REVERSE ENGINEERING BY MEANS OF GENETIC PROGRAMMING OF METABOLIC PATHWAYS FROM OBSERVED DATA

John R. Koza
Biomedical Informatics, Department of Medicine
Department of Electrical Engineering
Stanford University, Stanford, California, koza@stanford.edu
William Mydlowec
Genetic Programming Inc., Los Altos, California, myd@cs.stanford.edu
Guido Lanza
Genetic Programming Inc., Los Altos, California, guidissimo@hotmail.com
Jessen Yu
Genetic Programming Inc., Los Altos, California, jyu@cs.stanford.edu
Martin A. Keane
Econometrics Inc., Chicago, Illinois, makeane@ix.netcom.com

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. The concentrations of substances participating in networks of chemical reactions are modeled by non-linear continuous-time differential equations. This paper demonstrates that it is possible to automatically create (reverse engineer) a network of chemical reactions from observed time-domain data. Genetic programming starts with observed time-domain concentrations of substances and automatically creates both the topology of the network of chemical reactions and the rates of each reaction of a network such that the behavior of the automatically created network matches the observed time-domain data. Specifically, genetic programming automatically created a metabolic pathway involving four chemical reactions that consume glycerol and fatty acids as input, used ATP as a cofactor, and produced diacyl-glycerol as the final product. The metabolic pathway was created from 270 data points. The automatically created metabolic pathway contains three key topological features, including 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 topology and sizing of the entire metabolic pathway was automatically created using only the time-domain concentration values of diacyl-glycerol (the final product).

1. Introduction

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, Hashimoto, Takahashi, Shimizu, Matsuzaki, Miyoshi, Saito, Tanida, Yugi, Venter, and Hutchison 1999).

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; Yuh, Bolouri, and Davidson 1998).

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 from the data.

Genetic programming (Koza, Bennett, Andre, and Keane 1999) is a method for automatically creating a computer program whose behavior satisfies certain high-level requirements. Genetic programming starts with a primordial ooze of thousands of randomly created computer programs (program trees) and uses the Darwinian principle of natural selection, recombination (crossover), mutation, gene duplication, gene deletion, and certain mechanisms of developmental biology to breed a population of programs over a series of generations.

Recent work has demonstrated that genetic programming can automatically create complex networks that exhibit prespecified behavior in areas 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 the automatically created 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, Mydlowec, and Stiffelman 1999; 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.

Since the description herein of our methods and results is necessarily severely limited by space, the authors are simultaneously publishing a considerably longer technical report that provides additional details and explanatory figures (Koza, Mydlowec, Lanza, Yu, and Keane 2000).

Section 2 states an illustrative "proof of principle" problem. Section 3 presents a method of representing networks of chemical reactions with program trees. Section 4 presents the preparatory steps for applying genetic programming to the illustrative problem. Section 5 presents the results.

2. Statement of the Illustrative Problem

The goal is to automatically create (reverse engineer) *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 that incorporates three key 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, Hashimoto, Takahashi, Shimizu, Matsuzaki, Miyoshi, Saito, Tanida, Yugi, Venter, and Hutchison 1999). The network's external inputs are glycerol and fatty acids. The network's final product is diacyl-glycerol. The network's four reactions are catalyzed by Glycerol kinase (EC2.7.1.30), Glycerol-1-phosphatase (EC3.1.3.21), Acylglycerol lipase (EC3.1.1.23), and Triacylglycerol lipase (EC3.1.1.3).

3. Representation of Chemical Reaction Networks

This section describes a method for representing a network of chemical reactions as a program tree suitable for use in a run of genetic programming. Each program tree represents an interconnected network of chemical reactions involving various substances. A chemical reaction may consume one or two substances and produce one or two substances. The consumed substances may be external input substances or intermediate substances produced by reactions. The chemical reactions, enzymes, and substances of a network may be completely represented by a program tree that contains

- internal nodes representing chemical reaction functions,
- internal nodes representing selector functions that select the reaction's first versus the reaction's second (if any) product,
- external points (leaves) representing substances that are consumed and produced by a reaction,
- external points (leaves) representing enzyme that catalyzes a reaction, and.
- external points (leaves) representing numerical constants (reaction rates).

