

Automatic Synthesis of Both the Topology and Sizing of Metabolic Pathways using Genetic Programming

John R. Koza
Stanford University
Stanford, California
koza@stanford.edu

William Mydlowec
Genetic Programming Inc.
Los Altos, California
bill@pharmix.com

Guido Lanza
Genetic Programming Inc.
Los Altos, California
guido@pharmix.com

Jessen Yu
Genetic Programming Inc.
Los Altos, California
jyu@cs.stanford.edu

Martin A. Keane
Econometrics Inc.
Chicago, Illinois
makeane@ix.netcom.com

Abstract

The concentrations of substances participating in networks of chemical reactions are modeled by non-linear continuous-time differential equations. Recent work has demonstrated that genetic programming is capable of automatically creating complex networks (e.g., analog electrical circuits, controllers) whose behavior is modeled by linear and non-linear continuous-time differential equations and whose behavior matches prespecified output values. This paper describes how genetic programming can be used to automatically synthesize (reverse engineer) both the topology of the network of chemical reactions and the rates (sizing) of each reaction of a network such that the behavior of the automatically created network matches the observed time-domain data. Genetic programming has automatically created metabolic pathways that contain noteworthy topological features, such as an internal feedback loop, a bifurcation point where one substance is distributed to two different reactions, and an accumulation point where one substance is accumulated from two sources.

1 INTRODUCTION

Considerable amounts of time-domain data are now becoming available concerning the concentration of biologically important chemicals in living organisms. Such data include both gene expression data (obtained from microarrays) and data on the concentration of substances participating in metabolic pathways (Ptashne 1992; McAdams and Shapiro 1995; Loomis and Sternberg 1995; Arkin, Shen, and Ross 1997; Yuh, Bolouri, and Davidson 1998; Laing, Fuhrman, and Somogyi 1998; Mendes and Kell 1998; D'haeseleer,

Wen, Fuhrman, and Somogyi 1999; Bower and Bolouri 2000).

A living cell can be viewed as a dynamical system in which a large number of different substances react continuously and non-linearly with one another. In order to understand the behavior of a continuous non-linear dynamical system with numerous interacting parts, it is usually insufficient to study behavior of each part in isolation. Instead, the behavior must usually be analyzed as a whole (Tomita et al. 1999). The concentrations of substrates, products, and catalysts (e.g., enzymes) participating in chemical reactions are modeled by non-linear continuous-time differential equations, such as the Michaelis-Menten equations (Voit 2000).

The question arises as to whether it is possible to start with observed time-domain concentrations of substances and automatically create both the topology of the network of chemical reactions and the rates of each reaction that produced the observed data — that is, to automatically reverse engineer the network.

Recent work (Koza, Bennett, Andre, and Keane 1999; Koza, Bennett, Andre, Keane, and Brave 1999) has demonstrated that genetic programming can automatically create complex networks that exhibit prespecified behavior in fields where the network's behavior is governed by differential equations (both linear and non-linear). For example, genetic programming is capable of automatically creating both the topology and sizing (component values) for analog electrical circuits (e.g., filters, amplifiers, computational circuits) composed of transistors, capacitors, resistors, and other components merely by specifying the circuit's output — that is, the output data values that would be observed if one already had the circuit. This reverse engineering of circuits from data is performed by genetic programming even though there is no general mathematical method for creating the topology and sizing of analog electrical circuits from the circuit's desired (or observed) behavior (Koza, Bennett, Andre, and Keane 1999). Seven of these circuits infringe on previously issued patents. Others

duplicate the functionality of previously patented inventions in a novel way.

As another example, genetic programming is capable of automatically creating both the topology and sizing (tuning) for controllers composed of time-domain blocks such as integrators, differentiators, multipliers, adders, delays, leads, and lags merely by specifying the controller's effect on the to-be-controlled plant (Koza, Keane, Yu, Bennett, and Mydlowec 2000). This reverse engineering of controllers from data is performed by genetic programming even though there is no general mathematical method for creating the topology and sizing for controllers from the controller's behavior. Two of the automatically created controllers infringe on previously issued patents.

As yet another example, it is possible to automatically create antennas composed of a network of wires merely by specifying the antenna's high-level specifications (Comisky, Yu, and Koza 2000).

Our approach to the problem of automatically creating both the topology and sizing of a network of chemical reactions involves

- (1) establishing a representation involving program trees (composed of functions and terminals) for chemical networks,
- (2) converting each individual program tree in the population into an electrical circuit representing a network of chemical reactions,
- (3) obtaining the behavior of the network of chemical reactions by simulating the electrical circuit,
- (4) defining a fitness measure that measures how well the behavior of an individual network in the population matches the observed data, and

- (5) applying genetic programming to breed a population of improving program trees using the fitness measure.

2 STATEMENT OF PROBLEM

The goal is to automatically create *both* the topology and sizing of a network of chemical reactions.

