of reproducibility, the simulations described in this paper do not take full advantage of the possible time evolution of control variables. Instead of assigning a constant value to environmental factors such as the sugar concentrations, it is possible to specify the variation of these concentrations in time. This feature makes it possible to evaluate other traits of the model. For instance, it is possible to quantify the ability of the network to react to changes of the environment. The trait functions described in this work do not distinguish the networks that will quickly adapt to new conditions from the ones that will need more time to turn the galactose switch ON and OFF. In models of other regulatory networks, control variables had also been used to represent the effect of physical parameters of the environment such as temperature, volume, or light.

Environment	**Glu-6P**	**Glu**
Gal-Glu+	0.000000	2.000000
Gal+Glu-	1.994018	3.988036
Gal+Glu+	0.011255	2.022515
Performance = 0.025341	**T_1 = 0.012746**	**T_2 = 1.988163**

Table 5.2: Performance computation

In order to illustrate how performance is computed, the performance of the best performing individual generated across the entire simulation is computed in this table. This individual was found in the 35th generation of the 34th run of the simulation. Simulations were run in the 3 different environments containing sugars and the value of Glu-6P and Glu at $t=10^4$ are reported in this table. The two traits can be derived from these data by using equations (4.4) and (4.5). The performance score is the product of the two trait values.

Performance function: In order to assess the way individuals are scored by the performance value, we looked for the individual with the lowest performance value that was generated across the entire experiment. This individual was found at the 35th cycle of run 34. It is interesting that this individual was not found in the population generated at the end of the selection process (cycle 100). The performance value of the best individual is approximately 0.025 (see Table 5.2). The values achieved at the end of the selection process are typically close to 0.20. This 8-fold difference tends to indicate that a dramatic loss of performance occurred during the selection process. That is when it becomes necessary to examine how these performance values are achieved, *i.e.* the property of the trait and performance functions used in the simulation. The values in Table 5.2 show that the target values for Glu-6P are reached in the three environmental conditions and T_1 can reach a very low value. This is not the case for T_2. The target values for Glu are reached in Gal-Glu+ and Gal+Glu+ conditions but we cannot get close to the target value of 2 in the Gal+Glu- condition. When the best individual is compared to the best individuals typically found in the last cycle of selection, it turns out that their behavior is very comparable. Minimal changes in Glu-6P values result in a significant difference in the T_1 value, which propagates to the performance value. Even more interesting is the examination of the time-evolution of Glu-6P and Glu when the molecular network is integrated. Asymptotic values are quickly reached in Gal-Glu+ and Gal+Glu- but the system oscillates when placed in Gal+Glu+ conditions. The amplitudes of the oscillations are significant (0.5 for Glu and 0.3 for Glu-6P) but the values are close to the target values at $t = 10^4$. This shows how dependant is the outcome of the selection process on the trait and performance functions.

Simulations: Running such an experiment is a significant computational challenge. There are very significant differences of simulation time between runs since some simulations can be achieved in 1.7 hours while others would take up to 4.4 hours on a processor running at 2.8 GHz. Most of the time is spent evaluating the performance of 50,000 individuals generated during 100 populations of 500 individuals. Since the model needs to be simulated in three different environmental conditions, the differential equations are integrated 150,000 times in each run. In order to speed up the genetic algorithm, previously evaluated phenotypes are recorded in a cache. Some simulations are likely to explore larger regions of the genetic space than others. Even though the total number of individuals evaluated is the same for all simulations, some will evaluate more gentoypes than others, which explains the differences of simulation time. In order to complete the 1,000 runs in an acceptable time (approximately 15 hours), the simulations were distributed over the 56 nodes of a Linux cluster, each node having two processors. The selection process simulated in this experiment is extremely basic. Its implementation did not require much programming. In order to use the analyze the response of regulatory networks to actual breeding programs, we also interfaced the molecular network simulation environment with QU-GENE, an environment for simulating breeding strategies [23,24].

Response to selection: There are two ways to analyze the network response to selection. The time-evolution of performance is indicative of the effect of selection while the time-evolution of allele frequencies tells us how this effect is achieved. Two experiments (series of 1,000 simulations with identical parameters and initial conditions) were conducted. In Experiment 1, the only non-segregating loci were those corresponding to interactions that are often considered "outside" the Galactose switch. In a second experiment, Experiment 2, we also fixed the favorable alleles of loci having an additive effect in the results of Experiment 1 (see Table 5.1 for details).

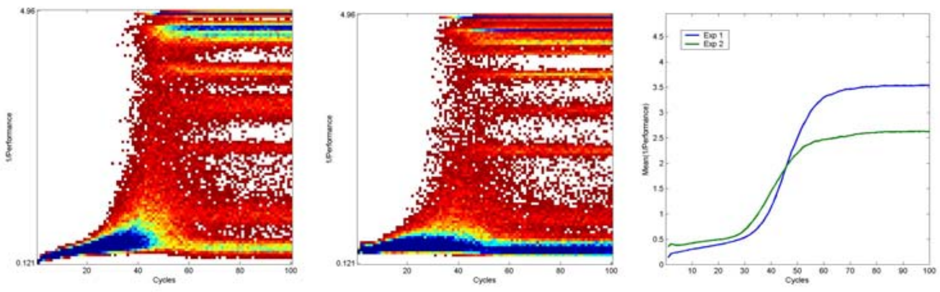


Figure 5.2: Time evolution of the population average performance distribution

Data were recorded during Experiment 1 and Experiment 2, each consisting of 1,000 simulations. Histograms of the inverse of the population average performance values were computed for each of the 100 cycles of selection and the frequency color-coded. Results from Experiment 1 (left) show that the distribution is clearly non-normal since it exhibits at least eight modes. Beyond cycle 80, the selection process has reached its asymptotic distribution. The distribution observed in Experiment 2 (center) is fairly similar to results of Experiment 1. The main difference is the weight of the bottom mode (blue peak) indicating that a large fraction of the simulations never achieved good performance values. In order to better compare these two distributions, the time evolutions of their mean values were plotted on the third graph (right). It shows that better performance is achieved in Experiment 2 (green line) for the early phases of the selection process. However, the long term response to selection in Experiment 2 is not as good as in Experiment 1.

The response to selection of this genetic system can be illustrated by graphing the evolution over cycles of selection of the mean performance value of the population. However, in the case of this experiment this graph did not appear the most appropriate. The best performers in our experiment have the lowest performance value. As a result the selection results in a reduction of performance values over time. The other problem is that the performance function has an absolute lower bound. So the plot of the mean performance values over cycles of selection is difficult to read since all the runs tend to accumulate toward 0. To overcome these difficulties, the statistical distribution of the inverse of the mean performance value was plotted (Figure 5.2).

Experiment 1: The statistical distribution of mean performance values is initially unimodal (Figure 5.2, left). Beyond cycle 50 or so up to eight modes can be identified. Interestingly, there is a mode corresponding to poor levels of performance. There are also two major modes corresponding to good performance and a few minor modes of intermediate values. Beyond cycle 50, the selection process appears to have reached its asymptotic distribution. However, the observation of individual trajectories indicates that despite a constant selection pressure, the populations can move from one mode to the other, resulting in quick gains or losses of performance even in the stationary regime. This pattern indicates that the performance landscape is complex with multiple local maxima and that the fluctuations of the selection process are large enough to move the population from one peak to the next.

Allele frequencies exhibit a fairly complex behavior at most loci (Figure 5.3, top). Fixation of one of the two alleles in more than 95% of the runs is observed for seven loci (D02, D05, R08, R14, R18, R19, R21). In the other cases the final allele frequencies are variable and are distributed between 0 and 1 with peaks at 0%, 50%, and 100%. Thus, either one of the homozygotes or the heterozygote could be favored depending on the replicate. So for most loci it is not possible to clearly identify a consistently favorable allele; the favorable allele is highly context dependent. Also, since a small percentage of the runs lead to retaining the heterozygous state, both alleles could be retained. Also included in Figure 5.3 is the time-evolution of the four parameters that are not polymorphic. They are the only ones exhibiting a random drift behavior. These loci can thus be considered as negative controls. All the polymorphic loci have some selective value in this experiment since none of them drift as the non-polymorphic ones do.

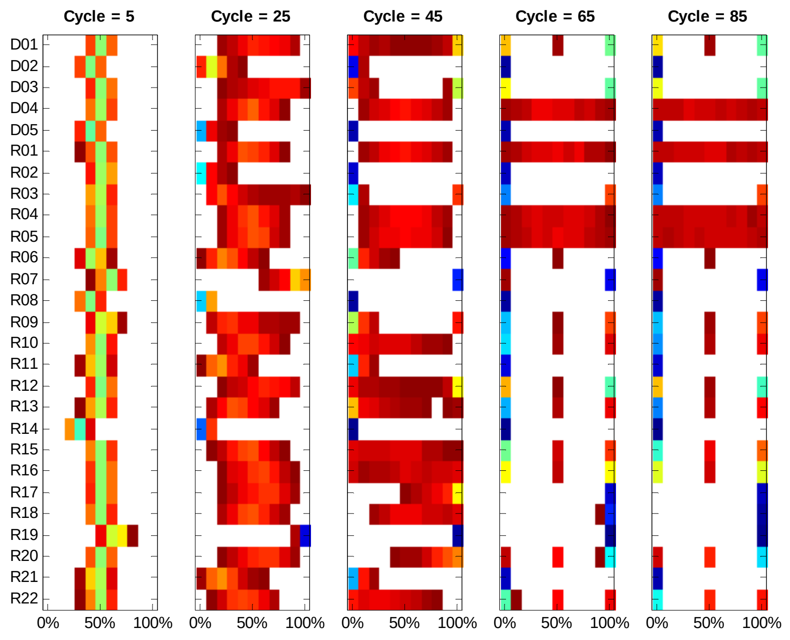


Figure 5.3: Evolution of alleles frequencies under selection

During the Monte-Carlo simulations corresponding to Figure 5.2, the frequencies of allele A1 at each of the 27 loci were recorded. Histograms of these frequencies were color-coded as in Figure 5.2. To illustrate the effect of the selection process on the genetic makeup of the population, five histograms corresponding to the selection cycles 5, 25, 45, 65, and 85 are displayed. In Experiment 1, D02, D05, R08, R14, R18, R19, R21 one of the two alleles is consistently fixed in more than 95% of the simulations. For most loci, however, no allele is fixed. Frequency distribution is multimodal with peaks at 0%, 100% and often 50%. Non-polymorphic loci (D04, R01, R04, and R05) exhibit a pattern indicative of genetic drift.

Experiment 2: In Experiment 2, the seven loci that had a favorable allele in Experiment 1 were fixed and thus made non-polymorphic (see Table 5.1) By fixing the favorable allele in the parameter file, it was anticipated that the transient phase of the selection process would be shortened. It turns out that the initial mean performance values are actually better (Figure 5.2, center) as anticipated. However, the asymptotic distributions are significantly different. The heavily loaded mode at the bottom of the plot indicates that a large fraction of the simulations never manage to achieve good levels of performance. This is confirmed by comparing the time evolution of the mean of these two distributions (Figure 5.2, left). The mean for Experiment 2 (green line) is initially higher than the mean of Experiment 1 (blue line). Since, initially, the performance response is slow, this results in almost a 10-cycle advantage provided by the fixation of favorable alleles. However, there is a long-time cost to this more limited genetic variability since the long-term response of Experiment 2 is not as good as in Experiment 1. The response of allele frequencies to selection is very similar in Experiment 1 and Experiment 2 even though some minor quantitative difference can be observed.

Future work will relate the peaks of the performance distribution (Figure 5.3) with the distributions of the allele frequencies. It appears that the context dependent combinations of alleles emphasized by the results of the different replicates of the selection process correspond to different peaks of performance on a moderately rugged landscape (data not shown).

Discussion

Molecular networks as GP maps: GP maps have traditionally been based on statistical models. In some cases we now have enough understanding of the molecular mechanisms to capture their dynamics into mathematical models. There are some indications in the recent literature that we now have models with some predictive power of the phenotype [25-28]. Analyzing the genetic properties of regulatory networks raises a number of theoretical and technical problems, which explains the limited numbers of articles dealing with this problem.

Non-linear GP maps: Introduction of non-linear terms in genotype to phenotype mappings leads to considerable theoretical difficulties that prevent any closed-form expression of the model properties. As suggested by Kempthorne [29], the development of software to simulate genotype-environment systems (e.g. plant breeding programs) has enabled geneticists to explore the genetic consequences of non-linear mappings *in silico* [24,30] without the need for an analytic result. The *E(NK)* framework provides a foundation for an *in silico* approach to genetic analysis of the properties of linear and non-linear gene-to-phenotype mappings at the individual and population levels [4]. It is specified as a generalization of Kauffman's *NK* gene network model [31], where a set of *N* genes are assumed to be under the control of, on average, *K* other genes in the network. The *E(NK)* framework incorporates GxE interactions through allowing a series of *NK* genotype to phenotype relationships corresponding to different environment types for a given target population of environment types. Here the target population of environments is defined as a mixture model of different environment-types. Within this generic modeling framework various types of genotype to phenotype mappings can be implemented [32,33]. So far we have examined a wide range of artificial gene networks, results from molecular map based genetic mapping of traits, and a combination of genetic analysis and crop growth models [32]. In this paper, we describe a way to build a genotype to phenotype map within the *E(NK)* framework that relies on our understanding of the molecular mechanisms of gene expression.

It is interesting to relate the results presented in this paper to previous work based on the *E(NK)* framework. In a broad perspective, molecular networks can be considered as *E(NK)* models. In the context of this paper we have N=27 loci and E=3 environments. Even though the loci in our model interact, quantifying the level of connections between genes, *K*, proves difficult. In molecular networks, interactions between genes often involved more than one reaction. Hence there is not straightforward way of computing *K*. This limitation does not really matter since it is often used as a summary statistics in experiments based on an ensemble approach to gene networks. Since in this paper the network we analyze is not random, the actual topology of the network is more meaningful than the parameter *K*.

Computational challenges: Simulating the evolution of a population of network models requires solving the model with a large number of different parameterizations (size of the population x number of generations). In order to estimate the fluctuations of the selection process, it is necessary to repeat the simulation of the network evolution a large number of times. Since dynamic models are orders of magnitude more expensive to simulate than a static model of a GP map, running an experiment such as the one described in this paper is a significant technical challenge.

Multiscale models: A major challenge in using regulatory gene networks or metabolic pathways as genotype to phenotype mappings is that gene networks are dynamical systems and consequently their properties are defined by reference to their time evolution. In contrast, the common genetics view is a more static vision of the relationship between the genotype of an individual and its phenotype. Time is included to describe the evolution of populations of individuals across generations. Analysis of the genetics of gene networks requires introducing a different time scale. By introducing a correspondence between genetic loci and the parameter space of a gene network on the one hand, and by defining trait functions to quantify the performance of a model parameterization on the other hand, we reconcile a theoretical framework that assumes a static relationship between phenotype and genotype with dynamical models of gene expression.

An important step of this approach is to reduce the time-evolution of the gene network into a set of static gene to phenotype relationships. So far, the performance of gene networks has been reduced to the asymptotic level of expression of one or few genes in one particular set of simulation conditions [34,35]. In this paper we have formalized and generalized the notion of trait and performance functions applied to models of molecular interactions. Instead of focusing on the level of expression of specific genes, the traits considered in this paper are derived from metabolite concentrations. These indicators integrate the effect of all genes in the system along with the effects of environmental parameters. This approach makes it possible to integrate the environment in the GP maps derived from molecular networks. In other simulations, we have defined on the same model, trait functions to quantify the ability of the model to quickly react to environmental perturbations or to quantify the stability and robustness of a network (not shown).

Trait and performance functions: We were surprised to find networks exhibiting oscillations in one environment at the end of the selection process. This observation illustrates the dependence of the selection outcome on the trait functions and performance index. By using a naïve expression of the trait that relied on a single data point rather than calculating a trend, the selection process lead to parameterization consistent with our specification of the selection target but more complex than we anticipated. Similarly, we illustrate that finding the right expression to combine several traits into a single performance index is challenging. Again, the examination of the outcome of the selection showed that the performance function we used in this experiment is not optimal. The choice of trait and performance functions is partly subjective since there is not one single way to quantify the properties that will be maximized by the selection process. By comparing the outcome of simulated selection using different performance functions, it might be possible to evaluate their relevance in the computer before using them in actual breeding programs.

Genetics of molecular networks: Even though the model of the galactose switch considered in this paper has not been validated by any experimental data, the results are probably representative of the results we would get from a model derived from molecular data. It will be necessary to apply the same approach to a number of molecular networks models to better understand the model topology and regulation translate into genetic properties.

Performance landscape: The multiple modes of the asymptotic distribution of the average performance values demonstrate that the outcome of the selection process is highly uncertain from a common starting point. In the context of plant breeding programs where there is only a single realization of the selection process, this observation raises a number of issues for risk management and breeding program design. From an evolutionary perspective, it is striking that given a deterministic genotype to phenotype mapping and a stable environment, the selection process can have a large diversity of outcomes. It would be interesting to investigate the properties of the performance landscape *in vivo*. This would require conducting a large number of selection experiments in parallel starting from identical conditions. Conducting such an experiment requires having first derived from a molecular network model, a GP map explaining a large number of observed genotype to phenotype relationships. Such a map should also have some prediction power on the unobserved regions of the genetic space. The derivation of validated GP maps from the understanding of molecular mechanisms controlling the expression of complex traits remains a major scientific challenge [36].

Exploration of the genetic space: Assuming that a GP map with a good prediction power is available, then another possible application of this type of simulations is the identification of the genotypes with outstanding levels of performance by exploring the genetic space *in silico* rather than *in vivo*. These genotypes could then be assembled by fixing alleles one locus at a time using genotyping techniques and marker based selection.

This application could be evaluated today by introducing a genetic variability in artificial gene networks [27,37].

Molecular noise: In the context of this article, the gene network dynamics have been represented by differential equations. It is recognized that the small copy number of some molecules involved in the mechanisms of gene expression (e.g. transcription complexes, genes) can result in molecular fluctuations responsible for some level of phenotypic variability. This has been addressed theoretically [17], numerically [16,38], and experimentally [39]. Using a stochastic model of the gene network dynamics might have a significant impact on the outcome of the selection process. It is likely to smooth the performance landscape. Having non-deterministic performance values would also reduce the likelihood of the process from being trapped on local performance minima. By modeling molecular interactions with mass-action equations as opposed to specialized biochemical kinetics, it is possible to simulate the fluctuations of molecular interactions without changing the model. In a follow-up paper we will show how molecular noise can influence the response to selection of a molecular network. It seems likely that molecular noise influences the expression of some complex traits in higher organisms [40,41]. The framework described in this paper makes it possible to investigate its evolutionary consequences.

Context-dependency of genetic effects: For seven of the loci, one of the alleles was fixed in more than 95% of the runs. These alleles can be regarded as favorable within the context of this parameterization of the genotype to phenotype mapping of the galactose pathway. In a first approximation, these alleles have a strong additive effect on performance. However, for the remaining polymorphic loci, the contribution of each allele to performance is context dependent and it is not possible to classify either of the alleles as favorable without specifying the context. At the individual level, the context refers to the alleles present at other loci associated to the trait. At this level, the context-dependency of allele values results from the non-linearity of the model of molecular interactions. Context-dependency can also be considered at the population level. The selective values of the allele at one particular locus depends on the allele frequencies of all other loci associated to the trait being selected.

Epistasis is a challenging concept with different meanings in molecular biology and genetics. At the molecular level, all the genes of the GAL system are engaged in some form of *cis* or *trans* interaction. Epistasis seems prevalent at this level. For geneticists, epistasis is associated with the limits of the additive model of gene action. If the complexity of the selection process indicates epistatic effects, it is nonetheless striking that most genetic gain takes place during the first 50 cycles of selection in our simulation experiment. This suggests that at the population level the system is initially in a largely additive state, despite these molecular interactions. However, following cycle 50 the results of the selection process are much less predictable. This indicates that the initial cycles of selection predictably fix particular alleles at seven loci. The additive genetic variation associated with these seven loci is exploited by selection. Following this additive gain, the population structure is such that the system moves into a state where there are more context-dependent, non-additive effects exploited by selection. The consequence is the many possible selection end points by cycle 100. It may thus be necessary to refine our understanding of the consequences of molecular interactions by, for instance, relating genetic epistasis to the control properties of the regulatory circuits of the gene network model [42]. Further, the results we observe reinforce that views of genetic variation based on the concepts of additive and non-additive (dominance, epistatic) components of variance for a trait are population specific and are therefore time dependent in relation to the cycles of selection [43]. The work presented in this paper paves the way to a more formal analysis of the genetic properties of molecular networks. In particular, it is necessary to analyze physiological and statistical genetic effects [44,45]. The techniques to analyze genetic interactions between more than two loci raise a number of theoretical and computational problems that are beyond the scope of this article.

It is an inspirational first step to use models of molecular interactions for gene networks and their gene-to-phenotype mappings, such as our representation of the galactose pathway, to consider the complex biological processes involved in the changes brought about by plant breeding. In turn this provides a demonstration of important issues that must be considered in the design of molecular plant breeding strategies.

Acknowledgements

We thank Howie Smith and two anonymous reviewers for valuable comments and suggestions. This work would not have been possible without the support of Bob Merrill and Roy Luedtke.

Literature cited

1. Omholt, S.W., Plahte, E., *et al*. 2000. Gene regulatory networks generating the phenomena of additivity, dominance and epistasis. *Genetics* **155**:969-80.
2. Frank, S.A. 1999. Population and quantitative genetics of regulatory networks. *J. Theor. Biol.* **197**:281-94.
3. Falconer, D.S. and MacKay, T.F.C. 1996. Quantitative Genetics. Longman Group Ltd., Harlow, United Kingdom.
4. Cooper, M. and Podlich, D.W. 2002. The E(NK) model: Extending the NK model to incorporate gene by environment interactions and epistasis for diploid genomes. *Complexity* **7**:31-47.
5. Eshed, Y. and Zamir, D. 1996. Less-than-additive epistatic interactions of quantitative trait loci in tomato. *Genetics* **143**:1807-17.
6. Li, Z., Pinson, S.R., *et al*. 1997. Epistasis for three grain yield components in rice. *Genetics* **145**:453-65.
7. Li, Z.K., Luo, L.J., *et al*. 2001. Overdominant epistatic loci are the primary genetic basis of inbreeding depression and heterosis in rice. I. Biomass and grain yield. *Genetics* **158**:1737-53.
8. Luo, L.J., Li, Z.K., *et al*. 2001. Overdominant epistatic loci are the primary genetic basis of inbreeding depression and heterosis in rice. II. Grain yield components. *Genetics* **158**:1755-71.
9. Leamy, L.J., Routman, E.J., and Cheverud, J.M. 2002. An epistatic genetic basis for fluctuating asymmetry of mandible size in mice. *Evolution* **56**:642-53.
10. Elena, S.F. and Lenski, R.E. 2001. Epistasis between new mutations and genetic background and a test of genetic canalization. *Evolution* **55**:1746-52.
11. Steinmetz, L.M., Sinha, H., *et al*. 2002. Dissecting the architecture of a quantitative trait locus in yeast. *Nature* **416**:326-30.

12. Perera, F.P. 1997. Environment and cancer: who are susceptible? *Science* **278**:1068-73.

13. Weatherall, D.J. 2001. Phenotype-genotype relationships in monogenic disease: lessons from the thalassaemias. *Nat. Rev. Genet.* **2**:245-55.

14. Rutherford, S.L. 2000. From genotype to phenotype: buffering mechanisms and the storage of genetic information. *Bioessays* **22**:1095-105.

15. De Jong, H. 2002. Modeling and simulation of genetic regulatory systems: A literature review. *J. Comp. Bio.* **9**:67-103.

16. Goss, P.J.E. and Peccoud, J. 1998. Quantitative modeling of stochastic systems in molecular biology using stochastic Petri nets. *Proc. Nat. Acad. Sci U.S.A.* **95**:6750-5.

17. Peccoud, J. and Ycart, B. 1995. Markovian modelling of gene products synthesis. *Theor. Pop. Bio.* **48**:222-34.

18. McAdams, H.H. and Arkin, A.P. 1999. It's a noisy business! Genetic regulation at the nanomolar scale. *Trends Gen.* **15**:65-9.

19. Arkin, A.P., Ross, J., and McAdams, H.H. 1998. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected *Escherichia coli* cells. *Genetics* **149**:1633-48.

20. Venkatesh, K.V., Bhat, P.J., *et al*. 1999. Quantitative model for Gal4p-mediated expression of the galactose/melibiose regulon in *Saccharomyces cerevisiae*. *Biotechnol. Prog.* **15**:51-7.

21. Erdi, P. and Toth, J. 1989. *Mathematical models of the chemical reaction*. Manchester University Press, Manchester.

22. Cohen, S.D. and Hindmarsh, A.C. 1996. CVODE, a stiff/nonstiff ODE solver in C. *Comp. Phys.* **10**:138-43.

23. Micallef, K.P., Cooper, M., and Podlich, D.W. 2001. Using clusters of computers for large QU-GENE simulation experiments. *Bioinformatics.* **17**:194-5.

24. Podlich, D.W. and Cooper, M. 1998. QU-GENE: a simulation platform for quantitative analysis of genetic models. *Bioinformatics.* **14**:632-53.

25. Elowitz, M.B. and Leibler, S. 2000. A synthetic oscillatory network of transcriptional regulators. *Nature* **403**:335-8.

26. Houchmandzadeh, B., Wieschaus, E., and Leibler, S. 2002. Establishment of developmental precision and proportions in the early Drosophila embryo. *Nature* **415**:798-802.

27. Kaern, M., Blake, W.J., and Collins, J.J. 2003. The engineering of gene regulatory networks. *Annu. Rev. Biomed. Eng.* **5**:179-206.

28. Hasty, J., McMillen, D., and Collins, J.J. 2002. Engineered gene circuits. *Nature* **420**:224-30.

29. Kempthorne, O. 1988. An overview of the field of quantitative genetics. 47-56. *In* B.S. Weir, E.J. Eisen, M.M. Goodman, and G. Namkoong (ed.) *Proceedings of the second international conference on quantitative genetics*. Sinauer Associates, Inc.

30. Micallef, K.P., Cooper, M., and Podlich, D.W. 2001. Using clusters of computers for large QU-GENE simulation experiments. *Bioinformatics.* **17**:194-5.

31. Kauffman, S.A. 1993. The origins of order: self-organization and selection in evolution. Oxford University Press, New York.

32. Cooper, M., Chapman, S.C., *et al*. 2002. The GP Problem: quantifying gene to phenotype relationships. *In Silico Biol.* **2**:151-64.

33. Podlich, D.W. and Cooper, M. 1999. Modelling plant breeding programs as search strategies on a complex response surface. *Lect. Notes Comp. Sci.* **1585**:171-8.

34. Omholt, S.W., Plahte, E., *et al*. 2000. Gene regulatory networks generating the phenomena of additivity, dominance and epistasis. *Genetics* **155**:969-80.

35. Frank, S.A. 1999. Population and quantitative genetics of regulatory networks. *J. Theor. Biol.* **197**:281-94.

36. Guet, C.C., Elowitz, M.B., *et al*. 2002. Combinatorial synthesis of genetic networks. *Science* **296**:1466-70.

37. Hasty, J., McMillen, D., and Collins, J.J. 2002. Engineered gene circuits. *Nature* **420**:224-30.

38. Arkin, A., Ross, J., and McAdams, H.H. 1998. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected *Escherichia coli* cells. *Genetics* **149**:1633-48.

39. Elowitz, M.B., Levine, A.J., *et al*. 2002. Stochastic gene expression in a single cell. *Science* **297**:1183-6.

40. Kemkemer, R., Schrank, S., *et al*. 2002. Increased noise as an effect of haploinsufficiency of the tumor-suppressor gene neurofibromatosis type 1 in vitro. *Proc. Natl. Acad. Sci. U.S.A.* **99**:13783-8.
41. Cook, D.L., Gerber, A.N., and Tapscott, S.J. 1998. Modeling stochastic gene expression: implications for haploinsufficiency. *Proc. Nat. Acad. Sci. U.S.A.* **95**:15641-6.
42. Thomas, R. 1999. Deterministic chaos seen in terms of feedback circuits: analysis, synthesis, "labyrinth chaos". *Inter. J. Bifurcation and Chaos* **9**:1889-905.
43. Carroll, S.P., Dingle, H., and Famula, T.R. 2003. Rapid appearance of epistasis during adaptive divergence following colonization. *Proc. R. Soc. Lond. B Biol. Sci.* **270S1**:S80-S83.
44. Cheverud, J.M. and Routman, E.J. 1995. Epistasis and its contribution to genetic variance components. *Genetics* **139**:1455-61.
45. Holland, J.B. 2001. Epistasis and plant breeding. *Plant Breed. Re.* **21**:27-92.
46. Ideker, T., Thorsson, V., *et al*. 2001. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science* **292**:929-34.
47. Ostergaard, S., Olsson, L., *et al*. 2000. Increasing galactose consumption by Saccharomyces cerevisiae through metabolic engineering of the GAL gene regulatory network. *Nat. Biotechnol.* **18**:1283-6.
48. Ostling, J. and Ronne, H. 1998. Negative control of the Mig1p repressor by Snf1p-dependent phosphorylation in the absence of glucose. *Eur. J. Biochem.* **252**:162-8.
49. Ronne, H. 1995. Glucose repression in fungi. *Trends Gen.* **11**:12-7.
50. Yano, K. and Fukasawa, T. 1997. Galactose-dependent reversible interaction of Gal3p with Gal80p in the induction pathway of Gal4p-activated genes of *Saccharomyces cerevisiae*. *Proc. Nat. Acad. Sci. U.S.A.* **94**:1721-6.

Appendix

The model of the GAL switch is given in this appendix in a format suitable for use with the MATLAB functions for numerically integrating differential equations.

```
Y0 = zeros(16, 1);                              % initial condition
Y0(1) = 0.000000;                               % GalExt
Y0(2) = 0.000000;                               % GluExt
Y0(3) = 0.000000;                               % Gal
Y0(4) = 0.000000;                               % Glu-6P
Y0(5) = 0.000000;                               % Gal80p
Y0(6) = 0.000000;                               % Gal4p
Y0(7) = 0.000000;                               % Gal3p
Y0(8) = 0.000000;                               % GAL-4-80
Y0(9) = 0.000000;                               % GAL-4-80-3
Y0(10) = 0.000000;                              % Glu
Y0(11) = 0.000000;                              % gal4g
Y0(12) = 1.000000;                              % gal4gX
Y0(13) = 0.000000;                              % E
Y0(14) = 1.000000;                              % GAL
Y0(15) = 0.000000;                              % GAL-4
Y0(16) = 0.000000;                              % Gal3p*
KC = zeros(27, 1);                              % vector of kinetic
KC(1) = 50.000000;                              % Gal80p → 0
KC(2) = 36.000000;                              % Gal4p → 0
KC(3) = 40.000000;                              % Gal3p → 0
KC(4) = 50.000000;                              % Glu → 0
KC(5) = 82.000000;                              % E → 0
KC(6) = 1.000000;                               % GalExt → Gal
KC(7) = 6.000000;                               % E + GalExt → Gal + E
KC(8) = 12.000000;                              % E + Gal → Glu-6P + E
KC(9) = 100.000000;                             % Glu-6P → Glu
KC(10) = 10.000000;                             % Glu + gal4g → gal4gX
KC(11) = 1.000000;                              % Glu + gal4g ← gal4gX
KC(12) = 23.000000;                             % gal4g → gal4g + Gal4p
KC(13) = 7.000000;                              % GAL + Gal4p → GAL-4
KC(14) = 9.000000;                              % GAL + Gal4p ← GAL-4
KC(15) = 3.500000;                              % GAL-4 + Gal80p ↔ GAL-4-
KC(16) = 5.000000;                              % GAL-4 + Gal80p ← GAL-4-
KC(17) = 8.000000;                              % Gal3p* + GAL-4-80 → GAL-
KC(18) = 10.000000;                             % Gal3p* + GAL-4-80 ← GAL-
KC(19) = 101.000000;                            % GAL-4 → GAL-4 + Gal80p
KC(20) = 2.000000;                              % GAL-4 → GAL-4 + Gal3p
KC(21) = 338.000000;                            % GAL-4-80-3 → GAL-4-80-3
KC(22) = 309.000000;                            % GAL-4-80-3 → GAL-4-80-3
KC(23) = 336.000000;                            % GAL-4-80-3 → GAL-4-80-3
KC(24) = 19.000000;                             % GAL-4 → GAL-4 + E
KC(25) = 10.000000;                             % GluExt → Glu
KC(26) = 1320.000000;                           % Gal + Gal3p → Gal3p*
KC(27) = 809.000000;                            % Gal + Gal3p ← Gal3p*
r1 = KC(1) * Y(5);                              % Gal80p → 0
r2 = KC(2) * Y(6);                              % Gal4p → 0
r3 = KC(3) * Y(7);                              % Gal3p → 0
r4 = KC(4) * Y(10);                             % Glu → 0
r5 = KC(5) * Y(13);                             % E → 0
r6 = KC(6) * Y(1);                              % GalExt → Gal
r7 = KC(7) * Y(1) * Y(13);                      % E + GalExt → Gal + E
```

```
r8 = KC(8) * Y(3) * Y(13);                  % E + Gal → Glu-6P + E
r9 = KC(9) * Y(4);                          % Glu-6P → Glu
r10 = KC(10) * Y(10) * Y(11);               % Glu + gal4g → gal4gX
r11 = KC(11) * Y(12);                       % Glu + gal4g ← gal4gX
r12 = KC(12) * Y(11);                       % gal4g → gal4g + Gal4p
r13 = KC(13) * Y(6) * Y(14);                % GAL + Gal4p → GAL-4
r14 = KC(14) * Y(15);                       % GAL + Gal4p ← GAL-4
r15 = KC(15) * Y(5) * Y(15);                % GAL-4 + Gal80p ↔ GAL-4-
r16 = KC(16) * Y(8);                        % GAL-4 + Gal80p ← GAL-4-
r17 = KC(17) * Y(8) * Y(16);                % Gal3p* + GAL-4-80 → GAL-
r18 = KC(18) * Y(9);                        % Gal3p* + GAL-4-80 ← GAL-
r19 = KC(19) * Y(15);                       % GAL-4 → GAL-4 + Gal80p
r20 = KC(20) * Y(15);                       % GAL-4 → GAL-4 + Gal3p
r21 = KC(21) * Y(9);                        % GAL-4-80-3 → GAL-4-80-3
r22 = KC(22) * Y(9);                        % GAL-4-80-3 → GAL-4-80-3
r23 = KC(23) * Y(9);                        % GAL-4-80-3 → GAL-4-80-3
r24 = KC(24) * Y(15);                       % GAL-4 → GAL-4 + E
r25 = KC(25) * Y(2);                        % GluExt → Glu
r26 = KC(26) * Y(3) * Y(7);                 % Gal + Gal3p → Gal3p*
r27 = KC(27) * Y(16);                       % Gal + Gal3p ← Gal3p*
Y2 = zeros(16, 1);                          % State of the system
Y2(3) = 1 * r6 + 1 * r7 + -1 * r8 + -1 *    % GalExt
Y2(4) = 1 * r8 + -1 * r9;                   % GluExt
Y2(5) = -1 * r1 + -1 * r15 + 1 * r16 + 1 *  % Gal
Y2(6) = -1 * r2 + 1 * r12 + -1 * r13 + 1 *  % Glu-6P
Y2(7) = -1 * r3 + 1 * r20 + 1 * r22 + -1 *  % Gal80p
Y2(8) = 1 * r15 + -1 * r16 + -1 * r17 + 1 * % Gal4p
Y2(9) = 1 * r17 + -1 * r18;                 % Gal3p
Y2(10) = -1 * r4 + 1 * r9 + -1 * r10 + 1 *  % GAL-4-80
Y2(11) = -1 * r10 + 1 * r11;                % GAL-4-80-3
Y2(12) = 1 * r10 + -1 * r11;                % Glu
Y2(13) = -1 * r5 + 1 * r23 + 1 * r24;       % gal4g
Y2(14) = -1 * r13 + 1 * r14;                % gal4gX
Y2(15) = 1 * r13 + -1 * r14 + -1 * r15 + 1  % E
Y2(16) = -1 * r17 + 1 * r18 + 1 * r26 + -1  % GAL
```

Chapter 6. Model parameter and topology fitting

Abstract

Motivation: Modeling of biochemical networks requires extensive knowledge of interactions and rates. However, the required interactions and rates are often incompletely understood. This creates an immediate need in synthetic biology for methods to fit models to experimental observations.

