

Modelling Coupled Evolution of Gene Regulatory and Metabolic Networks



Abstract

Cells we see today are the consequence of a selection process that has taken place over some billions of years for optimal survival and proliferation in the appropriate environments, which will have changed drastically during this time period. At shorter timescales, a cell is able to adapt to fluctuations in the environment by means of regulating gene expression. By modelling gene regulatory interactions in a graph-theoretic framework and using a constraints-based approach to measuring cell growth or by using dynamic graphical models and suitable learning algorithms, several studies have investigated the short-term behaviour of cell evolution. However, the modelling the long-term evolution of genetic information and the resulting metabolic behaviour (or metabolic network architecture) of a cell has received less attention. Here we propose a model of coupled regulatory-metabolic network evolution and discuss the theoretical and computational aspects of the model. By representing gene regulatory relationships as Boolean functions and applying constraints-based methods of assessing the robustness of an evolutionary event in the genome, the model attempts to predict the long-term metabolic behaviour of a cell.

Acknowledgements

I acknowledge Professor Jotun Hein and Dr. Gail Preston for presenting the idea of coupled evolution of regulatory-metabolic networks. I thank Dr. Adam Novak, Dr. Rune Lyngsø and Dr. Bhalchandra Thatte for their company in the Oxford Centre for Gene Function. I would like to thank Dr. Steven Kelly and Dr. Craig Maclean for some fun discussions. I am indebted to Dr. Aziz Mithani for his guidance throughout the project as well as his cheerful company. I am also very grateful to Professor David Fell for his invaluable guidance and feedback on flux balance analysis, which without, would have made the report less readable than what it currently is. Last but not least, I would like to thank my parents and my brother for their support of my endeavours in science. This work was presented as a short talk at the Lipari School on Bioinformatics and Computational Biology 2010 as well as at the Second RoSBNet Synthetic Biology Workshop 2010 as a poster.

Contents

1	Introduction	3
2	Gene Regulatory Networks	4
2.1	Modelling Gene Regulatory Networks	4
2.1.1	Boolean Network Structure	4
2.1.2	Boolean Network Dynamics	4
2.1.3	Probabilistic Boolean Network Structure	5
2.1.4	Probabilistic Boolean Network Dynamics	6
2.2	Evolution of Gene Regulatory Networks	8
2.2.1	Growth of Gene Regulatory Networks	8
2.2.2	Shrinkage of Gene Regulatory Networks	8
2.2.3	Example	10
3	Metabolic Networks	10
3.1	Modelling Metabolic Networks	11
3.1.1	Linear Programming	11
3.1.2	Flux Balance Analysis	12
3.1.3	Example	12
3.2	Evolution of Metabolic Networks	14
3.2.1	Edge Insertion/Deletion Model	14
3.2.2	Edge Insertion/Deletion and Flux Balance Analysis	15
4	Connecting Regulatory and Metabolic Networks	15
4.1	Explicit and Implicit Connections	17
4.2	Metabolic Network Evolution as a Consequence of Regulatory Evolution	17
4.3	Algorithm for Coupled Evolution	17
5	Simulations	18
5.1	Toy Regulatory-Metabolic Network	18
6	MATLAB Code	20
6.1	Stoichiometric Matrix for Toy Network of Section 3.1.3	20
6.2	Main Function for Algorithm 1	20
7	Evaluation and Remaining Work	23
	References	28

1 Introduction

The genetic material we see in cells today has come about as the result of a combination of random variation events to genetic information as it is passed from a cell to its descendants and a selection procedure, which favours phenotypes that survive and proliferate. These two processes have been taking place for some billions of years, and continues to do so this very moment. Even within a single generation, cells can adapt to changes in the environment for optimal growth by use of complex feedback loops between internal gene expression levels and extracellular conditions such as available metabolite concentrations or pH. For example, the transcription of the *lac* operon in *E. coli*, a set of contiguous genes that code for proteins that transport lactose into the cell and break it down, is controlled by extracellular glucose and lactose levels in a dual positive-negative control mechanism. The operon is only expressed if lactose levels are high *and* glucose levels are low. Simply reducing the level of glucose, or increasing the level of lactose will not result in the expression of the *lac* operon [1]. It is therefore evident that in order to effectively model the evolution of genetic information, both regulation of gene expression and metabolic processing of a cell must be considered and done so at both the within-generation and between-generation timescales.

In this study, we present the underlying theoretical and computational aspects of a mathematical model of the coupled evolution of regulatory and metabolic systems. In the model, we represent the regulatory interactions and metabolic reactions of a single generation as a *coupled network* and thereafter investigate the possible changes that can occur to the network topology and connectivity during evolution. Networks are inherently discrete, making it relatively easier to model evolutionary processes that are largely stochastic. Computations with probability mass functions are, in general, often easier to manipulate and analyse than those with density functions. Embedding biological processes into a graph-theoretic framework also provides summary statistics that may elucidate properties unseen if the systems in question were laid out linearly. By using the degree distribution as the summary statistic of interest, Jeong et al. showed that networks in which metabolites are arranged as nodes and chemical reactions as edges for various organisms similarly have a scale-free network topology [8]. S. S. Shen-Orr et al. found small subnetworks or ‘network motifs’ that occur at a higher frequency in the regulatory network of *E. coli* than those found in randomised networks with similar constitutional characteristics [19]. The motivation of both studies were to explore possible design principles of metabolic and regulatory circuits. The objective of the current study is not too dissimilar. Although it is well-understood that genomes undergo evolutionary changes, the exact role of metabolic constraints in assessing evolutionary fitness are yet unclear. The development of an evolutionary model and thereafter comparisons with experimental data could shed light on the *evolutionary* design principles of the genetic information observed today, and subsequently aid our understanding of future events to come.

The first three sections of the report describe the theoretical framework used for the description and evolution of regulatory and metabolic networks. Section 2 outlines the use of probabilistic Boolean logic to characterise functional and evolutionary properties of regulatory networks. In section 3, methods for assessing the fitness before and after an evolutionary event are discussed. Section 4 deals with modelling the connections between gene regulation and metabolism, and we show the pseudocode for the full model that has been so far partially implemented. Thereafter, in section 5, a short description of the toy network to be used to test the efficiency and parameter-sensitivity of the model is provided. MATLAB code so far implemented is given in section 6 and in section 7 we discuss evaluation of the current model and the

future work to be done.

2 Gene Regulatory Networks

Briefly, an expressed gene first undergoes transcription and thereafter translation to produce a protein. The rate of transcription is controlled by a region preceding the gene known as the *promoter*. Proteins, called *transcription factors* (TF's), bind to specific binding sites of promoters to increase or decrease the transcription rate of the gene. Since TF's are proteins, they too will have a gene that encodes the necessary information for its production [2]. It is already clear that depicting the genome as a long string of nucleotides is not optimal in describing regulatory interactions. As aforementioned in section 1, a more effective method of describing experimentally verified regulatory relationships between different locations in the genome is by means of a network, commonly termed *transcriptional* or *gene regulatory* networks (GRN).

In the network representation of gene regulation, transcription factors and genes are depicted as nodes and regulatory interactions as edges. Several mathematical frameworks have been suggested to model the action of TF's on target genes (TG's), including the use of linear ordinary differential equations (please see [11] and references therein) or the Hill function (please see [2] and references therein) to describe protein concentrations as functions of time and TF concentrations. For the purpose of this study, the action of TF's and the activity of the TG's are modelled as Boolean functions and nodes respectively. Since regulatory relationships between TF's and their TG's is expected to change with evolutionary events, probabilistic Boolean logic are used in a probabilistic Boolean network (PBN) framework [21].