Each program tree in the population is a composition of functions from the problem's function set and terminals from the problem's terminal set.

3.1. Repertoire of Functions

There are four chemical reaction functions and two selector functions.

The first argument of each chemical reaction function identifies the enzyme that catalyzes the reaction. The second argument 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 runs in this paper use a first-order and second-order rate law.

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 input substances (input substances, intermediate substances created by reactions, and output substances). Other terminals represent the enzymes that catalyze the chemical reactions. Still other terminals represent numerical constants for the velocity 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 velocity 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 a substance terminal. 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 illustrative figure is also 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).

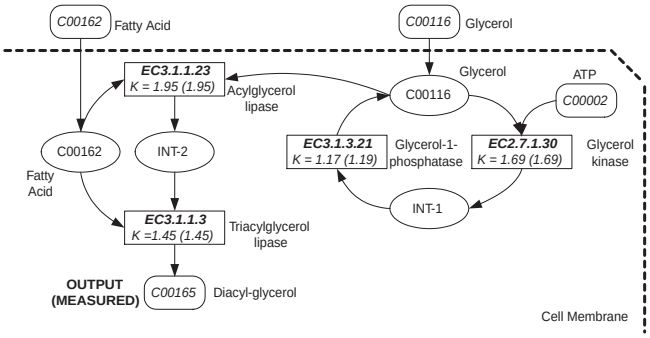

Figure 1 Best of run individual from generation 225.

Figure 2 shows the electrical circuit corresponding to the network of figure 1. The triangles in the figure represent integrators.

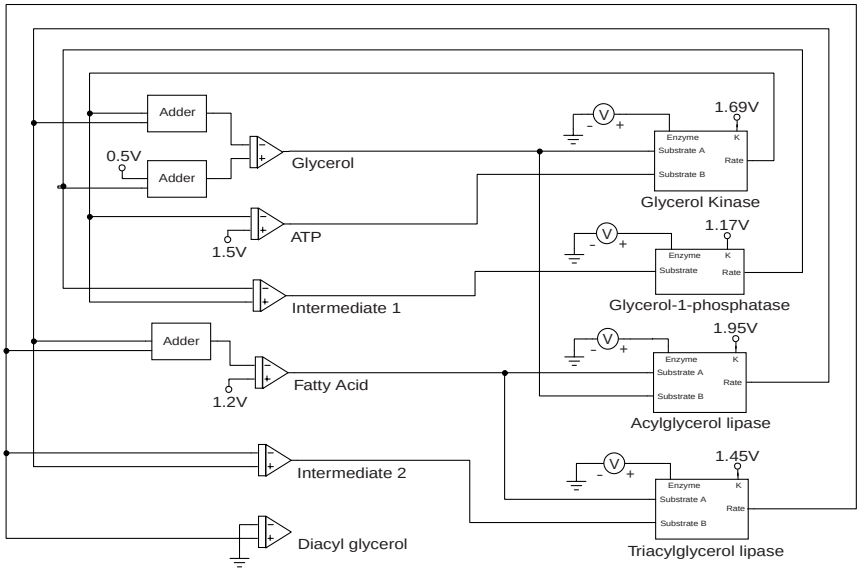

Figure 2 Electrical circuit corresponding to the chemical reaction network of figure 1.

For additional details, see Koza, Mydlowec, Lanza, Yu, and Keane 2000.

4. Preparatory Steps

Six major preparatory steps are required before applying genetic programming: (1) determine the architecture of the program trees, (2) identify the functions, (3) identify the terminals, (4) define the fitness measure, (5) choose control parameters for the run, and (6) choose the termination criterion and method of result designation. For additional details, see Koza, Mydlowec, Lanza, Yu, and Keane 2000.

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 = {CR1_1, CR1_2, CR2_1, CR2_2, FIRST, SECOND}.

4.3. Terminal Set

The terminal set, T, is

T = {ℜ, C00116, C00162, C00002, C00165, INT_1, INT_2, INT_3}.

ℜ denotes a perturbable numerical value. In the initial random generation (generation 0) of a run, each perturbable numerical value is set, individually and separately, to a random value in a chosen range (from 0.0 and 2.0 here).

In the illustrative problem herein, C00116 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).