Results: GenoFIT is an optimization and exploration tool aiding the development of models of biochemical networks. Complementing the GenoDYN modeling environment, GenoFIT provides, as an optimization tool, a means to optimize model parameters, topology and rates, to capture the dynamics of experimental observations. As an exploration tool, GenoFIT can produce ensembles of network models that express a desired set of dynamics. GenoFIT uses distributed computing for improved performance as well as a built-in scripting language for specifying evaluation functions.

Availability: Complete GPL licensed source code is available from the author.

Contact: kent.vandervelden@ieee.org

Introduction

A model of a biochemical network summarizes available knowledge of the network. However, the model has the potential to be more useful by estimating the response to external stimuli and other perturbations. Methods for simulating models[1-4] and directly solving their systems of equations for steady states[5-7] are well known. However, these methods require that a model, including its topology, kinetic parameters, and initial conditions, be fully specified. The topology can be estimated using techniques, such as gene knock-outs, to expose dependencies that define the network topology. Relative initial conditions may be possible to estimate; however, due to basins of attraction, models should be fairly robust to imprecise initial conditions.

Kinetic parameters are the greatest challenge in developing a fully specified model. The easiest way to measure kinetic rates is *in vitro,* but such an environment, free of the spatial constraints of a cell, can result in inaccurate kinetic rates at best[8]. Measurement of kinetic rates *in vivo* is more difficult and would be uncommon to perform for all interactions.

If experimental observations are available for the model species, an evaluation function can be formulated that measures the model's deviation from observations. The evaluation function may be as simple as sum of squared error. Alternatively, if no experimental measurements are available but if a qualitative description of the dynamics is, an evaluation function can also be constructed. The evaluation function transforms the process of finding parameters into an optimization or search process. Since the number of observed species is likely to be less than the number of model parameters, the optimization process will be under-constrained and thus many apparently equivalent sets of parameters will exist. The model parameters located during optimization are unlikely to be the true kinetic parameters but are still potentially useful for model prediction.

Much research has been applied to optimizing metabolic networks[9-11], and model fitting of gene regulatory networks is less refined[12-14]. One reason for this disparity is that metabolic networks are composed of largely static chemical reactions while regulatory networks are more transitory by their nature. The choice of modeling formalism also affects the ease of model fitting. A nonlinear formalism, such as one based on tightly coupled differential equations, will naturally be more difficult to optimize than one based on linear approximation such as s-systems. With nonlinear models, techniques may not exist that guarantee convergence to the global optimum.

Evolution produces complex systems from random mutations guided by selective pressure. Computer scientists, inspired by evolution, created the field of evolutionary computation that uses the same principles thought to be at work in evolution. Through evolution, nature has been successful, more than engineers, in constructing systems robust in noisy environments. It is therefore only natural to apply the concepts of evolutionary computation to the problem of network modeling.

Evolution of network models is an extension of model fitting that increases the degrees of freedom available. Model fitting considers alterations of the model's parameters alone with no impact on the model's topology. With network model evolution, the topology of the network is allowed to change as well as the parameters. While similar to network reverse engineering[15,16] in goal, evolutionary methods do not rely on statistical inference, but instead rely upon random changes and selective pressures. Similar approaches have been used successfully to direct the evolution of an engineered network *in vivo*[17].

Previous attempts to evolve networks include that of Sakamoto and Iba[18] which used a genetic programming[19] approach consisting of unconstrained parse trees representing differential equations of arbitrary mass-action reactions. This approach explores large areas of a search space containing models without analogues to known regulation mechanisms and ultimately may settle on such a model. Possibly key to the success of Francois and Hakim[20] was limiting the procedure to only reactions that correspond to viable regulation mechanisms, resulting in a smaller search space. However, without restricting the use of the building blocks, several unusual constructs were possible, such as transcription factors consisting of more than four subunits or unusual promoter–gene constructs. Regardless, the resulting networks are more likely to be realistic than those found by the Sakamoto and Iba's method. Also, neither method considered the effects of delays between transcription and translation known to be important for circadian clock networks[21].

GenoFIT is the tool we present next that combines the ability to fit the reaction rates of an existing model to experimental data with the ability to evolve new network topologies. The ability to evolve networks is particularly interesting, as it may help to develop initial models for new systems. In our discussion of GenoFIT, we will first present the methods used by GenoFIT followed by four examples. In closing, we discuss uses and limitations of GenoFIT and suggestions for future improvements.

Methods

GenoFIT has two modes of operation. In one mode, GenoFIT only optimizes the parameters, while in the second mode GenoFIT may also manipulate the topology. In the following sections we will discuss the methods for optimizing the parameters and topology separately. Following those discussions there are some final words on common operations that GenoFIT must perform regardless of its mode.

Optimization of parameters

GenoFIT searches for solutions using a genetic algorithm[22] (GA) for global optimization and a downhill simplex algorithm[23] (SA) for local refinement. The stochastic search of the GA quickly samples the parameter space, initially avoiding local minima, but it is not well suited for fine-tuning a particular solution within a local area. The downhill descent behavior of the SA quickly finds local minima of areas visited by the GA, but by itself it would quickly become trapped in a local minimum. The GA and SA complement each other, and this hybrid approach has been successfully applied previously[11].

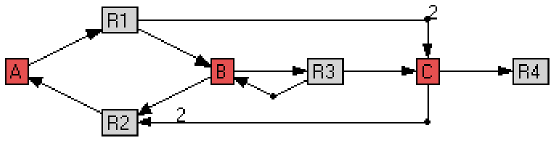
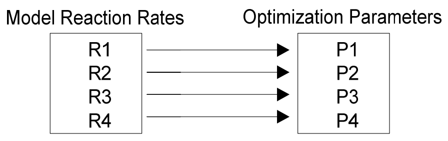


Figure 6.1: Example network model

To understand the encoding and the operators used by GenoFIT, consider the example shown in Figure 6.1. This model contains four reaction rates to optimize. Each reaction rate is mapped to an optimization parameter, allowing each reaction rate to have a unique optimization parameter (Figure 6.2) or multiple reaction rates to be associated with a single optimization parameter (Figure 6.3). This mapping is controlled by the GenoFIT parameter file.



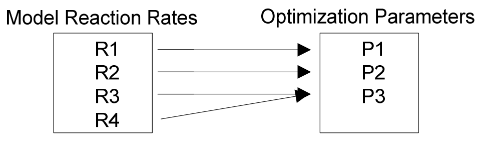


Figure 6.2: 1-to-1 parameter mapping

Figure 6.3: Many-to-1 parameter mapping

The GA maintains a population of *individuals* where each individual actually represents a set of model parameters. Each iteration of the GA produces a new population based upon the previous population by using a combination of crossover and mutation events. Crossover operates by selecting a pair of individuals, biased by their fitness compared to the rest of the population, and produces parameters of the new individuals by shuffling corresponding parameters of the selected individuals between crossover points. Two types of crossover operators are available in GenoFIT differing in the number of crossover points: two-point crossover (Figure 6.4), which always selects two points, and *n*-point crossover (Figure 6.5), which randomly chooses each parameter. If it is known that contiguous ranges of parameters behave as units, the two-point crossover may have an advantage.

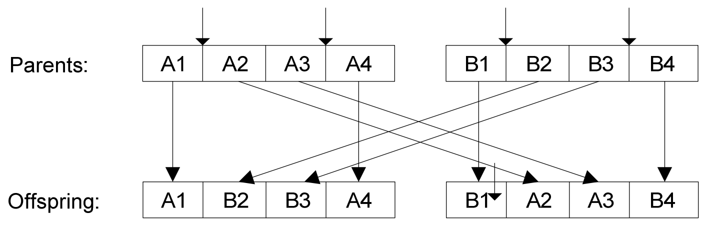


Figure 6.4: 2-point crossover

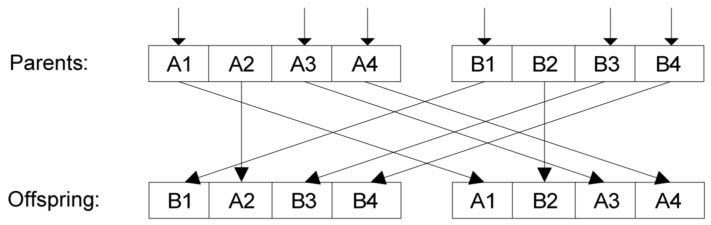


Figure 6.5: N-point crossover

In addition to the crossover event, random mutations (Figure 6.6) may also occur. A mutation can occur at any parameter, adjusting it by a random amount within a percentage, specified in the parameter file, of the original value.

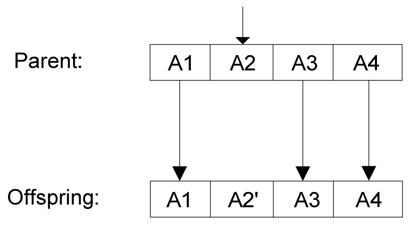


Figure 6.6: Point mutation

If individuals in the new population cannot be simulated properly, e.g. due to unrealistically stiff equations that cause the integrator called during simulation to fail, GenoFIT can either leave this individual, likely discarding it in the next generation due to the unlikeliness of its being selected, or immediately replace it with a new random individual. To avoid losing an individual with good fit, the top individual can optionally be ensured a place in the next generation.

Since the selection of individuals is biased by their relative fitness, ultimately the population will converge to highly similar individuals during the later cycles of the GA. The point where this will occur is dependent partially upon the population size as well as selection pressure. To avoid population stagnation altogether and having individuals trapped in local minima, the diversity of the population can be periodically examined and, if too low, the bottom half of population can be replaced with random individuals.

During each iteration, some number of the top individuals of the population, controllable from the parameter file, can be optimized using the SA. To avoid excess time being spent performing local optimization, an upper limit on the number of iterations or wall clock time can be set on the SA.

Key to any optimization is the proper specification of the evaluation function. GenoDYN includes a rich scripting language for defining evaluation functions. All the model parameters, except the topology, are accessible and can be modified from within scripts. Evaluation functions can set both a fitness score of an individual and optionally an objective score. The fitness score is a quantitative score, often some derivative of root-mean-square deviation (RMSD) from measured values, while the objective score is more often a qualitative score. Individuals are first sorted by the objective score and then by the fitness score. The need for two measures is to help specify the evaluation function for complex behavior that is hard to capture with RMSD alone. The observation driving the use of two scores is that qualitative dynamics are harder to obtain than the quantitative values, but after obtaining correct dynamics, the emphasis can shift to refining the values.

Models of biochemical systems can have many parameters, and with more parameters, the search space becomes larger, and the optimization problem is likely to become harder in general. Any reduction in the number of parameters should result in a simpler optimization problem. Not every parameter of a biochemical system is likely to be independent. For example, the rates of each instance of translation, transcription, and protein degradation could be assumed to be the same, potentially allowing many parameters to be collapsed. To facilitate the collapse of similar parameters, GenoFIT maps a set of model parameters to a set of optimization parameters with predefined ranges. It is the optimization parameters that are actually manipulated, and manipulation of an optimization parameter has the effect of changing each associated model parameter. Any number of model parameters is allowed to share the same optimization parameter.

Being able to specify ranges that are valid for optimization parameters helps to simplify the parameter search space, but being able to specify the range as a function of other parameters would be even more useful. In its current form, GenoFIT does not directly support this within its configuration file. However, we have accomplished this in a rudimentary way by using the fitness function scripting language to test for correctness of the parameters and by penalizing individuals that deviate from the desired functional relationship of the optimization parameters.

```
Parameter ranges:
# RxnName       Index into param table  Rxn Equation    Kinetic value
GFP      0           # GFP --> 0
TetR        1           # TetR --> 0
PL2      2   3  # (lambda cIg) + 4LacI <-> (lambda cIgX)
PL-a     4   5       # GFPg + 2(lambda cI) <-> GFPgX
PL-b     4   5       # LacIg + 2(lambda cI) <-> LacIgX
# Parameters
# Param Index   Min     Max                     -or-
# Param Index   Fixed Value
0      0.001       100
1      0.5
2      0.001       1
3      0.001       1
4      0.001       100
5      0.001       100
```

Figure 6.7: Example parameter specification

Many of the features of model and optimization parameters are shown in Figure 6.7, which is a section of a GenoFIT parameter file. Eight model parameters and six optimization parameters are specified. First, the model reactions are listed, including their name used in the model file and an index into the optimization parameter list. Each reaction that will be optimized, but not necessarily all reactions, must be assigned an optimization parameter; reversible reactions must have two optimization parameters assigned. In the example, reactions PL-a and PL-b involve the same transcription factor and promoter, are assumed to have a similar reaction rate, and thus have the same optimization parameters assigned. Following the list of reactions is the list optimization parameters which are tuples of index value and either a minimum-maximum pair or a single value. If a single value is used, the model parameters referencing that optimization parameter will be assigned that fixed value for the entire optimization. If a minimum–maximum pair present, GenoFIT is allowed to vary that parameters inclusively between those values.

Optimization of model topology

When evolving a network topology GenoFIT can no longer code the optimization parameters as a fixed length vector. Instead, individuals are graphs. Crossover operations are not used because there is no direct correspondence between individuals. The number of reactions and their order will vary. Mutation operators modify reactions rates, as before, and also are able to introduce new network components. Mutation operators modify reaction rates much more frequently than network components are added.

GenoFIT samples from a collection of building blocks when selecting a new component to add. The building blocks are an abstraction of an underlying mass-action reaction motif that is representative of a biochemical process. The collection could simply be a reaction node and a molecule node; this would allow the greatest freedom, but also allow many more unrealistic topologies. Instead, biologically relevant building blocks identified by Francois and Hakim[20] and shown in Figure 6.8 through Figure 6.14 are used.

These building blocks use four types of molecules: genes, proteins, gene–protein complexes, and protein-protein complexes. GenoFIT enforces that when a building block is added that only compatible molecule types are connected to compatible points on the building blocks. For instance, a gene node could not attach to a protein node on a building block.

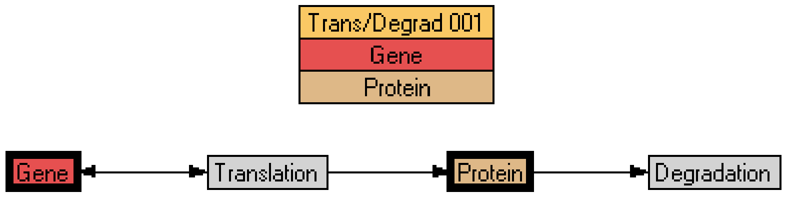


Figure 6.8: Protein production

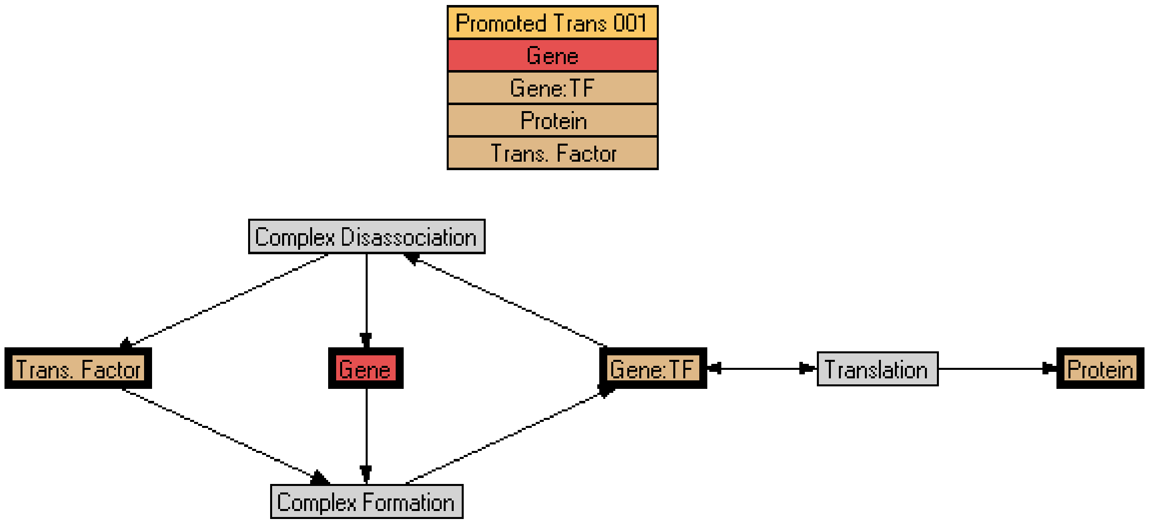


Figure 6.9: Protein production with bound promoter building block and model

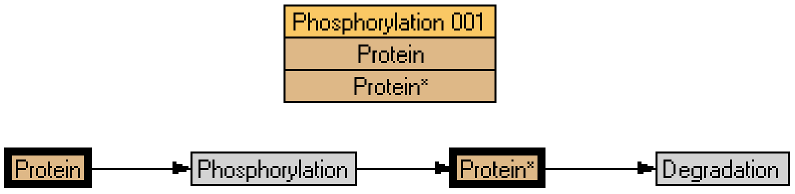


Figure 6.10: Phosphorylation building block and model

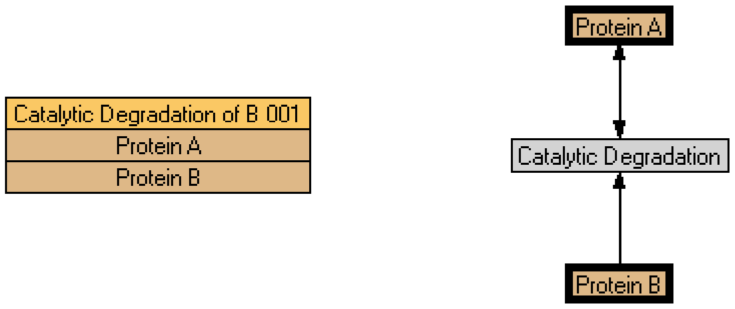


Figure 6.11: Catalytic protein degradation building block and model

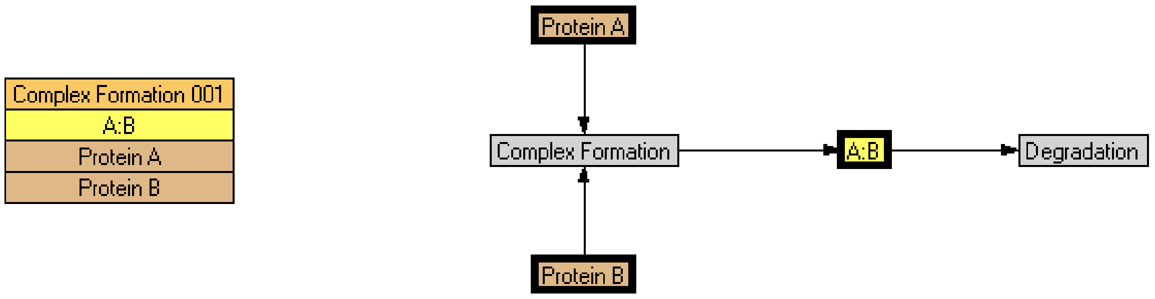


Figure 6.12: Protein complex formation building block and model

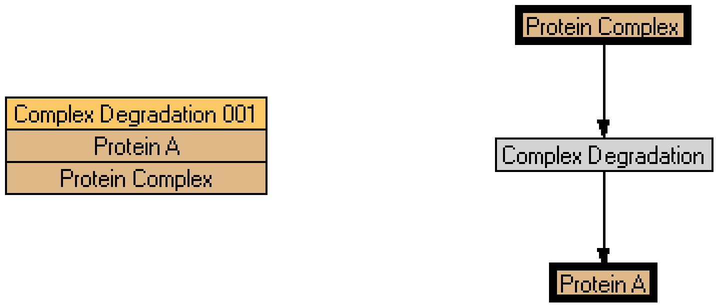


Figure 6.13: Partial complex degradation building block and model

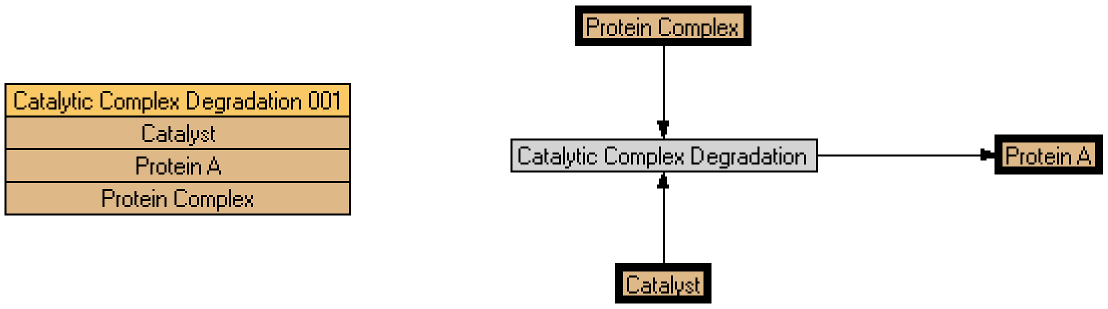


Figure 6.14: Catalytic partial complex degradation building block and model

GenoFIT introduces modules into a network randomly without bounds. By not explicitly removing modules during the evolution, a module that is introduced might eventually be made important as the result of a subsequent change, or an unimportant module may be rendered useless by its kinetic rates falling to zero. Regardless, the final models are likely to contain many unnecessary components. After all iterations are complete, GenoFIT can apply a pruning process. During the pruning process, modules are removed from the network iteratively and the model is reevaluated. If the removal did not severally alter the evaluation function and did not alter the objective function, then the removal is allowed and another round of removal is applied. If the evaluation function changes considerably or the objective score decreases then the removal is rejected and the removal of a different module is tested. This process is repeated until no further modules can be removed without severely affecting the evaluation and objective function severely negatively. Although GenoFIT only makes a single pruning attempt, alternative solutions are possible given the selection of the modules to be tested is random.

Common operations

Model fitting can be very time-consuming. To help with this, GenoFIT uses distributed computing. A single task is designated as the supervisor task, which is responsible for communicating with worker tasks, including distributing the model and subsequently the individuals. During each iteration, the population is divided between the workers that are responsible for performing the simulations, calculating fitness, and refining optional local areas. The results are returned to the supervisor, which is responsible for building the next generation. Simplifying management, the same binary is used for both the workers and the supervisor.

Upon completion, GenoFIT saves a GenoDYN compatible file containing the final model. If optimizing only the parameters, this model will be identical to the original but with updated parameters. It is also possible to save a controllable number of top individual in each generation. Sometimes it is interesting to examine the parameter space trajectory that GenoFIT took to arrive at the final configuration. Model parameter sets can be easily extracted from the network files and tabulated using tools found with GenoDYN.

Results

To demonstrate GenoFIT, we will consider four examples. Each example is optimized in triplicate. The population contains 100 individuals and the population evolves for 100 generations. Each example uses RMSD from target measurements as the evaluation function. The fourth example also includes an objective function.

The first two examples consider existing models where simulated results provide the target values for the evaluation function. Each specie is measured at three time points, t = 0, t = 10, and t = 100. It is likely that the data used for optimization will be steady state or time series measurements. To compare these two usage scenarios we compare optimizing each network using only the last time point (steady state) and using all three time points (time series).

The third example considers an existing model that exhibits bi-stability. This example demonstrates how an evaluation function alone can lead to misleading results.

The fourth example evolves a toggle-switch model *de novo* given only evaluation and objective functions. We examine how the top individual changes over time. Also, we examine how pruning identifies the functional core of the model. In Appendix B this particular example is extended and we examine how one can identify similar models between populations and how many alternatives models may exist which are all equally valid.

Each example considers a model which is synthetic. By considering only synthetic examples, we are assured that a solution exists and that valid parameters are known ahead of time, allowing evaluation of the quality of the optimized parameters with respect to the original parameters.

Example 1

The model shown in Figure 6.15 represents gene transcription and translation with negative feedback controlling transcription and protein degradation. This is a common motif present in models of regulatory networks. Node A represents the inaccessible gene, B represents the accessible gene, and C represents the protein product. Transcription and translation is captured in the R3 reaction, degradation of C is captured in the R4 reaction, and gene regulation is captured by the R1 and R2 reactions.

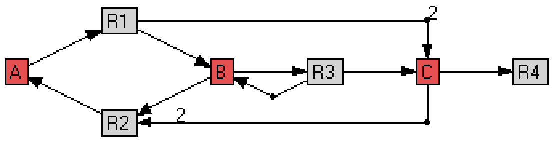


Figure 6.15: Model of self-regulated protein production

Time	A	B	C
1.00	0.62	9.38	0.72
10.00	6.05	3.95	1.11
100.00	9.83	0.17	5.38

Table 6.1: Target values for evaluation function

The values used for the evaluation function are shown in Table 6.1. Figure 6.16 shows species trajectories of the original and two sets of optimized parameters using only the last time point. While the trajectories appear quite different from the original, the last time point matches quite well.

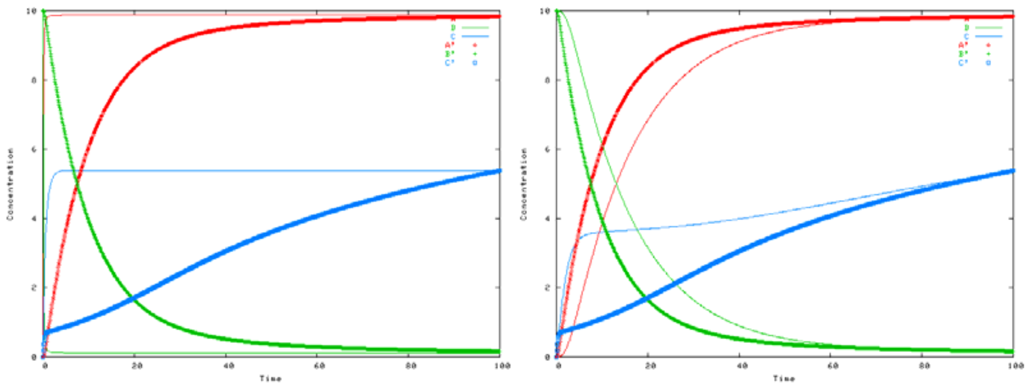


Figure 6.16: Original and optimized model trajectories (steady state)

As one might expect, with additional constraints of the time series data the optimized parameters more closely match the original dynamics (Figure 6.17). By examining the deviation of the highest ranked individual in each generation, it can be seen that GenoFIT quickly finds a rough set of parameters and then slowly refines them (Figure 6.18).

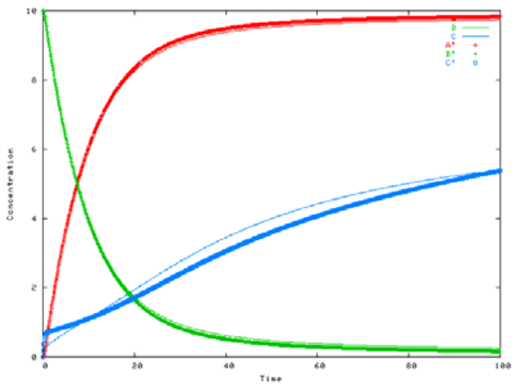


Figure 6.17: Original and optimized model trajectories (time series)

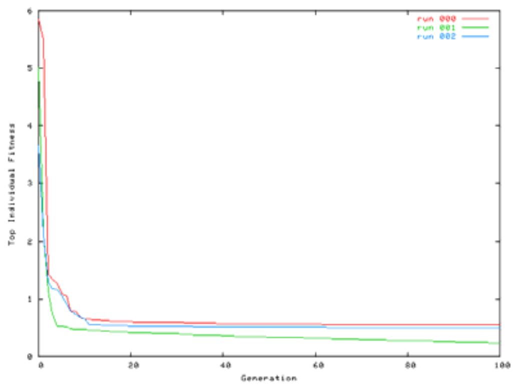


Figure 6.18: Deviation over time (time series)

Comparing the original and the optimized parameters (Table 6.2) further exposes differences, and quickly reaffirms that the original parameters are unlikely to be recovered due to the under-constrained nature of the optimization process. Not only are the same values not reconstructed, but the parameters are not even ranked in the same order, and there is considerable variation within each parameter. The use of time series data reduces the variation somewhat, especially among R3 and R4, which control protein production and protein degradation, respectively.

		Steady state			Time series		
Parameter	Target	Run 000	Run 001	Run 002	Run 000	Run 001	Run 002
R1	0.100	8.588	785.58	0.003	330.734	0.282	2.078
R2	0.200	23.056	1573.5	0.005	395.550	0.407	2.629
R3	0.200	19.630	621.17	0.135	0.201	0.201	0.202
R4	0.001	0.461	19.577	0.00001	0.007	0.005	0.006
Fitness		0.063	0.001	0.015	0.560	0.240	0.497

Table 6.2: Original and optimized kinetic parameters

Example 2

Example 2 (Figure 6.19) is similar to Example 1, containing two copies of the protein production and degradation motif. However, instead of self-regulation, in this example each protein regulates the other. The proteins are nodes C and F, which regulate the two genes between their active states, B and E, and their inactive states, A and D. Despite several of the model parameters having similar purposes (e.g. protein degradation), each model parameter was mapped to an independent optimization parameter.

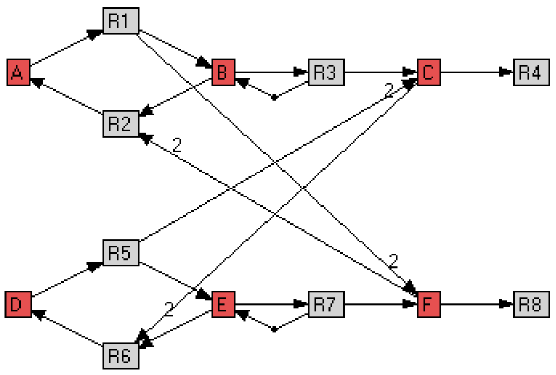


Figure 6.19: Model of production of two proteins that regulate each other

Time	A	B	C	D	E	F
1.00	1.00	9.00	1.55	0.19	9.81	0.91
10.00	9.34	0.66	1.79	3.77	6.23	3.16
100.00	9.98	0.02	1.65	4.05	5.95	17.88

Table 6.3: Target values for evaluation function

As with Example 1, we sample three time points from the simulation results of the original network to define the values for our fitness evaluation function (Table 6.3), and then we perform steady-state and time-series experiments in triplicate. Again, we can see, with a limited number of time points such as with the steady-state set, that GenoFIT has considerable freedom to find solutions, demonstrated by the deviation in the trajectories (Figure 6.20).

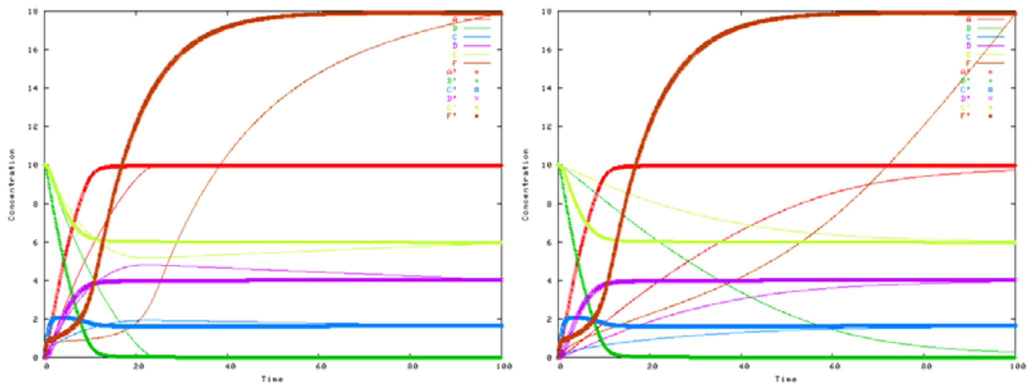


Figure 6.20: Original and opimized model trajectories (steady state)

The two additional time points of the time-series data set have a dramatic affect on the deviation of the trajectories, and it is difficult to find any discrepancy between the time course of the original and the optimization networks (Figure 6.21). The time-course deviation of the best individual (Figure 6.22) shows that this example stresses GenoFIT more than the previous example (Figure 6.18), as GenoDYN took longer to reach the point of slower progress.

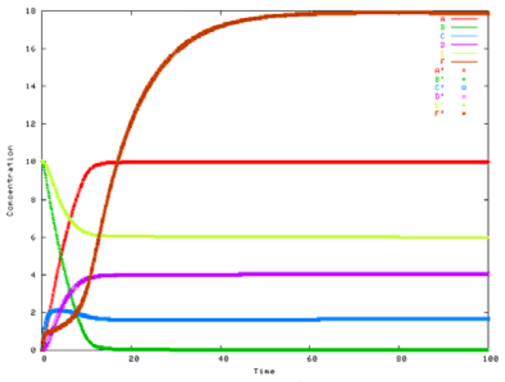


Figure 6.21: Trajectories of original and optimized kinetic parameters (time series)

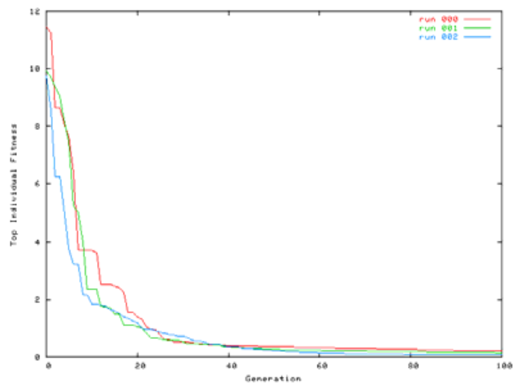


Figure 6.22: Deviation of optimized model over time (time series)

		Steady state			Time series		
Parameter	Target	Run 000	Run 001	Run 002	Run 000	Run 001	Run 002
R1	0.100	0.00001	3838.000	0.012	0.178	0.126	0.093
R2	0.200	0.113	427.674	0.020	0.287	0.230	0.194
R3	0.200	0.124	0.028	1.354	0.197	0.199	0.197
R4	0.001	0.013	0.00001	0.00001	0.0008	0.002	0.0002
R5	0.080	53.926	390.218	9287.00	0.066	0.058	0.071
R6	0.020	13.327	101.050	2319.07	0.017	0.016	0.018
R7	0.300	0.157	0.053	6.791	0.293	0.295	0.298
R8	0.100	0.050	0.00001	2.263	0.097	0.098	0.099
Fitness		0.031	0.392	0.005	0.210	0.148	0.071

Table 6.4: Original and optimized kinetic parameters

The amount of variability in the parameters for the steady-state case is high across all the parameters except R3, which is the production rate of one of the two proteins (Table 6.4). The additional data of the time series experiment reduces the variability and appears to split the parameters into two classes of variability (Table 6.4). Parameters R3, R6, R7, and R8 not only have a low amount of variability, but their values are also close to the corresponding values of the sampled model, while R1, R2, R5, and R4 have much greater variability. Each of the parameters corresponding to protein production, R3 and R7, has low variability, which was also the situation with Example 1. Beyond that, there is no clear commonality to explain the partitioning and it may simply be a coincidence.

Example 3

Here we consider how an evaluation function can be misleading if not properly designed. The model shown in Figure 6.23 is a toggle switch, which can switch between two states in response to control pulses. The nodes P1 and P2 represent the proteins of the model and by observing the concentration of these two proteins we will see the bi-modal behavior of this model. CV 1 and CV2 are the sources of environmental stimuli and via Rxn 1 and Rxn 2, respectively, they create a sudden increase in concentration of P1 and P2.