2.1 Modelling Gene Regulatory Networks

2.1.1 Boolean Network Structure

A Boolean network (BN) with n nodes $G_n(V, F)$ is defined by the set of node states $V = \{x_1, \dots, x_n\}$ where $x_i \in \{0, 1\}$, $i = 1, \dots, n$ and a set of Boolean functions $F = \{f_1, \dots, f_n\}$. Each Boolean function $f_i(x_1^i, \dots, x_k^i)$, where k is the total number of inputs and the superscript i indicates the input nodes for function f_i , is assigned at each node. The total number of inputs k can be either a function of the node index i , $k = k(i)$, or constant at $k = n$ such that each function considers the state of *all* nodes as its input. When $k = n$, nodes that do not make a difference to the outcome of the function are labelled *fictitious* and those that do *essential*. Therefore if the i^{th} node is fictitious to function f_j ,

$$f_j(x_1, \dots, x_{i-1}, x_i = 0, x_{i+1}, \dots, x_n) = f_j(x_1, \dots, x_{i-1}, x_i = 1, x_{i+1}, \dots, x_n)$$

for all $\mathbf{x} \setminus x_i$, $\mathbf{x} \in \{0, 1\}^n$, $\mathbf{x} = (x_1, \dots, x_n)$. In the above expression, we do not assign superscript j for each input node since k is fixed as constant n and hence all nodes are input nodes. Biologically, each node represents the expression of gene i , with $x_i = 1$ and $x_i = 0$ indicating high or low (zero) expression levels. Each function in the set F describes the regulatory interaction between genes [21].

2.1.2 Boolean Network Dynamics

Updating a BN between two time points t and $t + 1$ essentially involves inputting the state of the network at time t , V^t , into the set of Boolean functions F^t to determine the state of all

genes at the next time point, V^{t+1} . Therefore, $x(t+1)_i = f_i(x(t)_1^i, \dots, x(t)_k^i)$. Naturally, for a BN the set of possible functions F is fixed and $F^t = F^{t+1}$. Given an initial joint probability of each node state $D^0(\mathbf{x}(0))$, with $\mathbf{x}(0) \in \{0, 1\}^n$ since the initial state is fixed, we have for the joint probability distribution after one time step,

$$P\{f_1(\mathbf{x}(0)) = \mathbf{x}(1)_1, \dots, f_n(\mathbf{x}(0)) = \mathbf{x}(1)_n\} = \sum_{\substack{\mathbf{x}(0) \in \{0,1\}^n: \\ f_k(\mathbf{x}(0)) = \mathbf{x}(1)_k, \\ k=1, \dots, n}} D^0(\mathbf{x}(0)) \quad (2.1)$$

that is, we are summing over all joint probabilities at $t = 0$ that once input to node functions f_i , produces the desired outcome at $t = 1$. Equation (2.1) can then be used iteratively to find any joint distribution at any subsequent time because

$$P\{f_k(\mathbf{x}(t)) = \mathbf{x}(t+1)_k\} = D^{t+1}(\mathbf{x}(t+1)), \quad \mathbf{x}(t) \in \{0, 1\}^n, \quad t > 0$$

Thus, by induction,

$$D^{t+1} = \mathcal{F}(D^t) \quad (2.2)$$

with the function $\mathcal{F} : [0, 1]^{2^n} \mapsto [0, 1]^{2^n}$ defined as in (2.1). The mapping is from $[0, 1]^{2^n}$ to $[0, 1]^{2^n}$ because D^t and D^{t+1} account for probabilities in the range $[0, 1]$ of all 2^n possible network states. Now, define matrix $A \in \{0, 1\}^{2^n \times 2^n}$ as

$$A_{ij} = \begin{cases} 1, & \text{if } \exists \mathbf{x}(t) \in \{0, 1\}^n \text{ s.t. } \begin{cases} C(f_1(\mathbf{x}(t)), \dots, f_n(\mathbf{x}(t))) = j, \\ C(\mathbf{x}(t)) = i \end{cases} \\ 0, & \text{otherwise} \end{cases}$$

where $C(\mathbf{x}) = 1 + \sum_{j=1}^n 2^{n-j} \cdot x_j$, for $\mathbf{x} \in \{0, 1\}^n$. A_{ij} is an input (i) to output (j) matrix, where if the binary state corresponding to i results in the binary state corresponding to j after one time step then $A_{ij} = 1$ and 0 otherwise. Since one input state *must* result in an output state, matrix A will have one non-zero entry per row (input). Hence (2.2) can be rewritten as

$$\begin{aligned} D^{t+1} &= D^t \cdot A \\ &= D^0 \cdot A^{t+1} \end{aligned} \quad (2.3)$$

Equation (2.3) shows that the dynamics of the BN is Markovian with order one [21].

2.1.3 Probabilistic Boolean Network Structure

The main difference between BN's and PBN's is that in PBN's there is a *set* of functions possible for each node x_i . In other words at each node we have $F_i = \{f_j^{(i)}\}_{j=1, \dots, l(i)}$ where $f_j^{(i)}$ is the j^{th} possible function for node i , from a total of $l(i)$ functions. To account for the case where multiple function assignments at each node produce the same network state, a PBN at a given instant is described with N function vectors $\mathbf{f}_k = (f_{m_1}^{(1)}, \dots, f_{m_n}^{(n)})$, $k = 1, \dots, N$ where N is the number of network realisations, $1 \leq m_i \leq l(i)$ and $f_{m_i}^{(i)} \in F_i$, $i = 1, \dots, n$. Plainly put, each of the N function vectors assign a plausible function to the n nodes in the network, and which out of the $l(i)$ different functions for node i have been used in a function vector is given by \mathbf{f}_k . The function vector \mathbf{f}_k thus satisfies the mapping $\mathbf{f}_k(\mathbf{x}(t)) = \mathbf{x}(t+1)$. Note that when $k = 1$, this mapping is reduced to the case of the BN, with only one function assigned to each node.

If we let $\mathbf{f} = (f^{(1)} \in F_1, \dots, f^{(n)} \in F_n)$ be a random sample from the space $F_1 \times \dots \times F_n$, then the probability that the j^{th} function is assigned at gene i , denoted $c_j^{(i)}$, is found by

$$c_j^{(i)} = P\{f^{(i)} = f_j^{(i)}\} = \sum_{k: f_{m_i}^{(i)} = f_j^{(i)}} P\{\mathbf{f} = \mathbf{f}_k\} \quad (2.4)$$

that is, by summing the probabilities of obtaining network realisations where specifically the desired j^{th} function is assigned at node i . Since there are $l(i)$ possible functions that can be assigned to node i , it naturally follows that

$$\sum_{j=1}^{l(i)} c_j^{(i)} = 1 \quad (2.5)$$

The above idea can be extended to finding the probability of a specific sequence of functions being assigned to each of the n nodes by assuming independence of assignment events. For example, with the independence assumption, the joint probability of the j^{th} function being assigned to node a and the k^{th} function to node b is simply

$$P\{f^{(a)} = f_j^{(a)}, f^{(b)} = f_k^{(b)}\} = P\{f^{(a)} = f_j^{(a)}\} \cdot P\{f^{(b)} = f_k^{(b)}\} \quad (2.6)$$

Collectively, a PBN with n nodes $\tilde{G}_n(V, F)$ is defined by the set of nodes $V = \{x_1, \dots, x_n\}$ as with BN's but now the set of *possible* functions is given by $F = \{F_1, \dots, F_n\}$, with F_i as defined previously. The total number of network function assignments $N = \prod_{i=1}^n l(i)$ assuming independence of function assignment [21].

2.1.4 Probabilistic Boolean Network Dynamics

To describe the dynamics of PBN's, first define the matrix K of all possible function assignments at each node as in [21]

$$K = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & l(n) \\ \hline 1 & 1 & \cdots & 2 & 1 \\ 1 & 1 & \cdots & 2 & 2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 2 & l(n) \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ l(1) & l(2) & \cdots & l(n-1) & l(n) \end{bmatrix} \quad (2.7)$$

where each row corresponds to one of N network function configurations and the columns to the nodes of the network. Entry K_{ij} simply holds the index of the function assigned to node j in network function configuration i such that in this configuration, the function assigned to node j is $f_{K_{ij}}^{(j)}$. Therefore the first $l(n)$ rows of K contains function configurations $f^{(1, \dots, n-1)} = 1$, $f^{(n)} = K_{in}$, $i = 1, \dots, l(n)$, the next $l(n)$ rows $f^{(1, \dots, n-2)} = 1$, $f^{(n-1)} = 2$ and $f^{(n)} = K_{in}$,

$i = l(n) + 1, \dots, 2l(n)$. This pattern continues for all N possible network function configurations. Consequently, $1 \leq K_{ij} \leq l(j)$ and K is of size $N \times n$. With the assumption that functions are assigned to each node independently, the probability that network configuration i is selected is given by

$$P_i = P\{\text{network function configuration } i \text{ is selected}\} = \prod_{j=1}^n c_{K_{ij}}^{(j)} \quad (2.8)$$

Note that

$$\sum_{i=1}^N P_i \stackrel{(2.8)}{=} \sum_{i=1}^N \prod_{j=1}^n c_{K_{ij}}^{(j)} = \prod_{j=1}^n \sum_{i=1}^{l(j)} c_i^{(j)} \stackrel{(2.5)}{=} \prod_{j=1}^n 1 = 1 \quad (2.9)$$

with the second equality arising due to the fact that node j can only be assigned one of $l(j)$ functions.