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 velocity. 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 first product of a reaction is selected whenever a FIRST function is encountered in a branch. The second product of a reaction is selected whenever a SECOND function is encountered in a branch.

After the network is constructed, the pathway is converted into an electrical circuit. A SPICE netlist is then constructed to represent the electrical circuit. We provide SPICE with subcircuit definitions to implement all the Chemical reaction equations. This SPICE netlist is wrapped inside an appropriate set of SPICE commands to carry out analysis in the time domain (described below). The electrical circuit is then simulated using our modified version of the original 217,000-line SPICE3 simulator (Quarles, Newton, Pederson, and Sangiovanni-Vincentelli 1994). We have embedded our modified version of 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.

Table 2 Variations in the levels of the four enzymes.

Signal	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

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 data). 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.

4.5. Control Parameters for the Run

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 value, 6.5% one-offspring crossover on points of the program tree other than perturbable numerical value, 1% mutation on points of the program tree other than perturbable numerical value, 20% mutation on perturbable numerical value,

10% reproduction, 3% branch creation, and 2% subroutine deletion. The other parameters are the default values that we apply to a broad range of problems (Koza, Bennett, Andre, and Keane 1999).

4.6. Termination

The run was manually monitored and manually terminated when the fitness of many successive best-of-generation individuals appeared to have reached a plateau.

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 3) 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 4) 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 5) 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 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`) where a substance is bifurcated to two different reactions.

In generation 120, the fitness of the best individual (figure 6) 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 velocity constants of three of the four reactions match the correct velocities (to three significant digits) while the fourth velocity differs by only about 2% from the correct velocity (i.e., the velocity of EC 3.1.3.21 is 1.17 compared with 1.19 for the correct network).

In the best-of-run network, the rate of the two-substrate, one-product reaction catalyzed by Triacylglycerol lipase (EC 3.1.1.3) (found at the very bottom of figures 1 and 3) that produces the final product diacyl-glycerol (`C00165`) is

$$\frac{d[C00162]}{dt} = 1.45[C00162][INT\_2][\text{EC } 3.1.1.3] .$$

The rate of the two-substrate, one-product reaction catalyzed by Acylglycerol lipase (EC3.1.1.23) that produces intermediate substance `INT_2` is

$$\frac{d[INT\_2]}{dt} = 1.95[C00162][C00116][\text{EC } 3.1.1.23] - 1.45[C00162][INT\_2][\text{EC } 3.1.1.3] .$$

The rate of the two-substrate, one-product reaction catalyzed by Glycerol kinase (EC2.7.1.30) that produces intermediate substance `INT_1` in the internal loop is

$$\frac{d[INT\_1]}{dt} = 1.69[C00116][C00002][\text{EC } 2.7.1.30] - 1.17[INT\_1][\text{EC } 3.1.3.21] .$$

The rate of supply and consumption of cofactor ATP (`C00002`) is

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

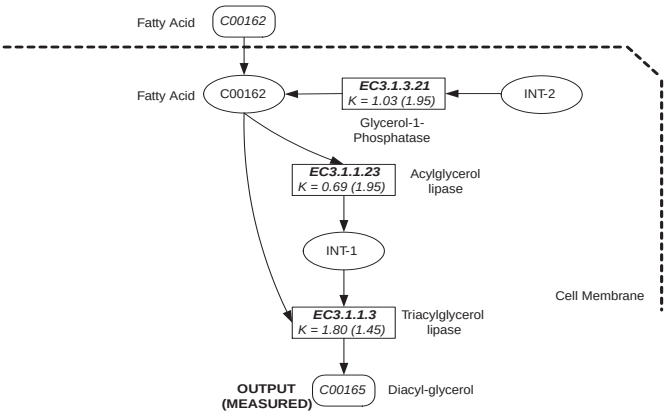

Figure 3 Best of generation 0.