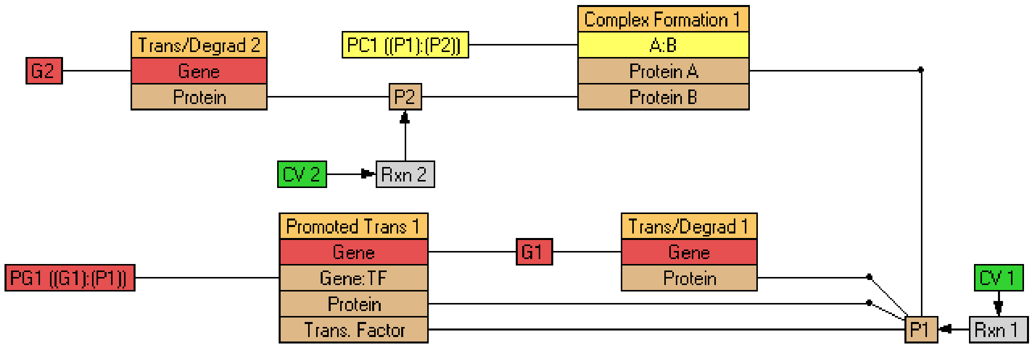


Figure 6.23: Toggle switch model

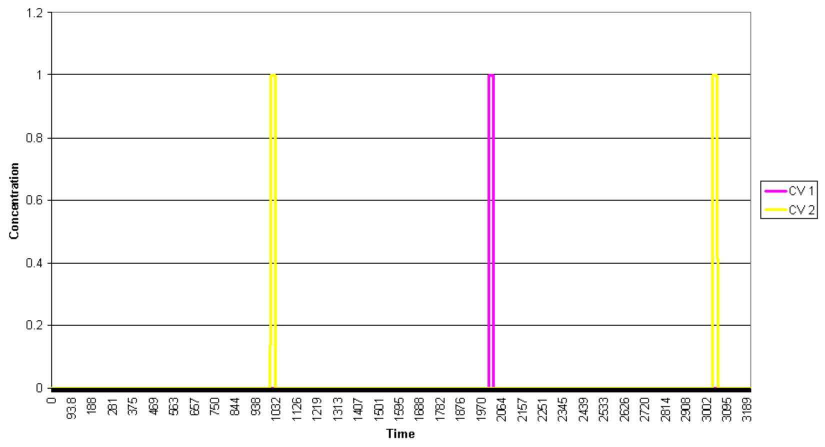


Figure 6.24: Environmental pulses

Pulses generated by CV 1 and CV 2 at times 1000, 2000, and 3000 (Figure 6.24) toggle the state of the model. The proper behavior of this model is shown in Figure 6.25. Following each pulse, the system quickly reestablishes steady state behavior.

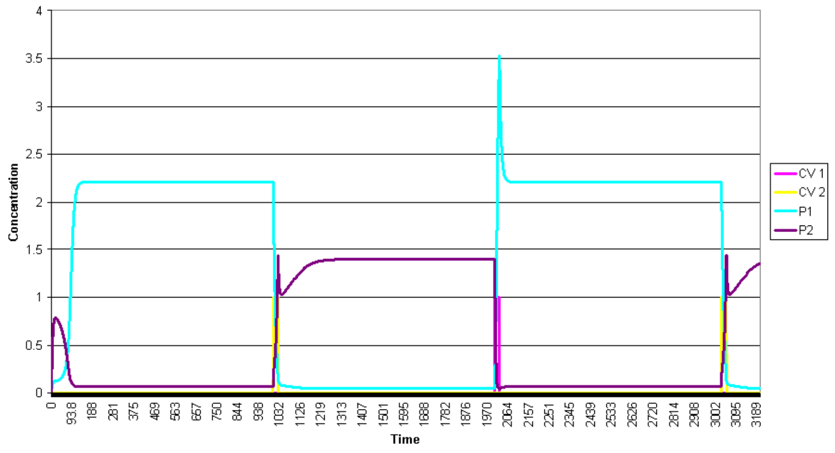


Figure 6.25: Desired toggle switch dynamics

An evaluation function can be defined that calculates the fit of any set of model parameters as the RSD from the observed concentrations for P1 at an assumed steady state (Figure 6.26).

```
    void main() {
      int i = 0;
      float time[8];
      float p1_target[8];
      float p2_target[8];
      time[0] =  950;  p1_target[0] = 2.20400;  p2_target[0] = 0.0632;
      time[1] = 1950;  p1_target[1] = 0.04276;  p2_target[1] = 1.3990;
      time[2] = 2950;  p1_target[2] = 2.20400;  p2_target[2] = 0.0632;
      time[3] = 3950;  p1_target[3] = 0.04276;  p2_target[3] = 1.3990;

      simulate();

      fitness = 0;
      for(i=0; i<4; i++) {
        fitness = fitness + sqr(p1_target[i] - result("P1", time[i]));
      }

     fitness = sqrt(fitness);
    }
```

Figure 6.26: Toggle switch evaluation function

With this evaluation function, GenoFIT quickly finds parameters that appear to exhibit bi-stability (Figure 6.27), though not identical to original dynamics (Figure 6.25). In particular, P2 does not exhibit the desired dynamics.

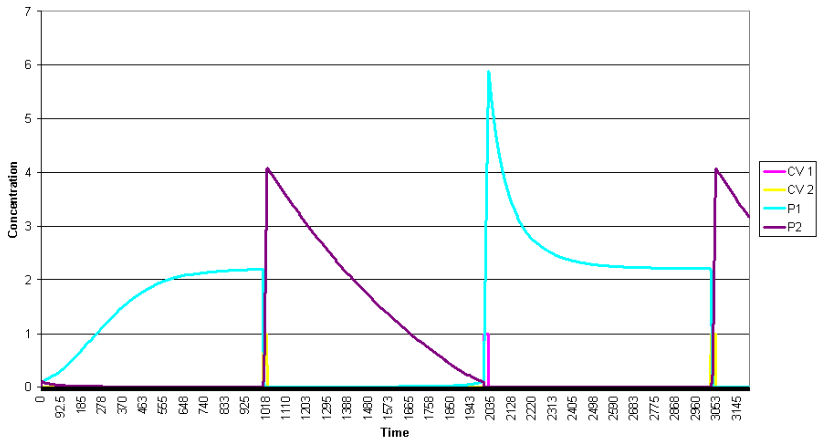


Figure 6.27: Optimized toggle switch dynamics given above fitness function

While GenoFIT quickly found this solution that the evaluation function ranks high, if the time scale is increased, allowing the model to simulate 10x longer between pulses we find that the previously observed low plateaus of P1 are unstable (Figure 6.28). While Figure 6.27 shows P1 at two distinct concentration levels, P1 had not yet reached steady state, a criterion that the evaluation function did not test for and that GenoFIT quickly exploited.

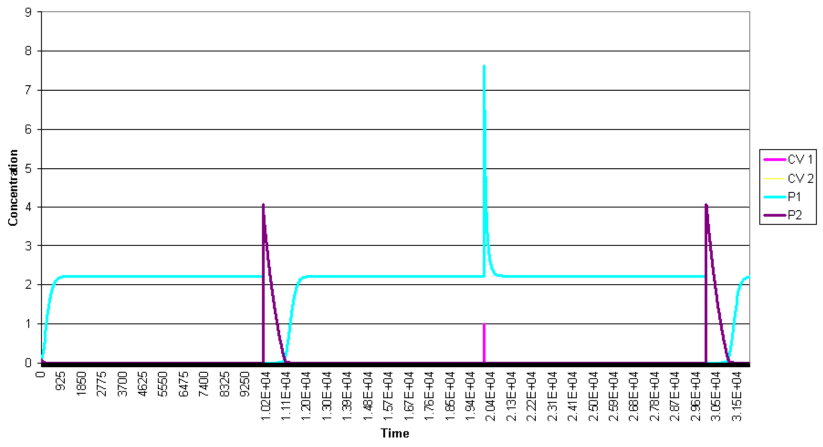


Figure 6.28: High fit solution simulated longer showing previously misleading steady states

Example 4

The last example considers a toggle switch again, but this time the topology is evolved using a collect of building blocks. The population is seeded with models containing two genes producing the indicator proteins (P000 and P001). Two control variables (CV1, CV2) act as environmental stimuli switching the state of the system. Correct behavior of the network is when P000 is in a high concentration, then P001 is in a low concentration. Likewise, when P000 is in a low concentration, then P001 is in a high concentration. The shift of P000 from low to high concentration is signaled by a pulse from CV1, and the shift of P001 from low to high concentration is signaled by a pulse from CV2.

The evaluation function, a quantitative measure of fit, is calculated using RMSD between the expected levels (10.0 in the "on" state and 0.0 in the "off" state) and the observed concentrations (Equation 6.1). The objective function (Equation 6.2), a qualitative measure of fit, is calculated by measuring the change in concentration between consecutive states for both indicator proteins just before transition when steady state is most likely to have been reached. The change in concentration is required to be greater than 1.0, and the sign indicates the direction of the switch between "on" and "off" states. Four values are possible for the objective function: 0: no toggle switch phenotype is observed; 1: P0 is exhibiting the toggle switch behavior alone; 2: P1 is exhibiting the toggle switch behavior alone; 3: both P0 and P1 are correctly exhibiting the toggle switch behavior. The optimal model will have a low evaluation score and the objective score of 3. Both functions are required because they test seemingly incompatible aspects of the model's behavior. Although a maximal objective score truly represents a successful network, the function is too coarse to guide the evolution process. The evaluation function, however, is continuous, allowing ranking and distinguishing slight improvements; however it can be misleading, as seen in Example 3. A low evaluation score does not necessarily mean a high objective score or vice versa. Also, an improvement in the objective score is more important than an improvement in the evaluation score.

$$EvaluationScore = \sqrt{\begin{bmatrix} \left(P000_A - 0\right)^2 + \left(P000_A - 10\right)^2 + \\ \left(P000_A - 0\right)^2 + \left(P000_A - 10\right)^2 + \\ \left(P000_B - 10\right)^2 + \left(P000_B - 0\right)^2 + \\ \left(P000_B - 10\right)^2 + \left(P000_B - 0\right)^2 \end{bmatrix}}$$

Equation 6.1: The toggle-switch evaluation function.

The steady state point after each transition is represented by A, B, C, D.

$$F_1 = \begin{cases} \begin{matrix} P000_A - P000_B < -1 \&\& \\ P000_B - P000_C > 1 \&\& \\ P000_C - P000_D < -1 \end{matrix} & 1 \\ \text{Otherwise} & 0 \end{cases}$$

$$F_2 = \begin{cases} \begin{matrix} P001_A - P001_B > 1 \&\& \\ P001_B - P001_C < -1 \&\& \\ P001_C - P001_D > 1 \end{matrix} & 2 \\ \text{Otherwise} & 0 \end{cases}$$

$$ObjectiveScore = F_1 + F_2$$

Equation 6.2: The toggle-switch objective function.

The steady state point after each transition is represented by A, B, C, D.

Figure 6.29 through Figure 6.33 show the best performing model every five generations, along with plots of the two indicator proteins and control variables. These figures demonstrate the evolution of the network and the emergence of the desired phenotype. The only change to the models from GenoFIT was to organize the layout of the nodes.

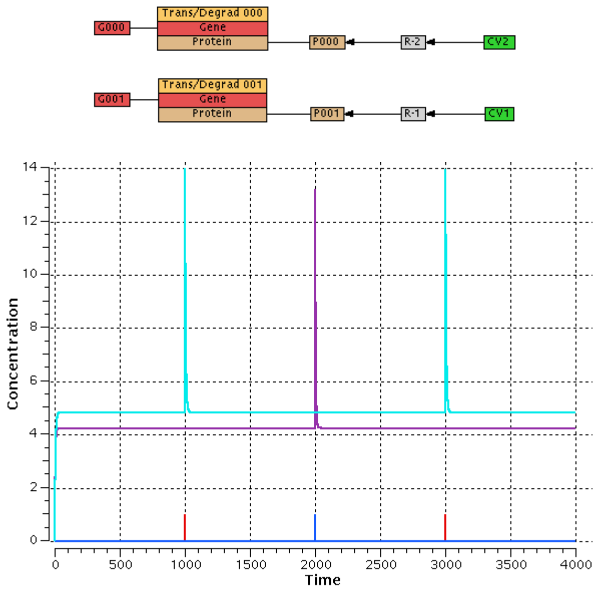


Figure 6.29: Best of Generation 0 – Evaluation score = 61.1 – Objective score = 0. The initial topology of all the individual networks in the population, each initialized with random kinetic constants. No sustained response to the control variables is observed.

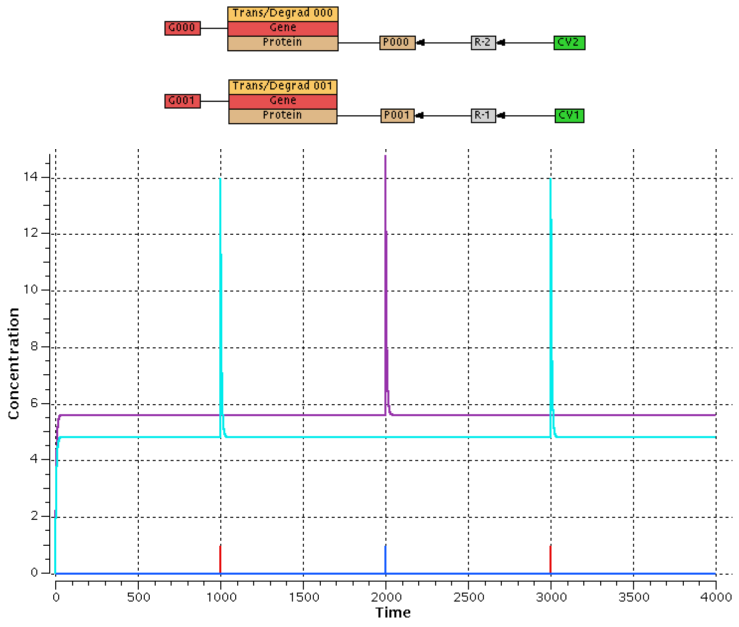


Figure 6.30: Best of Generation 5 – Evaluation score = 50.2 – Objective score = 0. The topology has not changed; all improvement has been a result of mutation of the kinetic constants. Qualitatively, no improvement has been achieved.

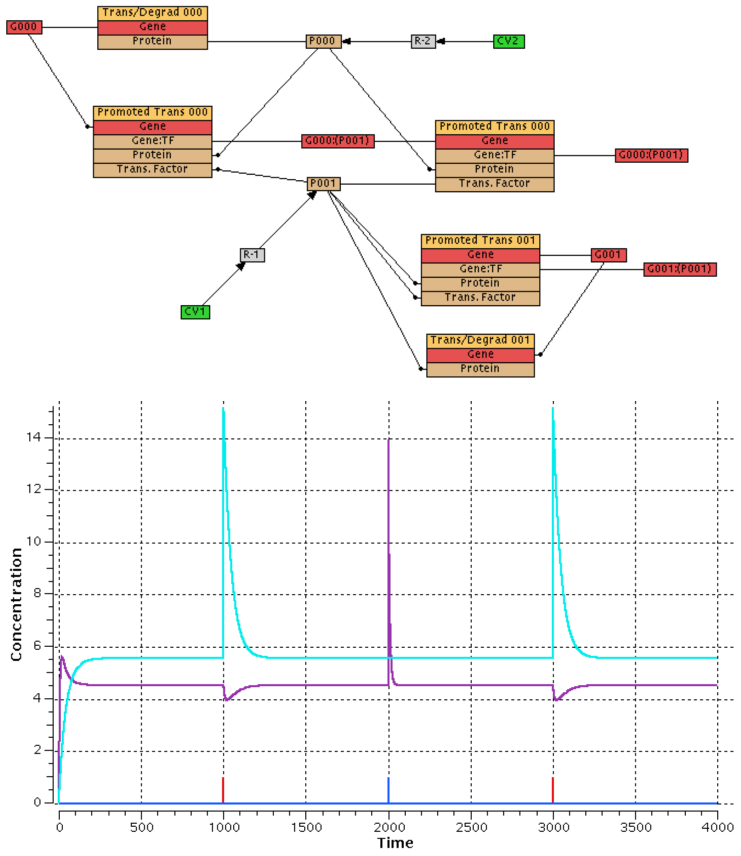


Figure 6.31: Best of Generation 10 – Evaluation score = 48.75 – Objective score = 0. Additional modules have been added to the network and the kinetic constants continue to be mutated. Some mutual regulation is starting to be demonstrated.

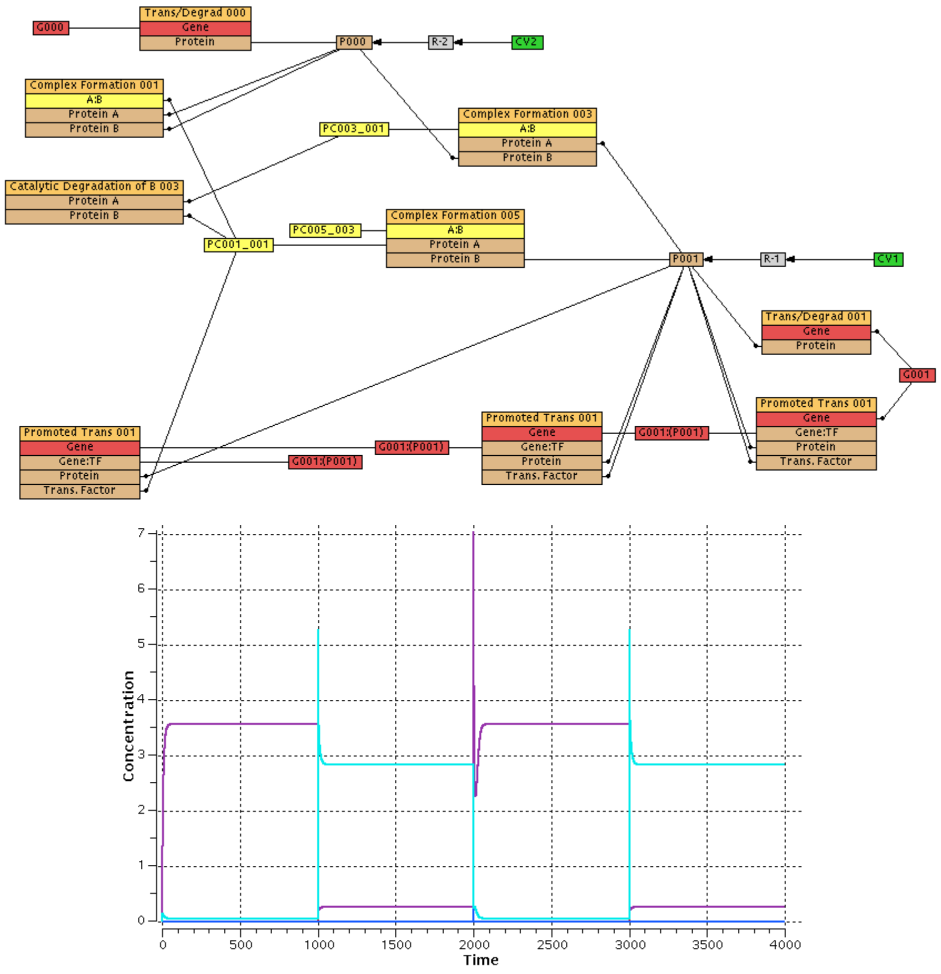
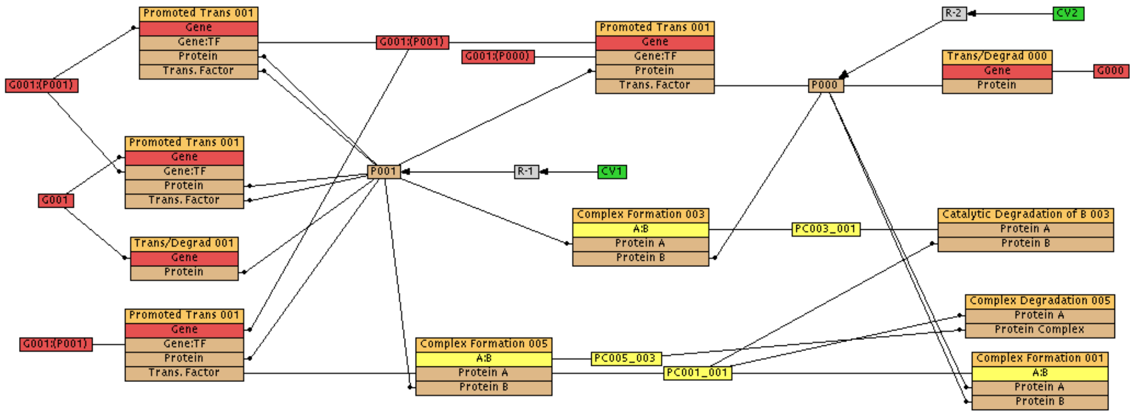


Figure 6.32: Best of Generation 15 – Evaluation score = 46.5 – Objective score = 3. The switching phenotype is observed. Additional network motifs have been added and kinetic constants mutated. The observed concentrations are not quite at the desired levels, but qualitatively this is a significant improvement.



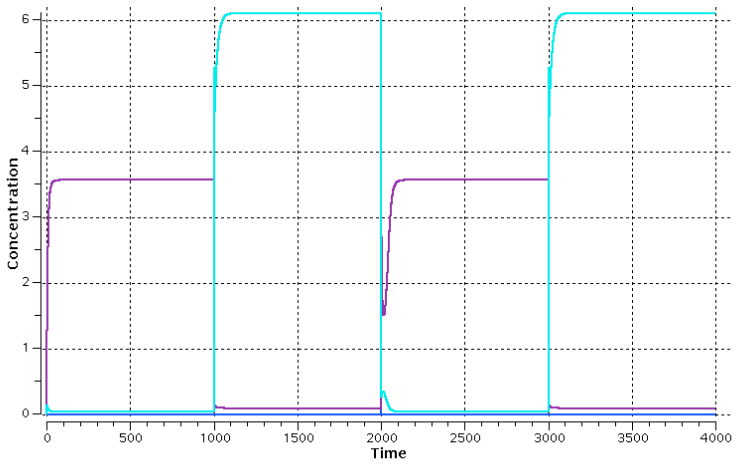


Figure 6.33: Best of Generation 20 – Evaluation score = 39.5 – Objective score = 3. The concentrations levels continue to be improved and there is less noise at transition.

After the model has captured the required phenotype, pruning is performed in order to remove modules which are not necessary. Figure 6.34 shows the final toggle switch model.

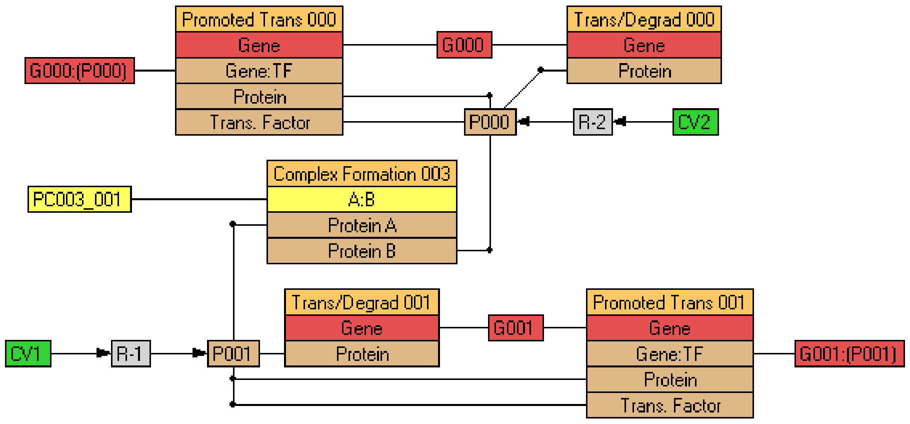


Figure 6.34: Final pruned toggle switch model

It is probable that a series of alternative networks exhibiting the desired phenotype will be found. Each of these alternatives is a possible solution itself, but better understanding may come from considering the collection as a whole. This is supported by the evolution hypothesis that details of observed biological systems are not uniquely inevitable[24,25]. (Alternative toggle switch solutions are considered in the appendix of this chapter.)

Discussion

The first two example models are simple, but they contain motifs present in more complex networks, namely protein production, degradation, and gene regulation. The more complex a model is, the more time GenoFIT will require though exact requirement is difficult to predict. Beside the number of parameters, the smoothness of the evaluation function and the quantity of local minima potentially have great influence on the computational time required.

The evaluation function is key to successful use of GenoFIT. Unintended optima will quickly be exposed if they are easier to reach than the intended optimum[26]. Evaluation functions must be crafted to avoid unintended optima. One approach that can help without making the evaluation functions overly complex is to define one evaluation function that describes the quantitative fit to data and a second that describes the qualitative fit. Individuals are first ranked by the qualitative fit and then within each qualitative class they are ranked by quantitative fit. This approach works quite well, especially when seeking networks with non-trivial qualitative behavior such as a toggle-switch.

In most cases, the evaluation function will be RMSD from experimental measurements. These data must therefore be of high quality, contain low noise, and be sufficiently information-rich to expose dynamics with measurements at time points that differentiate the species behavior. For instance, multiple values sampled from a period of stability contain may be redundant. However, if the desired behavior is a change in state in response to stimuli or oscillation, the observations must demonstrate this behavior.

Due to the under-constrained nature of the optimization being performed by GenoFIT, many parameter sets will have equal evaluation scores. However, they may vary in robustness. A particular parameter set that captures the experimentally observed dynamics may also allow the network to enter undesirable states exhibiting unexpected oscillatory, chaotic, or otherwise complex behavior. If such possibilities are of concern, subsequent analysis of candidate parameter sets by nonlinear methods may be necessary[7,27].

Although constraints reduce the size of the search space, care must be taken when constraining an evolutionary computation strategy used as a discovery method. Solutions from unconstrained searches can be interesting due to their exploitation of search space properties that were not obvious *a priori*. When forced to choose, the better choice is to expand the search space, within reasonable limits, instead of constricting the search space and removing viable solutions.

Future directions

The ability to map model parameters to an intermediate set of optimization parameters is a great benefit for reducing the parameter space of the optimization problem. Being able to specify allowed ranges for each optimization parameter further simplifies the task. However, additional restrictions can still be envisioned; for instance, some parameters may depend upon others. For example, it may be desirable to specify that protein degradation always be less than production. More generically, one could specify the allowed range of an optimization parameter to be a function of other parameters. This could be accomplished using some functional notation in the configuration file or by specifying a function written in the scripting language that is executed parallel to the evaluation function. A crude version of this is possible currently by checking, within the evaluation function, that the relationships are met and if not by assigning a low evaluation score.

GenoFIT is able to sample a regular grid of initial values to test for multiple steady states. When used in this fashion, the fitness function compares the target value to an average of the results across the observed steady states for each environment. This is suitable for situations where experimental measurements are made of an entire population of cells, effectively averaging the potentially distinctness of the individual cells. This approach exposed limitations in the ODE solver, as some of the identified steady states are actually the result of the ODE solver numerically failing without warning, must likely due to unmanaged floating-point error and the stiffness of the system of equations. Restriction of optimization parameters reduces the stiffness of the equations but does not entirely eliminate the problem. Research into the area of handling multiple steady states is quite important as non-trivial networks will contain feedback and thus be capable of exhibiting multiple steady states. The use of only ODEs, especially starting from only a single initial state, will be misleading in many cases. Further research, to improve performance of stochastic simulation, including hybrid stochastic simulation[28], is also necessary.

The current process of using GenoFIT requires models to be developed in GenoDYN, saved, and then optimized using GenoFIT. In the future we plan to unify both programs. The new program will provide an interface for manipulating models, writing objective descriptions used to specify desired behavior, and rating models produced by the exploratory process of topology evolution. This system is likely to support only discrete simulation instead of ODEs. We believe that the decision to focus on discrete simulation will enable consideration of alterative steady states and result in a more robust environment free from floating point failures. Focusing on stochastic simulation will require extending the scripting language to access statistical measures of the ensemble of trajectories.

Currently, GenoFIT is only available for Linux and other Unix-like systems by compiling on each system from the available source code. A binary could be built for Windows, but most computing clusters use Linux as their operating system, so a Windows binary has not been pursued.

Conclusion

GenoFIT combines hybrid optimization, a scripting language, and distributed computing to create a powerful model-fitting tool. Combining GenoFIT with GenoDYN yields a complete modeling environment for synthetic biology. We have shown how GenoFIT is able to fit models of various complexity, discussed pitfalls possible with evaluation functions, and discussed future directions for improvements. The ability to evolve the topology is a unique optimization method and potentially represents a novel way to develop constructs for synthetic biology.

Literature cited

1. de Jong, H. 2002. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.* **9**:67-103.
2. Gibson, M. and Bruck, J. 2000. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A* **104**:1876-89.
3. Mendes, P. 1997. Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3. *Trends Biochem. Sci.* **22**:361-3.
4. Turner, T.E., Schnell, S., and Burrage, K. 2004. Stochastic approaches for modelling in vivo reactions. *Comput. Biol. Chem.* **28**:165-78.
5. Kearfoot, R.B., Markus, N., et al. 2004. Libraries, tools, and interactive systems for verified computations four case studies. *Lect. Notes Comp. Sci.* **2991/2004**:36-63.
6. Kearfoot, R.B. 2008. Globsol user guide. J. Global Opt. To appear.
7. Ermentrout, B. 2002. *Simulating, analyzing, and animating dynamical systems: A guide to Xppaut for researchers and students.* Society for Industrial & Applied Mathematics, Philadelphia, PA, U.S.A.
8. Schnell, S. and Turner, T.E. 2004. Reaction kinetics in intracellular environments with macromolecular crowding: simulations and rate laws. *Prog. Biophys. Mol. Biol.* **85**:235-60.
9. Mendes, P. and Kell, D. 1998. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics.* **14**:869-83.
10. Moles, C.G., Mendes, P., and Banga, J.R. 2003. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res.* **13**:2467-74.
11. Yen, J., Liao, J.C., et al. 1995. A hybrid approach to modeling metabolic systems using genetic algorithm and simplex method. *Proc. 11th IEEE Conf. A. Intel. App.* 277-83.
12. Ronen, M., Rosenberg, R., et al. 2002. Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. *Proc. Natl. Acad. Sci. U.S.A.* **99**:10555-60.
13. Welch, S.M., Roe, J.L., and Dong, Z.S. 2003. A genetic neural network model of flowering time control in *Arabidopsis thaliana*. *Agron. J.* **95**:71-81.

14. Wong, P., Gladney, S., and Keasling, J.D. 1997. Mathematical model of the lac operon: inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose. *Biotechnol. Prog.* **13**:132-43.
15. Greller, L.D. and Somogyi, R. 2002. Reverse engineers map the molecular switching yards. *Trends Biotechnol.* **20**:445-7.
16. Friedman, N. 2004. Inferring cellular networks using probabilistic graphical models. *Science* **303**:799-805.
17. Yokobayashi, Y., Weiss, R., and Arnold, F.H. 2002. Directed evolution of a genetic circuit. *Proc. Natl. Acad. Sci. U.S.A.* **99**:16587-91.
18. Sakamoto, Erina and Iba, Hitoshi. 2001. Inferring stem of differential equations for a gene regulatory network by using genetic programming. *Proc. 2001 Congress Evol. Comp.* 720-726.
19. Koza, J.R. 1992. *Genetic programming: On the programming of computers by means of natural selection.* MIT Press, Cambridge, Massachusetts.
20. Francois, P. and Hakim, V. 2004. Design of genetic networks with specified functions by evolution in silico. *Proc. Natl. Acad. Sci. U.S.A.* **101**:580-5.
21. Millar, A.J. 1999. Tansley Review: Biological clocks in *Arabidopsis thaliana*. *New Phytol.* 141:175-97.
22. Goldberg, D.E. 1989. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Boston, MA, U.S.A.
23. Nelder, J.A. and Mead, R. 1965. A simplex method for function minimization. *Comput. J.* **7**:308-13.
24. Gould, S.J. 2000. *Wonderful life: The Burgess Shale and the nature of history.* W.W. Norton, New York, NY, U.S.A.
25. Fontana, W. and Buss, L.W. 1994. What would be conserved if "the tape were played twice"? *Proc. Natl. Acad. Sci. U.S.A.* **91**:757-61.
26. Bird, J. and Layzell, P. 2002. The evolved radio and its implications for modeling the evolution of novel sensors. *Proc. of the 2002 Congress on Evolutionary Comp.* **2**:1836-1841.

27. Elf, J. and Ehrenberg, M. 2003. Fast evaluation of fluctuations in biochemical networks with the linear noise approximation. *Genome Res.* **13**:2475-84.
28. Griffith, M., Courtney, T., et al. 2006. Dynamic partitioning for hybrid simulation of the bistable HIV-1 transactivation network. *Bioinformatics*. **22**:2782-9.

Appendix

Comparison of network topologies

Biochemical networks can be represented as directed graphs with nodes representing molecules and edges representing interactions or transformations of the molecules. Instead of building a network model from scratch, it may be possible to select one from a collection of networks exhibiting the correct phenotype. As additional knowledge is gained, the collection of topologies can be pruned by removing those that no longer fit the understanding of the system. For this method to be viable, a measure of graph similarity is required to discard topologies that are identical and possibly highly similar. Such a measure also provides insight into the flexibility of the network topology through measure of variation. Although the similarity could be measured considering the minimum score derived from an analysis of the maximum common subgraphs, such a process would be very slow. Instead we developed a heuristic that reasonably captures how one would rank the similarity of network topologies.

To build this heuristic, a comparison of six graph distance metrics derived from the Laplacian matrix was made on an exhaustive collection of non-isomorphic graphs up to and including size five. The graphs were generated using *geng* and *glist* composing *gtools* distributed with *nauty,* which implements specialized graph algorithms including the determination of isomorphism. Tables were constructed of measured distances between any two graphs for each metric and ranked. Examination of the most similar and especially the most dissimilar graphs was made by eye to decide if the heuristic reasonably ranked the graphs. The best heuristic compared the top eigenvalues in common between the unnormalized Laplacian matrices treating missing eigenvalues as zeroes.

Comparison of graph distance measures based on the Eigenvalues of the Laplacian Matrices

The Laplacian matrix (6.3) is defined for undirected, unweighted non-reflexive graphs to be the negative of the edge adjacency matrix with the vertex degrees along the diagonal. The Laplacian matrix is symmetric, each row sums to zero, and all the eigenvalues are nonnegative. The Laplacian matrix also has a normalized form (6.4) which considers the degrees of the nodes. The eigenvalues of the normalized Laplacian matrix are in the range [0, 2].

Definition of Laplacian matrix, where d_i is the degree of vertex i:

$$L_{i,j}(G) = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } \exists\ \mathit{edge(i,j)} \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

Definition of normalized Laplacian matrix, where d_i is the degree of vertex i:

$$L_{i,j}(G) = \begin{cases} 1 & \text{if } i = j \\ \frac{-1}{\sqrt{d_i d_j}} & \text{if } i \neq j \text{ and } \exists\ \mathit{edge(i,j)} \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

Measure comparison

Given two graphs $G1 = <N1,E1>$ and $G2 = <N2,E2>$ and the associated sorted vector of eigenvalues $\lambda 1$ and $\lambda 2$ of the Laplacian matrices for $G1$ and $G2$ respectively, we define three distance measures.