Now, with P_i defined as above, we can find the transition probability from $\mathbf{x}(t)$ to $\mathbf{x}(t+1)$ with a PBN as

$$\begin{aligned} P\{\mathbf{x}(t) \rightarrow \mathbf{x}(t+1)\} &= \sum_{i: f_{K_{i1}}^{(1)}(\mathbf{x}(t))=x(t+1)_1, \dots, f_{K_{in}}^{(n)}(\mathbf{x}(t))=x(t+1)_n} P_i \\ &= \sum_{i=1}^N P_i \cdot \left[\prod_{j=1}^n (1 - |f_{K_{ij}}^{(j)}(\mathbf{x}(t)) - x(t+1)_j|) \right] \end{aligned} \quad (2.10)$$

The first summation in (2.10) sums network function configuration probabilities over all configurations that return the desired output state $\mathbf{x}(t+1)$. Since the criteria that the configuration must satisfy is that *all* its n nodes are at the desired state, this summation can be expressed as a sum over all configurations, but with probabilities P_i multiplied by a binary function that is 1 if all node states at $t+1$ are the ones desired and 0 otherwise. Simply using the fact that $P(A) = \sum_b P(A | B = b) \cdot P(B = b)$, we can understand (2.10) as

$$P\{\mathbf{x}(t) \rightarrow \mathbf{x}(t+1)\} = \sum_{i=1}^N P\{\mathbf{x}(t) \rightarrow \mathbf{x}(t+1) \mid \text{network func. config. } i \text{ is selected}\} \cdot P_i$$

Note that just as in the case of BN's, for some input $\mathbf{x}(t)$, there *must* be an output $\mathbf{x}(t+1)$ and therefore there will be at least one network function configuration i for which $\prod_{j=1}^n (1 - |f_{K_{ij}}^{(j)}(\mathbf{x}(t)) - x(t+1)_j|) = 1$ in (2.10). We then know that $P\{\mathbf{x}(t) \rightarrow \mathbf{x}(t+1)\} = 1$ from (2.9). This ensures that the input-output matrix $A_{ij} \in [0, 1]$ and $A \in [0, 1]^{2^n \times 2^n}$, now not necessarily binary since one input can result in several outputs, satisfies

$$\sum_{j=1}^{2^n} A_{ij} = 1$$

for any input $i = 1, \dots, 2^n$. Thus, the dynamics of PBN's, like BN's, is also Markovian with order one. Moreover, it is clear that A for PBN's will have at most $N \cdot 2^n$ non-zero entries, when there is a one-to-one correspondence between the input and output states (2^n non-zero entries) for each configuration (N) [21].

2.2 Evolution of Gene Regulatory Networks

2.2.1 Growth of Gene Regulatory Networks

In this study we focus on gene duplication and thereafter divergence as the main method of GRN growth [5, 22]. The effects of duplication and divergence for a transcription factor-target gene (TF-TG) pair is shown in Figure 1.

- Case a): a TF is duplicated. Initial divergence will result in the two homologous TF's competitively regulating the same gene, that is, inheritance of the regulatory interaction to the new TF (case a)A). Further divergence can lead to the each TF regulating a separate gene where a new regulatory interaction occurs (case a)B).
- Case b): a TG is duplicated. Here divergence will first result in a single TF regulating two homologous TG's, again with inheritance of the regulatory interaction (case b)A). Further divergence leads to each gene being regulated by separate TF's (case b)B). As in the previous case, this further divergence brings a new regulatory interaction.
- Case c): both TF and TG are duplicated. Initial divergence results in inheritance of regulatory interaction where homologous TF's regulate homologous TG's (case c)A). Subsequent divergence can lead to another TF regulating the duplicated gene (case c)B, upper right) or the duplicated TF regulating another gene (case c)B, lower right). In both situations, a new regulatory interaction is gained. Since two TF's competing for a single gene and a single TF regulating two TG's can be unstable, the gained interaction may change again (case c)B left).

For a GRN of n nodes before duplication, the number of nodes (and therefore nodal functions) in cases a), b) and c) corresponds to $n + 1$, $n + 1$ and $n + 2$ respectively. The overall effect of divergence is to change the number of essential nodes (inputs) to the nodal functions of the network after growth. In this study, we assume the rate at which the network grows, denoted as λ is constant and assign equal probability to all nodes for duplication. Therefore, in the PBN (and equally the BN) setting when an evolutionary event is to occur, $P\{(\tilde{G}_n(V, F)(t) \rightarrow \tilde{G}_{n+1}(V, F)(t+1))\} = P\{\tilde{G}_n(V, F)(t) \rightarrow \tilde{G}_{n+2}(V, F)(t+1)\} = 0.5$. This corresponds to assuming that mutations that occur in the genome are random.

2.2.2 Shrinkage of Gene Regulatory Networks

As well as GRN growth, we also consider shrinkage, that is, the removal of nodes in the network. Genes that contribute little to the fitness of the cell are likely to be inactivated or undergo deletational mutations [16] (the 'Use It or Lose it' rule [14]). An extreme example is the loss of genes in the genomes of pathogenic prokaryotes that have evolved toward endosymbiosis [15, 14]. Here metabolic pathways producing metabolites that can be obtained from the host are eliminated, the pathogen relying on the metabolic capabilities of its host instead. However, the dynamics of evolution is not so straightforward. When the genetic population is drastically reduced, due to limited resources such as after infecting a new host, genes that are beneficial may also be lost (the 'Use It, but Lose It Anyway' rule) [16, 14]. Just as the number of nodes in the network increased with GRN growth, shrinkage is modelled by the loss of nodes. In the PBN (and equally the BN) framework, modelling the 'Use It or Lose It' rule of gene loss corresponds to removal of nodes that have been in the '0' state for a number of generations to

be specified. The ‘Use It, but Lose It Anyway’ rule can be modelled by the random loss of a node, independent of its previous states.

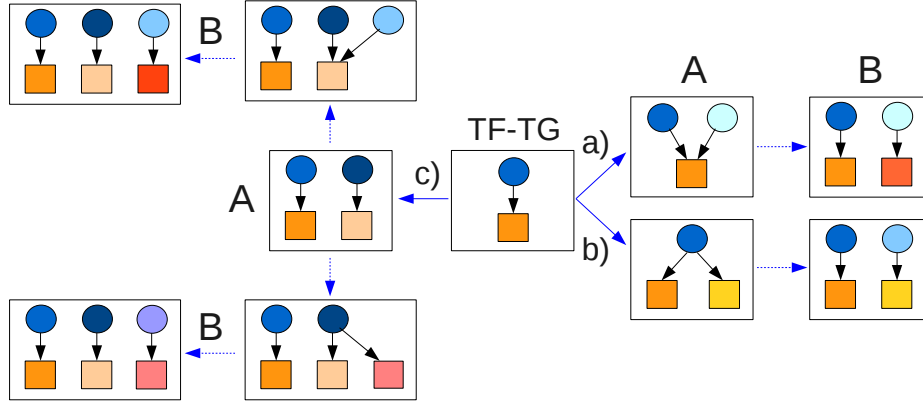


Figure 1: Possible growth pathways for a TF-TG pair (‘TF-TG’ above), modified from [5]. TF’s and TG’s are shown as circles and squares respectively. Blue arrows show all the possible changes to the GRN topology, with duplication events from the original pair represented by bold and subsequent outcomes due to divergence by dashed arrows. Black arrows represent regulatory relations. Case a)A shows duplication of a TF, where with divergence the regulatory interaction is inherited such that the two TF’s both regulate the same gene. In a)B, subsequent divergence from a)A results in the gain of a new regulatory interaction with the duplicated TF now regulating an existing or new gene. In case b)A, the TG is duplicated, such that with divergence a single TF regulates both the original and duplicated genes, that is, the regulatory interaction is inherited to the duplicated TG. Subsequent divergence again leads to a new regulatory interaction with an existing or new TF regulating the duplicated gene as shown in b)B. Case c)A shows duplication of both TF and TG, where divergence first results in homologous TF’s regulating homologous TG’s [22] and therefore inheritance of regulatory interactions. Further divergence then leads to a gain of regulatory interaction just as in a)A to a)B and b)A to b)B, shown collectively in c)B.