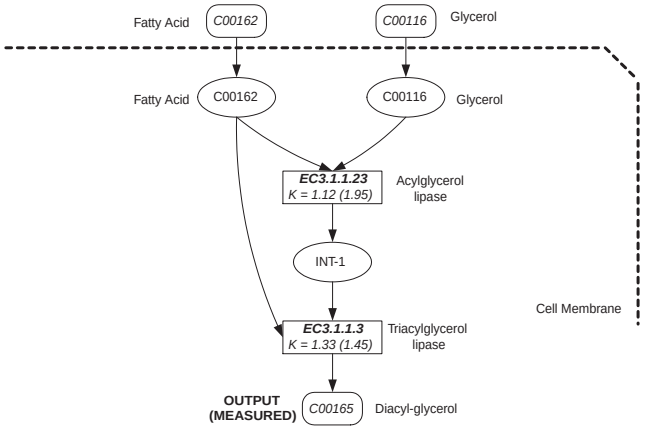

Figure 4 Best of generation 10.

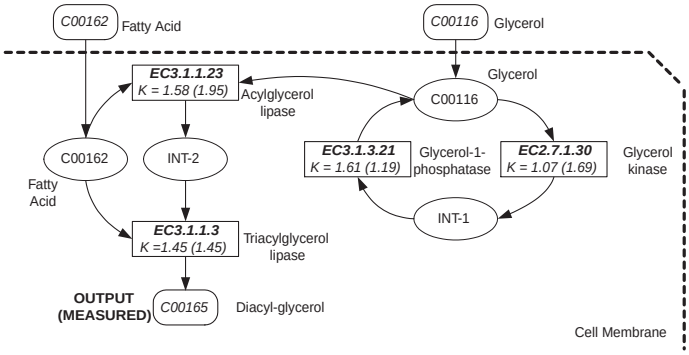

Figure 5 Best of generation 25.

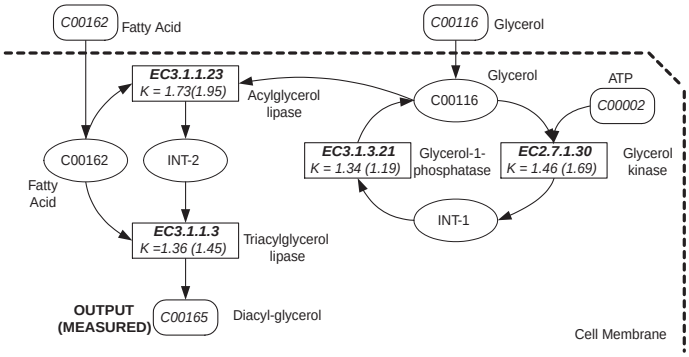

Figure 6 Best of generation 120.

The rate of supply and consumption of fatty acids (C00162) is

$$\frac{d[C00162]}{dt} = 1.2 - 1.95[C00162][C00116][\text{EC } 3.1.1.23] - 1.45[C00162][INT\_2][\text{EC } 3.1.1.3].$$

The rate of supply, consumption, and production of glycerol C00116 (the production being from the one-substrate, one-product reaction catalyzed by Glycerol-1-phosphatase (EC3.1.3.21) in the internal loop) is $1.95[C00162][C00116][\text{EC } 3.1.1.23]$

$$\frac{d[C00116]}{dt} = 0.5 + 1.17[INT\_1][\text{EC } 3.1.3.21] - 1.69[C00116][C00002][\text{EC } 2.7.1.30] - 1.95[C00162][C00116][\text{EC } 3.1.1.23].$$

Notice that genetic programming created the entire metabolic pathway, including topological features such as the internal feedback loop, the bifurcation point, and the accumulation point, and including all numerical parameter values (sizing) in the pathway, using only the time-domain concentration values of C00165 (i.e., diacyl-glycerol, the final product).

For additional details, see Koza, Mydlowec, Lanza, Yu, and Keane 2000.

6. Conclusion

Genetic programming automatically created a metabolic pathway involving four chemical reactions that took in glycerol and fatty acids as input, used ATP as a cofactor, and produced diacyl-glycerol as its final product. The metabolic pathway was created from 270 data points. The automatically created metabolic pathway contains three key topological features, including 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.

References

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.

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., 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., Keane, Martin A., Yu, Jessen, Bennett, Forrest H III, Mydlowec, William, and Stiffelman, Oscar. 1999. Automatic synthesis of both the topology and parameters for a robust controller for a non-minimal phase plant and a three-lag plant by means of genetic programming. *Proceedings of 1999 IEEE Conference on Decision and Control*. Pages 5292 - 5300.

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.

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.

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, CA. 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.