Distance measure $d1$ compare only the number of eigenvalues in common between both graphs:

$$d1 = \sqrt{\sum_{i \leq \min(|\lambda_1|,|\lambda_2|)} \left(\lambda_{1,i} - \lambda_{2,i}\right)^2} \tag{6.5}$$

Distance measure $d2$ treats eigenvalues that are not shared between the graphs as zeroes:

$$d2 = \sqrt{\sum_{i \leq \min(|\lambda_1|,|\lambda_2|)} \left(\lambda_{1,i} - \lambda_{2,i}\right)^2 + \begin{cases} |\lambda_1| > |\lambda_2| & \sum_{i=\min(|\lambda_1|,|\lambda_2|)+1}^{i<|\lambda_1|} \lambda_{1,i}^{\ 2} \\ |\lambda_1| < |\lambda_2| & \sum_{i=\min(|\lambda_1|,|\lambda_2|)+1}^{i<|\lambda_2|} \lambda_{2,i}^{\ 2} \end{cases}} \tag{6.6}$$

Distance measure $d3$ ignores eigenvalues which are zero:

$$d3 = \sqrt{\sum_{i \le \min(|\lambda_1|,|\lambda_2|)} \begin{cases} \lambda_{1,i} > 0, \lambda_{2,i} > 0 \left(\lambda_{1,i} - \lambda_{2,i}\right)^2 \\ 0 \end{cases}} \tag{6.7}$$

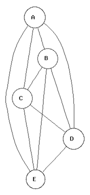
The differences between these distance measures will be noticed when comparing graphs with different numbers of nodes. All measures are symmetric, $d1(G1, G2) = d1(G2, G1)$.

Comparison of un-normalized distance measures

Given the graph *G*, characterize the distances reported by *d*1, *d*2, and *d*3 for a series of graphs *G*1–*G*9 based on an un-normalized Laplacian matrix.

Distance measure *d*3 is unusable since several of the graphs are misreported as being similar. Distance measure *d*1 distinguishes between *G*1, *G*2, and *G*3 while *d*2 treats them identically, likewise with *G*4 and *G*5. *G*6 appears the same using either *d*1 or *d*2 as desired. Any discrepancy between the two measures should occur when comparing graphs with different number of edges, with *d*2 being insensitive to differences in the number of unconnected nodes. If the desire is to have unconnected nodes affects the distance, then *d*1 appears the natural choice, but it seems intuitive that *d*1 is comparing objects of lower dimension in higher dimensional space, which may not be valid. All the distance measures, unfortunately, will indicate two graphs are identical regardless of the number of nodes if there are no edges. Neither *d*1 nor *d*2 scored *G*4 and *G*5 in desirable manner. Both only have a single edge, but *G*5 at least has a closer number of nodes to *G*, yet it is assigned a worse score. However, *d*2 scored both *G*4 and *G*5 identically instead of scoring *G*4 with a better score as *d*1 did. The progression of graphs demonstrated by *G*7, *G*8, and *G*9 also demonstrate now *d*1 inappropriately scores smaller graphs with smaller relative scores compared with the larger graphs. However, *d*2 shows an improving score as the graph is enlarged.

G:



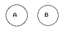
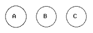
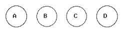
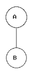
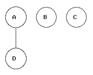
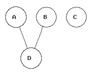
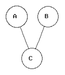
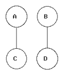
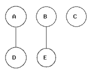
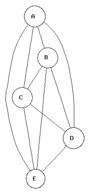
		d1	d2	d3
G1		7.1	10	0
G2		8.7	10	0
G3		10	10	0
G4		5.8	9.2	3
G5		9.2	9.2	3
G6		8.4	8.4	4.5
G7		6.7	8.4	4.5
G8		8.2	8.2	4.2
G9		8.2	8.2	4.2

Table 6.5: Comparison of un-normalized distance measures

Comparison of normalized distance measures

Given the graph *G*, characterize the distances reported by *d*1, *d*2, and *d*3 for a series of graphs *G*1–*G*9 based on a normalized Laplacian matrix.

As mentioned previously, the eigenvalues of the normalized Laplacian matrix are in the range [0, 2], which can be used to place an upper bound on the distance measure ($\sqrt{2n}$, where *n* is the number of eigenvalue pairs used in the calculation). The distance measures applied to normalized Laplcian matrices follow the same trends as seen with un-normalized Laplacian matrices. The only discrepancy is seen with $d2_n$ and $d3_n$ applied to *G*7, *G*8, and *G*9. While $d2(G,G7) > d2(G,G8)$ as desired, unfortunately $d2_n(G,G7) < d2_n(G,G8)$. In this particular case the distances associated with *G*8 are likely larger due to smaller degree vertices and thus smaller scaling factors when compared with *G*7.

G: 

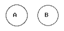
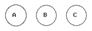
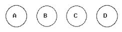
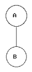
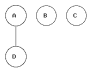
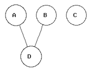
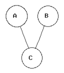
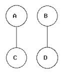
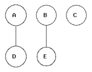
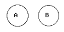
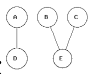
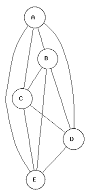
		$d1_n$	$d2_n$	$d3_n$
G1		1.8	2.5	0
G2		2.2	2.5	0
G3		2.5	2.5	0
G4		1.5	2.3	0.75
G5		2.3	2.3	0.75
G6		1.9	1.9	0.79
G7		1.5	1.9	0.79
G8		2.1	2.1	1.1
G9		2.1	2.1	1.1

Table 6.6: Comparison of normalized distance measures

The distance measures based on the normalized Laplacian matrix seem counterintuitive. Given the collection of the graphs presented on the following pages, one would expect that the graphs that *G*1 and *G*51 would be most dissimilar which is what *d*2 indicates ($d2 = 10.0$). However $d2_n$ reports *G*1 and *G*25 are the most dissimilar ($d2_n = 3.0$).

*G*1:  *G*25:  *G*51: 

Conclusion of distance measure comparison

It would seem that *d*2 is the best distance measure presented here. The use of a normalized Laplacian, while beneficial by setting an upper limit on the distance, does not appear to rank graphs in an expected way.

Subtleties of the *d*2 distance measure

Comparison of the most similar graphs of those generated is not very interesting since these have no edges and therefore have only zeroes for eigenvalues and thus $d1 = d2 = d3 = 0$. However, it is interesting for non-degenerate graphs to consider the most similar graphs. In the table below *d*2 is reported for all pairs of six graphs. When trying to find most similar graphs it may be necessary to compare in both directions for validation. For instance, if graph *A* is found to be most similar to graph *B* in a set of graphs, it might be informative to identify which graph is most similar to graph *B*.

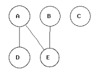
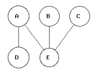
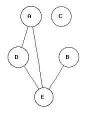
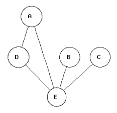
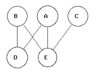
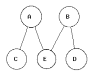
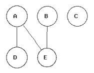
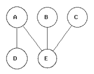
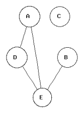
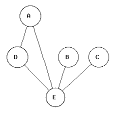
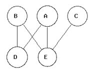
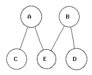
						
	0.00	1.05	1.23	2.16	2.07	1.10
	1.05	0.00	0.88	1.18	1.16	0.75
	1.23	0.88	0.00	1.41	1.42	0.77
	2.16	1.18	1.41	0.00	1.18	1.61
	2.07	1.16	1.42	1.18	0.00	1.15
	1.10	0.75	0.76	1.61	1.15	0.00

Table 6.7: Comparison of un-normalized distance measures

Identification of Alternative Switch Topologies

Using the GenoFIT, a search was performed to find alternative toggle switch topologies. Of the 220 independent runs of at most 30 generations, below are the eight successfully evolved topologies that exhibit the toggle switch phenotype. Each of these topologies has been pruned to be minimal (no further block can be removed without destroying the desired phenotype):

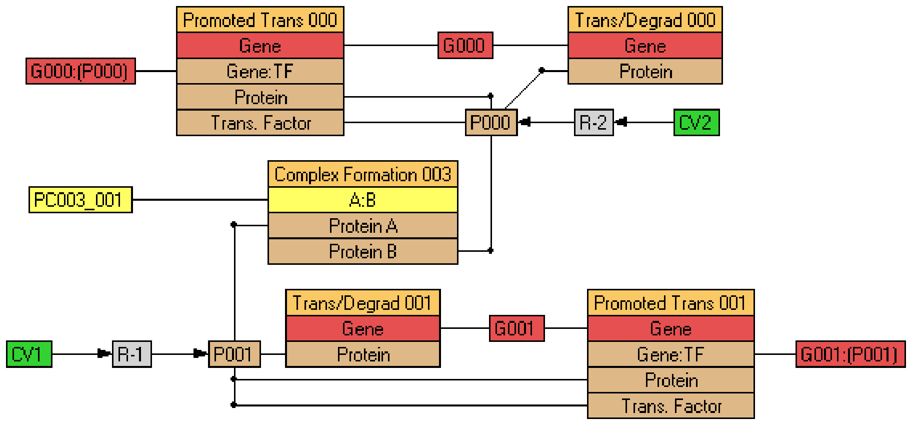


Figure 6.35: Toggle switch model S1

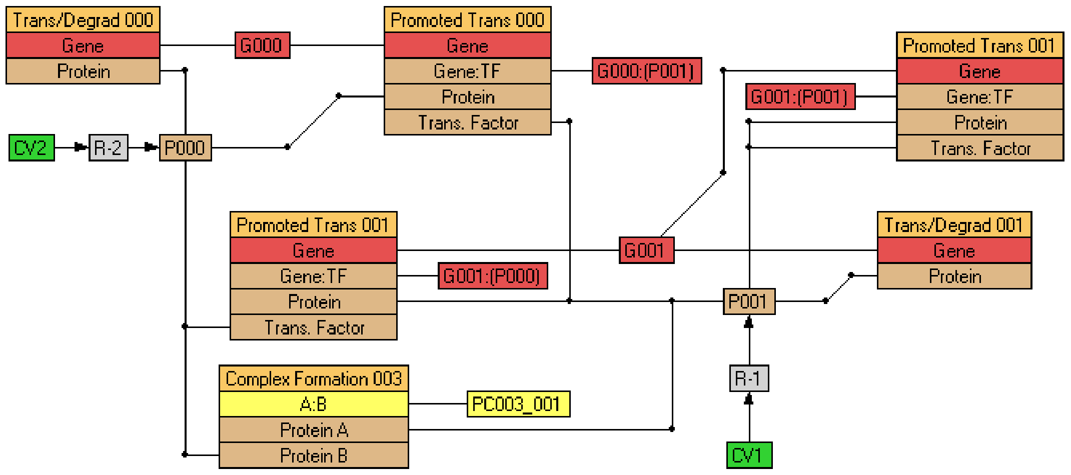


Figure 6.36: Toggle switch model S2

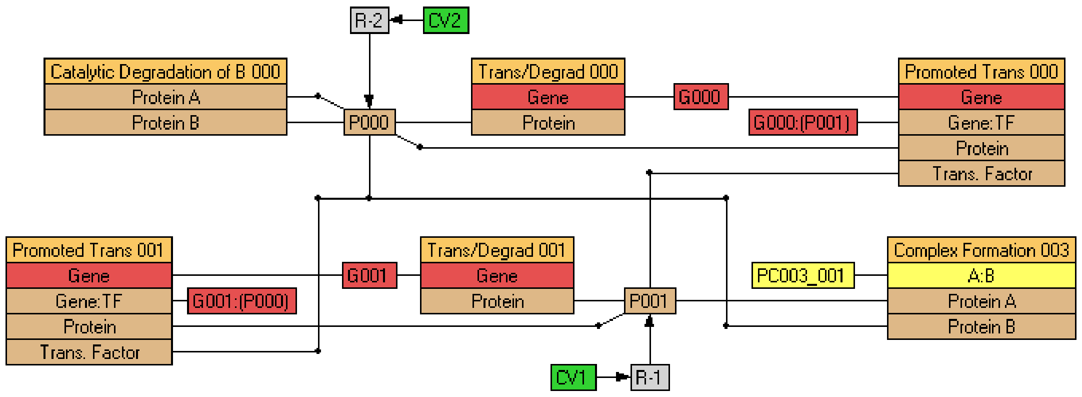


Figure 6.37: Toggle switch model S3

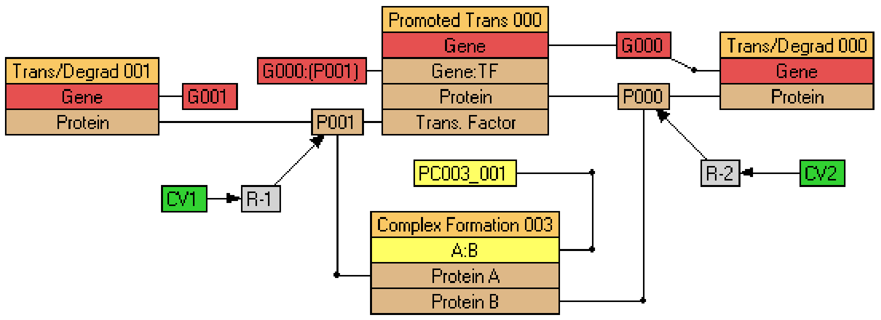


Figure 6.38: Toggle switch model S4

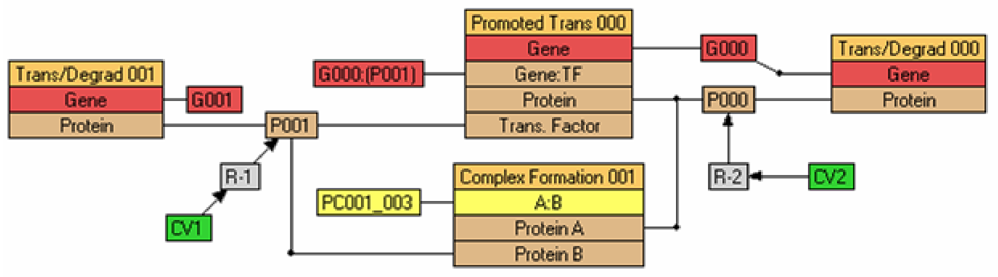


Figure 6.39: Toggle switch model S5

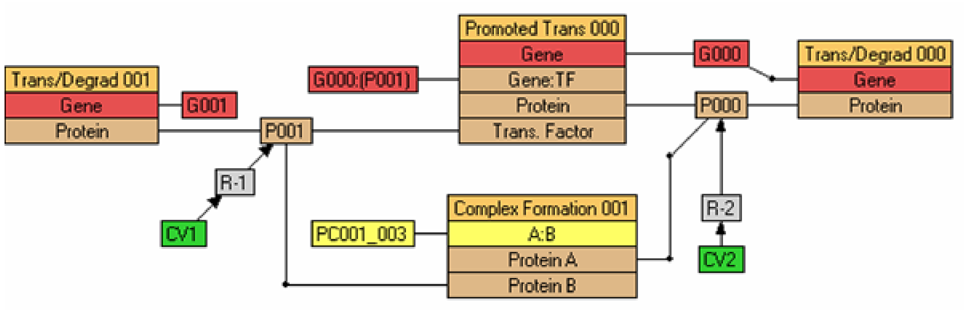


Figure 6.40: Toggle switch model S6

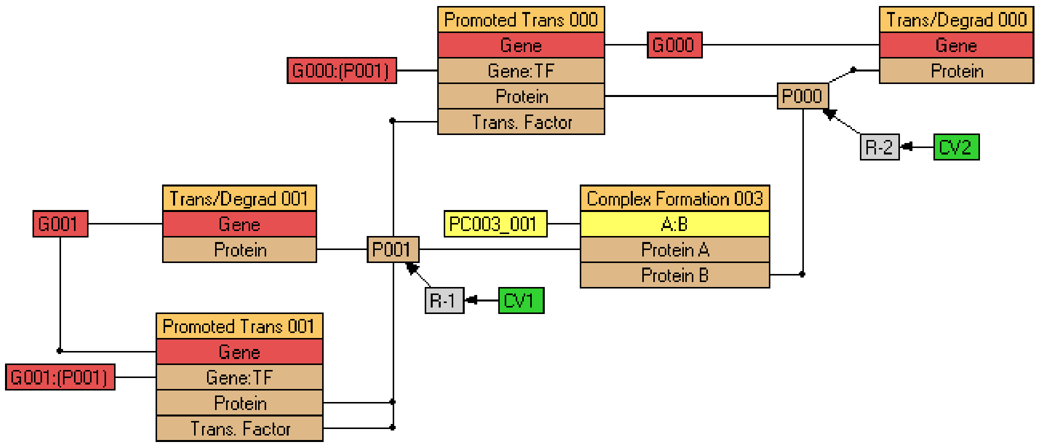


Figure 6.41: Toggle switch model S7

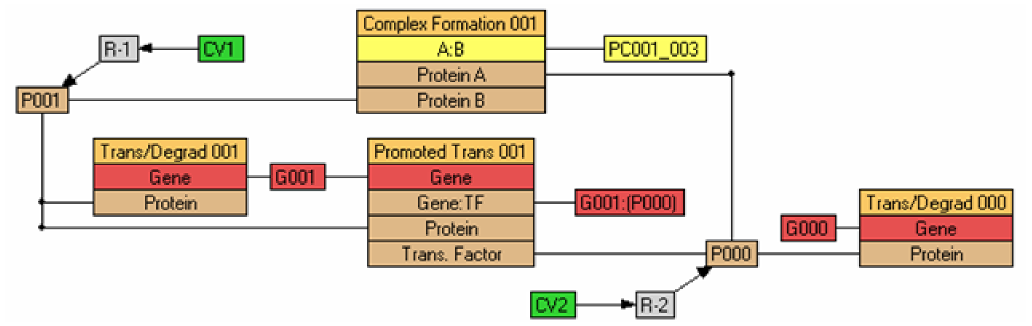


Figure 6.42: Toggle switch model S8

To identify unique networks, the *d*2 graph similarity metric was applied to build the following Table 6.8. Using this information, we are able to quickly identify toggle switches S5 (Figure 6.39), S6 (Figure 6.40), and S8 (Figure 6.42) as potential duplicates. In examining these network models we find that indeed they are duplicates and two can be discarded.

After performing the pruning of identical networks we are left with six unique network topologies that exhibit the toggle switch phenotype. If one was presented with a new system that exhibited this behavior, but they lacked sufficient knowledge of the actual topology, these possibilities may be used to guide their experiments either confirming or contradicting the suggestions. We have demonstrated the non-uniqueness of topology on the toggle switch example, but this approach could be applied to any other phenotype of interest.

	S1	S2	S3	S4	S5	S6	S7	S8
S1	0	2.726	2.072	2.533	2.533	2.533	1.063	2.533
S2	2.726	0	1.345	4.432	4.432	4.432	2.372	4.432
S3	2.072	1.345	0	3.617	3.617	3.617	1.556	3.617
S4	2.533	4.432	3.617	0	0	0	2.463	0
S5	2.533	4.432	3.617	0	0	0	2.463	0
S6	2.533	4.432	3.617	0	0	0	2.463	0
S7	1.063	2.372	1.556	2.463	2.463	2.463	0	2.463
S8	2.533	4.432	3.617	0	0	0	2.463	0

Table 6.8: Similarities between toggle switch models

GenoFIT parameter file

Below the GenoFIT parameter file for the toggle switch model considered in the text.

```
Network file: toggle.net

Population size: 100
# Must be even number if using 2pt crossover

Client-server mode: false
# Must be true or false

Work-unit size: 100
# The number of individuals sent to each client

#Master Address: 170.54.124.31
Master Address: 10.56.0.4
# The address of the machine that is responsible for dividing up work

Maximize fitness function: false
# Must be true or false

Elitism: true
# Ensure best individual survives - Must be true / false

Replace failures: true
# Replace any individual whose fitness could not be computed
# with a new random individual - Must be true / false

Simplex size: 1
# The number of top individuals to perform local area optimization on
# using the simplex algorithm.  x<=0 => disable

Time limit type: Timer
# The interpretation of the time limit placed on generation - Must be None
/ Iteration / Timer
# If limit type = Iteration => this is the maximum number of iterations
the simplex will make
# If limit type = Timer => this is the maximum amount of time the simplex
method will take
# If limit type = Timer && CS_Mode => this is the maximum amount of time
allocated to the
#                                    generation including GA specific and
Simplex

Time limit: 60
# The time limit in either seconds or iterations placed on the generation.
# See "Time limit type" for more details

Number of generations: -1
# -1 => never end

Selection pressure: 0.2

Crossover type: NPt
# Options are None, NPt, TwoPt

Shift mutation range: 0.05
# Shift values by +- 5%

Save best interval: 1
Save best prefix: best
Print population interval: 0
Print population stats interval: 1

Population identity check interval: 1
Population identity limit: 0.50

Random seed: -1
# -1 => generate one

Integer parameters: false
Parameter ranges:
# Fields _MUST_ be separated by tabs
# RxnName Index into param table Rxn Equation Kinetic value
Trans/Degrad 1::Translation 0 # Gene --> Gene + Protein
Trans/Degrad 1::Degradation 1 # Protein -->
Trans/Degrad 2::Translation 2 # Gene --> Gene + Protein
Trans/Degrad 2::Degradation 3 # Protein -->

# Gene + (Trans. Factor) --> Gene:TF
Promoted Trans 1::Complex Formation 4
```

```
# Gene:TF --> Gene + (Trans. Factor)
Promoted Trans 1::Complex Disassociation 5

# Gene:TF --> Gene:TF + Protein
Promoted Trans 1::Translation 6

# (Protein A) + (Protein B) --> A:B
Complex Formation 1::Complex Formation 7

Complex Formation 1::Degradation 8  # A:B -->

Rxn 1 9  # (CV 1) --> P1
Rxn 2 9  # (CV 2) --> P2

# Parameters
# Param Index Min Max   -or-
# Param Index Fixed Value
0 0.001 10 # 0.034376
1 0.001 10 # 0.238531
2 0.001 10 # 0.140899
3 0.001 10 # 0.058583
4 0.001 10 # 0.437839
5 0.001 10 # 0.492626
6 0.001 10 # 0.983715
7 0.001 10 # 0.985091
8 0.001 10 # 0.224507
9 0.3  # 0.300000
```

Chapter 7. Modeling the Guet library of networks

Abstract

In 2002 Guet *et al*. described the combinatorial construction of a library of synthetic networks and characterized their behavior[1]. The data from this experiment attracted the interest of several in systems biology, but little progress has been made to model these networks. Since each network is built from a common set of transcription factors and promoters, it should be possible to model them with a common framework. Here, we present our attempts to model the networks of the Guet library using knowledge of each of the components of these networks. Some of the networks can be modeled independently, a few can be modeled sharing common motifs, and some simply defy explanation.

The discussion begins with a description of the library, including components and construction approach. These details are necessary to build models that are presented next. Results of stochastic optimization of kinetic constants for these models are then presented.

2. The Experiment of Guet *et al.*

Overview

Published network diagrams make modeling regulatory networks look deceptively simple. In practice, most regulation mechanisms are simply not understood well enough to accurately model an arbitrary network, a consequence of having limited observations of complex interactions between arbitrary elements in an incompletely understood network. Guet sought to better understand regulation by removing several of these unknowns through construction of a library of artificial gene networks[1] containing all possible topologies between three of the best understood transcription factors. Each network was then placed in four different environments and their response was measured. This exhaustive approach provides several new observations, providing data for new insight into regulation and demonstrating the diversity of phenotypes possible with just a small number of regulatory elements.

Transcription factors

Each network in Guet's library contains the three prokaryotic transcription factors: LacI[2-4], TetR, and λcI[5-7]. These three transcription factors are among the most extensively studied to date, improving the chance of understanding the resulting networks.

The transcription factors were modified by the addition of an ssrA tag, a carboxy-terminal tag that reduces protein stability by allowing certain *E. coli* proteases to target the protein for early destruction[8,9]. For instance, the half-life of the λ repressor is reduced from 60 min. to 4 min[10]. This modification brings protein half-lives closer to that of mRNA, reducing the latency of regulation, and helps to avoid toxic affects due to over-expression.

Promoters

Promoters, specific to the three transcription factors, regulate the expression of the transcription factors. Each network is built using combinatorial fusion polymerase chain reaction (fusion PCR) from five available promoters (the same promoter might be used multiple times in the same network). LacI repressed two of the promoters and TetR repressed another. The remaining two promoters were regulated by λcI, one positively and the other negatively. A promoter also controlled expression of the reporter green fluorescent protein (GFP)[11], and that promoter was constant for all networks. These promoters were mutant forms engineered in previous studies to enhance the effects of regulation[12,13].

Controlling the expression of the three transcription factors are five promoters listed in Table 7.1. Several of the promoters used had mutations to enhance their regulatory control.

Reporting

Through the presence or absence of IPTG (an inhibitor of LacI) and aTc (an inhibitor of TetR) four different environments are created to observe the response of the network. The colony is assumed to reach steady state by growing overnight. The phenotype of the network is measured through GFP fluorescence. The particular GFP gene is a mutant variety called gfpmut3 which has increased half-life and 20-fold greater florescence than the wild-type[14,15].

Guet's name	Model's names	Function	Specific name	Reference

P_1^L	PL1	Repressed by LacI	$P_L$lacO1	12
P_2^L	PL2	Repressed by LacI	Poid70.5	13
P^T	PT	Repressed by TetR	$P_L$tetO1	12
P_+^λ	PL+	Positively regulated by λcI	P_{RM} + or3-r3 mutation	
P_-^λ	PL-	Negatively regulated by λcI	P_R	

Table 7.1: Name and behavior of promoters

Plasmid construction

Guet's library is constructed using a modular cloning strategy. There are five promoters and three transcription factors that are arranged to form each of the possible 125 networks. In addition, genetic support elements insure that each promoter affects the expression of only a single downstream gene and that transcription stops promptly, yielding mRNA for only one gene. To construct each of the networks by hand would be too laborious. Instead Guet developed a clever technique to build all possible networks simultaneously through directed combinatorial ligation of promoters and transcription factors.

In the initial constructs, a trailing ribosomal binding site (RBS) follows each of the five promoters and leads each of the three transcription factors. In addition, a T1 termination site, an intrinsic terminator that does not require a rho factor, follows the transcription factor. T1 has also been called an attenuator of transcription or a partial terminator which suggests that its termination can be controlled and is not guaranteed. Rho dependent termination uses rho to force dissociation of the RNA polymerase from the DNA once the polymerase pauses at a termination site due to the formation of a hairpin RNA structure. Without successful termination, a promoter may affect regulation of genes downstream from the gene associated with the promoter (Figure 7.1). Please note in the following figures that $P_i \in \{P_1^L, P_2^L, P^T, P_+^\lambda, P_-^\lambda\}$ and $TF_j \in \{LacI, \lambda cI, TetR\}$.

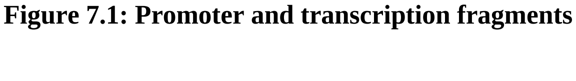


Figure 7.1: Promoter and transcription fragments

Next, portions of each of the promoter fragments, including a forward primer that contains a Bgl I restriction enzyme site, are amplified, as is the transcription factor fragment, including a reverse primer containing a Bgl I restriction site (figure 7.2). The specific sequence used for the Bgl I sites depends on the associated transcription factor. This is further explained below.

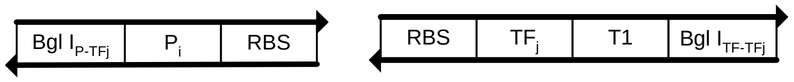


Figure 7.2

Now, a mixture of the transcription factor construct and the five associated promoter constructs is created and the dsDNA is denatured (Figure 7.3).

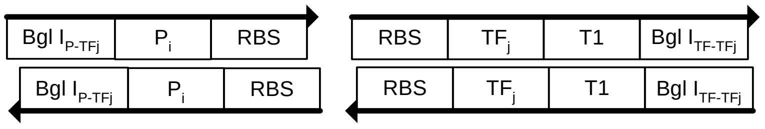


Figure 7.3

During the annealing stage, the RBS site acts as a common tag to ligate the promoter fragment with the transcription factor fragment (Figure 7.4).

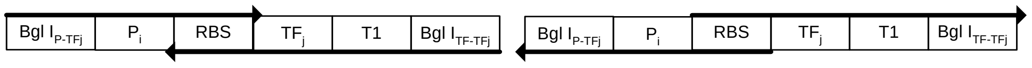


Figure 7.4

Ligation is followed with an amplification step to clean up the dsDNA creating a mixture of five different promoter:transcription factor constructs with flanking restriction sites (Figure 7.5). These steps are done for each of the three transcription factors independently, resulting in each of the possible promoter:transcription factor gene pairings.

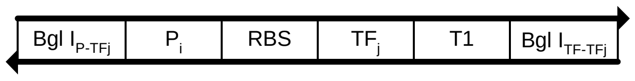


Figure 7.5

Now, we must return to the specifics of the Bgl I restriction enzyme. Bgl I cleaves dsDNA in the following way, where N represents an indiscriminate site that can be occupied by any nucleotide (Figures 7.6 and 7.7).

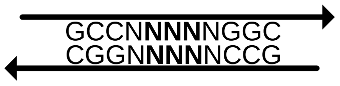


Figure 7.6

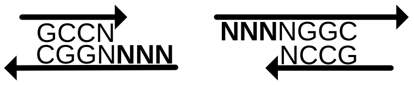


Figure 7.7: After addition of Bg1 I restriction enzyme

The overhanging sections are characteristic of a restriction enzyme that leaves sticky ends, with the sticky ends forming specific tags since Bgl I ignores these nucleotides. These sticky ends are used in a later step to enforce a particular arrangement of the promoter:transcription factor pairs and insertion into a plasmid. The ordering is achieved through neighboring promoter:transcription factor constructs having complementary sticky tags (Figure 7.8).

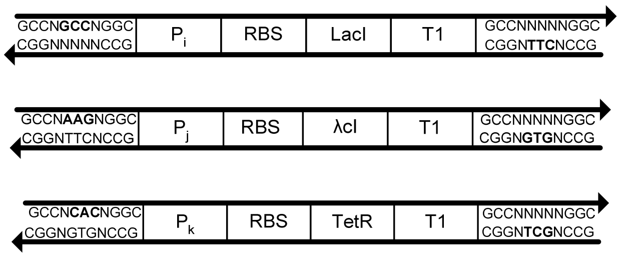


Figure 7.8

The final phase in the library creation ligates together the three promoter:transcription factor pairs. A mixture of all the promoter:transcription factor pairs is created and Bgl I restriction enzyme added producing sticky ends. The mixture contains the following fragments in addition to several smaller fragments from the restriction enzyme (Figure 7.9).

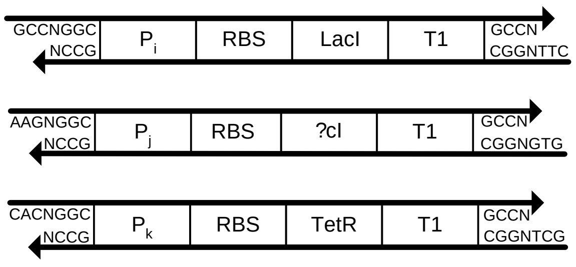


Figure 7.9

The dsDNA fragments ligate together in an order enforced by complementation of the sticky ends.

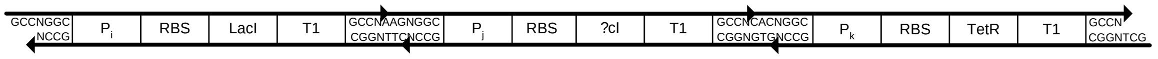


Figure 7.10

This full length strand is then ligated into an expression vector and cloned into plasmids. The plasmids are designed to have Dra III sites (cac**nnn**/gtg) around the sacrificial kan^R gene. These Dra III sites are designed such that after digestion there are sticky ends compatible with the ends of the promoter-gene constructs. The digested plasmid has the structure P_-^{λ}-gfpmut3-T1-SC101*-bla. SC101* is the origin of replication. The bla gene confers ampicillin resistance to the host and can be used to purify the colony. It is likely that more than one plasmid was introduced into each cell.

E. coli Strains

Each network is inserted into two different *E. coli* strains: CMW101 and DH10B. These strains differ by the presence of a wild-type lacI gene, with CMW101 being lacI- and DH10B being lacI+. Both strains are tetR-. Data for two networks D038 and D052 is available for both strains. However, only lacI- (strain CMW101) data are available for the other networks. As one would expect, the data do demonstrate a significant difference between GFP expression for the same network in different strains.

An assumption that the lacI- mutant did not affect the other components of the lac operon has been made here regarding CMW101, i.e. lacZ (β-galactosidase), lacY (the permease), and lacA (transacetylase). Since LacI is a trans-acting transcription factor, its presence on the plasmid will also regulate of the lac operon on the host chromosome.

The CMW101 strain uses MC1061 as a base and transduction is with bactriophage P1. Guet's paper indicates specifically "transduction of a recA::cam marker from CLC90." We assume this means that a section of DNA from CLC90 is inserted into MC1061 that includes the recA gene (the product of which is an enzyme that encourages recombination events and is used to displace homolog sequences), a cam gene to provide chloramphenicol resistance to select transformed cells, and a lacI mutant.

Environments

The transcription factors LacI and TetR can be inhibited by isopropyl β-D-thiogalactopyranside (IPTG) and anhydrotetracycline (aTc), respectively. These inducers bind to LacI and TetR changing the conformational state of the protein and preventing binding to their respective promoters. By controlling the presence or absence of IPTG and aTc, the *E. coli* colonies can be in four distinct environments (IPTG-/aTc-, IPTG+/aTc-, IPTG-/aTc+, and IPTG+/aTc+ at concentrations of 1mM IPTG and 100 ng/mL aTc). Within each environment, the network's phenotype is determined by measuring GFP florescence.

Measurements

Two different types of analysis are performed. The florescence measurement of a colony is performed on 30 different networks. FACS (florescence activated cell sorter) data is also available for two networks (D038 and D052). FACS data provides a distribution of the florescence by measuring the florescence of each cell individually instead of averaging the measurements as is done in the other style of experiment. Unfortunately, the actual FACS data are not available.

Topologies

Any of the five promoters may be associated with each of the three transcription factors, with repetition, yielding 5^3=125 possible networks before mutations. Some of the networks will have unconnected components, and this is allowed since the only affect is loss of control by one or both inducers. The four environments in which each network is phenotyped and the two different *E. coli* strains in which the plasmids are inserted increase the number of possible data points to 125 x 4 x 2=1000. However, genotype and phenotype data for only 30 networks in one strain across all four environments in triplicate are available. Care must be taken with the available data since mutants are included. The effects of mutations include phenotypic variation for the same network topology complicating otherwise clean experimental data.

The construction of these networks was not without error. The constructed plasmids were sequenced and mutations in the regulatory elements and gene sequence were noticed. These mutations affected the phenotype of the network.

Limitations

Guet's library includes several networks that may exhibit multi-state or oscillatory behavior. Most obvious is d123 which has the same structure as the repressilator, a network constructed and observed to oscillate *in vivo*[10], and d180, which should exhibit bi-stability[16]. Unfortunately, the GFP florescence measurement method measurement used by Guet averages the expression of the total colony and could not have detected these unless the expression of the individuals in the colony was in sync. This results in expression data missing the presence of interesting dynamics.

3. Modeling the Guet networks with knowledge

Overview

The experiment by Guet *et al*. provides new insight into how apparently simple prokaryotic regulatory mechanisms operate when placed in different network topologies and different environments. This exhaustive exercise may fully elucidate the behavior of an entire class of network, by phenotyping every network. However, this approach is undesirable if one is trying to build a gene network for a specific purpose. The traditional engineering approach would be first to design and model the system computationally before the actual construction. It is the modeling of these networks that is the goal of this work.

Guet's networks were chosen instead of networks of commercial or therapeutic interest for the same reason that components of which the networks were constructed were chosen: to remove unknowns by starting with a well-understood system. Before tackling complex eukaryotic systems comprised of genes of which little is known, a convincing case has to be built supporting the efforts of modeling. To do this a system built of well-understood elements, having lots of experimental data, and made up of a less complex prokaryotic system is ideal.

Initial models of each of the thirty networks for which Guet provided expression data for have been constructed. These models have been improved as details of mechanisms of the transcription factors with the promoters and with the environment (IPTG and aTc) have become better understood.