The *topology* of a network of chemical reactions comprises (1) the number of substrates consumed by each reaction, (2) the number of products produced by each reaction, (3) the pathways supplying the substrates (either from external sources or other reactions) to the reactions, and (4) the pathways dispersing the reaction's products (either to other reactions or external outputs). The *sizing* of a network of chemical reactions consists of the numerical values representing the rates of each reaction.

We chose, as an illustrative problem, a network (figure 1) that incorporates three noteworthy topological features. These features include an internal feedback loop, a bifurcation point (where one substance is distributed to two different reactions), and an accumulation point (where one substance is accumulated from two sources). The particular chosen network is part of a phospholipid cycle, as presented in the E-CELL cell simulation model (Tomita et al. 1999). The network's external inputs are glycerol and fatty acid. The network's final product is diacyl-glycerol. The network's four reactions are catalyzed by the enzyme Glycerol kinase (called EC2.7.1.30 by the Enzyme Nomenclature Commission), Glycerol-1-phosphatase (EC3.1.3.21), Acylglycerol lipase (EC3.1.1.23), and Triacylglycerol lipase (EC3.1.1.3).

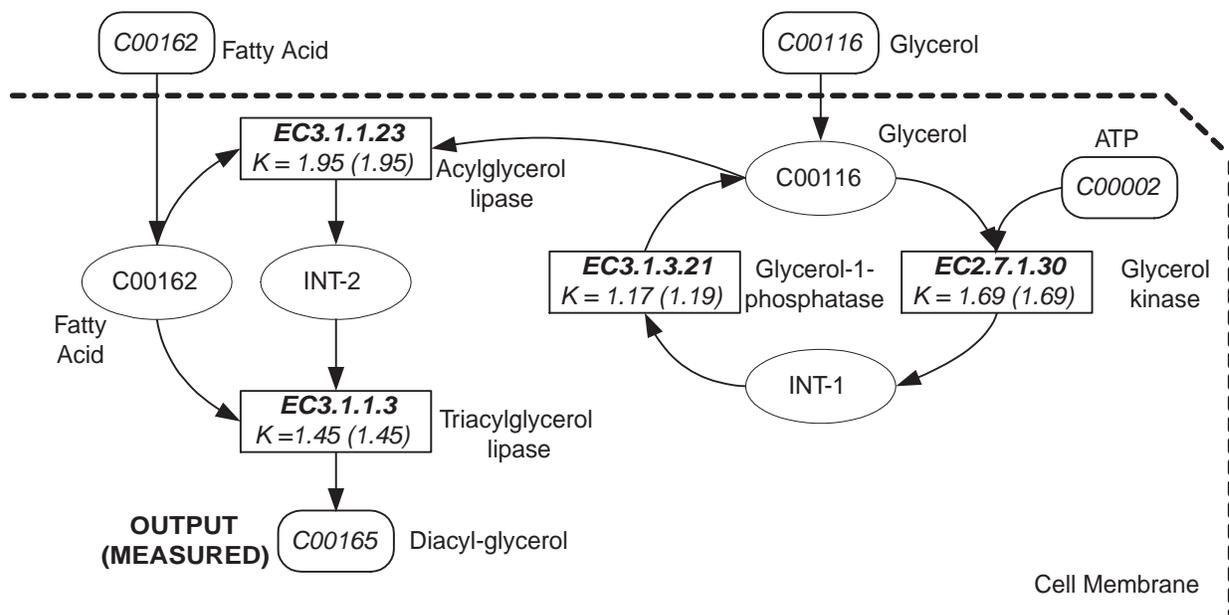


Figure 1 Best of run individual from generation 225.