Modelling GRN evolution is inherently a stochastic process, similar to the dynamics of PBN’s. In both network growth and shrinkage, all functions to which the created or deleted node was essential to will require an update. This can be most easily depicted using truth tables. If we denote $\epsilon_i(t)$ as the number of essential nodes of function $f^{(i)}$ at time t , the truth table for $f^{(i)}$ will essentially have $2^{\epsilon_i(t)}$ distinct input-output relationships, which is assumed to be known. After an evolutionary event, $2^{\epsilon_i(t+1)}$ distinct relationships are required. However, these are *unknown* and thus $2^{2^{\epsilon_i(t+1)}}$ possible *sets* of input-output relationships are possible. The stochasticity of the GRN evolution arises from choosing one of the $2^{2^{\epsilon_i(t+1)}}$ sets. This stochastic process can be incorporated in both BN’s or PBN’s, but lends itself nicely to the PBN framework simply by assigning the $2^{2^{\epsilon_i(t+1)}}$ input-output relationships to the set $F_i(t+1)$ (which accordingly changes the size of K (2.7) and the distribution $c_j^{(i)}$ at time $t+1$). Computationally, the change to $c_j^{(i)}$ from time t to time $t+1$ occurs as follows. When the total number of nodes changes for the node whose essential node number has changed such that $\epsilon_i(t) \neq \epsilon_i(t+1)$, $l(i)(t+1)$ is set to $2^{2^{\epsilon_i(t+1)}}$ where recall $l(i)(t+1)$ is the total number of functions node i can take at time $t+1$. This change in $l(i)$ from t to $t+1$ then changes the size of K . If at time t , $K \in \mathbb{Z}^{N(t) \times n(t)}$, after

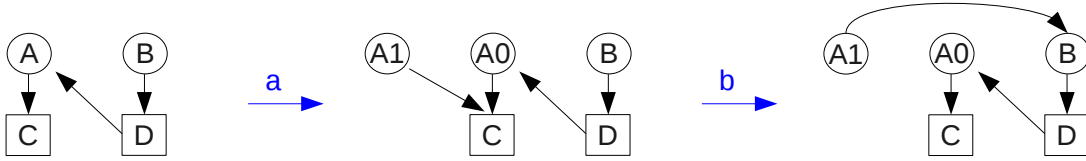


Figure 2: Effect of duplication and subsequent divergence on an isolated regulatory system. TF's A and B regulate TG's C and D . Before duplication (left) functions $f^{(A)}$, $f^{(C)}$ and $f^{(D)}$ have essential nodes $\{x_D\}$, $\{x_A\}$ and $\{x_B\}$ respectively. After TF A is duplicated to A_0 (the original) and A_1 (the duplicate), divergence may result in TF A_1 also regulating gene C (arrow a). Now $f^{(C)}$ has two essential nodes $\{x_{A_1}\}$ and $\{x_{A_0}\}$ (centre). Further divergence may then lead to TF A_1 regulating another gene in the network, gene B (arrow b). Now, $f^{(C)}$ has essential node $\{x_{A_0}\}$ and a new set of essential nodes, $\{x_{A_1}\}$ has formed for $f^{(B)}$ (right).

an evolutionary event the number of columns of K changes from $n(t)$ to $n(t+1) = n+1$, $n-1$ or $n+2$. Since $N(t)$, the total number of network function configurations at time t , is, in general, $N(t) = \prod_{i=1}^n l(i)$, at time $t+1$ where an evolutionary event has occurred, the number of rows of K changes from $N(t)$ to $N(t+1) = \prod_{i=1}^{n(t+1)} l(i)(t+1) = 2^{2^{\epsilon_i(t+1)}} \times \prod_{j \in \Omega} l(j)$, with Ω defined as the set of indices for which $\epsilon_j(t) = \epsilon_j(t+1)$. The matrix $K = K(t)$ is then used in (2.8) to find the probability of each network configuration. With this, we can find the PBN transition probability incorporating GRN evolutionary events from $\mathbf{x}(t) \rightarrow \mathbf{x}(t+1)$ with (2.10). Finding these transition probabilities for all $2^{n(t)}$ and $2^{n(t+1)}$ states will return the transition matrix $A \in [0, 1]^{2^{n(t)} \times 2^{n(t+1)}}$ with (2.1.4).

2.2.3 Example

Consider a small network of two TF's (nodes A and B) and two TG's (nodes C and D), with regulatory relationships as shown in Figure 2, left. In a BN framework, functions f_A , f_C and f_D have essential nodes $\{x_D\}$, $\{x_A\}$ and $\{x_B\}$ respectively. If A is duplicated to A_0 (the original TF) and A_1 (Figure 2, centre), divergence may result in TF A_1 also regulating TG C (arrow a). Now f_C has two essential nodes $\{x_{A_1}, x_{A_0}\}$. As aforementioned, further divergence (Figure 2 arrow b) may lead to TF A_1 regulating another gene in the network, in this case gene B . Here the set of essential nodes for f_C returns to $\{x_{A_0}\}$, and a new set of essential nodes, $\{x_{A_1}\}$, is formed for f_B (Figure 2, right). If we assign times t , $t+1$ and $t+2$ to Figure 2 left, centre and right respectively, we have $\epsilon_C(t) = 1$, $\epsilon_C(t+1) = 2$ and $\epsilon_C(t+2) = 1$. The truth table for $f^{(C)}$ at $t+2$ is given in Table 1. As shown, the input-output relationships for only the essential nodes are required. This results in four bits that are to be determined, one each for $f^{(C)}(0, 0, **)$, $f^{(C)}(0, 1, **)$, $f^{(C)}(1, 0, **)$ and $f^{(C)}(1, 1, **)$, with $*$ represents nodal values (bits) that can be either 0 or 1. Thus, 2^4 different configurations for these function values are possible for $f^{(C)}$.

3 Metabolic Networks

Metabolic networks (MN's) are essentially biochemical networks in which nodes represent metabolites and edges represent reactions. Chemical reactions have directionality owing to thermodynamic constraints and furthermore multiple metabolites (nodes) can be involved in a single

Table 1: Truth table arrangement for $f^{(C)}$ at $t + 2$.

x_{A0}	x_{A1}	x_B	x_C	x_D	$f^{(C)}(x_{A0}, x_{A1}, x_B, x_C, x_D) \in \{0, 1\}$
0	0	0	0	0	$f^{(C)}(0, 0, 0, 0, 0) = f^{(C)}(0, 0, ***)$
		⋮			$f^{(C)}(0, 0, ***)$
0	0	1	1	1	$f^{(C)}(0, 0, 1, 1, 1) = f^{(C)}(0, 0, ***)$
0	1	0	0	0	$f^{(C)}(0, 1, 0, 0, 0) = f^{(C)}(0, 1, ***)$
		⋮			$f^{(C)}(0, 1, ***)$
0	1	1	1	1	$f^{(C)}(0, 1, 1, 1, 1) = f^{(C)}(0, 1, ***)$
1	0	0	0	0	$f^{(C)}(1, 0, 0, 0, 0) = f^{(C)}(1, 0, ***)$
		⋮			$f^{(C)}(1, 0, ***)$
1	0	1	1	1	$f^{(C)}(1, 0, 1, 1, 1) = f^{(C)}(1, 0, ***)$
1	1	0	0	0	$f^{(C)}(1, 1, 0, 0, 0) = f^{(C)}(1, 1, ***)$
		⋮			$f^{(C)}(1, 1, ***)$
1	1	1	1	1	$f^{(C)}(1, 1, 1, 1, 1) = f^{(C)}(1, 1, ***)$

reaction (edge). Therefore MN’s are most effectively modelled as directed hypergraphs. The purpose of the MN in the coupled evolutionary system is to evaluate the ‘fitness’ of an evolutionary change in the GRN.