Several assumptions have been made in the models to reduce their complexity, but also to avoid constraining them by introducing incompletely understood mechanisms. The main assumptions are 1) the network is independent from the rest of the plasmid and the chromosome; 2) the transcription factors and promoters are building blocks shared between the networks, with their behavior being constant regardless of context; and 3) time delays for transcription and translation were ignored as were effects of mutation. The models of the lac operon[3], λcI[17-19], and gfp[20] have been previously described.

Reactions are explicitly modeled rather than using the restricted Michaelis-Menten enzyme rate law, which is common in the literature although questioned[21]. However, any modeling formalism is likely to be inaccurate in some circumstances though. For instance, the use of mass-action equations assumes uniform environment and accessibility, not present in a cell where physical obstacles limit molecular mobility.

Limitations

The choice of mass-action-only equations presents a problem when the exact mechanism is not fully understood. In such a case, if differential equations are being manipulated directly, such modifications can be easily approximated with a new term. Using only mass-action equations forces one to find a plausible explanation of the desired term.

Degradations are modeled as first order reactions – the greater the concentration of the molecule or the larger the kinetic parameter the faster the degradation. This ignores the limited supply of protease in the cell. Alternatively, degradations can rely on an explicit limited pool of protease or approximate this by using a Michaelis-Menten reaction and the appropriate value of $k_{\max}$[20].

One possible problem with the networks presented here is that there is only a single copy of each gene. However, in Guet's paper there were roughly 15 copies of the plasmid per cell.

Example

What follows is an example of how the knowledge of the various promoters can be used to construct a network model, in this particular case one that Guet entitled D038. Represented as a graph (Figure 7.11), the network model is composed of edges, representing flow of molecules indicated by the direction of the arrow, and nodes, representing molecules or reactions. Edges can have weights, shown near the termination of the edge, indicating the number of molecules of a particular type moving along this edge. In the Guet models the numbers capture whether the molecule is acting alone, as a dimer, or as a tetramer. Molecules can only be directly connected to reactions, shown in gray. Transcription/translation reactions are labels starting with T1 through T4. The R1 and R2 reactions represent the diffusion of IPTG and aTc across the cell membrane. The other reactions model promoters, PTa, PTb, PL2, and PL-a, where P stands for promoter, T for TetR, L for LacI, and L- and L+ for negative and positive promoters associated with lambda cI. The transcription factors and GFP are shown in orange and genes are shown in red. Given that the promoters attenuate gene transcription, there are two states for each gene, an on and an off state. The inactive form is designated with a trailing capital gX while the active form of the gene has a trailing g. LacI and TetR can also be in two states depending on the presence of IPTG and aTc hence the two yellow molecules that represent the inhibited forms. Two control variables, the green nodes, model external activators aTc and IPTG used to create the four environments.

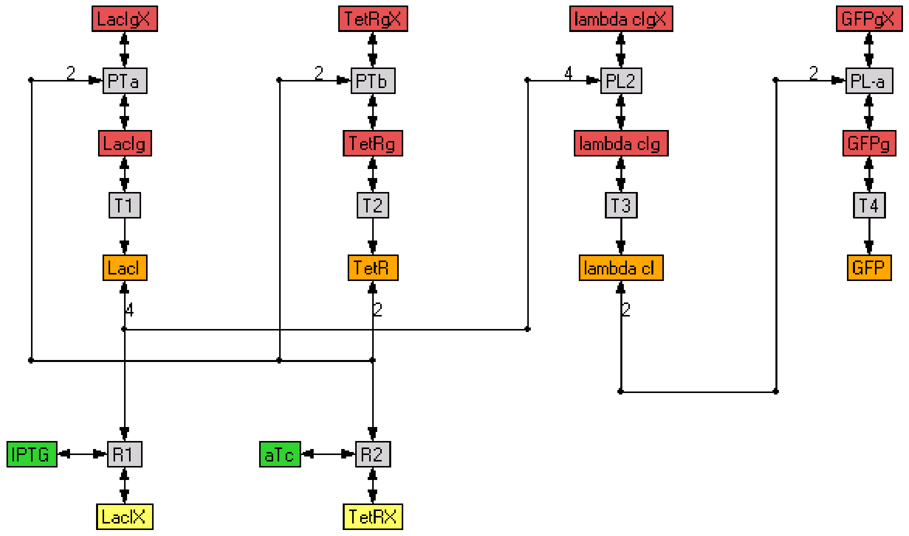


Figure 7.11: Example of network model

The signals of the two control variables are seen in Figure 7.12 as are the corresponding concentrations of each of the molecules with each transition and over time. For the D038 model to have the correct phenotype it must have a high concentration of GFP when aTc is high and IPTG is low. In all other environments it must have a low concentration. The graph shows that the model exhibits this behavior strongly. To emphasize the relationship between the concentrations of the control molecules and GFP, these variables are plotted with respect to each other in Figure 7.13.

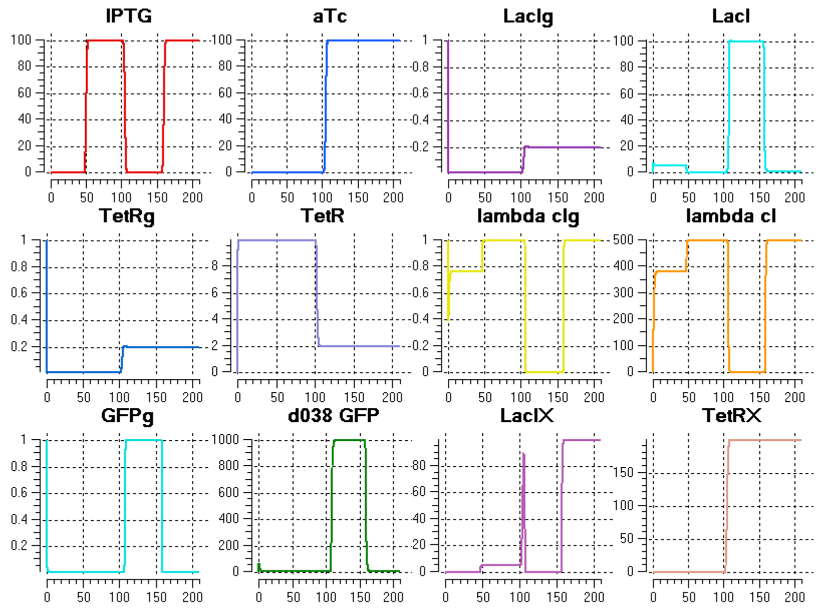


Figure 7.12: Simulated results of network model

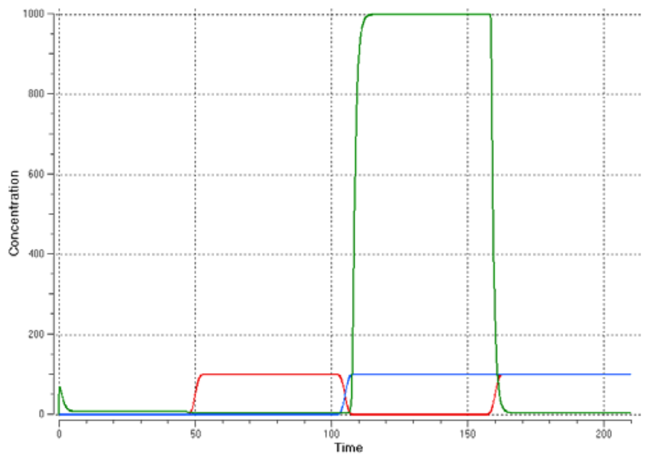


Figure 7.13: Simulated results of network model emphasizing GFP and environment

Optimization of individual Guet networks

Each of the following networks is optimized independently for twenty generations. After optimizing each network independently, pairs of networks with solutions identified are optimized. This increase in constraints helps to remove incorrect solutions.

The networks are constructed in two strains, *lacI*- and *lacI+*, that differ by the "presence of a wild-type copy of the *lacI* gene at a chromosomal locus." We focus on the lacI- data since the extra copy is controlled by unknown mechanisms.

It is possible, due to the limited number of optimization iterations applied to each network, that the GA component of GenoFIT was insignificant and that most of the improvement came through simplex optimization. It would be interesting to perform simplex optimization on a large number of random parameter sets and to compare the performance with GA+Simplex or even with GA by itself. The simplex method was enabled for this runs because it improved the rate of convergence but also because that it improved CPU utilization when run in a distributed mode.

Tables 7.2 and 7.3 below show, for each network, the mean of the experimental measurements as well as the values from the best network to result from the model fitting process. Subjectively large differences between the experimental and computational results are highlighted.

The "Feedback Present" column indicates if the network contains feedback; if so, it gives the minimal number of steps along in the cycle. For example, D012 contains the feedback cycle of LacI controlling LamCI which in turn controls LacI, and therefore the number of steps is 2. Compare this with D019 in which TetR self regulates and thus the number of steps is only 1. A network such as D016 has no feedback involved in the portion controlling GFP and therefore all controlling signals are feed-forward.

Network	Feedback Present	Measure	Env1	Env2	Env3	Env4
D012	Y(2)	Mean	9799.553	135.4443	384.7536	207.8155
		Opt	9797.253	270.4756	270.5110	270.4756
D016	N[1]	Mean	28855.19	1058.356	769.8561	633.8722
		Opt	14812.52	846.114	14812.5213	846.1142
D018	Y(2)	Mean	5746.876	4954.329	6977.721	6682.387

[1] Feedback is present but on a disconnected section that should have no influence on GFP

		Opt	6090.328	6090.328	6090.328	6090.328
D019	Y(1)	Mean	331.4371	529.6587	10787.62	168.6702
		Opt	343.2553	343.2553	10787.62	343.2553
D028	Y(2)	Mean	571.3948	658.9516	284.0915	255.054
		Opt	615.1732	615.1732	269.5727	269.573
D032	Y(2)	Mean	48101.94	1047.139	48895.48	737.5005
		Opt	48498.71	892.320	48498.71	892.3197
D038	Y(1)	Mean	427.4511	550.8609	13733.83	235.2039
		Opt	427.4706	393.0426	13733.83	393.0426
D052	Y(1)	Mean	13557.13	467.7863	435.0814	686.2104
		Opt	7012.46	7012.4552	434.6700	686.3325
D066	Y(3)	Mean	7853.719	1621.622	410.1682	435.9176
		Opt	4737.332	4737.332	419.3673	428.6299
D078	Y(3)	Mean	102.9218	146.2082	48589.24	407.9522
		Opt	124.5649	124.5650	48589.24	407.9522
D090	Y(2)	Mean	203.558	162.6739	92.84085	125.2698
		Opt	148.199	143.9718	148.19943	143.9718
D101	N[2]	Mean	38861.47	591.6231	4654.478	885.175
		Opt	11248.19	11248.1892	11248.189	11245.659
D104	Y(1)	Mean	215.8781	210.1672	234.766	343.5589
		Opt	251.0925	251.0925	251.093	251.0925
D113	Y(2)	Mean	35163.88	1836.717	38297.85	1125.72
		Opt	36730.98	1481.107	36730.98	1481.11
D114	Y(2)	Mean	218.8726	276.0236	446.396	416.1289
		Opt	339.3553	339.3553	339.355	339.3553
D117	Y(2)	Mean	45398.87	46253.63	51533.72	52351.76
		Opt	48884.32	48884.50	48884.50	48884.50
D123	Y(3)	Mean	653.3381	866.5852	402.461	282.9939
		Opt	653.5507	866.5706	342.659	342.6601
D133	Y(2)	Mean	43474.82	44461.52	47795.19	53051.28
		Opt	47195.70	47195.70	47195.70	47195.70
D135	Y(1)	Mean	6088.122	5654.586	5820.305	5765.542
		Opt	5832.139	5832.139	5832.139	5832.139
D143	Y(1)	Mean	14218.99	14124.03	568.3894	451.3978
		Opt	14171.51	14171.51	509.8936	509.8936
D180	Y(2)	Mean	47496.43	48890.79	791.2396	885.5921
		Opt	48090.86	48829.78	1021.5488	1043.0039
D250	Y(1)	Mean	795.7318	296.989	354.3591	182.9879
		Opt	575.0454	239.988	575.0454	239.9884
D253	Y(1)	Mean	885.4587	799.1516	325.6498	204.3227
		Opt	553.6457	553.6457	553.6457	553.6457
C024	Y(1)	Mean	13922.28	439.0846	538.8381	345.9892
		Opt	7229.77	391.9829	7229.7711	391.7693
C101	Y(3)	Mean	176.7035	9121.306	297.036	226.886
		Opt	233.5376	9121.306	233.538	233.550

[2] Feedback is present but on a disconnected section that should have no influence on GFP

C103	Y(3)	Mean	11305.01	2440.715	818.0481	653.8954
		Opt	6872.86	6872.862	735.9717	735.9717
C113	Y(3)	Mean	13454.04	1798.284	447.3287	530.8328
		Opt	7626.22	7626.219	489.1356	489.1523
C144	Y(1)	Mean	31623.44	4857.925	534.1963	656.0131
		Opt	18240.68	18240.678	534.1875	656.0676
C195	Y(2)	Mean	46271.42	18231.93	68665.37	56802.93
		Opt	45019.03	37746.30	57351.96	37746.30
C242	Y(1)	Mean	17403.9	715.8412	567.6562	442.334
		Opt	9059.9	9059.8707	504.9951	504.995

Table 7.2

Network	Feedback Present	Measure	Env1	Env2	Env3	Env4
D012	Y(2)	Mean	9800	135	385	208
		Opt	9797	270	271	270
		Error	-0.02%	99.70%	-29.69%	30.15%
D016	N(2)	Mean	28855	1058	770	634
		Opt	14813	846	14813	846
		Error	-48.67%	-20.05%	1824.06%	33.48%
D018	Y(2)	Mean	5747	4954	6978	6682
		Opt	6090	6090	6090	6090
		Error	5.98%	22.93%	-12.72%	-8.86%
D019	Y(1)	Mean	331	530	10788	169
		Opt	343	343	10788	343
		Error	3.57%	-35.19%	0.00%	103.51%
D028	Y(2)	Mean	571	659	284	255
		Opt	615	615	270	270
		Error	7.66%	-6.64%	-5.11%	5.69%
D032	Y(2)	Mean	48102	1047	48895	738
		Opt	48499	892	48499	892
		Error	0.82%	-14.78%	-0.81%	20.99%
D038	Y(1)	Mean	427	551	13734	235
		Opt	427	393	13734	393
		Error	0.00%	-28.65%	0.00%	67.11%
D052	Y(1)	Mean	13557	468	435	686
		Opt	7012	7012	435	686
		Error	-48.27%	1399.07%	-0.09%	0.02%
D066	Y(3)	Mean	7854	1622	410	436
		Opt	4737	4737	419	429
		Error	-39.68%	192.14%	2.24%	-1.67%
D078	Y(3)	Mean	103	146	48589	408
		Opt	125	125	48589	408
		Error	21.03%	-14.80%	0.00%	0.00%

D090	Y(2)	Mean	204	163	93	125
		Opt	148	144	148	144
		Error	-27.20%	-11.50%	59.63%	14.93%
D101	N(2)	Mean	38861	592	4654	885
		Opt	11248	11248	11248	11246
		Error	-71.06%	1801.24%	141.66%	1170.44%
D104	Y(1)	Mean	216	210	235	344
		Opt	251	251	251	251
		Error	16.31%	19.47%	6.95%	-26.91%
D113	Y(2)	Mean	35164	1837	38298	1126
		Opt	36731	1481	36731	1481
		Error	4.46%	-19.36%	-4.09%	31.57%
D114	Y(2)	Mean	219	276	446	416
		Opt	339	339	339	339
		Error	55.05%	22.94%	-23.98%	-18.45%
D117	Y(2)	Mean	45399	46254	51534	52352
		Opt	48884	48885	48885	48885
		Error	7.68%	5.69%	-5.14%	-6.62%
D123	Y(3)	Mean	653	867	402	283
		Opt	654	867	343	343
		Error	0.03%	0.00%	-14.86%	21.08%
D133	Y(2)	Mean	43475	44462	47795	53051
		Opt	47196	47196	47196	47196
		Error	8.56%	6.15%	-1.25%	-11.04%
D135	Y(1)	Mean	6088	5655	5820	5766
		Opt	5832	5832	5832	5832
		Error	-4.20%	3.14%	0.20%	1.16%
D143	Y(1)	Mean	14219	14124	568	451
		Opt	14172	14172	510	510
		Error	-0.33%	0.34%	-10.29%	12.96%
D180	Y(2)	Mean	47496	48891	791	886
		Opt	48091	48830	1022	1043
		Error	1.25%	-0.12%	29.11%	17.77%
D250	Y(1)	Mean	796	297	354	183
		Opt	575	240	575	240
		Error	-27.73%	-19.19%	62.28%	31.15%
D253	Y(1)	Mean	885	799	326	204
		Opt	554	554	554	554
		Error	-37.47%	-30.72%	70.01%	170.97%
C024	Y(1)	Mean	13922	439	539	346
		Opt	7230	392	7230	392
		Error	-48.07%	-10.73%	1241.73%	13.23%
C101	Y(3)	Mean	177	9121	297	227
		Opt	234	9121	234	234
		Error	32.16%	0.00%	-21.38%	2.94%

C103	Y(3)	Mean	11305	2441	818	654
		Opt	6873	6873	736	736
		Error	-39.21%	181.59%	-10.03%	12.55%
C113	Y(3)	Mean	13454	1798	447	531
		Opt	7626	7626	489	489
		Error	-43.32%	324.08%	9.35%	-7.85%
C144	Y(1)	Mean	31623	4858	534	656
		Opt	18241	18241	534	656
		Error	-42.32%	275.48%	0.00%	0.01%
C195	Y(2)	Mean	46271	18232	68665	56803
		Opt	45019	37746	57352	37746
		Error	-2.71%	107.03%	-16.48%	-33.55%
C242	Y(1)	Mean	17404	716	568	442
		Opt	9060	9060	505	505
		Error	-47.94%	1165.63%	-11.04%	14.17%

Table 7.3

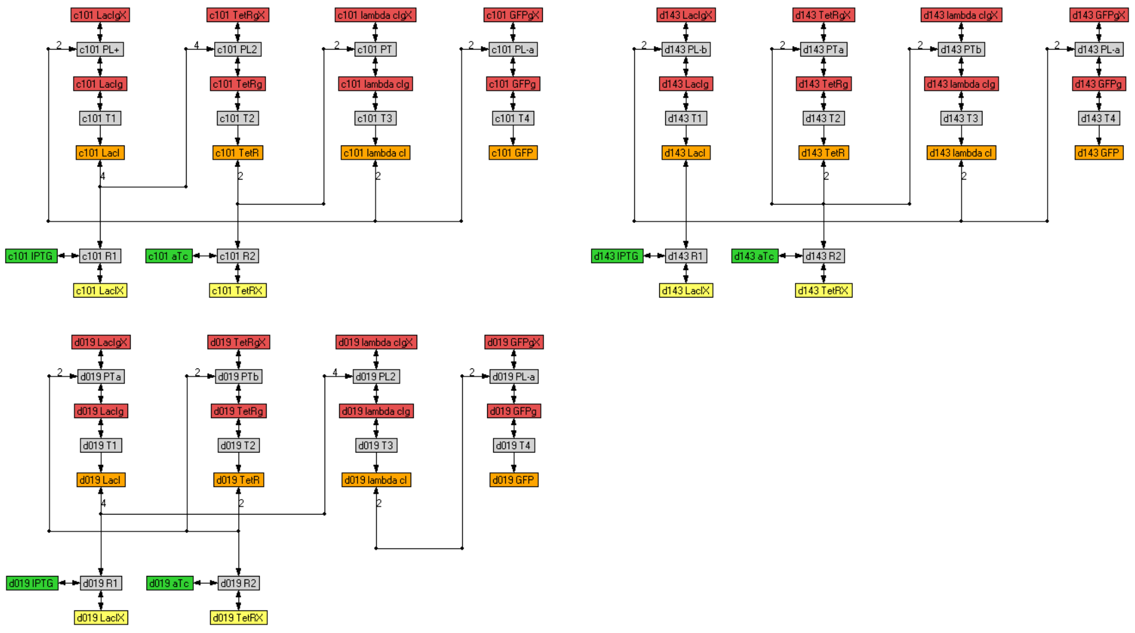


Figure 7.14: Reconciliation of three of Guet's networks into a single model

Figure 7.14 shows a model composed of three independent networks based on shared parameters and regulation motifs. Despite its complexity, one can see the main features. Each box represents either a reaction (grey boxes) or a molecule (colored boxes). Lines represent the flow of molecules with the arrowheads indicating the direction of flow. Molecules required by a reaction flow into the reaction box and the reaction product flows out. The structure of this network helps to identify regulation motifs that have appear as vertical structures of molecules and reactions. The labels help to identify the individual networks by the prefix (e.g. c101, d143, d019) and the label suffixes identify the network component. Like suffixes (ignoring the lowercase ‘a’ and ‘b’ that are required only to give each entity a unique name) within and between networks identify reactions or molecules with similar constraints, i.e. the building blocks.

Conclusions

Models of thirty networks from Guet’s library have been created in the modeling environment and fitted to the experimental data provided by Guet. While most models do not fit well in all environments, at least three of those that could be fitted individually could also be reconciled into a single model supporting the possibility of model independence of regulation mechanisms. Reviewing the literature surrounding the specific promoters, transcription factors, and protocols surrounding the original experiment helped to identify inaccuracies in the modeling of certain regulatory mechanisms that could be responsible for some of the failures. This work should be revisited from a synthetic view, i.e. disregarding the experimental data initially and trying to fit generated data to the models, much like what will be done in the characterization phase. This removes the possibility that the model is wrong but still allows testing whether regulation motifs can be modeled independently.

Guet used a combinatorial approach to build artificial networks by treating transcription factors and promoters as building blocks in a similar way as this research tests if network models can be built using regulation motifs as building blocks. Although the identical components may be shared between the various networks in Guet’s library, it does not immediately mean that they behave identically regardless of their context.

The final goal of a single unifying framework is still elusive but hopefully this work represents an additional step forward.

Literature cited

1. Guet, C.C., Elowitz, M.B., *et al.* 2002. Combinatorial synthesis of genetic networks. *Science* **296**:1466-70.

2. Muller-Hill, B. 1996. *The lac operon: A short history of a genetic paradigm.* Walter de Gruyter & Co. Koln, Germany.
3. Vilar, J.M., Guet, C.C., and Leibler, S. 2003. Modeling network dynamics: the lac operon, a case study. *J. Cell Biol.* **161**:471-6.
4. Wong, P., Gladney, S., and Keasling, J.D. 1997. Mathematical model of the lac operon: inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose. *Biotechnol. Prog.* **13**:132-43.
5. Ptashne, M. 1992. *Genetic switch: Phage lambda and higher organisms*. Blackwell Science Inc.
6. Ptashne, M. 2004. *Genetic switch: phage lambda revisited*. Cold Spring Harbor Press, Cold Spring Harbor, N.Y.
7. Ptashne, M. and Gann, A. 2001. Genes & signals. Cold Spring Harbor Press, Cold Spring Harbor, N.Y.
8. Gottesman, S., Roche, E., *et al*. 1998. The ClpXP and ClpAP proteases degrade proteins with carboxy-terminal peptide tails added by the SsrA-tagging system. *Genes Dev.* **12**:1338-47.
9. Herman, C., Thevenet, D., *et al*. 1998. Degradation of carboxy-terminal-tagged cytoplasmic proteins by the Escherichia coli protease HflB (FtsH). *Genes Dev.* **12**:1348-55.
10. Elowitz, M.B. and Leibler, S. 2000. A synthetic oscillatory network of transcriptional regulators. *Nature* **403**:335-8.
11. Chalfie, M., Tu, Y., *et al*. 1994. Green fluorescent protein as a marker for gene expression. *Science* **263**:802-5.
12. Lutz, R. and Bujard, H. 1997. Independent and tight regulation of transcriptional units in Escherichia coli via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements. *Nucleic Acids Res.* **25**:1203-10.
13. Muller, J., Oehler, S., and Muller-Hill, B. 1996. Repression of lac promoter as a function of distance, phase and quality of an auxiliary lac operator. *J. Mol. Biol.* **257**:21-9.

14. Cormack, B.P., Valdivia, R.H., and Falkow, S. 1996. FACS-optimized mutants of the green fluorescent protein (GFP). *Gene* **173**:33-8.
15. Andersen, J.B., Sternberg, C., *et al*. 1998. New unstable variants of green fluorescent protein for studies of transient gene expression in bacteria. *Appl. Environ. Microbiol.* **64**:2240-6.
16. Gardner, T.S., Cantor, C.R., and Collins, J.J. 2000. Construction of a genetic toggle switch in Escherichia coli. *Nature* **403**:339-42.
17. Ackers, G.K., Johnson, A.D., and Shea, M.A. 1982. Quantitative model for gene regulation by lambda phage repressor. *Proc. Natl. Acad. Sci. U.S.A.* **79**:1129-33.
18. Hasty, J., McMillen, D., *et al*. 2001. Computational studies of gene regulatory networks: *in numero* molecular biology. *Nat. Rev. Genet.* **2**:268-79.
19. Thieffry, D. and Thomas, R. 1995. Dynamical behavior of biological regulatory networks--II. Immunity control in bacteriophage lambda. *Bull. Math. Biol.* **57**:277-97.
20. Leveau, J.H. and Lindow, S.E. 2001. Predictive and interpretive simulation of green fluorescent protein expression in reporter bacteria. *J. Bacteriol.* **183**:6752-62.
21. Hill, C.M., Waight, R.D., and Bardsley, W.G. 1977. Does any enzyme follow the Michaelis-Menten equation? *Mol. Cell Biochem.* **15**:173-8.

Appendix

Introduction: There may be a number of explanations why it was not possible to model the dynamics of all of the Guet constructs. Here we will consider possibilities not requiring validation of the biological experimental data.

The method employed in this chapter to measure dynamics of the network models is simulation, which relies on numerical integration of ODEs formulated from the model. How the ODEs are constructed and how they may be used to perform the simulation are topics covered in earlier chapters. This method of simulation is very fast and thus is most often employed by modelers. However, there are two major deficiencies with this method: numerical stability and solution completeness.

Numerical integration is implemented using floating point arithmetic. If the integrator is not carefully implemented, it will be prone to the compounding influence of seemingly small numerical error. While the integrator used in our experiments, CVODE, has a long and respected history, even high quality integrators can be susceptible to error accumulation. This problem is most pronounce when system variables have widely different magnitudes, a situation that occurs during simulation of a model containing low copy-number genes and high concentration proteins. Numerical integration is a complex subject of applied mathematics and thus the best we can do is try to avoid problematic situations.

When simulating models based on ODEs with a numerical integrator, the final solution is dependent on the initial values of the parameters, i.e. the initial concentrations. Viewing a model as a function in multidimensional space, there are ranges of values for the parameters that will result in the same solution, known as basins of attraction. The simplest models will only have a single solution regardless of the initial conditions. However, many models, including many of the models of the Guet library, have feedback allowing multimodal behavior. Nonlinear dependencies between parameters complicate the identification of separate basins of attraction.

Feedback is common in regulatory network models due to properties such as self and mutual regulation. The feedbacks allow the ODEs to potentially have multiple steady states. However, numerical integration, given a single set of initial conditions, will only identify one steady state. When optimizing the parameters of a model, reporting only a single solution may be misleading. Ideally, we need to identify all solutions and compare experimental observations to each of these solutions even when there is only a single experimental observation. However, to do this the observations must be made supporting this approach.

The Guet networks clearly have the ability to exhibit multimodal behavior given that several have topologies that are similar to toggle-switch and repressor like networks. However, the experimental observations used in this chapter were collected from a population of cells and averaged over a period of time. Any variation in florescence between cells or over time is lost by the averaging. Even with each measurement being made in triplicate, since they were made over a population any variation, due to multimodal behavior is lost. If the data would have been collected on a cell-by-cell, perhaps it would be possible to observe the multi-modal behavior. The population of two networks were examined using a cell sorter and a distribution of florescence was observed. The distribution did not show multimodal behavior, but at least one these networks not present difficulty in modeling.

There is a need to develop methods that can find all steady state solutions of network models, but there is a competing need that the solutions be found quickly. Slow generation of solutions during modeling building is a nuisance, but slow solution generation will quickly make parameter optimization impractical. Stochastic or hybrid optimization methods are most often employed, and these methods will test tens of thousands of parameter sets (at least). The solution sets of each model must therefore be found very quickly.

Methods: In Chapter 4 we introduced a method that samples the parameter space of a model during optimization to identify multiple steady states. This method could miss some solutions if the sampling is not sufficiently fine and can be time consuming. Our implementation makes this method impractical for use during optimization. However, there are alternatives. Here we describe some methods that we have tried and our impressions. The rough idea, with each method is to, within our modeling environment, extract the system of equations from the models, manipulate them to be suitable for a particular method, and then call on an external package to find the actual solution.

The equations in the systems of ODEs used throughout this dissertation have consisted of polynomials of second degree. Each term in the polynomial consists of at least one variable. Many of these equations, but not all, can be solved by Mathematica directly. For example, take network S1, a toggle switch, considered in Chapter 6 (Figure 7.15). This network is designed to have bimodal behavior but this is completely hidden by simulation. Finding the steady state solution of this model using Mathematica requires exporting the ODEs and finding where all the ODEs, derivatives of species concentrations, are zero (Figure 7.16), which by definition is a steady state.

Complicating solving the ODEs for steady state solutions is the fact that the equations are not entirely independent. This happens due to mass conservation of certain species. For example, a gene network will conserve abundance of genes. While the protein concentration is free to change, gene copy number is normally not. However, genes may change between various states, such as inhibited and uninhibited, but the total count does not change. To express this, mass conservation terms are necessary. These are shown in Figure 7.16 as bolded terms.

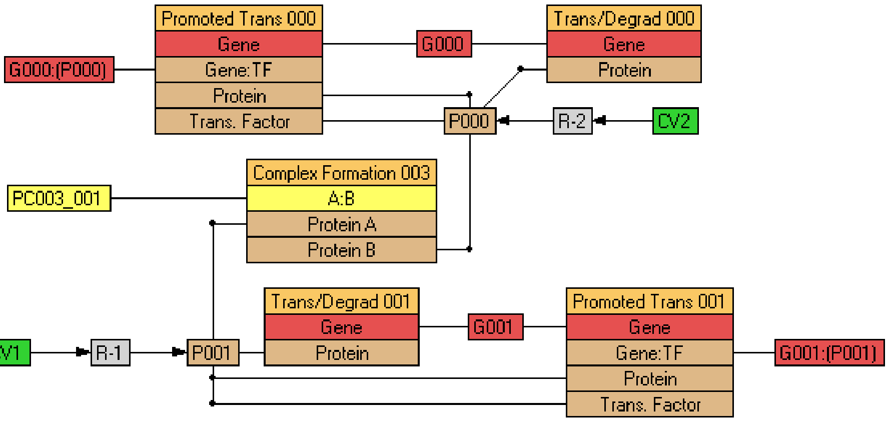


Figure 7.15: Toggle switch model S1

```
k1 = 0.106696;
k2 = 0.15061;
k3 = 0.328247;
k4 = 0.0666803;
k5 = 0.939691;
k6 = 0.862253;
k7 = 0.968658;
k8 = 0.655576;
k9 = 0.0256774;
k10 = 0.369641;
k11 = 0.983843;
k12 = 0.416003;
pa = -k5*a*A+k6*cp1;
pA = -k2*A-k5*a*A-k8*A*B+k1*a+k6*cp1+k7*cp1;
pb = -k10*b*B+k11*cp2;
pB = -k4*B-k8*A*B-k10*b*B+k3*b+k11*cp2+k12*cp2;
pcp1 = -k6*cp1+k5*a*A;
pC = -k9*C+k8*A*B;
pcp2 = -k11*cp2+k10*b*B;
Chop[TableForm[NSolve[{a+cp1==1, b+cp2==1, pa == 0, pA == 0, pb == 0, pB
== 0, pcp1 == 0, pC == 0, pcp2 == 0},
{a, A, b, B, cp1, C, cp2}]]]
```

Figure 7.16: Mathematica code to solve S1

Mathematica is able to quickly locate the steady states (Figure 7.17), but not all of them are realistic. Solutions involving negative or imaginary terms have no physical interpretation and must be discarded. This example shows how a simple network, which would only exhibit a single steady state through simulation from many initial conditions, has three steady states:

two corresponding to the bimodal states of the toggle-switch and one corresponding to an unstable steady state separating the two stable ones. Mathematica has located all solutions.

```
{a = -1.69872, A = -1.45776, b = -23.3759, B = -2.77548, cp1 = 2.69872, C =
103.299, cp2 = 24.3759},
{a = 1.05387, A = -0.0469053, b = 5.63688, B = -2.18944, cp1 = -0.0538716,
C = 2.62196, cp2 = -4.63688},
{a = 0.238771, A = 2.92539, b = 0.940617, B = 0.168032, cp1 = 0.761229, C =
12.5501, cp2 = 0.0593825},
{a = 0.775587, A = 0.265502, b = 0.640395, B = 1.4946, cp1 = 0.224413, C =
10.1313, cp2 = 0.359605},
{a = 0.923089, A = 0.0764527, b = 0.452397, B = 3.22174, cp1 = 0.0769107, C
= 6.28863, cp2 = 0.547603}
```

Figure 7.17: Mathematica solutions to S1

To implement a tight connection to Mathematica, it is necessary to identify the mass conservation relationships. This may be possible through analysis of the network model. However, an alterative is to require the modeler to specify the type of each specie and then apply specie type specific rules to construct the mass conservation rules.

When performing stochastic optimizations the optimization time can be greatly reduced by using a cluster of machines. However, using Mathematica as the solver would require a license for each machine, which may be prohibitively expensive. Given the commercial licensing of Mathematica one may want to consider alternatives.

Internally, Mathematica is likely using Groebner basis. Another tool, Reduce, is considered by many to contain the most sophisticated algorithms for Groebner basis and solving systems of equations. While not as fast as Mathematica, source code is available for Reduce allowing tighter integration, and its license is much more flexible.

The methods discussed so far find multiple solutions but they do little to help with the possibility of numerical error. Mathematica, specializing in symbolic manipulation, may be less prone to floating point error, but the model constants are expressed as floating point values and this may trigger Mathematica to use numerical methods. Expressing the constants as rational numbers may help.

An alternative is to use interval arithmetic methods. Unlike floating point arithmetic, which can only represent a finite number of infinitesimal points, interval methods operate using intervals of values. This allows floating point error to be explicitly captured as an expansion of the interval. Functions may also be implemented using interval methods. A Newton method implemented using interval methods has the amazing property of being able to find all solutions to a system of equations along with error bounds on the solutions affectively solving both problems inherent with simulation.

GLOBSOL is a well known interval analysis package that implements a Newton method. The same requirements are placed on the modeling environment when exporting a system of equations to GLOBSOL as with Mathematica. GLOBSOL is a slower than Mathematica in some cases, but still has good performance. GLOBSOL has the least restrictive license of the packages considered here and is available freely, including source code, from its author who continues to develop it. Interval methods are slower than point methods because, at the very least, the interval has to be tested at both extremes of each interval instead of only a single point. However, this is a simple view of interval arithmetic and many additional operations are necessary to maintain tight intervals and correctness throughout the calculations.

Finally, stochastic simulation can also help avoid problems with the traditional simulation. Traditional simulation uses a numerical integrator and thus is susceptible to numerical error and is deterministic. The determinism guarantees that the same steady state will be located for a given set of initial conditions, regardless of the number of possible steady states. Floating point error is not a problem with stochastic simulation because each specie has an integer concentration. This also provides a more physically realistic interpretation to the concentration results and greatly simplifies implementation. In fact, implementation of the Gillespie method can be done in perhaps 100 lines of code and requires no advanced mathematics. Multiple steady states are exposed during stochastic simulation because even though the simulation begins from a point of initial concentrations, there is always some probability of the system transition to another state, even in steady state.