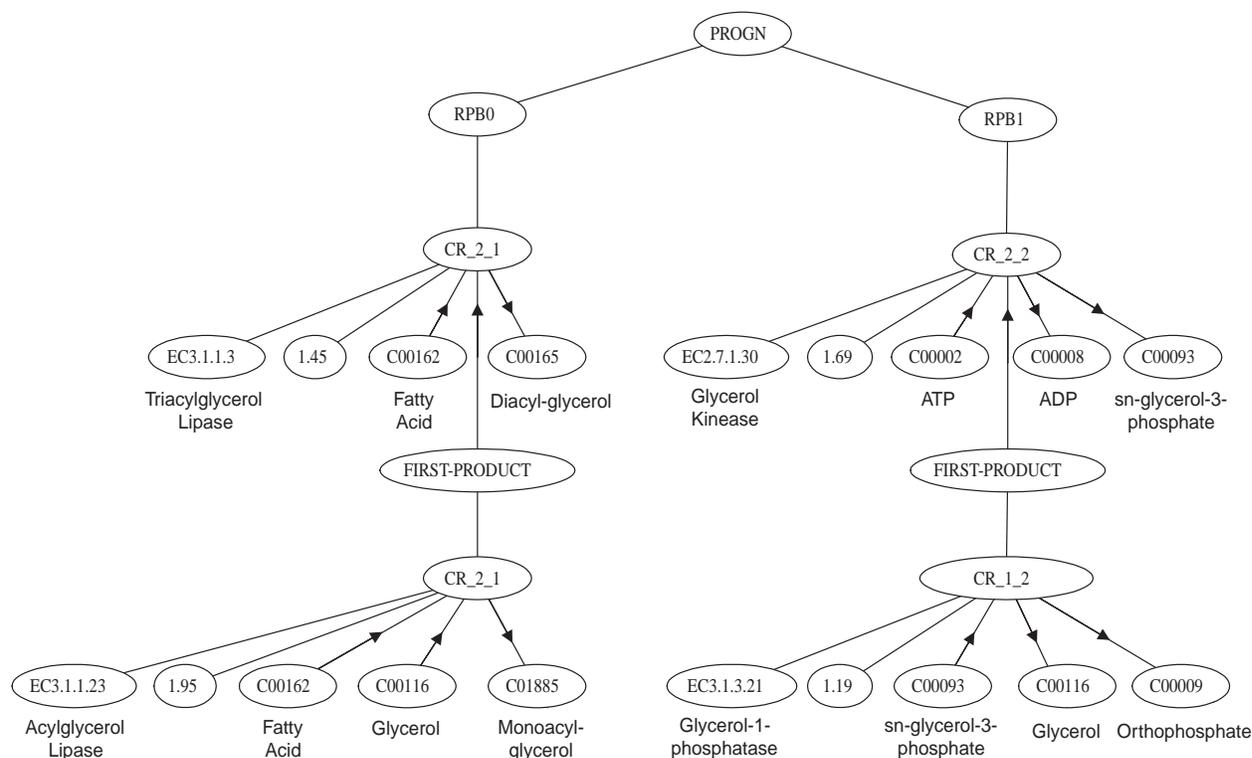


Figure 2 Program tree corresponding to network of chemical reactions of figure 1.

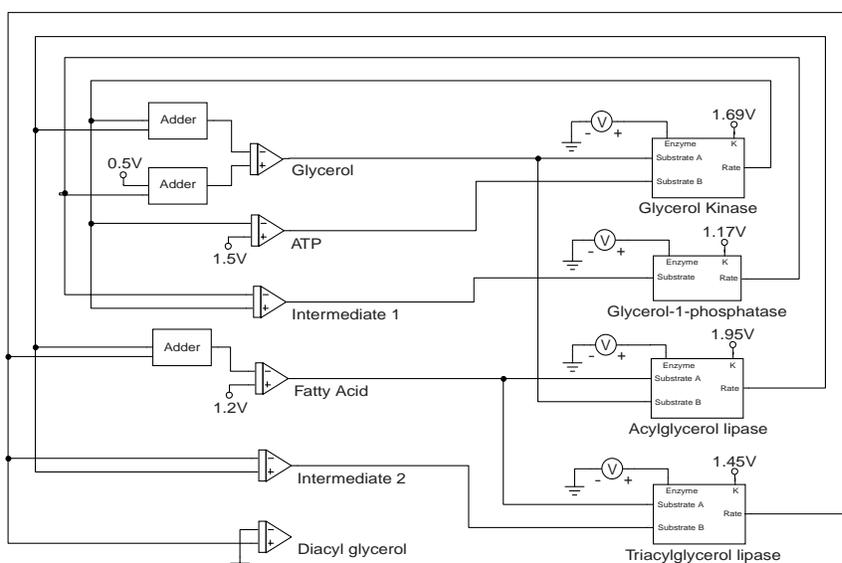


Figure 3 Electrical circuit corresponding to the chemical reaction network of figure 1.

3 REPRESENTATION OF A NETWORK OF CHEMICAL REACTIONS

Each program tree represents an interconnected network of chemical reactions involving various substances. A chemical reaction may consume one or two substances and may produce one or two substances. The consumed substances may be external input substances or may be intermediate

substances produced by reactions. The reactions, enzymes, and substances of a network may be completely represented by a program tree that contains

- internal nodes representing reaction functions,
- internal nodes representing selector functions that select the reaction's first versus the reaction's second product (if any),

- external points (leaves) representing substances that are consumed and produced by a reaction,
- external points representing enzymes that catalyze a reaction, and,
- external points representing numerical constants (reaction rates).

3.1 REPERTOIRE OF FUNCTIONS

The first argument of each chemical reaction function identifies the enzyme that catalyzes the reaction. The second argument is a numerical value that specifies the reaction's rate. In addition, there are two, three, or four arguments specifying the substrate(s) and product(s) of the reaction. Table 1 shows the number of substrate(s) and product(s) and overall arity for each of the four chemical reaction functions. The functions are first-order and second-order rate laws.