3.1 Modelling Metabolic Networks

3.1.1 Linear Programming

A general nonlinear programming problems takes the form

$$\begin{aligned} \max_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geq b_i, \quad i = 1, \dots, m \end{aligned} \tag{3.1}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, \dots, p \tag{3.2}$$

where $\mathbf{x} \in \mathbb{R}^n$. The problem is termed a *linear programming* (LP) problem if f , g_i and h_j are linear, that is, if $f(x)$, $g_i(x)$ and $h_j(x)$ are of the form $\mathbf{c}^T \mathbf{x}$, $\mathbf{G} \mathbf{x}$ and $\mathbf{H} \mathbf{x}$ respectively, with $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{G} \in \mathbb{R}^{m \times n}$, $m < n$ and $\mathbf{H} \in \mathbb{R}^{p \times n}$, $p < n$ and superscript T denoting the transpose. A notable fact about LP problems is that the search space is described by a convex polytope with candidate solutions as its vertices, guaranteeing that the local optimum is also global [18].

3.1.2 Flux Balance Analysis

Flux Balance Analysis (FBA) is a method with which a linear combination of the fluxes of metabolic reactions in a MN can be maximised or minimised given upper and lower bounds on the flux assuming the system is in steady-state [17]. Formally, for a set of m metabolites that undergo n reactions, denote the stoichiometric matrix $\mathbf{S} \in \mathbb{Z}^{m \times n}$ and the vector of fluxes for each reaction $\mathbf{v} \in \mathbb{R}^{n \times 1}$. If the concentrations of the metabolites are entries of vector $\mathbf{x} \in \mathbb{R}^{m \times 1}$, we have for the mass balance equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{S}\mathbf{v} \quad (3.3)$$

Now, at steady-state $d/dt = 0$ and therefore 3.3 simply becomes

$$\mathbf{S}\mathbf{v} = \mathbf{0} \quad (3.4)$$

It is often the case that we have known bounds on the flux in each reaction from experimental data. Therefore $\mathbf{a} \leq \mathbf{v} \leq \mathbf{b}$, where $\mathbf{a} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$ contain the lower and upper bounds of the flux in all reactions. If the objective of the cell is to maximise some linear combination of the flux in all (or some) reactions at steady-state, finding the flux distribution \mathbf{v}^* which does so is a LP problem of the form [17]

$$\max_{\mathbf{v}} \quad \mathbf{c}^T \mathbf{v} \quad (3.5)$$

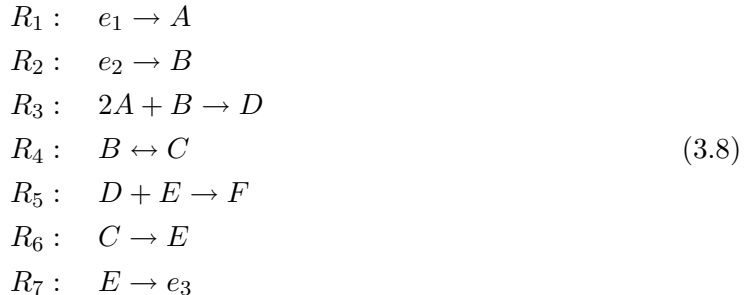
$$\text{s.t.} \quad \mathbf{a} \leq \mathbf{v} \leq \mathbf{b} \quad (3.6)$$

$$\mathbf{S}\mathbf{v} = \mathbf{0} \quad (3.7)$$

Constraint (3.7) narrows down the search space for \mathbf{v}^* to the nullspace of the stoichiometric matrix \mathbf{S} . Clearly, for a non-zero vector solution to (3.7), the nullspace of \mathbf{S} must not be empty. Additional linear constraints defined in (3.6) further reduce the search space to a convex polytope, a vertex of which is the optimal solution \mathbf{v}^* . Physically, the constraints laid down by the stoichiometric matrix \mathbf{S} imposes mass balance constraints on metabolite production and consumption whilst finer bounds on flux values is applied by \mathbf{a} and \mathbf{b} [17].

3.1.3 Example

Consider the hypothetical MN shown in Figure 3. A, B, C, D, E and F are intracellular metabolites, while e_1, e_2 and e_3 are extracellular metabolites that are transported into and out of the cell. Chemical reactions for all internal metabolites in Figure 3 are shown in (3.8).



Inclusion of reactions that transport extracellular metabolites into and out of the cell is done by introducing additional reactions that result in fictitious fluxes that only transport metabolites

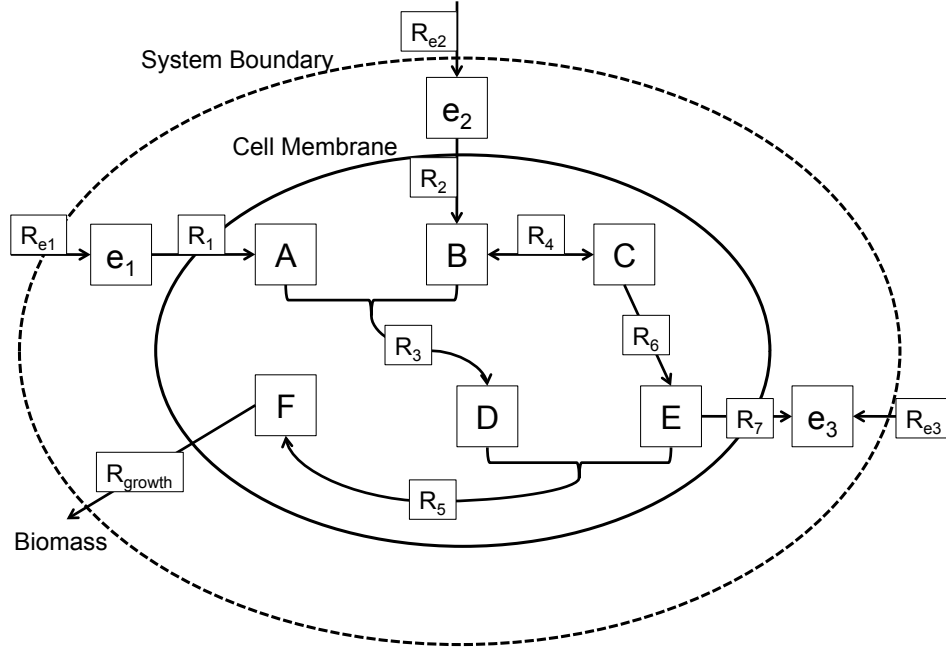


Figure 3: Hypothetical MN for a cell, modified from [4]. A, B, C, D, E and F are intracellular metabolites whilst e_1, e_2 and e_3 are extracellular metabolites. The boundary of the system which we are considering is drawn to include the extracellular metabolites

across the system boundary [4]. These reactions, which we label R_{e_1}, R_{e_2} and R_{e_3} , simply introduce the extracellular metabolites to the system. The hypothetical MN shown in Figure 3 is therefore represented by the *partitioned* stoichiometric matrix

$$\mathbf{S} = \begin{matrix} & R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 & R_{\text{growth}} & R_{e_1} & R_{e_2} & R_{e_3} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ e_1 \\ e_2 \\ e_3 \end{matrix} & \begin{pmatrix} 1 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} & \end{matrix} \quad (3.9)$$

where reversible reaction R_4 has been taken to be a single reaction in the left-to-right direction. In general, a partitioned stoichiometric matrix \mathbf{S}' can be expressed as the block matrix

$$\mathbf{S}' = [\mathbf{S}_{\text{reactions}} | \mathbf{S}_{\text{use}} | \mathbf{U}]$$

matrices $\mathbf{S}_{\text{reactions}}, \mathbf{S}_{\text{use}}$ and \mathbf{U} represent metabolic reactions that take place within the system boundary, metabolic reactions for growth and maintenance purposes and those reactions that involve metabolites that are transported into and out of the system respectively. Now mass

balance at steady-state results in fluxes that satisfy

$$\mathbf{S}'\mathbf{v} = [\mathbf{S}_{reactions} | \mathbf{S}_{use} | \mathbf{U}] \begin{bmatrix} \mathbf{v}_{reactions} \\ \mathbf{v}_{use} \\ \mathbf{b} \end{bmatrix} = \mathbf{0} \quad (3.10)$$

with \mathbf{b} being the vector of metabolic exchange fluxes for metabolites that are transported into and out of the system. Flux distributions that satisfy (3.10) are those that do not violate mass, energy or redox balance constraints. As aforementioned, such vectors lie in the null space of \mathbf{S}' , and the role of FBA is to find, by applying further constraints for physiologically feasible flux values, the flux distribution that maximises some linear combination of all fluxes. These further constraints for flux values are imposed as linear inequalities with the following convention [4]