With the benefits of stochastic simulation there are disadvantages. Stochastic simulations require ensembles of trajectories, perhaps tens of thousands, to be constructed. Each trajectory is independent of each other so parallel hardware can help, but it is still time consuming. Also, because stochastic simulation is still a simulation, there is still some dependence on initial conditions. It is conceivable that certain steady states may be difficult to reach from some initial conditions. Without knowledge of the number of steady states, it becomes difficult to know if sufficient simulation has been performed. Also, interpretation of the simulation results is not so clear. Given a transition may occur anytime, it is unclear when steady state has been reached. It is also unclear how to go about identifying the steady states from the distributions of trajectory ensembles.

Conclusions: If modeling of biochemical networks, such as regulatory networks, is to become a viable part of synthetic biology, foundational methods must be developed that allow the dynamics of networks to be fully observed in the presence of multiple steady states. In addition, methods must be unsusceptible or highly resistant to numerical error. To allow optimization, these methods must be fast. Parameter optimization places a large demand on any method. However, allowing the topology to also be altered during optimization increases the degrees of freedom and the demands grow considerably. We have examined a few methods that are available now and hope that others will identify and develop other methods to replace traditional simulation.

Chapter 8. General conclusions

Overview

The benefits of modeling are well recognized throughout engineering. Most systems constructed by people today are simply too complex to fully understand in all situations. The disciplines of systems biology, genetic engineering, and synthetic biology have been influenced by the engineering approach, and as such, as they mature these fields will also benefit from modeling. However, biological systems may prove to be more complicated and so there is an urgent need for advancement of methods for modeling biological systems.

This dissertation chronicles one group's attempt to apply modeling methods to improve their understanding of genotype to phenotype mapping as well as to identify new constructs for synthetic biology. When we began, there was little existing work to leverage, so we started at foundational levels: characterizing a formalism, developing a modeling environment and optimization methods, and applying these methods to several independent problems.

Accomplishments

Throughout this dissertation, we have advanced a framework for modeling biochemical constructs with the goal of aiding synthetic biology. Starting with a mathematical formalism of mass-action reactions and showing its application to networks of DNA and proteins, we built the GenoDYN modeling environment. Our modeling environment differs from others by relying on hierarchical modeling to compose existing constructs into more complicated and sophisticated ones. This makes GenoDYN extensible in general and practical for models with considerable replication, such as explicit modeling of individual cells in a population. The extensibility is demonstrated by deriving a new modeling tool that focuses on Petri networks instead of mass-action reactions.

Model fitting is particularly important. It is the main method of finding parameters to make the model consistent with observations. We designed a method of searching a model's parameter space by recursive partitioning. While impractical for large models, this method was published and is important for evaluating subsequent techniques.

To demonstrate the applicability of this work to genotype-to-phenotype maps and breeding programs, we subjected a population containing our model of the yeast galactose switch to selective pressure. The results of this paper concluded that genes are context dependent and so it is impossible to select the best allele for a particular gene independent of interacting genes. This has strong implications to a mature breeding program that has likely lost alleles that may be optimal in different contexts.

More recent exploratory results appear in the last chapters. Using networks constructed and characterized by Guet *et al*., we attempt to reconcile a dataset of network topologies and experimental measurements into a collection of network models sharing common motifs. Many of the networks are successfully modeled, but not all of them. Given the inconsistencies observed in the experimental results, the constructs that could not be modeled may have sequencing errors or the impact of multiple equilibria may not be fully appreciated.

Given the difficulty of modeling the Guet networks, we introduce an additional method that increases the degrees of freedom during model fitting. Now the topology is allowed to change in addition to the model parameters. Effectively, we evolve a network model given a description of the desired phenotype. With the increase in degrees of freedom also comes an increase in the computational requirement satisfied by clusters of computers. The unique models produced by this system are interesting to study for alternative solutions. For networks without models, the evolved networks can provide inspiration for new models.

Future directions

The next step in the continuation of this research, which may be quickest to obtain, is solving the problems of multiple equilibria. Continuous simulation is used throughout this dissertation because of its performance, however its limitations require one to be careful not to be misled. Relying on continuous simulation will, at best, result in an incomplete understanding of the dynamics of the model. In particular, continuous simulation is dependent on initial conditions and will only identify one of possibly many equilibria. This is particularly troublesome when trying to fit a model since only one steady state is examined, potentially the incorrect one.

Possible alternatives to continuous simulation include directly solving the system of equations, stochastic simulation, and interval analysis based Newton methods. However, these methods are not entirely equivalent. It is difficult to know when enough stochastic simulation has been performed, and as a simulation it is still biased by initial conditions. Directly solving the equations and interval methods are the most thorough methods and may be the best candidates for replacing continuous simulation. While these methods may not yield trajectories, in most situations trajectories are uninteresting because the initial conditions do not realistically represent any particular state of an individual cell.

After enumerating multiple equilibria, the next area requiring research is optimization and model fitting. Stochastic optimization, such as genetic algorithms, is typically used when little is known about the problem. However, the quality of the outcome of stochastic optimization is not guaranteed. Again, interval methods may help. Beyond treating the model fitter as a black-box, perhaps structure in the problem will allow improvements in the optimization methods.

The Guet *et al*. dataset is an impressive piece of work and deserves a more thorough characterization. Reproducing these constructs, characterizing each construct, and releasing the data to the public would be a huge contribution to synthetic biology and to those developing modeling methods. It is important that the constructs be characterized in such a way to measure if multiple equilibria are present. It is unclear why Guet did not perform these measurements originally. Perhaps it was due to expense at the time or because he did not know how influential the dataset would become. Key datasets such as an improved Guet-like dataset will be vital for developing and benchmarking new methods and can be done independently of the previous suggestions for future work.

Final thoughts

Modeling of the actual cellular processes for engineering purposes is a relatively new activity. It is impossible not to acknowledge the impact that engineering, based on the physical sciences, has had on the modern world. If ethically approached, engineering based on the life sciences may ultimately have a larger beneficial impact.

Appendix A. PPN: A Petri Net simulation tool

Preface

Chapter 3 of this dissertation describes a modeling environment that is instrumental in the development of the subsequent chapters. Originally called PNE, for Pioneer Network Editor, reflecting the sponsorship of Pioneer Hi-Bred during its development, and later renamed as GenoDYN to complement a coauthor's existing tool, GenoCAD, this package is actually more than a single tool, but is a framework, allowing entirely new modeling environments to be created. In this appendix we describe one such extension, PPN, for modeling Petri nets. Much of the graphical framework is reused, but the simulation engine is completely new. The description of the simulation engine may also help those with similar interests, since such implementation details are rare.

Introduction

Petri nets[1] have been used to model systems in a wide range of fields including dependability, communication, music[2], and biology[3,4]. The broad acceptance of Petri nets likely has a lot to do with the understandable discrete nature of Petri nets and their amenability to accommodate application in new fields. Here we present a new Petri net modeling environment that is meant to be generic. While the field of Petri net modeling environments is a crowded one, we believe that our application has novel features including hierarchical structures of complex models. Our system is made more generic by being able to simulate models that contain any number of both deterministic and stochastic transitions.

Methods

In this section we describe several of the key features of PPN and the implementation of these features. To ensure maximum performance, PPN is implemented in C++. To increase portability to other operating systems, including Microsoft Windows, Linux, and Mac OS/X, the Trolltech's Qt widget set underlies the GUI. Where possible, operating system specific functions, such as thread management and network communication, have been implemented using Qt to increase portability.

Modeling environment

PPN presents a canvas on which networks may be drawn using a combination of nodes and edges (Figure A.1). The PPN canvas appears similar to other visual Petri net modeling tools, but there are differences in how networks are presented. Places and transitions are rectangles instead of the traditional circles, the names are contained within the places and transitions, and places display their token count below their name.

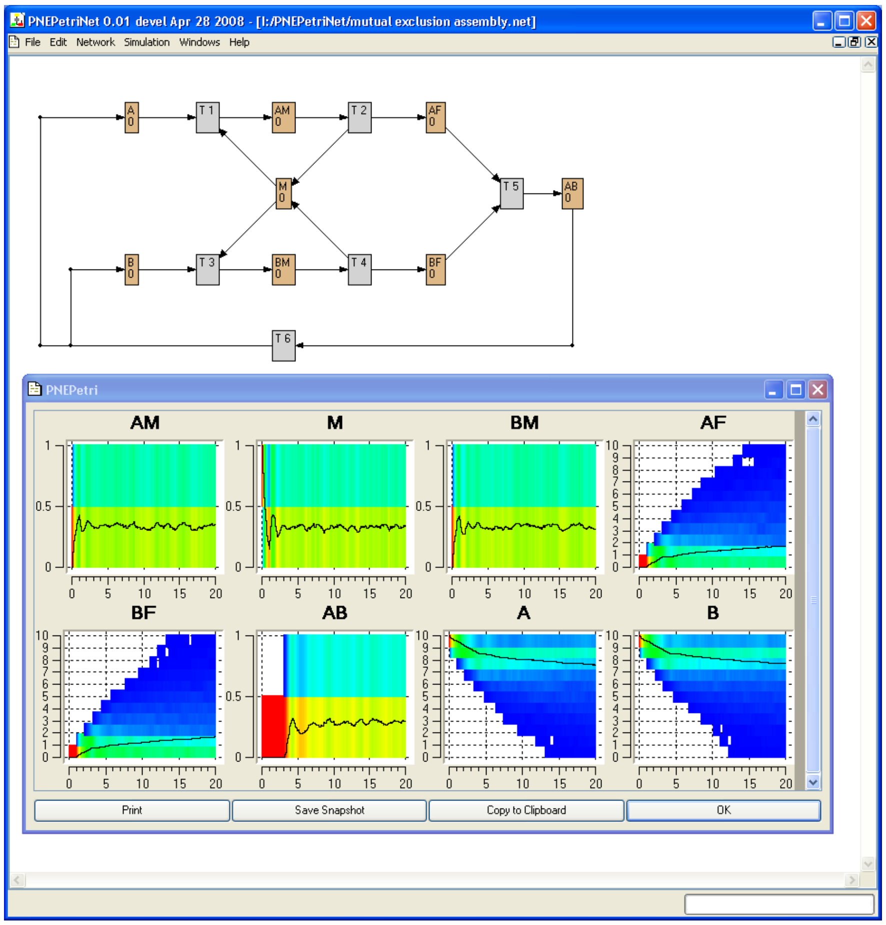

PNEPetriNet 0.01 devel Apr 28 2008 - [I:/PNEPetriNet/mutual exclusion assembly.net]
File Edit Network Simulation Windows Help
A 0
T 1
AM 0
T 2
AF 0
M 0
T 5
AB 0
B 0
T 3
BM 0
T 4
BF 0
T 6
PNEPetri
AM
M
BM
AF
BF
AB
A
B
Print
Save Snapshot
Copy to Clipboard
OK

Figure A.1: PPN screenshot

Perhaps unique to PPN is the ability to model networks hierarchically using subnetworks. A subnetwork is a network motif that is encapsulated and referenced by a higher level network. Places that are exported from the subnetwork are exposed to the higher level network where they can be tied to other places. Subnetworks offer a convenient way to compartmentalize a repetitive motif of a network. By abstracting the motif, it becomes possible to simplify the appearance of the design and reuse components in other networks. A new instance of a subnetwork is created when it is copied, and modification of the copy only affects the copy.

Hybrid simulation algorithm

Both places and transitions have single values associated with them. The value associated with a place represents the number of tokens present in the place. While the value associated with transitions represent the expected waiting time. Transitions come in two flavors in PPN, deterministic and stochastic. With deterministic transitions, the value can be considered a time delay. Once a transition is ready, because all the required tokens are available, the transition will delay by the specified waiting time. Stochastic transitions are similar, although the actual time delay is sampled from an exponential distribution centered about the waiting time. Below are the main algorithm steps.

Step 1: Initialize a state vector, *state*. The state vector contains, for each place, the current number of tokens present. The required initialization simply requires copying the token values from each place present in the model.

Step 2: Initialize results matrix, *results*, that contains a copy of the *state* for each sampled time point. The sampling rate could be different from the actual transition rate, but to best observe the dynamics of a model, the sample rate should be twice that of the fastest transition.

Step 3: Initialize deadline vector, *deadline*. For each transition there is an associated time at which the transition may actually occur. The initialize is performed by filling the vector with the values of $t0$ plus the transition specific delay.

Step 4: While $t \leq t_{end}$ perform the following steps:

Step a: For each of the transitions *trans*, check to see if $deadline_{trans} < t$. If so, add *trans* to a list of ready transitions, *ready*.

Step b: Randomize *ready*, which is necessary for deterministic models with race conditions. Without this step, transitions that are regularly ready at the same time would always fire in the same order.

Step c: For each of the $trans \in ready$, check to see if *trans* can still occur by checking if the required tokens are available and inhibitory tokens are not present. If so, update *state* based on the movement of tokens. To update *state,* decrement the source places by the number of tokens taken and increment the destination places with the number of tokens generated.

Step d: Update *deadline* with times relative to *t* when transitions that can occur should be attempted.

Step e: Examine *deadline* and find the time the next transition will occur, t_{next}. Set $t = t_{next}$. If no transition is scheduled to occur then set $t = t_{end}$. This step allows the simulation to jump to the next time point of interest without examining intermediate times where no events occur.

Step f: As needed, based on the sampling rate and *t*, update results matrix by copying results from the state vector into the results vector.

Generation of results

When simulating strictly deterministic Petri nets, the results will be the same with each simulation. However, if there is any non-determinism present in the model, such as with race conditions or if the model has stochastic transitions, the results may change with each simulation. In such a case, each simulation represents only a single trajectory that the system could take and is of little independent value. However, combine several trajectories and the breadth of the system dynamics is exposed.

Presentation of results

The graphs produced by PPN visualize the ensemble of trajectories, but there are currently no metrics for summarizing the results. However, results can be exported for analysis in statistical packages.

The graphs are generated using a modified version of the Qwt graphing package for Qt. The main differences include the addition of a novel histogram representation and context menus for controlling common activities for each graph.

If a simulation results in only a single trajectory, the graph will be familiar line plots representing the token count of each place during the simulation. However, if a simulation generates an ensemble of trajectories, then a 3D histogram is presented. In the histogram, time and token count are still represented along the horizontal and vertical axes respectively. However, there are also vertical bins for each time point, with each bin assigned a color depending on its relative count. Cooler blue colors represent low values and warmer red colors represent high values. The mean is superimposed as a line plot.

By viewing the histogram plot of a large number of simulations, it becomes possible to find if a model exhibits rare and unexpected multi-model behavior. For instance, perhaps a system fails only rarely. The difference between the correctly functioning state and the failure state will appear as an additional mode. The time when this mode appears, and its relative frequency, provides clues as to the expected frequency of the failure.

Discussion

In this section we present several examples of Petri nets that have been modeled in PPN and discuss their dynamics. The goal of these examples is to act as a demonstration of Petri nets as well as PPN.

Example: Communication protocol

In this example we consider a Petri net of a communication protocol (Figure A.2). This particular protocol is modeled as having *send* and *receive* queues and two channels for communication. Messages are only sent on the first link, while the second link is reserved for sending acknowledgments. All transitions are deterministic with fixed time delays. The simulation results show the expected decrease of tokens in the *sendQueue* and the corresponding increase of tokens in the *rcvQueue* place. Steady state occurs at t = 60 (Figure A.3).

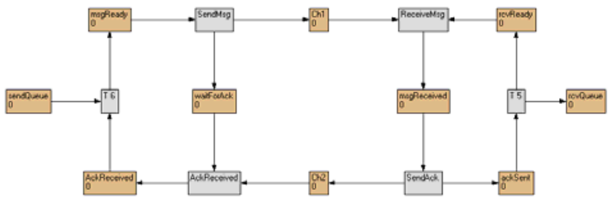


Figure A.2: Communication protocol

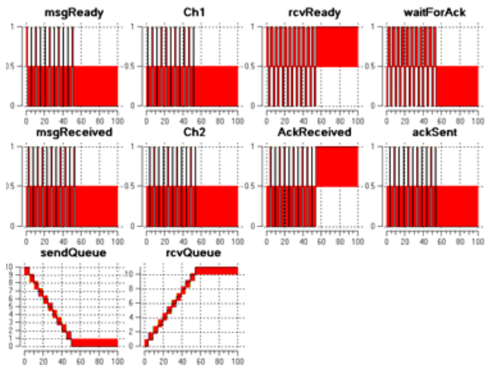


Figure A.3: Simulation of comm. protocol

To continue the exploration of the dynamics of this system, PPN can color-code the places with their relative token counts to show the patterns in the flow of tokens. Shown below are three time points that have been selected to show the initial state (Figure A.4), an intermediate state (Figure A.5), and the steady-state configuration (Figure A.6).

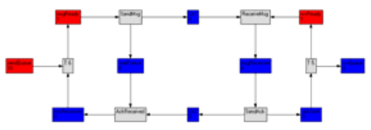


Figure A.4: State at *t*=0

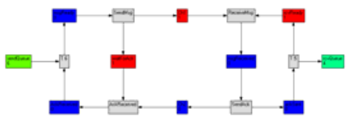


Figure A.5: State at *t* = 23.7

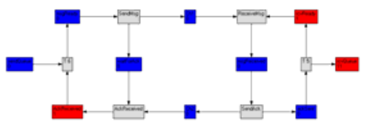


Figure A.6: State at *t* = 63.9

Example: Stochastic vs. deterministic

To examine the differences between stochastic and deterministic transitions, consider Figure A.7, which is a model that contains two identical networks, with the top network consisting of deterministic transitions and the bottom network consisting of stochastic transitions. All tokens start in the two N1 places and flow to the N1 and N2 places. Also, the transition rate of T1 is five-fold faster than T5.

Steady state occurs at approximately *t* = 12 (Figure A.8) with N5 containing roughly 1/5 the number of tokens as N1. This is expected considering the difference in transition rate. However, the graph of the stochastic network differs from the deterministic network. In fact, with each simulation the stochastic network results will differ (compare Figure A.8 to Figure A.9). It is therefore important to perform several simulations and generate ensembles (Figure A.10). On average, this model will reach steady state with eight tokens in N1 and two tokens in N5, although there are rare contradictory trajectories.

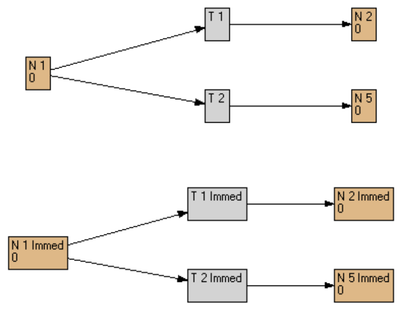


Figure A.7: Stochastic and deterministic models of same network

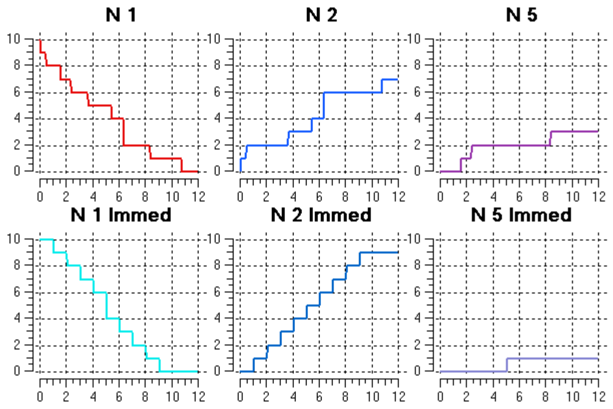


Figure A.8: Stochastic and deterministic simulation of the same network

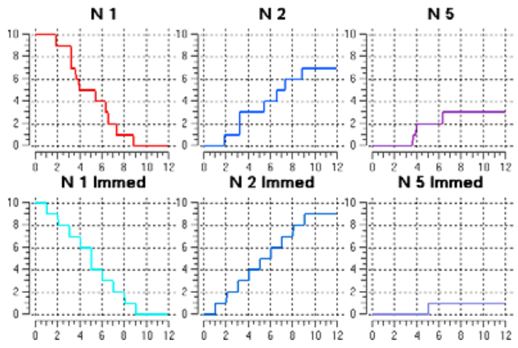


Figure A.9: Second simulation showing how stochastic results differ each time

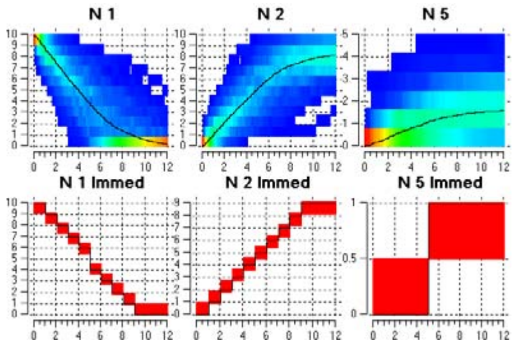


Figure A.10: Histogram representation of trajectory ensembles

Example: Identification of deadlock and race conditions

Simulations of Petri nets can also identify unexpected, potentially dangerous, behavior. Consider the Petri net in Figure A.11, which passes tokens from N1 to N2, then stochastically transfers the token to either N3 or N4. Once tokens are present in both N3 and N4, a pair of tokens from each place is combined and recycled back to N1. This may be similar to a process in a factory. The network appears benign, but the ensemble of thousands of simulations shows an unexpected steady state around *t* = 1500 (Figure A.12).

Examining one of the trajectories exposes the root cause of this bi-model distribution. Starting from the initial conditions (Figure A.13), at an intermediate point the tokens are split between places N3 and N4 (Figure A.14), but eventually, at steady state, all the tokens have become lodged in N4 (Figure A.15). In a different trajectory, the tokens may have become lodged in N3. This network demonstrates the detection of race conditions, here present at N2, and deadlock, the network has a steady state where none should exist.

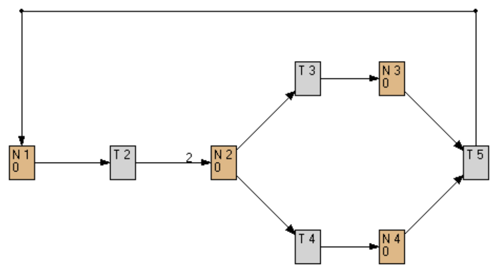


Figure A.11: Example model with deadlock possible

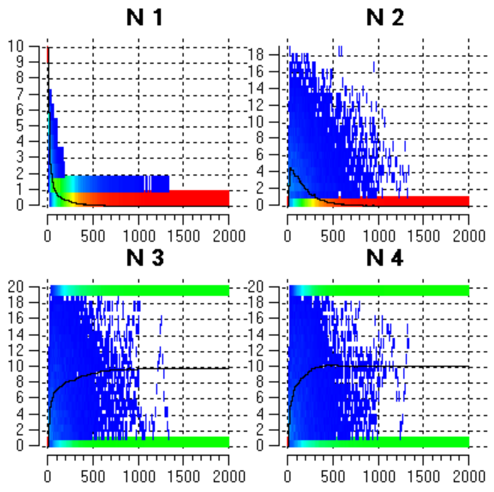


Figure A.12: Simulation results showing that a deadlock is possible in N3 or N4

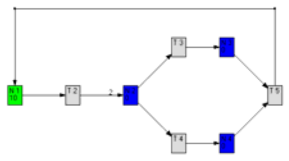


Figure A.13: State at *t* = 0

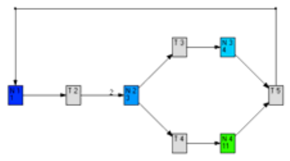


Figure A.14: State at *t* = 3.0

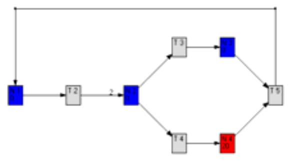


Figure A.15: State at *t* = 4.8

Example: Reliability impact of serial modules

In this example we consider how Petri nets can be used to model reliability using a network of modules in series. In subsequent examples, we will examine how the reliability changes when the modules are in parallel.

First, let's define a subnetwork (Figure A.16) to contain the network in Figure A.17. This subnetwork allows tokens to transition deterministically in a single time step if the node is active. However, the node can stochastically fail, after which no tokens can transition.

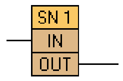


Figure A.16: Subnetwork module for serial networks

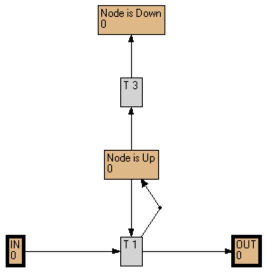


Figure A.17: Internals of subnetwork module

We can use this subnetwork to compose a series of networks containing one, two, and three modules (Figure A.18). Given tokens recycle from the destination to the source places in these networks, failures can be detected by identifying steady states when the destination place is drained of tokens. With one module steady state occurs at approximately 100 time units, with two modules this happens at approximately 40 time units, and with three modules total system failure occurs at approximately 20 time units (Figure A.19). While these results do not exactly reflect the expected times, they are reasonable, and with additional trajectories the simulation results should asymptotically reach the expected times.

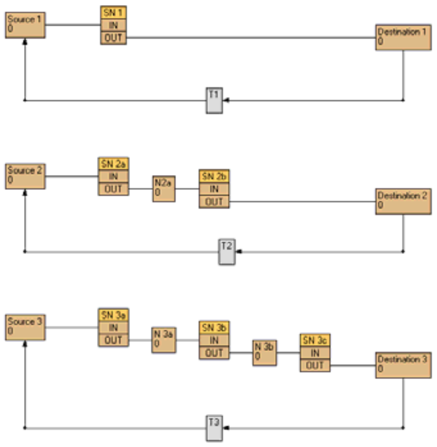


Figure A.18: Three networks with an increasing number of modules in series

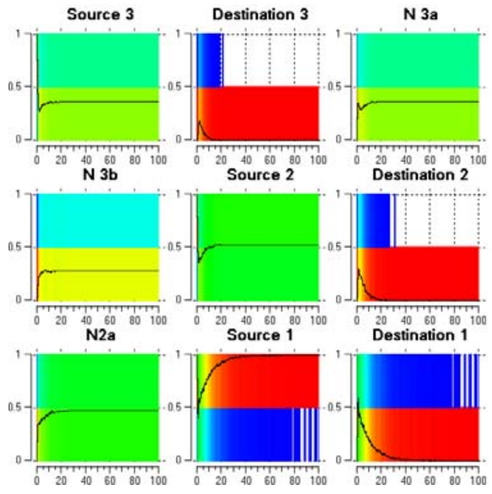


Figure A.19: Simulation results showing how reliability decreases with increase in modules

Example: Reliability impact of parallel modules

Using a similar subnetwork as with the serial example, the next networks consider the impact of parallel modules on reliability. In the first example, we have only one module (Figure A.20), and failure occurs at $t = 70$ (Figure A.21), but the second example (Figure A.22) has five modules in parallel increasing the time before total system failure to $t = 90$ (Figure A.23), as expected.

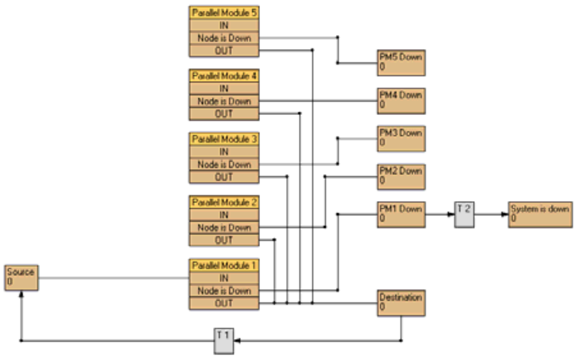


Figure A.20: Network with modules in parallel (only one module active)

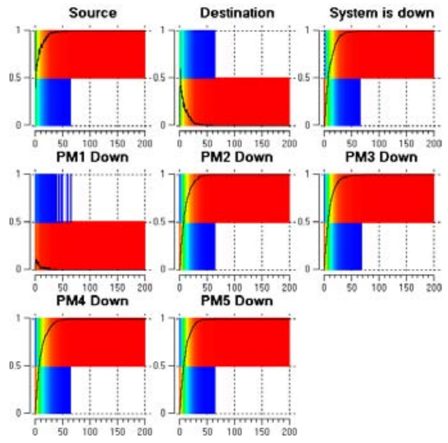


Figure A.21: Simulation results of network of parallel modules (only one module active)

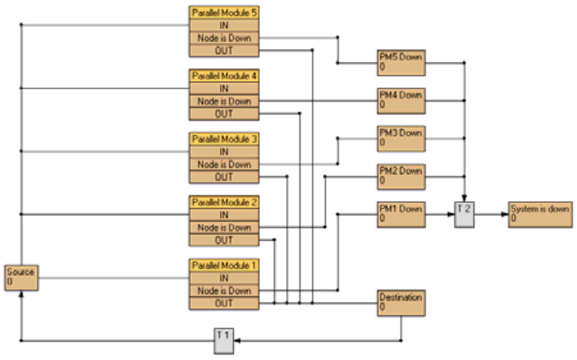


Figure A.22: Network with modules in parallel (five modules active)

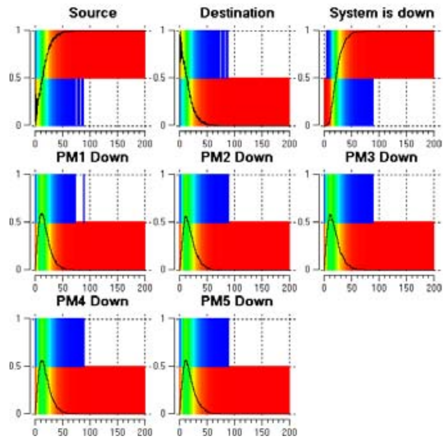


Figure A.23: Simulation results of network of parallel modules (five modules active)

Example: Reliability impact of modules with failover

The previous example (Figure A.22) has all parallel modules active from the start. Using Petri nets we can test how the reliability of the system changes if only a single module is active at any point. Failure of an active module triggers rollover to the next module. By adding an extra signal from the subnetwork to trigger a power-on event in the subnetwork (Figure A.24 and Figure A.25), a parallel network with failover capability can be created (Figure A.26). Simulation of this network shows that failure does not occur until *t* = 170 (Figure A.27), which is significantly better than *t* = 90 when all modules were active from the start in the previous example. This configuration is clearly superior from the perspective of reliability if one can tolerate system downtime to rollover. One could relate this to real-world modules with limited usefulness while powered-on.

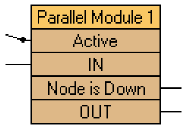


Figure A.24: Subnetwork module for parallel networks

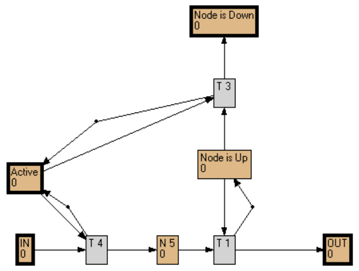


Figure A.25: Internals of subnetwork module

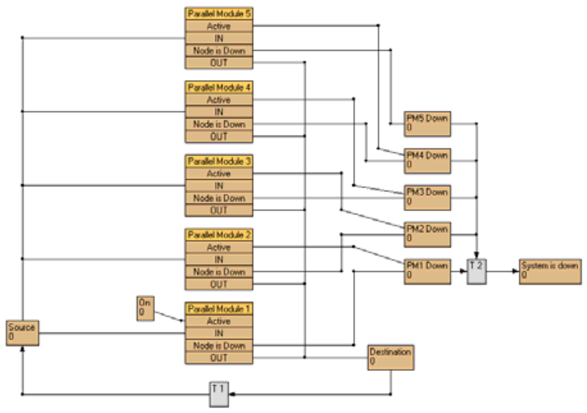


Figure A.25: Network of parallel modules with failover

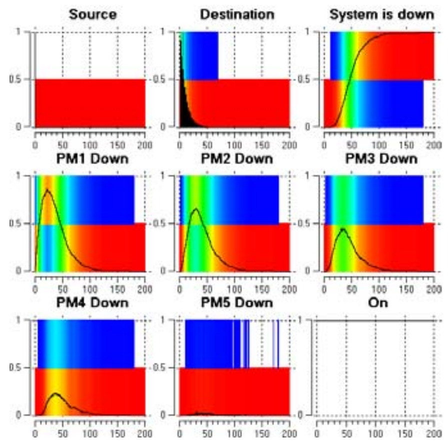


Figure A.26: Simulation results of parallel modules with failover

Future improvements

Currently, to make a change to any property of a place or a transition one must access a property dialog of that particular node. This is inconvenient when many changes must be made. We would like to create a single table where any property of a model could be changed. This change will also allow sets of parameters to be saved and recalled.

The graphical presentation of the simulation results is useful for gaining an appreciation of the overall behavior of the system; however, there is currently no way to statistically summarize these results. Similarly, there is currently no way to know when sufficient simulation has been performed. By adding some statistical measures of the simulation results, both of these deficits may be addressed.

Conclusions

We have developed a new Petri net modeling package, PPN, capable of modeling hybrid deterministic and stochastic systems. We have demonstrated PPN using examples modeling systems, estimating reliability, and testing for unexpected behavior. PPN can be used as a modeling tool or a teaching aid.

Literature cited

1. Peterson, J.L. 1977. Petri nets. *ACM Comp. Sur.* **9**:223-52.
2. Barate, A. 2008. Music description and processing: an approach based on Petri Nets and XML. 525-34. *In* V.Kordic (ed.) *Petri Net: Theory and Applications*. I-Tech Education and Publishing, Vienna, Austria.
3. Pinney, J.W., Westhead, D.R., and McConkey, G.A. 2003. Petri Net representations in systems biology. *Biochem. Soc. Trans.* **31**:1513-5.
4. Goss, P.J. and Peccoud, J. 1998. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc. Natl. Acad. Sci. U.S.A.* **95**:6750-5.

(This page was intentionally left blank for proper layout of Appendix B.)

Appendix B. Additional examples

What follows is a summary of modeling and optimization experiments for each network in the Guet library. In the network model diagrams, an X over a section of the network model highlights a disconnected section that should not have any influence. The following simulation parameters were present in all the reports and have been factored out for brevity.

```
Simulation:
  Simulation Method: ODE (0)
  Time Interval: [0,100]
  Sampling Rate: 0.1
  Absolute Tolerance: 1e-008
  Relative Tolerance: 0.0001
  Max ODE Step Size: 1
  Min ODE Step Size: 0
  Single Trajectory: false
  Number of Replicates: 20
  Histogram Update Rate: -1
  Current Environment: 1 (of 4 total)
  Current Fitness Function: Null (0)
```

The networks in the Guet collection respond to stimuli from two external chemical signals. Given the high concentrations of these chemical signals when present, they can be considered as Boolean signals. Setting a threshold for the florescence of GFP allows the response of each network, its phenotype, to be interrupted as a Boolean function. Table B.1 is a summarization of the Guet constructions, their phenotypes, and how they relate to Boolean functions of two inputs. Some Boolean functions have no analogue in the network library.