Table 1 Four chemical reaction functions.

Function	Substrates	Products	Arity
CR_1_1	1	1	4
CR_1_2	1	2	5
CR_2_1	2	1	5
CR_2_2	2	2	6

Each function returns a list composed of the reaction's one or two products. The one-argument `FIRST` function returns the first of the one or two products produced by the function designated by its argument. The one-argument `SECOND` function returns the second of the two products (or the first product, if the reaction produces only one product).

3.2 REPERTOIRE OF TERMINALS

Some terminals represent substances (input substances, intermediate substances created by reactions, or output substances). Other terminals represent the enzymes that catalyze the chemical reactions. Still other terminals represent numerical constants for the rate of the reactions.

3.3 CONSTRAINED SYNTACTIC STRUCTURE

The trees are constructed in accordance with a constrained syntactic structure. The root of every result-producing branch must be a chemical reaction function. The enzyme that catalyzes a reaction always appears as the first argument of its chemical reaction function. A numerical value representing a reaction's rate always appears as the second argument of its chemical reaction function. The one or two input arguments to a chemical reaction function can be either a substance terminal or selector function (`FIRST` or `SECOND`). The result of having a selector function as an input argument is to create a cascade of reactions. The one or two output arguments to a chemical reaction function must be substance terminals. The argument to a one-argument selector

function (`FIRST` or `SECOND`) is always a chemical reaction function.

3.4 EXAMPLE

Figure 1 shows an illustrative network of chemical reactions represented by a program tree. In fact, this figure is the outcome of the run (section 5) as well as the desired network (with the desired rates of each reaction being in parenthesis and the genetically evolved rate outside the parenthesis).

Figure 2 is a program tree corresponding to network of chemical reactions of figure 1. Figure 3 shows the electrical circuit corresponding to the network of figure 1 (where triangles represent integrators). For additional details, see Koza, Mydlowec, Lanza, Yu, and Keane 2000.

4 PREPARATORY STEPS

4.1 PROGRAM ARCHITECTURE

Each program tree in the initial random population (generation 0) has one result-producing branch. In subsequent generations, the architecture-altering operations (patterned after gene duplication and gene deletion in nature) may insert and delete result-producing branches to particular individual program trees in the population. Each program tree may have four result-producing branches.

4.2 FUNCTION SET

The function set, F , is

$$F = \{\text{CR1_1, CR1_2, CR2_1, CR2_2, FIRST, SECOND}\}.$$

4.3 TERMINAL SET

The terminal set, T , is

$$T = \{\mathfrak{R}, \text{C00116, C00162, C00002, C00165, INT_1, INT_2, INT_3, EC2_7_1_30, EC3_1_3_21, EC3_1_1_23, EC3_1_1_3}\}.$$

\mathfrak{R} denotes a perturbable numerical value. In generation 0, each perturbable numerical value is set, individually and separately, to a random value in a chosen range (from 0.0 and 2.0 here).

C00116 (following the notation of the E-CELL cell simulation model) is the concentration of glycerol. C00162 is the concentration of fatty acid. These two substances are inputs to the illustrative overall network of interest herein. C00002 is the concentration of the cofactor ATP. C00165 is the concentration of diacyl-glycerol. This substance is the final product of the illustrative network herein. INT_1, INT_2, and INT_3 are the concentrations of intermediate substances 1, 2, and 3 (respectively).

EC2_7_1_30, EC3_1_3_21, EC3_1_1_23, and EC3_1_1_3 are enzymes.

4.4 FITNESS MEASURE

Genetic programming is a probabilistic algorithm that searches the space of compositions of the available functions and terminals under the guidance of a fitness measure. In order to evaluate the fitness of an individual program tree in the population, the program tree is converted into a directed graph representing the network. The result-producing branches are executed from left to right. The functions in a particular result-producing branch are executed in a depth-first manner. One reactor (representing the concentration of the substances participating in the reaction) is inserted into the network for each chemical reaction function that is encountered in a branch. The reactor is labeled with the reaction's enzyme and rate. A directed line entering the reactor is added for each of the reaction's one or two substrate(s). A directed line leaving the reactor is added for each of the reaction's one or two product(s). The reaction's first product is selected whenever a FIRST function is encountered in a branch. The reaction's second product is selected whenever a SECOND function is encountered in a branch.

After the network of chemical reactions is constructed, its behavior in the time-domain must be ascertained. Our approach is to convert the network into an electrical circuit. A SPICE netlist is constructed to represent the circuit. We provide SPICE with subcircuit definitions to implement the chemical reaction equations. The SPICE netlist is wrapped inside an appropriate set of SPICE commands and the circuit is simulated using our modified version of the original 217,000-line SPICE3 simulator (Quarles, Newton, Pederson, and Sangiovanni-Vincentelli 1994). We have embedded SPICE as a submodule within our genetic programming system.