- All internal metabolic reactions are unconstrained in the forward direction.
- For all reversible reactions, $a_i = -\infty$. For all irreversible reactions, $a_i = 0$.
- Fluxes for inorganic chemicals such as phosphates, ammonia, carbon dioxide, sulphate, potassium and sodium are unconstrained.
- Fluxes for carbon sources and oxygen are constrained, where exact upper and lower bound values are to be found experimentally.
- Fluxes of metabolites not available in the environment are set to have zero upper and lower bounds.
- Fluxes for metabolites that can leave the metabolic network, for example acetate, ethanol, lactate, succinate, formate and pyruvate, are unconstrained in the outward direction.
- Exchange fluxes, the fictitious fluxes of additional reactions introduced to the system, were defined as reversible with the positive direction being inward.

With the rules above, for hypothetical system shown in Figure 3 has the flux constraint vectors

- $\mathbf{a} = [0 \ 0 \ 0 \ -\infty \ 0 \ 0 \ 0 \ 0 \ V_1 \ V_2 \ -\infty]^T$.
- $\mathbf{b} = [\infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ \infty \ V_1 \ V_2 \ 0]^T$.

where V_1 and V_2 are fixed fluxes for metabolites e_1 and e_2 respectively. Once both the stoichiometric matrix and the flux constraints are set up, a vector \mathbf{c} is chosen to maximise (or minimise) some linear combination of reaction fluxes. For maximum growth in the hypothetical system of Figure 3, $\mathbf{c}^T\mathbf{v} = R_{\text{growth}}$.

3.2 Evolution of Metabolic Networks

3.2.1 Edge Insertion/Deletion Model

The model for the evolution of MN's used in this study is largely based on [13, 12], namely the edge insertion/deletion (EID) model. We define a MN with n nodes (metabolites) and m edges (reactions), $M_{(n,m)}(W, R)$, with the set of nodes $W = \{y_1, \dots, y_n\}$ and the set of edges $R = \{R_1, \dots, R_m\}$. Each $R_i \subset R$ represents a reaction with a non-empty input (reactant) set of nodes

and a non-empty output (product) set of nodes. For example, in (3.8), $R_1 = \{(A, B), (C, D)\}$, $R_2 = \{(B, C), A\}$ and $R_3 = \{D, E\}$. In each evolutionary step, an edge (reaction) can be inserted (if it is not already present) or deleted (if it is already present). The number of nodes (metabolites) remains fixed. Since insertion and deletion of an edge to an existing network are discrete events, the evolution process is ideally modelled as a discrete space continuous time Markov process.

The EID model described so far is far from biological reality. Some metabolic reactions may be crucial to the survival of the cell and therefore cannot be deleted. Conversely, some reactions, if inserted in the presence of certain reactions may be prohibited. Therefore, we introduce the set of *core edges* $C \subset R$, edges (reactions) that cannot be removed, and the set of *prohibited edges* $\Gamma \subset R$, edges (reactions) that cannot be added for a specific organism during evolution [13]. For a MN $M_{(n,m)}$, assuming at least one metabolite must undergo a reaction to produce another and that all metabolites (nodes) can reach all other metabolites (nodes), there can be at most $(2^n - 1)^2$ possible edges (reactions). Then, the following clearly holds

- $m \leq (2^n - 1)^2$ where it is normally expected that equality is never reached.
- $|C| \leq m$, with equality when all m edges are core edges. When equality holds, no more edges may be removed.
- $|\Gamma| + m \leq (2^n - 1)^2$, with equality when all reactions not included are prohibited. When equality holds, no more edges may be added.

We note here an important point. Due to chemical and biological constraints, not all of the $(2^n - 1)^2$ reactions can be realised in an organisms. A ‘reference pathway’ for carbohydrate metabolism, available from KEGG [9, 10] is shown in Figure 4. A metabolite, represented by either a rounded rectangle (end nodes in the network) or a small empty circle (intermediary nodes in the network) does not connect to all other metabolites. Instead, the figure shows the reference network where reactions from all organisms within the database have been included. In general, a single organism will not require every reaction in the reference network. For example, as shown in Figure 4, only those reactions for which the relevant enzymes have been highlighted are found in *Escherichia coli* K-12 MG1655.

3.2.2 Edge Insertion/Deletion and Flux Balance Analysis

In this study, we combine the EID model of MN evolution with FBA, denoting it the EIDFBA model. The EIDFBA model accounts for edge insertion/deletion by filling (when a reaction is inserted) and emptying (when a reaction is deleted) columns of the stoichiometric matrix \mathbf{S} in (3.7) and changing the entries of \mathbf{a} , \mathbf{b} in (3.6) and \mathbf{c} in (3.5) accordingly. Therefore \mathbf{S} , \mathbf{a} , \mathbf{b} and \mathbf{c} are now $\mathbf{S}(t)$, $\mathbf{a}(t)$, $\mathbf{b}(t)$ and $\mathbf{c}(t)$. If we denote \mathcal{R} as the set of all possible reactions in the reference network, $\mathbf{S} \in \mathbb{Z}^{m \times |\mathcal{R}|}$ and at time t we have $\mathbf{S}_{i\{\mathcal{R} \setminus R(t)\}}(t) = \{0\}^n$ in (3.7). In other words, columns corresponding to all reactions not present in the MN at time t are set to zero columns. Similarly, each entry of $\mathbf{a}_{\{\mathcal{R} \setminus R(t)\}}$, $\mathbf{b}_{\{\mathcal{R} \setminus R(t)\}}$ and $\mathbf{c}_{\{\mathcal{R} \setminus R(t)\}}$ are all set to 0 in (3.6) and (3.5).

4 Connecting Regulatory and Metabolic Networks

There are several examples of integrated regulatory-metabolic network models currently available in the literature. For example, Shlomi et al. present ‘steady-state regulatory flux balance