$f(x,y)$# Class	$x \wedge y$	$\overline{x} \wedge y$	$x \wedge \overline{y}$	$\overline{x} \wedge \overline{y}$	Logic name	Textual name	Guet constructs
0	0	0	0	0	False	False	D028 D090 D104 D114 D123 D250 D253(1)
1	0	0	0	1	$\overline{x \vee y}$	Nor	
2	0	0	1	0	$x \wedge \overline{y}$	Inhibition	D019 D038 D078
3	0	0	1	1	$\overline{y}$	Not Y	
4	0	1	0	0	$\overline{x} \wedge y$	Inhibition	C101
5	0	1	0	1	$\overline{x}$	Not X	
6	0	1	1	0	$x \otimes y$	Exclusive Or	
7	0	1	1	1	$\overline{x \wedge y}$	Nand	
8	1	0	0	0	$x \wedge y$	And	D012 **D016** **D052** **D066**(2) **C024** **C103**(2) **C113**(2) C133(2) **C242**
9	1	0	0	1	$x = y$	Equivalence	
10	1	0	1	0	x	x	D032 **D101** D113(1)
11	1	0	1	1	$y \rightarrow x$	Implication	
12	1	1	0	0	y	y	**D066**(1) D143 D180 D253(2) **C103**(1) **C113**(1) **C144**
13	1	1	0	1	$x \rightarrow y$	Implication	
14	1	1	1	0	$x \vee y$	Or	
15	1	1	1	1	True	True	D118 D113(2) D117 D133 D135 **C195**

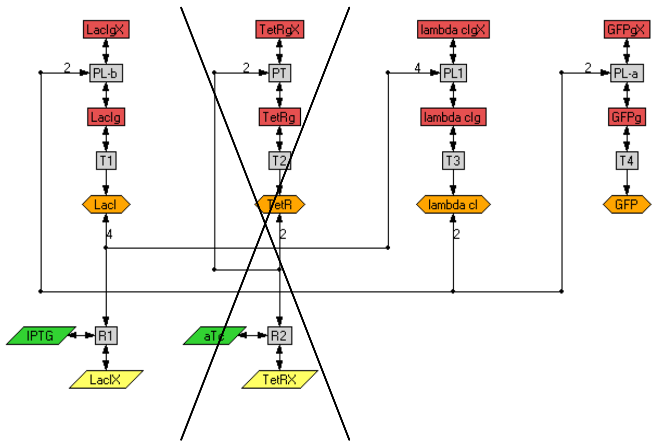
Table: B.1: Interpretation of Guet networks as Boolean functions

d012

The experimental data of this network are not consistent with the theory of how the network should behave. Since tetR only regulates itself, the expression of tetR should have no impact on the expression of GFP. Therefore one would expect GFP_{env1} ~= GFP_{env3} and GFP_{env2} ~= GFP_{env4}. Effectively, this network should act as only a function of IPTG. However, the experimental data clearly show aTc having an influence. The authors might have been trying to make this point when pointing out that identical "topologies" with different promoters yield different logical functions.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	9658	134	463	259
Exp2	9995	198	263	210
Exp3	9745	74	429	154
Mean	9800	135	385	208
Opt	9797	270	271	270

```
Molecules (16):
  GFP  DR=80.7071
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=243508
  LacIX  DR=20711.6
  LacIg  IC=1
  LacIgX
  TetR  DR=431930
  TetRX  DR=546715
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=378665
  lambda cIg  IC=1
  lambda cIgX
```



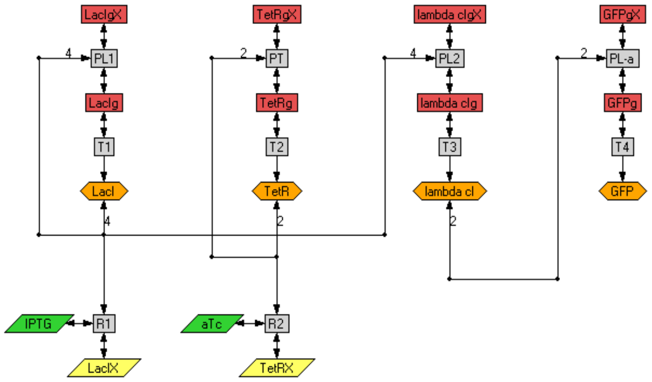
```
Reactions (22):
  D1:  LacI --> 0  Kf=243508
  D2:  TetR --> 0  Kf=431930
  D3:  lambda cI --> 0  Kf=378665
  D4:  GFP --> 0  Kf=80.7071
  D5:  LacIX --> 0  Kf=20711.6
  D6:  TetRX --> 0  Kf=546715
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=870724  Kr=54674.6
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=870724  Kr=54674.6
  PL1:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=295651  Kr=691023
  PT:  TetRg + 2TetR <-> TetRgX  Kf=454467  Kr=137193
  R1:  IPTG + LacI <-> LacIX  Kf=816165  Kr=692311
  R2:  aTc + TetR <-> TetRX  Kf=373474  Kr=847260
  T1:  LacIg --> LacIg + LacI  Kf=990824
  T2:  TetRg --> TetRg + TetR  Kf=19294.5
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=563886
  T4:  GFPg --> GFPg + GFP  Kf=792745
```

d016

The experimental data of this network are not consistent with the theory of how the network should behave. See discussion in d012. This particular network is acknowledged in Figure 5 of Guet's paper as exhibiting a NOR function

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	28391	1190	698	652
Exp2	28998	1160	748	484
Exp3	29176	825	863	766
Mean	28855	1058	770	634
Opt	14813	846	14813	846

```
Molecules (16):
  GFP  DR=59.5489
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=74549.8
  LacIX  DR=244179
  LacIg  IC=1
  LacIgX
  TetR  DR=385194
  TetRX  DR=637180
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=446273
  lambda cIg  IC=1
  lambda cIgX
```



```
Reactions (22):
  D1:  LacI --> 0  Kf=74549.8
  D2:  TetR --> 0  Kf=385194
  D3:  lambda cI --> 0  Kf=446273
  D4:  GFP --> 0  Kf=59.5489
  D5:  LacIX --> 0  Kf=244179
  D6:  TetRX --> 0  Kf=637180
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=649138  Kr=120499
  PL1:  LacIg + 4LacI <-> LacIgX  Kf=3259.68  Kr=946545
  PL2:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=575560  Kr=505127
  PT:  TetRg + 2TetR <-> TetRgX  Kf=677606  Kr=867500
  R1:  IPTG + LacI <-> LacIX  Kf=711270  Kr=743979
  R2:  aTc + TetR <-> TetRX  Kf=117110  Kr=958953
  T1:  LacIg --> LacIg + LacI  Kf=682183
  T2:  TetRg --> TetRg + TetR  Kf=535086
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=781232
  T4:  GFPg --> GFPg + GFP  Kf=882170
```

d018

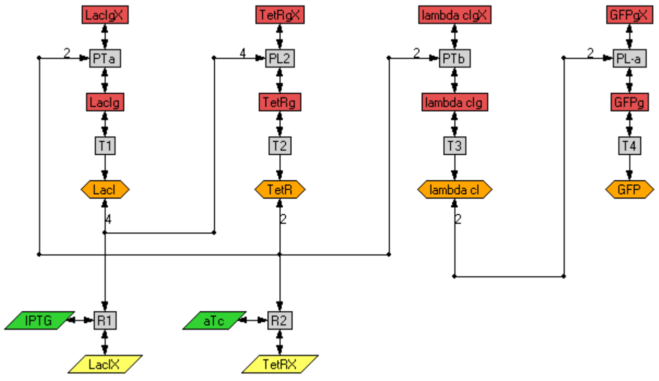
tetR and LacI act to repress each other although not completely. Some tetR is able to also repress lamCI. This subtl repression of lamCI results in a subtle repression of GFP.

Env2: When IPTG is present there is a slight increase in GFP, which would require a decrease in lamCI, which would require an increase in tetR, which would require a decrease in LacI. LacI is being inhibited by IPTG.

Env3: With no IPTG to inhibit LacI and aTc to inhibit tetR, lamCI levels should be their highest and GFP levels should be their lowest. Does not agree.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	5714	4935	7096	6454
Exp2	5989	4800	6656	6875
Exp3	5537	5128	7181	6719
Mean	5747	4954	6978	6682
Opt	6090	6090	6090	6090

```
Molecules (16):
  GFP  DR=134.998
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=157215
  LacIX  DR=136089
  LacIg  IC=1
  LacIgX
  TetR  DR=350608
  TetRX  DR=467069
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=976142
  lambda cIg  IC=1
  lambda cIgX
```

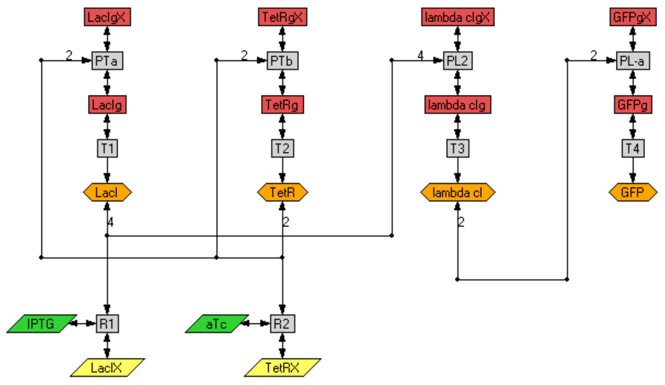


```
Reactions (22):
  D1:  LacI --> 0  Kf=157215
  D2:  TetR --> 0  Kf=350608
  D3:  lambda cI --> 0  Kf=976142
  D4:  GFP --> 0  Kf=134.998
  D5:  LacIX --> 0  Kf=136089
  D6:  TetRX --> 0  Kf=467069
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=233959  Kr=537530
  PL2:  TetRg + 4LacI <-> TetRgX  Kf=586015  Kr=64471.8
  PTa:  LacIg + 2TetR <-> LacIgX  Kf=523640  Kr=0.00011
  PTb:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=523640  Kr=0.00011
  R1:  IPTG + LacI <-> LacIX  Kf=542136  Kr=383598
  R2:  aTc + TetR <-> TetRX  Kf=121444  Kr=914971
  T1:  LacIg --> LacIg + LacI  Kf=769807
  T2:  TetRg --> TetRg + TetR  Kf=964088
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=12312.9
  T4:  GFPg --> GFPg + GFP  Kf=822182
```

d019

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	386	609	10550	125
Exp2	300	561	10796	117
Exp3	308	419	11017	264
Mean	331	530	10788	169
Opt	343	343	10788	343

```
Molecules (16):
  GFP  DR=72.0054
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=60450.8
  LacIX  DR=784418
  LacIg  IC=1
  LacIgX
  TetR  DR=4322.02
  TetRX  DR=827155
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=179980
  lambda cIg  IC=1
  lambda cIgX
```



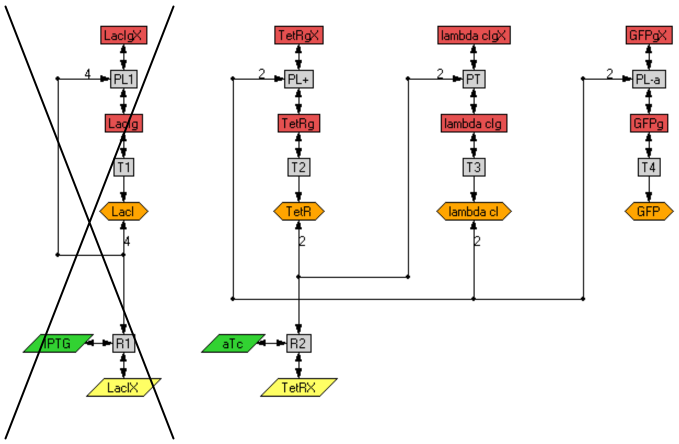
```
Reactions (22):
  D1:  LacI --> 0  Kf=60450.8
  D2:  TetR --> 0  Kf=4322.02
  D3:  lambda cI --> 0  Kf=179980
  D4:  GFP --> 0  Kf=72.0054
  D5:  LacIX --> 0  Kf=784418
  D6:  TetRX --> 0  Kf=827155
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=991594  Kr=437304
  PL2:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=698082  Kr=855382
  PTa:  LacIg + 2TetR <-> LacIgX  Kf=999944  Kr=3.8043
  PTb:  TetRg + 2TetR <-> TetRgX  Kf=999944  Kr=3.8043
  R1:  IPTG + LacI <-> LacIX  Kf=503098  Kr=822883
  R2:  aTc + TetR <-> TetRX  Kf=522304  Kr=987878
  T1:  LacIg --> LacIg + LacI  Kf=909974
  T2:  TetRg --> TetRg + TetR  Kf=732717
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=680721
  T4:  GFPg --> GFPg + GFP  Kf=826437
```

d028

Similar to d012 and d016 the theory of this network says that the expression of GFP should be independent of one of the environmental signals. In this case, d028 should be dependent only on aTc since LacI does not regulate any other transcription factor. Although the experimental data do not provide a very strong case, it does appear that this idea may be supported. When aTc is present there is a ~50% reduction in GFP expression compared to the environments without aTc.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	741	526	246	271
Exp2	475	786	219	265
Exp3	499	665	387	229
Mean	571	659	284	255
Opt	615	615	270	270

```
Molecules (16):
  GFP  DR=1425.76
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=927732
  LacIX  DR=469867
  LacIg  IC=1
  LacIgX
  TetR  DR=116581
  TetRX  DR=616171
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=375725
  lambda cIg  IC=1
  lambda cIgX
```



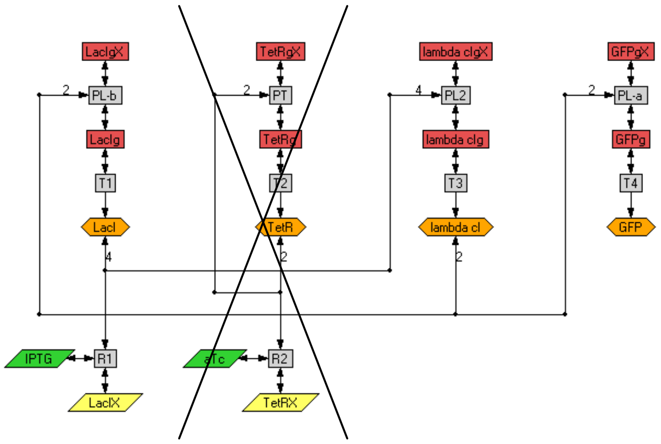
```
Reactions (22):
  D1:  LacI --> 0  Kf=927732
  D2:  TetR --> 0  Kf=116581
  D3:  lambda cI --> 0  Kf=375725
  D4:  GFP --> 0  Kf=1425.76
  D5:  LacIX --> 0  Kf=469867
  D6:  TetRX --> 0  Kf=616171
  PL+:  TetRg + 2(lambda cI) <-> TetRgX  Kf=6467.86  Kr=966913
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=637688  Kr=698314
  PL1:  LacIg + 4LacI <-> LacIgX  Kf=1467.59  Kr=367764
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=260214  Kr=596849
  R1:  IPTG + LacI <-> LacIX  Kf=472325  Kr=552886
  R2:  aTc + TetR <-> TetRX  Kf=902657  Kr=941680
  T1:  LacIg --> LacIg + LacI  Kf=977.144
  T2:  TetRg --> TetRg + TetR  Kf=765403
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=446575
  T4:  GFPg --> GFPg + GFP  Kf=879981
```

d032

Identical to d012 except PL2 instead of PL1. These two might be a good first pair.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	47156	1049	47627	654
Exp2	48978	1008	48928	806
Exp3	48172	1084	50132	753
Mean	48102	1047	48895	738
Opt	48499	892	48499	892

```
Molecules (16):
  GFP  DR=20.3931
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=9565.06
  LacIX  DR=556231
  LacIg  IC=1
  LacIgX
  TetR  DR=141536
  TetRX  DR=547239
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=303790
  lambda cIg  IC=1
  lambda cIgX
```



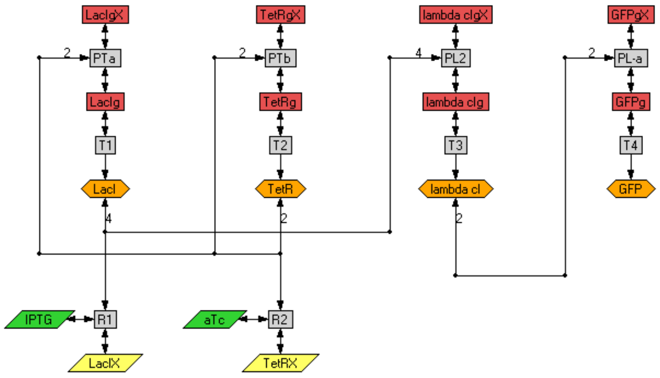
```
Reactions (22):
  D1:  LacI --> 0  Kf=9565.06
  D2:  TetR --> 0  Kf=141536
  D3:  lambda cI --> 0  Kf=303790
  D4:  GFP --> 0  Kf=20.3931
  D5:  LacIX --> 0  Kf=556231
  D6:  TetRX --> 0  Kf=547239
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=879296  Kr=88370.6
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=879296  Kr=88370.6
  PL2:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=866991  Kr=595419
  PT:  TetRg + 2TetR <-> TetRgX  Kf=779032  Kr=78355.8
  R1:  IPTG + LacI <-> LacIX  Kf=88788.1  Kr=500960
  R2:  aTc + TetR <-> TetRX  Kf=504998  Kr=742455
  T1:  LacIg --> LacIg + LacI  Kf=747025
  T2:  TetRg --> TetRg + TetR  Kf=715170
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=703448
  T4:  GFPg --> GFPg + GFP  Kf=989039
```

d038

Same topology as d019. Env3 differs. Roughly the same GFP expression. Mutation? Not indicated by Guet as such. Very different parameters.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	614	654	13903	173
Exp2	399	541	13421	207
Exp3	269	458	13878	325
Mean	427	551	13734	235
Opt	427	393	13734	393

```
Molecules (16):
  GFP  DR=56.0518
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=97406.3
  LacIX  DR=402256
  LacIg  IC=1
  LacIgX
  TetR  DR=11394.2
  TetRX  DR=568344
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=164168
  lambda cIg  IC=1
  lambda cIgX
```



```
Reactions (22):
  D1:  LacI --> 0  Kf=97406.3
  D2:  TetR --> 0  Kf=11394.2
  D3:  lambda cI --> 0  Kf=164168
  D4:  GFP --> 0  Kf=56.0518
  D5:  LacIX --> 0  Kf=402256
  D6:  TetRX --> 0  Kf=568344
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=693406  Kr=282444
  PL2:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=789120  Kr=566996
  PTa:  LacIg + 2TetR <-> LacIgX  Kf=265668  Kr=225839
  PTb:  TetRg + 2TetR <-> TetRgX  Kf=265668  Kr=225839
  R1:  IPTG + LacI <-> LacIX  Kf=726516  Kr=233644
  R2:  aTc + TetR <-> TetRX  Kf=536712  Kr=37503.7
  T1:  LacIg --> LacIg + LacI  Kf=370007
  T2:  TetRg --> TetRg + TetR  Kf=267284
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=610549
  T4:  GFPg --> GFPg + GFP  Kf=770114
```

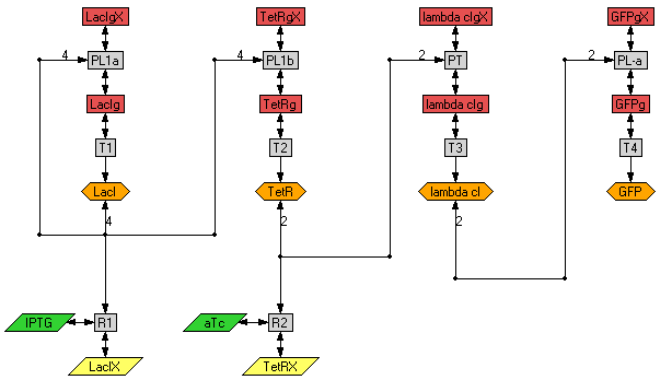
d052

This network has always caused a problem for the Optimization Engine. Three of the four environments could be matched but the experimental data from Env2 could never be matched. Jean Peccoud attempted to construct this network by hand and ran into the same problem. I do not recall the reasoning, but at the time there seemed to be something contradictory about that the data were suggesting and the theory of the network.

Isomorphic to d019 which could be optimized. If IPTG and aTc have same role both networks should optimize. Should be able to take d019 and swap TetR and LacI and have a solution.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	13119	554	392	577
Exp2	13841	517	492	646
Exp3	13711	331	421	836
Mean	13557	468	435	686
Opt	7012	7012	435	686

```
Molecules (16):
  GFP  DR=120.506
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=238622
  LacIX  DR=974606
  LacIg  IC=1
  LacIgX
  TetR  DR=9565.21
  TetRX  DR=615382
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=224105
  lambda cIg  IC=1
  lambda cIgX
```



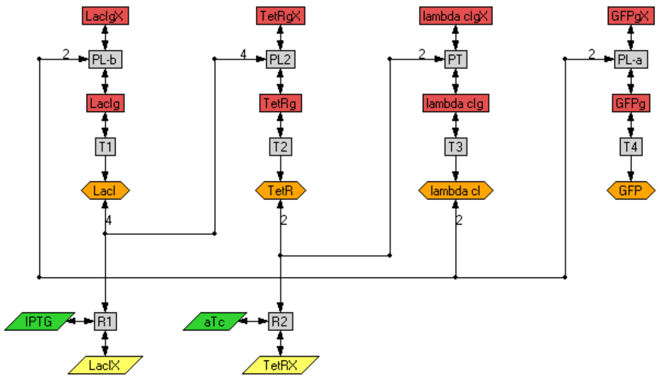
```
Reactions (22):
  D1:  LacI --> 0  Kf=238622
  D2:  TetR --> 0  Kf=9565.21
  D3:  lambda cI --> 0  Kf=224105
  D4:  GFP --> 0  Kf=120.506
  D5:  LacIX --> 0  Kf=974606
  D6:  TetRX --> 0  Kf=615382
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=854293  Kr=576795
  PL1a:  LacIg + 4LacI <-> LacIgX  Kf=628261  Kr=334889
  PL1b:  TetRg + 4LacI <-> TetRgX  Kf=628261  Kr=334889
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=716071  Kr=2317.44
  R1:  IPTG + LacI <-> LacIX  Kf=267066  Kr=10327.5
  R2:  aTc + TetR <-> TetRX  Kf=600864  Kr=992330
  T1:  LacIg --> LacIg + LacI  Kf=349374
  T2:  TetRg --> TetRg + TetR  Kf=890490
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=817812
  T4:  GFPg --> GFPg + GFP  Kf=845043
```

d066

Repressalator like.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	7673	1490	425	470
Exp2	7574	1628	387	387
Exp3	8314	1746	418	451
Mean	7854	1622	410	436
Opt	4737	4737	419	429

```
Molecules (16):
  GFP  DR=171.767
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=178964
  LacIX  DR=587690
  LacIg  IC=1
  LacIgX
  TetR  DR=23.3899
  TetRX  DR=625877
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=410721
  lambda cIg  IC=1
  lambda cIgX
```

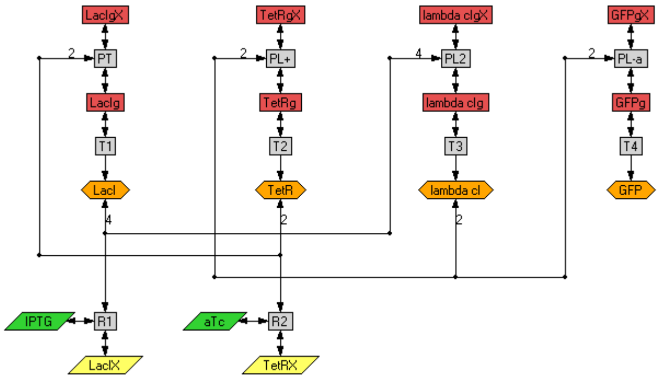


```
Reactions (22):
  D1:  LacI --> 0  Kf=178964
  D2:  TetR --> 0  Kf=23.3899
  D3:  lambda cI --> 0  Kf=410721
  D4:  GFP --> 0  Kf=171.767
  D5:  LacIX --> 0  Kf=587690
  D6:  TetRX --> 0  Kf=625877
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=991691  Kr=352617
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=991691  Kr=352617
  PL2:  TetRg + 4LacI <-> TetRgX  Kf=448874  Kr=244670
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=350211  Kr=8719.81
  R1:  IPTG + LacI <-> LacIX  Kf=329971  Kr=245577
  R2:  aTc + TetR <-> TetRX  Kf=270977  Kr=657054
  T1:  LacIg --> LacIg + LacI  Kf=880152
  T2:  TetRg --> TetRg + TetR  Kf=693412
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=862305
  T4:  GFPg --> GFPg + GFP  Kf=813716
```

d078

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	155	156	47482	380
Exp2	28	130	47895	430
Exp3	126	152	50391	413
Mean	103	146	48589	408
Opt	125	125	48589	408

```
Molecules (16):
  GFP  DR=14.4561
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=170030
  LacIX  DR=156698
  LacIg  IC=1
  LacIgX
  TetR  DR=2255.74
  TetRX  DR=313775
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=165843
  lambda cIg  IC=1
  lambda cIgX
```

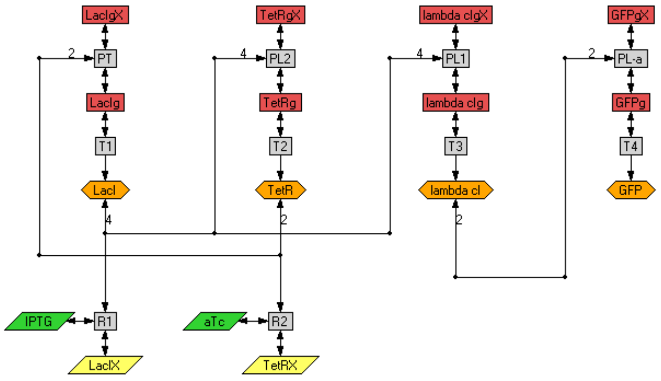


```
Reactions (22):
  D1:  LacI --> 0  Kf=170030
  D2:  TetR --> 0  Kf=2255.74
  D3:  lambda cI --> 0  Kf=165843
  D4:  GFP --> 0  Kf=14.4561
  D5:  LacIX --> 0  Kf=156698
  D6:  TetRX --> 0  Kf=313775
  PL+:  TetRg + 2(lambda cI) <-> TetRgX  Kf=306208  Kr=999970
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=590022  Kr=38906.5
  PL2:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=154017  Kr=29460.3
  PT:  LacIg + 2TetR <-> LacIgX  Kf=622448  Kr=37503.2
  R1:  IPTG + LacI <-> LacIX  Kf=88754.6  Kr=912393
  R2:  aTc + TetR <-> TetRX  Kf=572750  Kr=93924
  T1:  LacIg --> LacIg + LacI  Kf=924437
  T2:  TetRg --> TetRg + TetR  Kf=500922
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=840024
  T4:  GFPg --> GFPg + GFP  Kf=702424
```

d090

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	212	188	205	122
Exp2	183	205	45	198
Exp3	215	95	28	56
Mean	204	163	93	125
Opt	148	144	148	144

```
Molecules (16):
  GFP  DR=4255.32
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=164421
  LacIX  DR=659470
  LacIg  IC=1
  LacIgX
  TetR  DR=631411
  TetRX  DR=296473
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=688076
  lambda cIg  IC=1
  lambda cIgX
```



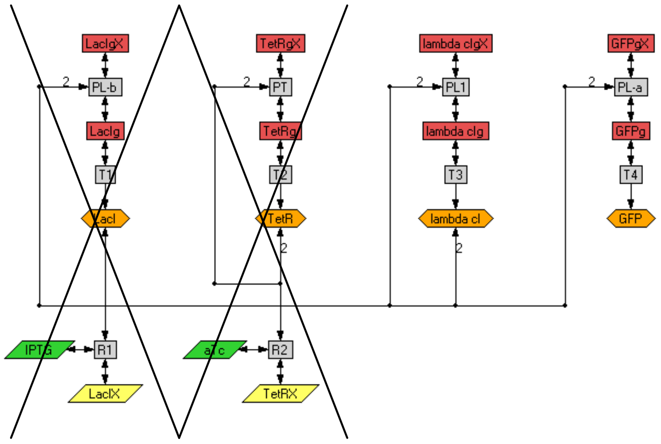
```
Reactions (22):
  D1:  LacI --> 0  Kf=164421
  D2:  TetR --> 0  Kf=631411
  D3:  lambda cI --> 0  Kf=688076
  D4:  GFP --> 0  Kf=4255.32
  D5:  LacIX --> 0  Kf=659470
  D6:  TetRX --> 0  Kf=296473
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=413694  Kr=361919
  PL1:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=427833  Kr=395052
  PL2:  TetRg + 4LacI <-> TetRgX  Kf=772882  Kr=778335
  PT:  LacIg + 2TetR <-> LacIgX  Kf=76309.8  Kr=698737
  R1:  IPTG + LacI <-> LacIX  Kf=773631  Kr=114916
  R2:  aTc + TetR <-> TetRX  Kf=511440  Kr=442122
  T1:  LacIg --> LacIg + LacI  Kf=873969
  T2:  TetRg --> TetRg + TetR  Kf=183488
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=110284
  T4:  GFPg --> GFPg + GFP  Kf=630636
```

d101

This is another wonderful example of the experimental data showing that there are errors in the theory. Neither of the environmental inputs have any apparent connection to the expression of GFP. Neither LacI nor tetR are involved in regulating λcI, but there is certainly a strong Boolean function being expressed in the observed data. Not surprisingly, there is no regulation present in the optimized network. Whatever mechanism that is responsible for the observed data is certainly not represented in the model. The presence of IPTG must be tying up λcI transcription factor protein somehow. Expression should have not be dependent on environmental signals. IPTG takes LacI out of the system, so lambda cI is not required. Without IPTG lambda cI is tied up. Why env1 != env3?

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	38940	547	4577	796
Exp2	38402	711	4635	905
Exp3	39242	518	4752	954
Mean	38861	592	4654	885
Opt	11248	11248	11248	11246

```
Molecules (16):
  GFP  DR=41.2634
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=628543
  LacIX  DR=676431
  LacIg  IC=1
  LacIgX
  TetR  DR=626336
  TetRX  DR=189727
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=802316
  lambda cIg  IC=1
  lambda cIgX
```



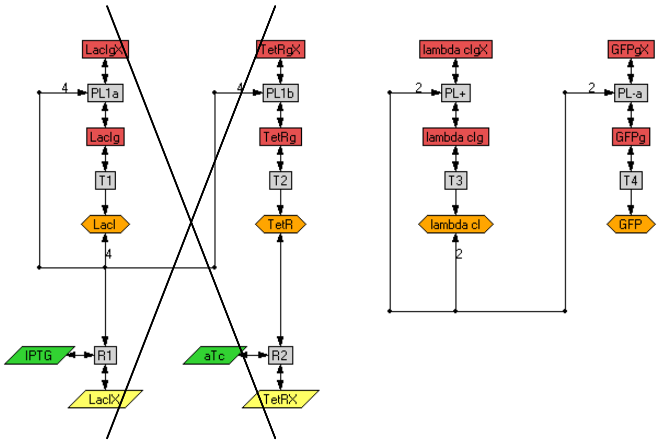
```
Reactions (22):
  D1:  LacI --> 0  Kf=628543
  D2:  TetR --> 0  Kf=626336
  D3:  lambda cI --> 0  Kf=802316
  D4:  GFP --> 0  Kf=41.2634
  D5:  LacIX --> 0  Kf=676431
  D6:  TetRX --> 0  Kf=189727
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=591742  Kr=591733
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=591742  Kr=591733
  PL1:  (lambda cIg) + 2(lambda cI) <-> (lambda cIgX)  Kf=600641
Kr=0.001191
  PT:  TetRg + 2TetR <-> TetRgX  Kf=532494  Kr=695404
  R1:  IPTG + LacI <-> LacIX  Kf=198217  Kr=482402
  R2:  aTc + TetR <-> TetRX  Kf=800901  Kr=203937
  T1:  LacIg --> LacIg + LacI  Kf=904276
  T2:  TetRg --> TetRg + TetR  Kf=497727
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=617.831
  T4:  GFPg --> GFPg + GFP  Kf=464138
```

d104

Similar to d101, neither environmental factor should have any influence on this network. Strangely the expression of GFP is considerably lower than in d101. Again, there may be something to λcI being tied up in regulation, in this case, of itself. Optimized network exhibited no regulation at all, which is what is expected. Perhaps a little higher in the fourth environment would have been nice. This network agrees with theory. Lambda CI is self inducing and represses GFP. Result GFP is always repressed. Similar to d101 but lambda cI has no requirement to bind to LacI.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	295	190	225	320
Exp2	222	254	264	352
Exp3	130	186	215	359
Mean	216	210	235	344
Opt	251	251	251	251

```
Molecules (16):
  GFP  DR=3980.38
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=663608
  LacIX  DR=773766
  LacIg  IC=1
  LacIgX
  TetR  DR=826580
  TetRX  DR=182648
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=937136
  lambda cIg  IC=1
  lambda cIgX
```



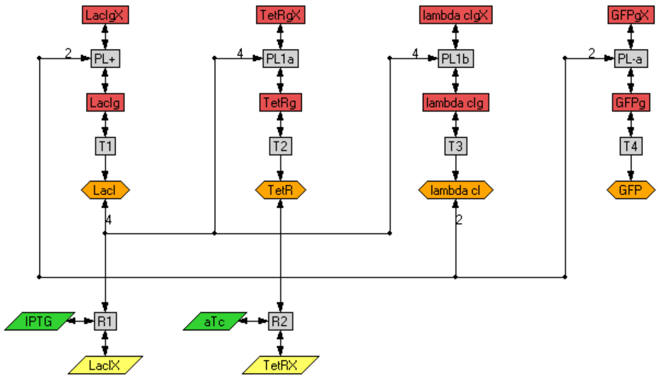
```
Reactions (22):
  D1:  LacI --> 0  Kf=663608
  D2:  TetR --> 0  Kf=826580
  D3:  lambda cI --> 0  Kf=937136
  D4:  GFP --> 0  Kf=3980.38
  D5:  LacIX --> 0  Kf=773766
  D6:  TetRX --> 0  Kf=182648
  PL+:  (lambda cIg) + 2(lambda cI) <-> (lambda cIgX)  Kf=983641
Kr=667247
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=931113  Kr=695206
  PL1a:  LacIg + 4LacI <-> LacIgX  Kf=348546  Kr=197833
  PL1b:  TetRg + 4LacI <-> TetRgX  Kf=348546  Kr=197833
  R1:  IPTG + LacI <-> LacIX  Kf=349493  Kr=404070
  R2:  aTc + TetR <-> TetRX  Kf=331007  Kr=967923
  T1:  LacIg --> LacIg + LacI  Kf=556080
  T2:  TetRg --> TetRg + TetR  Kf=722975
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=11605.2
  T4:  GFPg --> GFPg + GFP  Kf=999648
```

d113

Would expect {env1, env2} > {env3, env4}

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	34153	1878	37329	1140
Exp2	35760	1768	38170	1060
Exp3	35579	1864	39394	1177
Mean	35164	1837	38298	1126
Opt	36731	1481	36731	1481

```
Molecules (16):
  GFP  DR=20.3362
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=63613.9
  LacIX  DR=560147
  LacIg  IC=1
  LacIgX
  TetR  DR=454711
  TetRX  DR=526902
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=622375
  lambda cIg  IC=1
  lambda cIgX
```

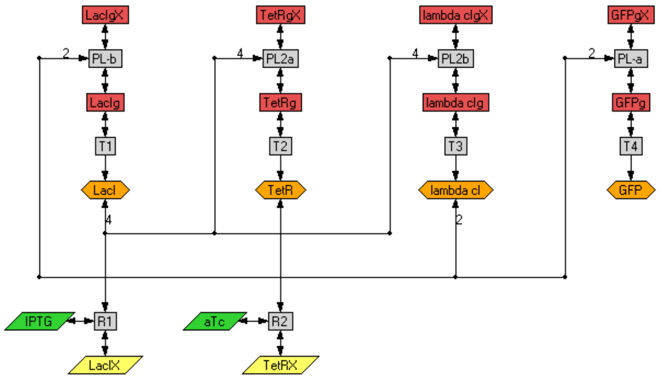


```
Reactions (22):
  D1:  LacI --> 0  Kf=63613.9
  D2:  TetR --> 0  Kf=454711
  D3:  lambda cI --> 0  Kf=622375
  D4:  GFP --> 0  Kf=20.3362
  D5:  LacIX --> 0  Kf=560147
  D6:  TetRX --> 0  Kf=526902
  PL+:  LacIg + 2(lambda cI) <-> LacIgX  Kf=461033  Kr=945892
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=554282  Kr=43361.1
  PL1a:  TetRg + 4LacI <-> TetRgX  Kf=224615  Kr=912496
  PL1b:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=224615  Kr=912496
  R1:  IPTG + LacI <-> LacIX  Kf=776916  Kr=289291
  R2:  aTc + TetR <-> TetRX  Kf=485438  Kr=211240
  T1:  LacIg --> LacIg + LacI  Kf=795358
  T2:  TetRg --> TetRg + TetR  Kf=539722
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=849225
  T4:  GFPg --> GFPg + GFP  Kf=746969
```

d114

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	185	326	493	467
Exp2	171	200	385	369
Exp3	300	302	462	412
Mean	219	276	446	416
Opt	339	339	339	339

```
Molecules (16):
  GFP  DR=2022.03
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=372063
  LacIX  DR=639956
  LacIg  IC=1
  LacIgX
  TetR  DR=235765
  TetRX  DR=554113
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=904719
  lambda cIg  IC=1
  lambda cIgX
```