Each individual chemical reaction network is exposed to nine time-domain signals (table 2) representing the time-varying concentrations of four enzymes (EC2.7.1.30, EC3.1.3.21, EC3.1.1.23, and EC3.1.1.3) over 30 half-second time steps. There are 270 fitness cases (9 test cases, each consisting of 30 time steps). Each of these time series patterns has been structured so as to vary the concentrations between 0 and 2.0 in a pattern to which a living cell might conceivably be exposed. None are extreme. Each of the nine test cases is constructed by choosing four different time series from a set of six time series as the concentration for the four enzymes. For example, the slope-up time series starts at a concentration of 0.5 at time step 0 and increases linearly to a concentration of 1.75 at time step 30.

Fitness is the sum, over the 270 fitness cases, of the absolute value of the difference between the

concentration of the end product of the individual reaction network and the observed concentration of diacyl-glycerol (C00165). The smaller the fitness, the better. An individual that cannot be simulated by SPICE is assigned a high penalty value of fitness (10^8). The number of hits is defined as the number of fitness cases (0 to 270) for which the concentration of the measured substances is within 5% of the observed data value.

Table 2 Variations in levels of the four enzymes.

	EC2.7.1.30	EC3.1.3.21	EC3.1.1.23	EC3.1.1.3
1	Slope-Up	Sawtooth	Step-Down	Step-Up
2	Slope-Down	Step-Up	Sawtooth	Step-Down
3	Step-Down	Slope-Up	Slope-Down	Step-Up
4	Step-Up	Slope-Down	Step-Up	Step-Down
5	Sawtooth	Step-Down	Slope-Up	Step-Up
6	Sawtooth	Step-Down	Knock-Out	Slope-Up
7	Sawtooth	Knock-Out	Slope-Up	Step-Down
8	Knock-Out	Step-Down	Slope-Up	Sawtooth
9	Step-Down	Slope-Up	Sawtooth	Knock-Out

4.5 CONTROL PARAMETERS

The population size, M , is 100,000. A generous maximum size of 500 points (for functions and terminals) was established for each result-producing branch. The percentages of the genetic operations for each generation is 58.5% one-offspring crossover on internal points of the program tree other than perturbable numerical values, 6.5% one-offspring crossover on points of the program tree other than perturbable numerical values, 1% mutation on points of the program tree other than perturbable numerical values, 20% mutation on perturbable numerical values, 10% reproduction, 3% branch creation, and 2% subroutine deletion. The other parameters are default values that have been used on a broad range of problems (Koza, Bennett, Andre, and Keane 1999).

5 RESULTS

The population for the initial random generation (generation 0) of a run of genetic programming is created at random. The fitness of the best individual (figure 4) from generation 0 is 86.4. This individual scores 126 hits (out of 270). Substance C00162 (fatty acid) is used as an input substance to this metabolic pathway; however, glycerol (C00116) and ATP (C00002) are not. Two of the four available reactions (EC 3.1.1.23 and EC 3.1.1.3) are used. However; a third reaction (EC 3.1.3.21) consumes a non-existent intermediate substance (INT_2) and the fourth reaction (EC 2.7.1.30) is not used at all. This metabolic pathway contains one important topological feature, namely the bifurcation of C00162 to two different reactions. However, this

metabolic pathway does not contain any of the other important topological features of the correct metabolic pathway.

In generation 10, the fitness of the best individual (figure 5) is 64.0. This individual scores 151 hits. This metabolic pathway is superior to the best individual of generation 0 in that it uses both C00162 (fatty acid) and glycerol (C00116) as external inputs. However, this metabolic pathway does not use ATP (C00002). This metabolic pathway is also defective in that it contains only two of the four reactions.

In generation 25, the fitness of the best individual (figure 6) is 14.3. This individual scores 224 hits. This metabolic pathway contains all four of the available reactions. This metabolic pathway is more complex than previous best-of-generation individuals in that it contains two topological features not previously seen. First, this metabolic pathway contains an internal feedback loop in which one substance (glycerol C00116) is consumed by one reaction (catalyzed by enzyme EC 2.7.1.30), produced by another reaction (catalyzed by enzyme EC 3.1.3.21), and then supplied as a substrate to the first reaction. Second, this metabolic pathway contains a place where there is an addition of quantities of one substance. Specifically, glycerol (C00116) comes from the reaction catalyzed by enzyme EC 3.1.3.21 and is also externally supplied. This metabolic pathway also contains two substances (C00116 and C00162) which are each bifurcated to two different reactions.

In generation 120, the fitness of the best individual (figure 7) is 2.33. The cofactor ATP (C00002) appears as an input to this metabolic pathway. This pathway has the same topology as the correct network. However, the numerical values (sizing) is not yet correct and this individual scores only 255 hits.