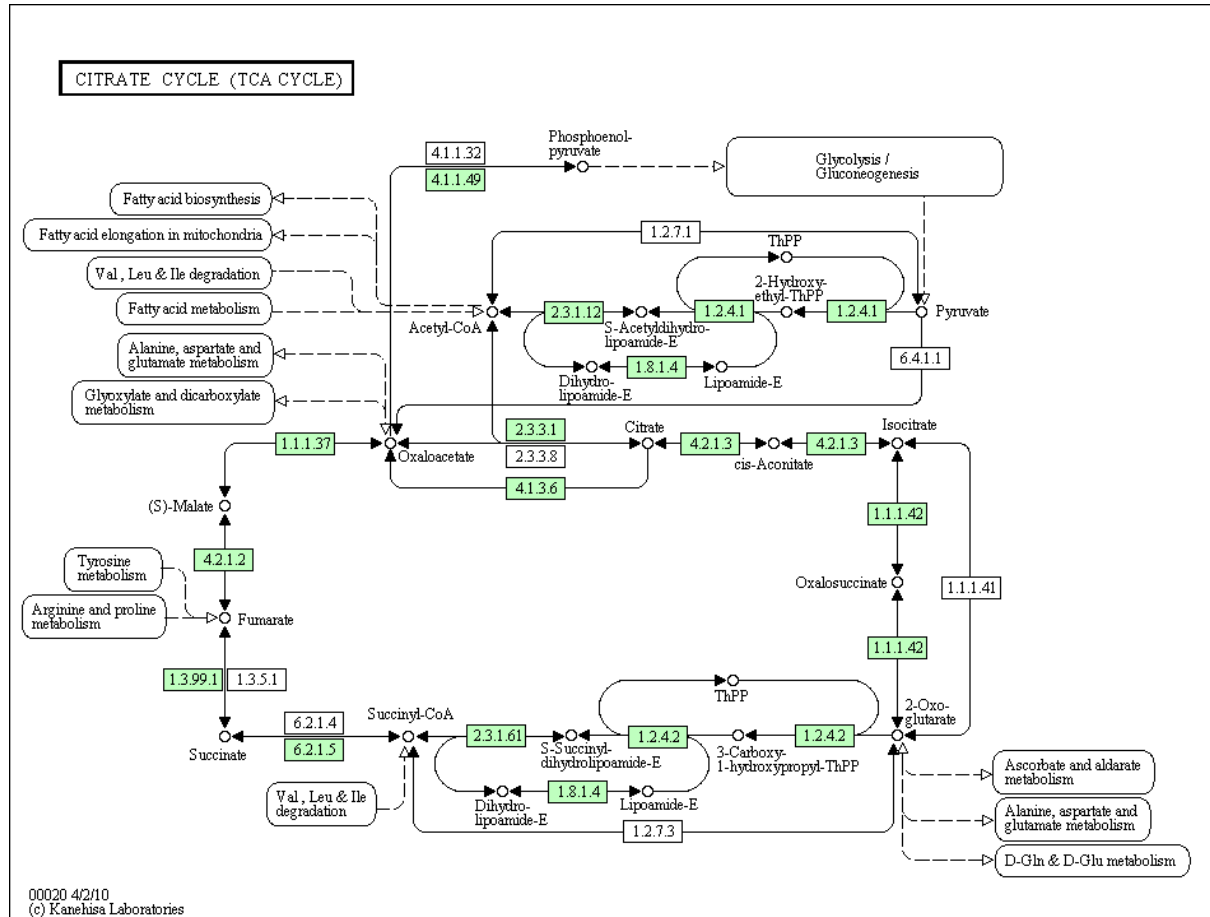


Figure 4: Reference network for carbohydrate metabolism constructed as the union of all networks available in the KEGG database [9, 10]. The enzymes that catalyse reactions present in the network for *Escherichia coli* K-12 MG1655 are highlighted. Clearly not all metabolites (rounded rectangles for end metabolites and small empty circles for intermediary metabolites) are not connected to all other metabolites.

analysis (SRFBA)', in which essentially the regulatory and metabolic steady-states are updated in an iterative manner given changing levels of extracellular metabolite concentrations [20]. In the study by Yeang and Vingron, a factor graph is essentially trained with perturbation data to identify the exact form of functional links between metabolites and regulators (TF's) [23]. Furthermore, as mentioned in sections 2.2 and 3.2, studies have considered the evolution of GRN's and MN's alone. Despite these studies, mathematical models of coupled regulatory-metabolic network models have received less attention. Consequently, we propose a novel approach to implementing evolutionary events into a coupled regulatory-metabolic network model similar to that proposed in [20].

4.1 Explicit and Implicit Connections

Connections between GRN's and MN's are classified into what will be called explicit or implicit connections. Explicit connections are directed from the GRN to the MN, where it is assumed that the presence and absence of an enzyme correlates exactly with high and low expression levels of the gene that is responsible for the enzyme. Implicit connections describe feedback from the MN to the GRN by means of the steady-state flux distribution calculated from FBA. Implementing explicit connections simply involves drawing the metabolic connectivity given the GRN node states when evolutionary events take place. To apply implicit connections, once the steady-state flux distribution which maximises biomass is calculated from FBA, this distribution is used as a measure of cell 'fitness' when determining whether to accept or reject an evolutionary event. If an evolutionary event results in a GRN and therefore MN which increases the biomass production of the cell, it is assumed that cell populations with these biochemical circuits have the competitive edge and survive. If the event reduces the biomass production, it is then assumed that this cell line *rarely* survives, or the evolutionary change simply ignored.

4.2 Metabolic Network Evolution as a Consequence of Regulatory Evolution

In using the EID model for MN evolution alone, Mithani, Preston and Hein [13] considered two models of MN evolution: the independent edge model, where the rates of edge insertion and deletion are both constant for all edges and the neighbour-dependent model, in which edges are more often added or removed to nodes with higher in- and out-degrees than those more poorly connected. The latter model is analogous to the preferential attachment models [3] that give rise to connectivities that follow the so-called 'scale-free' distribution [8]. In both settings, the MN evolves independently of any evolutionary events in the GRN responsible for the MN in question. Here we consider this effect, and therefore model MN evolution as the *consequence* of GRN evolution. The EID model of MN evolution is kept, but now whether a reaction (edge) is inserted or deleted after an evolutionary event is dictated by the GRN that governs the expression levels of the enzyme responsible for the reaction.

4.3 Algorithm for Coupled Evolution

The algorithm for GRN-MN coupled evolution is shown in Algorithm 1, where the following applies

- T (line 1) is the maximum time to which we simulate evolution. To save computational resources, it is assumed that after each iteration an evolutionary event takes place with

probability one, irrespective of whether the resulting circuitry is to be kept (the cell population survives) or discarded (the cell population dies out).

- α (line 3) is a parameter which controls the rate at which an evolutionary event results in duplication or deletion. Previously, in section 2.2.1 α was taken to be 0.5.
- When a node duplication event occurs, the parameter β (line 5) controls the rate at which one of the three duplication scenarios: TF, TG or TF-TG pair duplication occurs.
- Parameter δ (line 13) dictates whether the ‘Use It or Lose It’ rule or the ‘Use It, but Lose It Anyway’ rule is used to remove a node from the GRN.
- The function \mathcal{MN} (line 25) simply maps the MN provided the expression states of the GRN nodes, $V(t+1)$ using explicit connections. If the event has resulted in node duplication, and $\text{Uniform}[0, 1] \geq \gamma$ such that these new nodes code for enzymes that catalyse reactions in the MN, \mathcal{MN} will generate the appropriate MN.
- The function \mathcal{FBA} (line 26) takes as input $M_{(p,q)}(W, R)(t+1)$ and finds the steady-state flux distribution over all present reactions $R(t+1)$ that maximises an objective function defined by \mathbf{c} . As aforementioned, the constraint vectors \mathbf{a} and \mathbf{b} as well as the objective vector \mathbf{c} may change with time.

The values of parameters α , β , γ and δ mentioned above will differ depending on the organism and environment in question.

5 Simulations

Ideally, a full-scale evolutionary model requires knowledge of the regulatory interactions and metabolic pathways of a whole organism. Though integrated regulatory-metabolic network models exist [6, 23] the algorithm must first be tested on a toy coupled regulatory-metabolic network architecture. For the purpose of this short project report, the MATLAB code for the evolution of a given GRN topology following the procedures of algorithm 1 (lines 2-24) as well as that for generating the stoichiometric matrix for the hypothetical MN of section 3.1.3 has been prepared (available in sections 6.2 and 6.1 respectively).

5.1 Toy Regulatory-Metabolic Network

A random PBN was generated using the `pbnRnd.m` function in the BN/PBN MATLAB Toolbox [21] with $n = 5$ nodes and $l(i) \leq 4$, $i = 1, \dots, n$. At time t_0 , out of the five nodes, nodes 3, 4 and 5 were taken to be genes that code for enzymes that catalyse reactions R_3 , R_4 and R_5 of the hypothetical MN in section 3.1.3. As the algorithm progresses, these reactions will be removed and reinserted into the stoichiometric matrix.