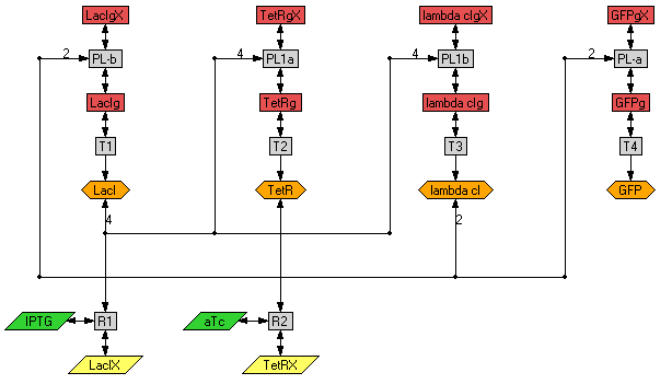


```
Reactions (22):
  D1:  LacI --> 0  Kf=372063
  D2:  TetR --> 0  Kf=235765
  D3:  lambda cI --> 0  Kf=904719
  D4:  GFP --> 0  Kf=2022.03
  D5:  LacIX --> 0  Kf=639956
  D6:  TetRX --> 0  Kf=554113
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=0.011607  Kr=630858
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=0.011607  Kr=630858
  PL2a:  TetRg + 4LacI <-> TetRgX  Kf=520435  Kr=165635
  PL2b:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=520435  Kr=165635
  R1:  IPTG + LacI <-> LacIX  Kf=931394  Kr=596329
  R2:  aTc + TetR <-> TetRX  Kf=616791  Kr=566215
  T1:  LacIg --> LacIg + LacI  Kf=132772
  T2:  TetRg --> TetRg + TetR  Kf=893129
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=11881.7
  T4:  GFPg --> GFPg + GFP  Kf=686187
```

d117

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	43974	47048	52142	53199
Exp2	45900	44002	50953	52515
Exp3	46322	47710	51506	51342
Mean	45399	46254	51534	52352
Opt	48884	48884	48884	48884

```
Molecules (16):
  GFP  DR=20.2238
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=605608
  LacIX  DR=898995
  LacIg  IC=1
  LacIgX
  TetR  DR=599154
  TetRX  DR=265194
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=617496
  lambda cIg  IC=1
  lambda cIgX
```



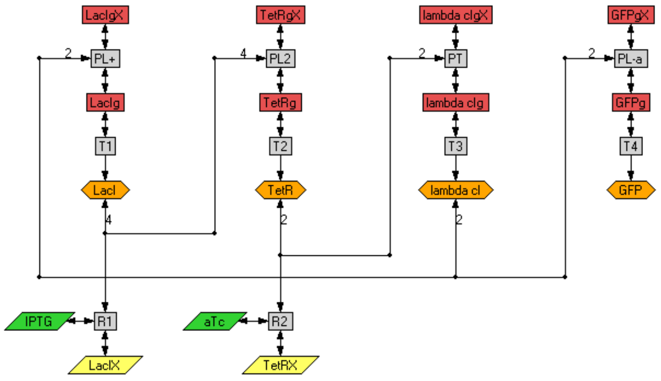
```
Reactions (22):
  D1:  LacI --> 0  Kf=605608
  D2:  TetR --> 0  Kf=599154
  D3:  lambda cI --> 0  Kf=617496
  D4:  GFP --> 0  Kf=20.2238
  D5:  LacIX --> 0  Kf=898995
  D6:  TetRX --> 0  Kf=265194
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=151820  Kr=542077
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=151820  Kr=542077
  PL1a:  TetRg + 4LacI <-> TetRgX  Kf=478051  Kr=912048
  PL1b:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=478051  Kr=912048
  R1:  IPTG + LacI <-> LacIX  Kf=0.467327  Kr=833851
  R2:  aTc + TetR <-> TetRX  Kf=353974  Kr=861973
  T1:  LacIg --> LacIg + LacI  Kf=188121
  T2:  TetRg --> TetRg + TetR  Kf=474191
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=0.403215
  T4:  GFPg --> GFPg + GFP  Kf=988632
```

d123

Repressalator like

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	570	723	397	388
Exp2	629	900	387	305
Exp3	762	977	423	157
Mean	653	867	402	283
Opt	654	867	343	343

```
Molecules (16):
  GFP  DR=958.766
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=251783
  LacIX  DR=244878
  LacIg  IC=1
  LacIgX
  TetR  DR=202925
  TetRX  DR=700063
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=555925
  lambda cIg  IC=1
  lambda cIgX
```

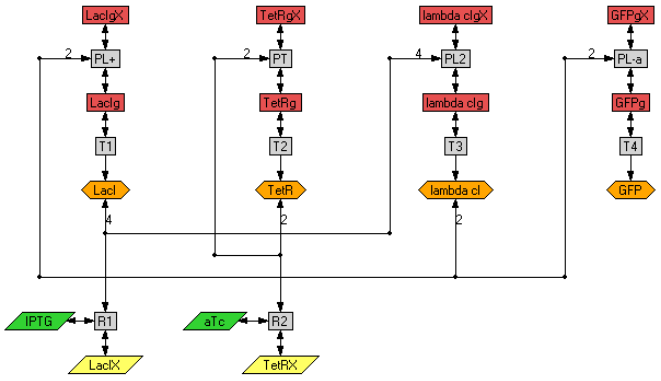


```
Reactions (22):
  D1:  LacI --> 0  Kf=251783
  D2:  TetR --> 0  Kf=202925
  D3:  lambda cI --> 0  Kf=555925
  D4:  GFP --> 0  Kf=958.766
  D5:  LacIX --> 0  Kf=244878
  D6:  TetRX --> 0  Kf=700063
  PL+:  LacIg + 2(lambda cI) <-> LacIgX  Kf=883720  Kr=199628
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=558657  Kr=719031
  PL2:  TetRg + 4LacI <-> TetRgX  Kf=691591  Kr=692971
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=492835  Kr=701118
  R1:  IPTG + LacI <-> LacIX  Kf=956356  Kr=220841
  R2:  aTc + TetR <-> TetRX  Kf=898519  Kr=412222
  T1:  LacIg --> LacIg + LacI  Kf=839707
  T2:  TetRg --> TetRg + TetR  Kf=562499
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=805156
  T4:  GFPg --> GFPg + GFP  Kf=863885
```

d133

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	44918	44056	45511	52095
Exp2	42908	44440	47256	52900
Exp3	42598	44889	50619	54159
Mean	43475	44462	47795	53051
Opt	47196	47196	47196	47196

```
Molecules (16):
  GFP  DR=1.1651
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=8094.8
  LacIX  DR=0.511133
  LacIg  IC=1
  LacIgX
  TetR  DR=485995
  TetRX  DR=602087
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=279468
  lambda cIg  IC=1
  lambda cIgX
```

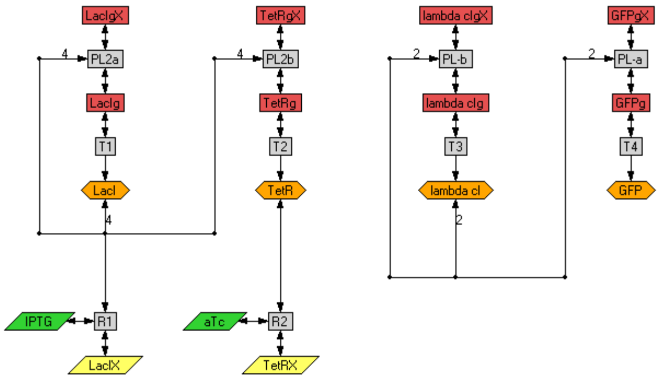


```
Reactions (22):
  D1:  LacI --> 0  Kf=8094.8
  D2:  TetR --> 0  Kf=485995
  D3:  lambda cI --> 0  Kf=279468
  D4:  GFP --> 0  Kf=1.1651
  D5:  LacIX --> 0  Kf=0.511133
  D6:  TetRX --> 0  Kf=602087
  PL+:  LacIg + 2(lambda cI) <-> LacIgX  Kf=126465  Kr=721532
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=15486.4  Kr=798191
  PL2:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=680064  Kr=333962
  PT:  TetRg + 2TetR <-> TetRgX  Kf=882468  Kr=739535
  R1:  IPTG + LacI <-> LacIX  Kf=0.323142  Kr=747716
  R2:  aTc + TetR <-> TetRX  Kf=126973  Kr=201933
  T1:  LacIg --> LacIg + LacI  Kf=948202
  T2:  TetRg --> TetRg + TetR  Kf=824955
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=925667
  T4:  GFPg --> GFPg + GFP  Kf=54987.9
```

d135

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	6115	5763	5795	5894
Exp2	6217	5621	5579	5974
Exp3	5933	5579	6087	5428
Mean	6088	5655	5820	5766
Opt	5832	5832	5832	5832

```
Molecules (16):
  GFP  DR=160.072
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=437457
  LacIX  DR=778747
  LacIg  IC=1
  LacIgX
  TetR  DR=586675
  TetRX  DR=726431
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=977181
  lambda cIg  IC=1
  lambda cIgX
```

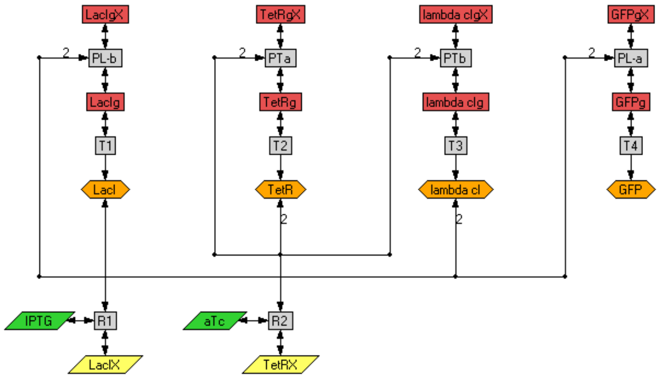


```
Reactions (22):
  D1:  LacI --> 0  Kf=437457
  D2:  TetR --> 0  Kf=586675
  D3:  lambda cI --> 0  Kf=977181
  D4:  GFP --> 0  Kf=160.072
  D5:  LacIX --> 0  Kf=778747
  D6:  TetRX --> 0  Kf=726431
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=3613.25  Kr=812462
  PL-b:  (lambda cIg) + 2(lambda cI) <-> (lambda cIgX)  Kf=3613.25
Kr=812462
  PL2a:  LacIg + 4LacI <-> LacIgX  Kf=822439  Kr=313557
  PL2b:  TetRg + 4LacI <-> TetRgX  Kf=822439  Kr=313557
  R1:  IPTG + LacI <-> LacIX  Kf=495632  Kr=687284
  R2:  aTc + TetR <-> TetRX  Kf=626813  Kr=379905
  T1:  LacIg --> LacIg + LacI  Kf=619234
  T2:  TetRg --> TetRg + TetR  Kf=795419
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=6944.09
  T4:  GFPg --> GFPg + GFP  Kf=933564
```

d143

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	14403	14629	621	464
Exp2	13999	13728	556	449
Exp3	14256	14016	528	441
Mean	14219	14124	568	451
Opt	14172	14172	510	510

```
Molecules (16):
  GFP  DR=63.7527
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=577670
  LacIX  DR=721273
  LacIg  IC=1
  LacIgX
  TetR  DR=174.206
  TetRX  DR=822797
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=230394
  lambda cIg  IC=1
  lambda cIgX
```

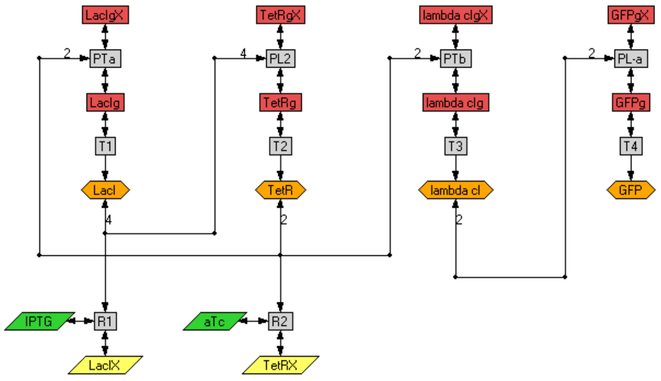


```
Reactions (22):
  D1:  LacI --> 0  Kf=577670
  D2:  TetR --> 0  Kf=174.206
  D3:  lambda cI --> 0  Kf=230394
  D4:  GFP --> 0  Kf=63.7527
  D5:  LacIX --> 0  Kf=721273
  D6:  TetRX --> 0  Kf=822797
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=774386  Kr=209093
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=774386  Kr=209093
  PTa:  TetRg + 2TetR <-> TetRgX  Kf=996155  Kr=60315.3
  PTb:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=996155  Kr=60315.3
  R1:  IPTG + LacI <-> LacIX  Kf=963074  Kr=203528
  R2:  aTc + TetR <-> TetRX  Kf=381229  Kr=649722
  T1:  LacIg --> LacIg + LacI  Kf=458237
  T2:  TetRg --> TetRg + TetR  Kf=555520
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=626527
  T4:  GFPg --> GFPg + GFP  Kf=903553
```

d180

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	46591	48572	753	824
Exp2	46652	47987	831	855
Exp3	49246	50113	790	978
Mean	47496	48891	791	886
Opt	48091	48830	1022	1043

```
Molecules (16):
  GFP  DR=9.70114
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=82765
  LacIX  DR=884439
  LacIg  IC=1
  LacIgX
  TetR  DR=824161
  TetRX  DR=351625
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=388291
  lambda cIg  IC=1
  lambda cIgX
```

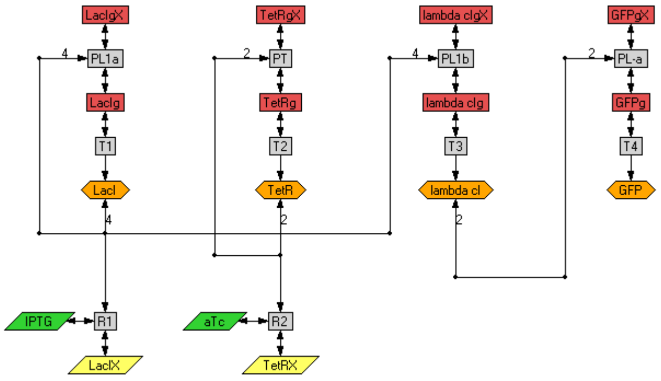


```
Reactions (22):
  D1:  LacI --> 0  Kf=82765
  D2:  TetR --> 0  Kf=824161
  D3:  lambda cI --> 0  Kf=388291
  D4:  GFP --> 0  Kf=9.70114
  D5:  LacIX --> 0  Kf=884439
  D6:  TetRX --> 0  Kf=351625
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=665825  Kr=46133.1
  PL2:  TetRg + 4LacI <-> TetRgX  Kf=132111  Kr=797110
  PTa:  LacIg + 2TetR <-> LacIgX  Kf=893773  Kr=120550
  PTb:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=893773  Kr=120550
  R1:  IPTG + LacI <-> LacIX  Kf=754150  Kr=116260
  R2:  aTc + TetR <-> TetRX  Kf=604750  Kr=548887
  T1:  LacIg --> LacIg + LacI  Kf=409254
  T2:  TetRg --> TetRg + TetR  Kf=922826
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=943623
  T4:  GFPg --> GFPg + GFP  Kf=854629
```

d250

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	774	329	400	145
Exp2	821	269	449	171
Exp3	791	294	214	233
Mean	796	297	354	183
Opt	575	240	575	240

```
Molecules (16):
  GFP  DR=1512.57
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=12547.5
  LacIX  DR=43377.7
  LacIg  IC=1
  LacIgX
  TetR  DR=420233
  TetRX  DR=465376
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=448462
  lambda cIg  IC=1
  lambda cIgX
```

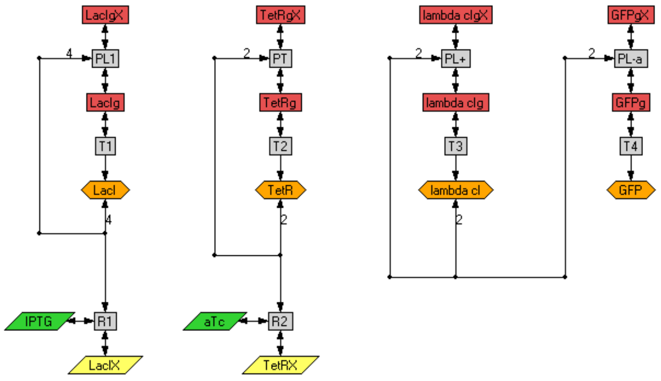


```
Reactions (22):
  D1:  LacI --> 0  Kf=12547.5
  D2:  TetR --> 0  Kf=420233
  D3:  lambda cI --> 0  Kf=448462
  D4:  GFP --> 0  Kf=1512.57
  D5:  LacIX --> 0  Kf=43377.7
  D6:  TetRX --> 0  Kf=465376
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=380168  Kr=992123
  PL1a:  LacIg + 4LacI <-> LacIgX  Kf=26208.8  Kr=655176
  PL1b:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=26208.8  Kr=655176
  PT:  TetRg + 2TetR <-> TetRgX  Kf=628449  Kr=723560
  R1:  IPTG + LacI <-> LacIX  Kf=529351  Kr=581313
  R2:  aTc + TetR <-> TetRX  Kf=851647  Kr=369405
  T1:  LacIg --> LacIg + LacI  Kf=972058
  T2:  TetRg --> TetRg + TetR  Kf=49799.3
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=859654
  T4:  GFPg --> GFPg + GFP  Kf=873912
```

d253

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	912	792	436	324
Exp2	1014	723	262	157
Exp3	730	883	279	131
Mean	885	799	326	204
Opt	554	554	554	554

```
Molecules (16):
  GFP  DR=1226.27
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=357066
  LacIX  DR=471867
  LacIg  IC=1
  LacIgX
  TetR  DR=891995
  TetRX  DR=734408
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=971965
  lambda cIg  IC=1
  lambda cIgX
```

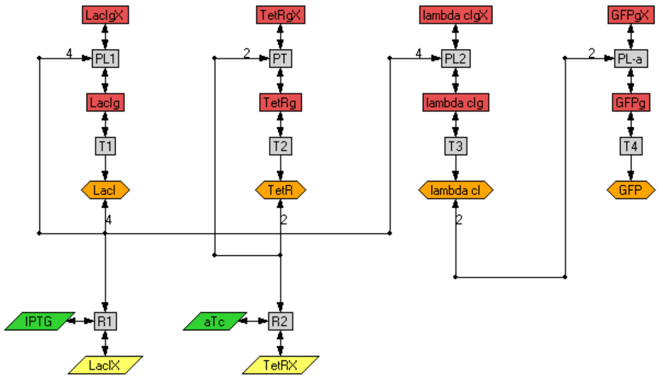


```
Reactions (22):
  D1:  LacI --> 0  Kf=357066
  D2:  TetR --> 0  Kf=891995
  D3:  lambda cI --> 0  Kf=971965
  D4:  GFP --> 0  Kf=1226.27
  D5:  LacIX --> 0  Kf=471867
  D6:  TetRX --> 0  Kf=734408
  PL+:  (lambda cIg) + 2(lambda cI) <-> (lambda cIgX)  Kf=503524
Kr=819094
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=6727.2  Kr=285612
  PL1:  LacIg + 4LacI <-> LacIgX  Kf=210194  Kr=160178
  PT:  TetRg + 2TetR <-> TetRgX  Kf=452829  Kr=314019
  R1:  IPTG + LacI <-> LacIX  Kf=305309  Kr=862778
  R2:  aTc + TetR <-> TetRX  Kf=39390.1  Kr=238919
  T1:  LacIg --> LacIg + LacI  Kf=592507
  T2:  TetRg --> TetRg + TetR  Kf=114365
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=15662.1
  T4:  GFPg --> GFPg + GFP  Kf=678922
```

c024

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	14090	471	575	517
Exp2	14355	422	502	311
Exp3	13322	424	540	211
Mean	13922	439	539	346
Opt	7230	392	7230	392

```
Molecules (16):
  GFP  DR=122.416
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=313487
  LacIX  DR=383603
  LacIg  IC=1
  LacIgX
  TetR  DR=252578
  TetRX  DR=961257
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=138530
  lambda cIg  IC=1
  lambda cIgX
```



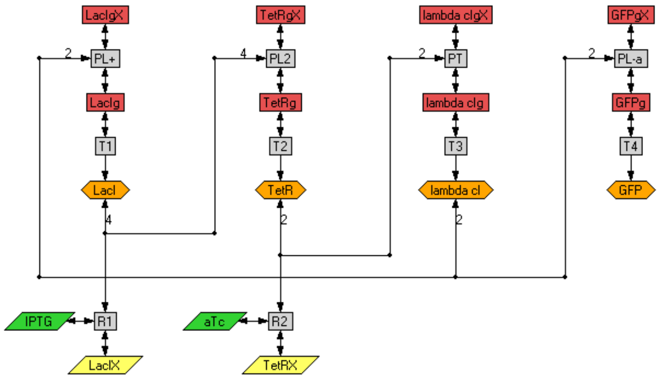
```
Reactions (22):
  D1:  LacI --> 0  Kf=313487
  D2:  TetR --> 0  Kf=252578
  D3:  lambda cI --> 0  Kf=138530
  D4:  GFP --> 0  Kf=122.416
  D5:  LacIX --> 0  Kf=383603
  D6:  TetRX --> 0  Kf=961257
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=774422  Kr=271148
  PL1:  LacIg + 4LacI <-> LacIgX  Kf=885325  Kr=966934
  PL2:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=868220  Kr=0.007801
  PT:  TetRg + 2TetR <-> TetRgX  Kf=228857  Kr=172739
  R1:  IPTG + LacI <-> LacIX  Kf=652946  Kr=123384
  R2:  aTc + TetR <-> TetRX  Kf=621296  Kr=344235
  T1:  LacIg --> LacIg + LacI  Kf=569136
  T2:  TetRg --> TetRg + TetR  Kf=522528
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=832479
  T4:  GFPg --> GFPg + GFP  Kf=885036
```

c101

Repressalator like

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	185	8792	238	241
Exp2	176	9264	314	174
Exp3	170	9309	339	266
Mean	177	9121	297	227
Opt	234	9121	234	234

```
Molecules (16):
  GFP  DR=101.364
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=17551.4
  LacIX  DR=71389.3
  LacIg  IC=1
  LacIgX
  TetR  DR=17542.3
  TetRX  DR=864751
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=174406
  lambda cIg  IC=1
  lambda cIgX
```



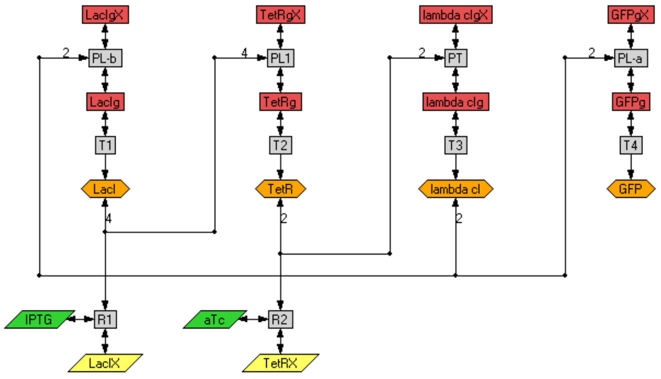
```
Reactions (22):
  D1:  LacI --> 0  Kf=17551.4
  D2:  TetR --> 0  Kf=17542.3
  D3:  lambda cI --> 0  Kf=174406
  D4:  GFP --> 0  Kf=101.364
  D5:  LacIX --> 0  Kf=71389.3
  D6:  TetRX --> 0  Kf=864751
  PL+:  LacIg + 2(lambda cI) <-> LacIgX  Kf=28199.7  Kr=507462
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=687110  Kr=332990
  PL2:  TetRg + 4LacI <-> TetRgX  Kf=464604  Kr=291449
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=607930  Kr=464387
  R1:  IPTG + LacI <-> LacIX  Kf=968381  Kr=33058.1
  R2:  aTc + TetR <-> TetRX  Kf=530238  Kr=670970
  T1:  LacIg --> LacIg + LacI  Kf=620472
  T2:  TetRg --> TetRg + TetR  Kf=136641
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=751270
  T4:  GFPg --> GFPg + GFP  Kf=930047
```

c103

Repressalator like

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	11489	2538	883	679
Exp2	11207	2360	811	714
Exp3	11220	2425	760	569
Mean	11305	2441	818	654
Opt	6873	6873	736	736

```
Molecules (16):
  GFP  DR=130.302
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=478905
  LacIX  DR=458104
  LacIg  IC=1
  LacIgX
  TetR  DR=12437.6
  TetRX  DR=820482
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=433949
  lambda cIg  IC=1
  lambda cIgX
```



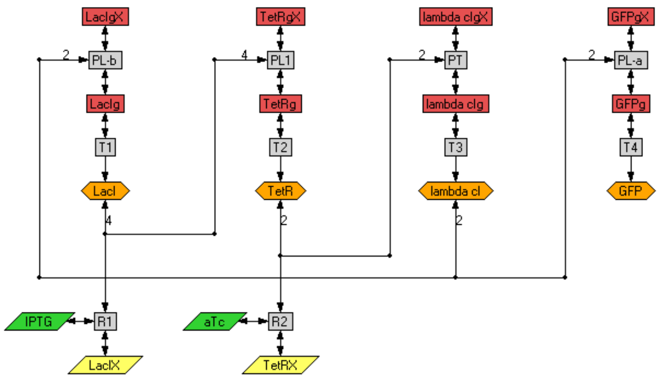
```
Reactions (22):
  D1:  LacI --> 0  Kf=478905
  D2:  TetR --> 0  Kf=12437.6
  D3:  lambda cI --> 0  Kf=433949
  D4:  GFP --> 0  Kf=130.302
  D5:  LacIX --> 0  Kf=458104
  D6:  TetRX --> 0  Kf=820482
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=915203  Kr=262809
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=915203  Kr=262809
  PL1:  TetRg + 4LacI <-> TetRgX  Kf=75972.3  Kr=946042
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=955199  Kr=52756.8
  R1:  IPTG + LacI <-> LacIX  Kf=287193  Kr=595386
  R2:  aTc + TetR <-> TetRX  Kf=955178  Kr=372583
  T1:  LacIg --> LacIg + LacI  Kf=12102.8
  T2:  TetRg --> TetRg + TetR  Kf=963035
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=674109
  T4:  GFPg --> GFPg + GFP  Kf=895548
```

c113

Repressalator like. Example of contradictory network.

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	13326	1784	395	529
Exp2	13442	1751	446	504
Exp3	13594	1860	501	560
Mean	13454	1798	447	531
Opt	7626	7626	489	489

```
Molecules (16):
  GFP  DR=119.564
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=585969
  LacIX  DR=274465
  LacIg  IC=1
  LacIgX
  TetR  DR=933.293
  TetRX  DR=33773.6
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=265282
  lambda cIg  IC=1
  lambda cIgX
```

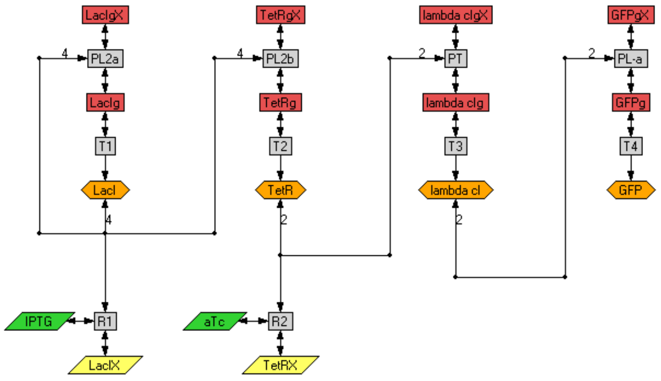


```
Reactions (22):
  D1:  LacI --> 0  Kf=585969
  D2:  TetR --> 0  Kf=933.293
  D3:  lambda cI --> 0  Kf=265282
  D4:  GFP --> 0  Kf=119.564
  D5:  LacIX --> 0  Kf=274465
  D6:  TetRX --> 0  Kf=33773.6
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=818723  Kr=297459
  PL-b:  LacIg + 2(lambda cI) <-> LacIgX  Kf=818723  Kr=297459
  PL1:  TetRg + 4LacI <-> TetRgX  Kf=823991  Kr=410017
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=459434  Kr=37165.3
  R1:  IPTG + LacI <-> LacIX  Kf=155224  Kr=580634
  R2:  aTc + TetR <-> TetRX  Kf=800675  Kr=345053
  T1:  LacIg --> LacIg + LacI  Kf=645601
  T2:  TetRg --> TetRg + TetR  Kf=957150
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=746506
  T4:  GFPg --> GFPg + GFP  Kf=911824
```

c144

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	33708	4795	518	679
Exp2	30473	4786	586	605
Exp3	30690	4993	499	684
Mean	31623	4858	534	656
Opt	18241	18241	534	656

```
Molecules (16):
  GFP  DR=49.8234
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=700408
  LacIX  DR=502657
  LacIg  IC=1
  LacIgX
  TetR  DR=21150.1
  TetRX  DR=822905
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=334013
  lambda cIg  IC=1
  lambda cIgX
```

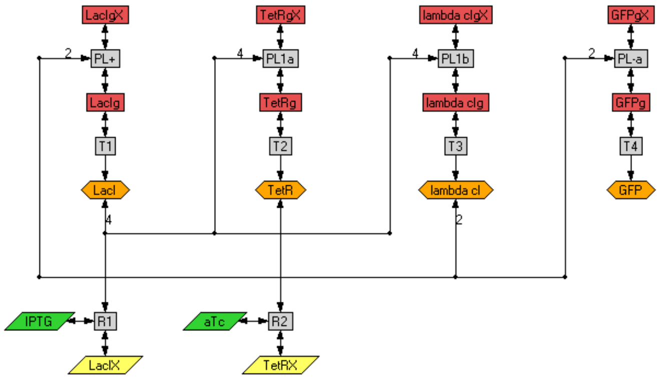


```
Reactions (22):
  D1:  LacI --> 0  Kf=700408
  D2:  TetR --> 0  Kf=21150.1
  D3:  lambda cI --> 0  Kf=334013
  D4:  GFP --> 0  Kf=49.8234
  D5:  LacIX --> 0  Kf=502657
  D6:  TetRX --> 0  Kf=822905
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=312709  Kr=25039.6
  PL2a:  LacIg + 4LacI <-> LacIgX  Kf=793075  Kr=866721
  PL2b:  TetRg + 4LacI <-> TetRgX  Kf=793075  Kr=866721
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=878762  Kr=643.341
  R1:  IPTG + LacI <-> LacIX  Kf=450815  Kr=909866
  R2:  aTc + TetR <-> TetRX  Kf=537540  Kr=925684
  T1:  LacIg --> LacIg + LacI  Kf=529362
  T2:  TetRg --> TetRg + TetR  Kf=543542
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=797378
  T4:  GFPg --> GFPg + GFP  Kf=908813
```

c195

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	46152	17778	66269	57863
Exp2	45979	18337	69483	56723
Exp3	46683	18581	70244	55822
Mean	46271	18232	68665	56803
Opt	45019	37746	57352	37746

```
Molecules (16):
  GFP  DR=10.9113
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=160761
  LacIX  DR=264858
  LacIg  IC=1
  LacIgX
  TetR  DR=435654
  TetRX  DR=857612
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=841397
  lambda cIg  IC=1
  lambda cIgX
```

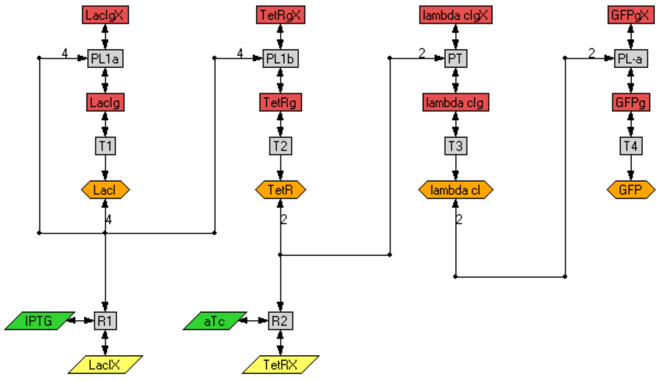


```
Reactions (22):
  D1:  LacI --> 0  Kf=160761
  D2:  TetR --> 0  Kf=435654
  D3:  lambda cI --> 0  Kf=841397
  D4:  GFP --> 0  Kf=10.9113
  D5:  LacIX --> 0  Kf=264858
  D6:  TetRX --> 0  Kf=857612
  PL+:  LacIg + 2(lambda cI) <-> LacIgX  Kf=283743  Kr=95286.9
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=373531  Kr=381048
  PL1a:  TetRg + 4LacI <-> TetRgX  Kf=322569  Kr=463059
  PL1b:  (lambda cIg) + 4LacI <-> (lambda cIgX)  Kf=322569  Kr=463059
  R1:  IPTG + LacI <-> LacIX  Kf=143810  Kr=48128.9
  R2:  aTc + TetR <-> TetRX  Kf=438482  Kr=599483
  T1:  LacIg --> LacIg + LacI  Kf=261675
  T2:  TetRg --> TetRg + TetR  Kf=832260
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=635933
  T4:  GFPg --> GFPg + GFP  Kf=642495
```

c242

	Env1	Env2	Env3	Env4
IPTG/aTc	-/-	+/-	-/+	+/+
Exp1	17494	737	585	436
Exp2	17498	749	598	442
Exp3	17220	662	519	449
Mean	17404	716	568	442
Opt	9060	9060	505	505

```
Molecules (16):
  GFP  DR=87.1502
  GFPg  IC=1
  GFPgX
  IPTG  (Control Variable)
  LacI  DR=955302
  LacIX  DR=504769
  LacIg  IC=1
  LacIgX
  TetR  DR=12040.9
  TetRX  DR=402519
  TetRg  IC=1
  TetRgX
  aTc  (Control Variable)
  lambda cI  DR=319297
  lambda cIg  IC=1
  lambda cIgX
```



```
Reactions (22):
  D1:  LacI --> 0  Kf=955302
  D2:  TetR --> 0  Kf=12040.9
  D3:  lambda cI --> 0  Kf=319297
  D4:  GFP --> 0  Kf=87.1502
  D5:  LacIX --> 0  Kf=504769
  D6:  TetRX --> 0  Kf=402519
  PL-a:  GFPg + 2(lambda cI) <-> GFPgX  Kf=661761  Kr=336800
  PL1a:  LacIg + 4LacI <-> LacIgX  Kf=566771  Kr=612637
  PL1b:  TetRg + 4LacI <-> TetRgX  Kf=566771  Kr=612637
  PT:  (lambda cIg) + 2TetR <-> (lambda cIgX)  Kf=714408  Kr=824584
  R1:  IPTG + LacI <-> LacIX  Kf=849315  Kr=262558
  R2:  aTc + TetR <-> TetRX  Kf=340725  Kr=342762
  T1:  LacIg --> LacIg + LacI  Kf=14984.6
  T2:  TetRg --> TetRg + TetR  Kf=772419
  T3:  (lambda cIg) --> (lambda cIg) + (lambda cI)  Kf=938979
  T4:  GFPg --> GFPg + GFP  Kf=789570
```