The best-of-run individual (figure 1) appears in generation 225. Its fitness is almost zero (0.054). This individual scores 270 hits (out of 270). In addition to having the same topology as the correct metabolic pathway, the rate constants of three of the four reactions match the correct rates (to three significant digits) while the fourth rate differs by only about 2% from the correct rate (i.e., the rate of EC 3.1.3.21 is 1.17 compared with 1.19 for the correct network).

In the best-of-run network from generation 225, the rate of production of the network's final product, diacyl-glycerol (C00165), is given by [1].

Note that genetic programming has correctly determined that the reaction that produces the network's final product diacyl-glycerol (C00165) has two substrates and one product. It has correctly

identified enzyme EC3.1.1.3 as the catalyst for this final reaction.

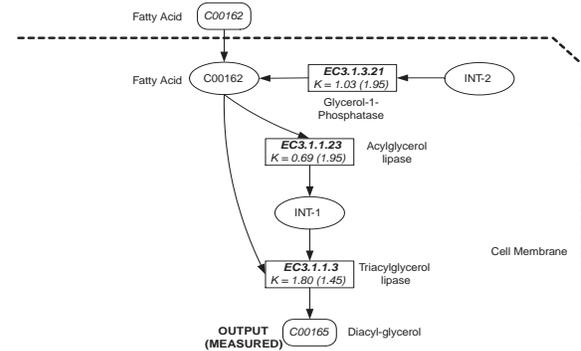


Figure 4 Best of generation 0.

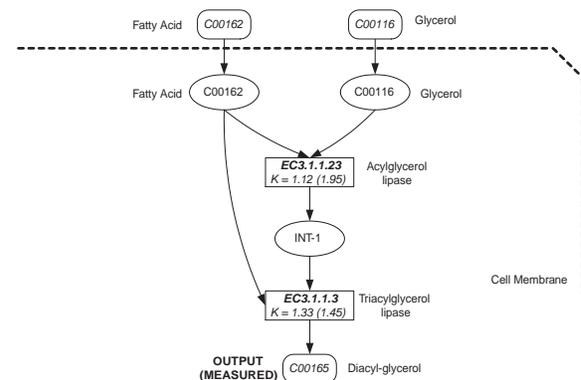


Figure 5 Best of generation 10.

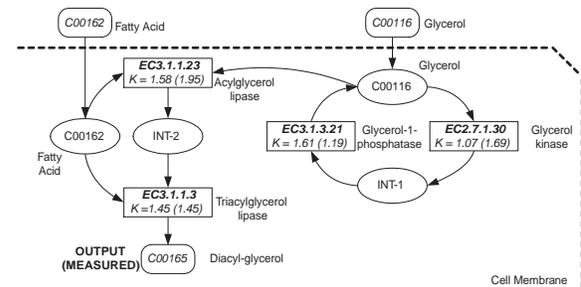


Figure 6 Best of generation 25.

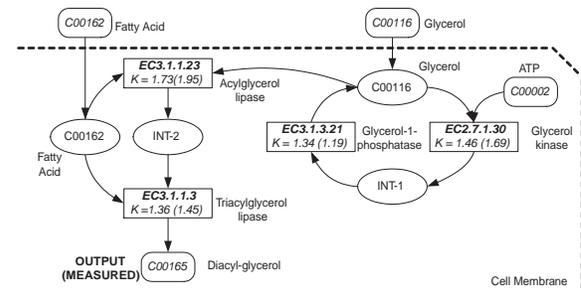


Figure 7 Best of generation 120.

It has correctly determined the rate of this final reaction as 1.45. It has correctly identified the externally supplied substance, fatty acid (C00162), as one of the two substrates for this final reaction. None

of this information was supplied *a priori* to genetic programming.

Of course, genetic programming has no way of knowing that biochemists call the intermediate substance (INT_2) by the name Monoacyl-glycerol (C01885) (as indicated in figure 1). It has, however, correctly determined that an intermediate substance is needed as one of the two substrates of the network's final reaction and that this intermediate substance should, in turn, be produced by a particular other reaction (described next).

In the best-of-run network from generation 225, the rate of production and consumption of the intermediate substance INT_2 is given by [2]. Again, genetic programming has correctly determined that the reaction that produces the intermediate substance (INT_2) has two substrates and one product; it has correctly identified enzyme EC3.1.1.23 as the catalyst for this reaction; it has correctly determined the rate of this reaction as 1.95; it has correctly identified two externally supplied substances, fatty acid (C00162) and glycerol (C00116), as the two substrates for this reaction.

In the best-of-run network from generation 225, the rate of production and consumption of the intermediate substance INT_1 in the internal feedback loop is given by [3]. Note that the numerical rate constant of 1.17 in the above equation is slightly different from the correct rate (as shown in Figure 1).

Here again, genetic programming has correctly determined that the reaction that produces the intermediate substance (INT_1) has two substrates and one product; it has correctly identified enzyme EC2.7.1.30 as the catalyst for this reaction; it has almost correctly determined the rate of this reaction to be 1.17 (whereas the correct rate is 1.19, as shown in figure 1); it has correctly identified two externally supplied substances, glycerol (C00116) and the cofactor ATP (C00002), as the two substrates for this reaction.

Genetic programming has no way of knowing that biochemists call the intermediate substance (INT_1) by the name sn-Glycerol-3-Phosphate (C00093) (as indicated in figure 1). Genetic programming has, however, correctly determined that an intermediate substance is needed as the single substrate of the reaction catalyzed by Glycerol-1-phosphatase (EC3.1.3.21) and that this intermediate substance should, in turn, be produced by the reaction catalyzed by Glycerol kinase (EC2.7.1.30).

In the best-of-run network from generation 225, the rate of supply and consumption of ATP (C00002) is given by [4].

The rate of supply and consumption of fatty acid (C00162) in the best-of-run network is given by [5].

The rate of supply, consumption, and production of glycerol (C00116) in the best-of-run network is given by [6]. Again, note that the numerical rate constant of 1.17 in the above equation is slightly different from the correct rate (as shown in Figure 1).

In summary, driven only by the time-domain concentration values of the final product C00165 (diacyl-glycerol), genetic programming created both the topology and sizing for an entire metabolic pathway whose time-domain behavior closely matches that of the naturally occurring pathway, including

- the total number of reactions in the network,
- the number of substrate(s) consumed by each reaction,
- the number of product(s) produced by each reaction,
- an indication of which enzyme (if any) acts as a catalyst for each reaction,
- the pathways supplying the substrate(s) (either from external sources or other reactions in the network) to each reaction,
- the pathways dispersing each reaction's product(s) (either to other reactions or external outputs),
- the number of intermediate substances in the network,
- emergent topological features such as
 - internal feedback loops,
 - bifurcation points,
 - accumulation points, and
- numerical rates (sizing) for all reactions.

Genetic programming did this using only the 270 time-domain concentration values of the final product C00165 (diacyl-glycerol).

This example demonstrates the principle that it is possible to reverse engineer a metabolic pathway using only observed data for the concentration values of the pathway's final product.

6 CONCLUSION

Genetic programming automatically created (from 270 data points) a metabolic pathway involving four chemical reactions that took in two substances and produced another substance as the final product.

7 FUTURE WORK

7.1 IMPROVED REPRESENTATION

Although the representation herein yielded the desired results, the authors believe that alternative representations for the program tree would significantly improve efficiency of the search. The authors are currently working on this.

7.2 DESIGNING ALTERNATIVE METABOLISMS

$$\frac{d[C00165]}{dt} = 1.45[C00162][INT_2][EC\ 3.1.1.3] \quad [1]$$

$$\frac{d[INT_2]}{dt} = 1.95[C00162][C00116][EC\ 3.1.1.23] - 1.45[C00162][INT_2][EC\ 3.1.1.3] \quad [2]$$

$$\frac{d[INT_1]}{dt} = 1.69[C00116][C00002][EC\ 2.7.1.30] - 1.17[INT_1][EC\ 3.1.3.21] \quad [3]$$

$$\frac{d[ATP]}{dt} = 1.5 - 1.69[C00116][C00002][EC\ 2.7.1.30] \quad [4]$$

$$\frac{d[C00162]}{dt} = 1.2 - 1.95[C00162][C00116][EC\ 3.1.1.23] - 1.45[C00162][INT_2][EC\ 3.1.1.3] \quad [5]$$

$$\frac{d[C00116]}{dt} = 0.5 + 1.17[INT_1][EC\ 3.1.3.21] - 1.69[C00116][C00002][EC\ 2.7.1.30] - 1.95[C00162][C00116][EC\ 3.1.1.23] \quad [6]$$

They observed that the naturally occurring pathway for the non-oxidative stage of the pentose phosphate is especially favorable in several respects to the alternatives that they generated. Specifically, the naturally occurring pathway has a comparatively small number of steps, does not use any reducing or oxidizing compounds, and requires only one ATP in one direction of flux.

Mendes and Kell (1998) have also suggested that novel pathways might be artificially constructed.

Conceivably, genetic programming could also be used to generate realizable and advantageous alternatives to naturally occurring pathways.

In one approach, the fitness measure might be oriented toward duplicating the final output(s) of the naturally occurring pathway (as was done in this paper). However, instead of harvesting only the individual from the population with the very best value of fitness, individuals that are slightly inferior could be examined to see if they simultaneously possess other desirable characteristics.

In a second approach, the fitness measure might be specifically oriented to factors such as the pathway's efficiency or use or non-use of certain specified reactants or enzymes.

In a third approach, the fitness measure might be specifically oriented toward achieving novelty. Genetic programming has previously been used as an invention machine by employing a two-part fitness measure that incorporates both the degree to which an individual satisfies performance requirements and the degree to which the individual does not possess the key

Mittenthal, Ao Yuan, and Scheeline (1998) presented a method for generating alternative biochemical pathways.

characteristics of previously known solutions (Koza, Bennett, Andre, and Keane 1999).

ACKNOWLEDGMENTS

Douglas B. Kell of the University of Wales and the GECCO-2001 reviewers made extremely useful comments.

REFERENCES

- Arkin, Adam, Shen, Peidong, and Ross, John. 1997. A test case of correlation metric construction of a reaction pathway from measurements. *Science*. 277: 1275 - 1279.
- Bower, James M. and Bolouri, Hamid. 2000. *Computational Modeling of Genetic and Biochemical Networks*. Cambridge, MA: MIT Press.
- Comisky, William, Yu, Jessen, and Koza, John. 2000. Automatic synthesis of a wire antenna using genetic programming. *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference, Las Vegas, Nevada*. Pages 179 - 186.
- D'haeseleer, Patrik, Wen, Xiling, Fuhrman, Stefanie, and Somogyi, Roland. 1999. Linear modeling of mRNA expression levels during CNS development and injury. In Altman, Russ B. Dunker, A. Keith, Hunter, Lawrence, Klein, Teri E., and Lauderdale, Kevin (editors). *Pacific Symposium on Biocomputing '99*. Singapore: World Scientific. Pages 41 - 52.
- Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1999. *Genetic Programming III: Darwinian Invention and Problem Solving*. San Francisco, CA: Morgan Kaufmann.

- Koza, John R., Bennett III, Forrest H, Andre, David, Keane, Martin A., and Brave Scott. 1999. *Genetic Programming III Videotape: Human-Competitive Machine Intelligence*. San Francisco, CA: Morgan Kaufmann.
- Koza, John R., Keane, Martin A., Yu, Jessen, Bennett, Forrest H III, and Mydlowec, William. 2000. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*. (1) 121 - 164.
- Koza, John R., Mydlowec, William, Lanza, Guido, Yu, Jessen, and Keane, Martin A. 2000. *Reverse Engineering and Automatic Synthesis of Metabolic Pathways from Observed Data Using Genetic Programming*. Stanford Medical Informatics Technical Report SMI-2000-0851.
- Laing, Shoudan, Fuhrman, Stefanie, and Somogyi, Roland. 1998. REVEAL: A general reverse engineering algorithm for inference of genetic network architecture. In Altman, Russ B. Dunker, A. Keith, Hunter, Lawrence, and Klein, Teri E. (editors). *Pacific Symposium on Biocomputing '98*. Singapore: World Scientific. Pages 18 - 29.
- Loomis, William F. and Sternberg, Paul W. 1995. Genetic networks. *Science*. 269. 649. August 4, 1995.
- McAdams, Harley H. and Shapiro, Lucy. 1995. Circuit simulation of genetic networks. *Science*. 269. 650-656. August 4, 1995.
- Mendes, Pedro and Kell, Douglas B. 1998. Non-linear optimization of biochemical pathways: Applications to metabolic engineering and parameter estimation. *Bioinformatics*. 14(10): 869 - 883.
- Mittenthal, Jay E., Ao Yuan, Bertrand Clarke, and Scheeline, Alexander. 1998. Designing metabolism: Alternative connectivities for the pentose phosphate pathway. *Bulletin of Mathematical Biology*. 60: 815 - 856.
- Ptashne, Mark. 1992. *A Genetic Switch: Phage λ and Higher Organisms*. Second Edition. Cambridge, MA: Cell Press and Blackwell Scientific Publications.
- Quarles, Thomas, Newton, A. R., Pederson, D. O., and Sangiovanni-Vincentelli, A. 1994. *SPICE 3 Version 3F5 User's Manual*. Department of Electrical Engineering and Computer Science, University of California. Berkeley. March 1994.
- Tomita, Masaru, Hashimoto, Kenta, Takahashi, Kouichi, Shimizu, Thomas Simon, Matsuzaki, Yuri, Miyoshi, Fumihiko, Saito, Kanako, Tanida, Sakura, Yugi, Katsuyuki, Venter, J. Craig, Hutchison, Clyde A. III. 1999. E-CELL: Software environment for whole cell simulation. *Bioinformatics*. Volume 15 (1) 72-84.
- Voit, Eberhard O. 2000. *Computational Analysis of Biochemical Systems*. Cambridge: Cambridge University Press.
- Yuh, Chiou-Hwa, Bolouri, Hamid, and Davidson, Eric H. 1998. Genomic cis-regulatory logic: Experimental and computational analysis of a sea urchin gene. *Science*. 279. 1896 - 1902.