Algorithm 1 Coupled Evolution of GRN and MN

```

1: for  $t = t_0$  to  $T$  do
2:   Make an evolutionary change such that  $|V(t)| - |V(t+1)| \neq 0$ .
3:   if  $\text{Uniform}[0, 1] \geq \alpha$  then
4:     A duplication event occurs in the GRN such that  $|V(t+1)| > |V(t)|$ .
5:      $n(t+1) \leftarrow n(t) + 1 + \lceil \text{Uniform}[0, 1] - \beta \rceil$ 
6:     if  $\text{Uniform}[0, 1] \geq \gamma$  then
7:       New node(s) (genes) code for enzyme present in MN
8:     else
9:       New node(s) (genes) do not code for enzyme present in MN
10:    end if
11:  else
12:    A deletion event occurs in the GRN such that  $|V(t+1)| < |V(t)|$ .
13:    if  $\text{Uniform}[0, 1] \geq \delta$  then
14:      ‘Use It, but Lose It Anyway’ rule applies
15:    else
16:      ‘Use It or Lose It’ rule applies
17:    end if
18:     $n(t+1) \leftarrow n(t) - 1$ 
19:  end if
20:  if  $n(t+1) > n(t)$  then
21:    Generate random GRN with  $n(t+1)$  nodes; return  $\tilde{G}'_{n(t+1)}(V, F)(t+1)$ 
22:  else if  $n(t+1) < n(t)$  then
23:    Remove node according to  $\delta$ ; return  $\tilde{G}'_{n(t+1)}(V, F)(t+1)$ 
24:  end if
25:   $M'_{(p,q)}(W, R)(t+1) \leftarrow \mathcal{MN}(V(t+1), \gamma)$ 
26:   $\mathbf{v}^*(t+1) \leftarrow \mathcal{FBA}(M'_{(p,q)}(W, R)(t+1))$ 
27:  if  $\mathbf{v}^*(t+1)/\mathbf{v}^*(t) \geq 1$  then
28:    Accept GRN resulting from evolutionary event, since it increases ‘fitness’
29:     $\tilde{G}_{n(t+1)}(V, F)(t+1) \leftarrow \tilde{G}'_{n(t+1)}(V, F)(t+1)$ .
30:  else if  $\mathbf{v}^*(t+1)/\mathbf{v}^*(t) \geq \text{Uniform}[0, 1]$  then
31:    Accept GRN resulting from evolutionary event even if it decreases ‘fitness’ with probability  $\mathbf{v}^*(t+1)/\mathbf{v}^*(t)$ .
32:     $\tilde{G}_{n(t+1)}(V, F)(t+1) \leftarrow \tilde{G}'_{n(t+1)}(V, F)(t+1)$ 
33:  else
34:    Evolutionary event has occurred but the resulting cell population have died. The majority of the population are those that have not undergone the evolutionary event.
35:     $n(t+1) \leftarrow n(t)$ 
36:     $\tilde{G}_{n(t+1)}(V, F)(t+1) \leftarrow \tilde{G}_{n(t)}(V, F)(t)$ 
37:     $M'_{(p,q)}(W, R)(t+1) \leftarrow M_{(p,q)}(W, R)(t)$ 
38:  end if
39:   $M_{(p,q)}(W, R)(t+1) \leftarrow M'_{(p,q)}(W, R)(t+1)$ 
40: end for

```

6 MATLAB Code

6.1 Stoichiometric Matrix for Toy Network of Section 3.1.3

```

1  %% Stoichiometric matrix for Toy Network
2
3  S = sparse(9,11);
4  % R1
5  S(1,1) = 1; S(7,1) = -1;
6  % R2
7  S(1,2) = 1; S(8,2) = -1;
8  % R3
9  S(1,3) = -2; S(2,3) = 1; S(4,3) = 1;
10 % R4
11 S(2,4) = -1; S(3,4) = 1;
12 % R5
13 S(4,5) = -1; S(5,5) = -1; S(6,5) = 1;
14 % R6
15 S(3,6) = -1; S(5,6) = 1;
16 % R7
17 S(5,7) = -1; S(9,7) = 1;
18 % R_growth
19 S(6,8) = -1;
20 % R_e1
21 S(7,9) = 1;
22 % R_e2
23 S(8,10) = 1;
24 % R_e3
25 S(9,11) = 1;
26
27 %% Objective function
28
29 c_toy = zeros(11,1);
30 c_toy(8) = 1;

```

6.2 Main Function for Algorithm 1

```

1  clear; close all; clc;
2  %% Create MN Structure
3      % The stoichiometric matrix for the galactose metabolism reference
4      % network from KEGG is used such that the scale-free network structure
5      % of the MN is kept.
6
7  load S_toy.mat
8
9  S_t = struct([]);
10 c_t = struct([]);
11
12 S_t{1} = S;
13 c_t{1} = c_toy;
14
15 %% Algorithm
16 T = 1000;

```

```

17 alpha = rand; beta = rand; Δ = rand;
18
19 n = zeros(1,T);
20 max_func = 4; % maximum number of nodal functions of the PBN.
21 nf = struct([]);
22 nv = struct([]);
23
24 cij_t = struct([]);
25 varF = struct([]);
26 F = struct([]);
27
28 x = struct([]);
29 v_star = struct([]);
30
31 remove = zeros(1,T-1);
32
33 for t = 1:T-1
34     %% Part A1. Initialise GRN structure
35
36     if t == 1
37         n(t) = 5;
38         nf{t} = randi(max_func,1,n(t));
39         nv{t} = randi(n(t),1,sum(nf{t}));
40         [F1,varF1,cij1] = pbnRnd(n(t),nf{t},nv{t});
41         F{t} = F1; varF{t} = varF1; cij_t{t} = cij1;
42         x{t} = round(rand(1,n(t))); % random node states at t0.
43     end
44
45     %% Part A2. Make a change to the GRN
46     if rand ≥ alpha
47         n(t+1) = n(t)+1+ceil(rand-beta); % n(t+1) = n(t)+1 or n(t)+2
48         remove(t) = 0;
49     else
50         n(t+1) = n(t)-1;
51         if rand ≥ Δ
52             remove(t) = randi(n(t),1); % 'Use It, but Lose It Anyway'
53         else
54             tmp = find(x{t}==0);
55             remove(t) = tmp(randi(length(tmp),1)); % 'Use It or Lose It'
56         end
57     end
58     clear tmp
59
60     cnf = cumsum(nf{t});
61     snf = cnf(end);
62     nv2 = nv{t};
63
64     %% Find candidate state of the GRN at t+1
65
66     if remove(t) == 0
67         % if there has been a duplication
68         gamma = rand; % for function MN
69
70         varF2 = [varF{t}; -1*ones(n(t+1)-n(t),snf)];
71         for i = (n(t)+1):n(t+1)
72             tmp = randperm(snf);
73             tmp2 = tmp(1:randi(snf));
74             for j = 1:length(tmp2)
75                 varF2(nv2(tmp2(j))+1,tmp2(j)) = i;

```

```

76         end
77         nv2(tmp2) = nv{t}(tmp2)+1;
78     end
79     clear tmp tmp2
80
81     extra_f = randi(max_func,1,n(t+1)-n(t));
82     nf2 = [nf{t} extra_f];
83     nv2 = [nv2 randi(n(t),1,sum(extra_f))];
84     [F2,varF2,cij2] = pbnRnd(n(t+1),nf2,nv2);
85
86     for i = 1:length(nv{t})
87         if (nv2(i)-nv{t}(i)) == 0
88             F2(1:2^nv{t}(i),i) = F{t}(1:2^nv{t}(i),i);
89             F2(2^nv{t}(i)+1:end,i) = -1;
90         end
91     end
92     x2 = [x{t} round(rand(1,n(t+1)-n(t)))];
93 else
94     % if there has been a deletion
95
96     varF2 = varF{t};
97     varF2(varF2==remove(t)) = -1;
98     nv2 = zeros(1,snf);
99     nv2(1:snf) = sum(varF2(:,1:snf)~=1);
100    no_var = find(nv2==0);
101
102    nf2 = nf{t};
103    nf2(remove(t)) = [];
104
105    F2 = F{t};
106    if remove(t) == 1
107        nv2(1:cnf(1)) = [];
108        varF2(:,1:cnf(1)) = [];
109        F2(:,1:cnf(1)) = [];
110    else
111        nv2(cnf(remove(t)-1)+1:cnf(remove(t))) = [];
112        varF2(:,cnf(remove(t)-1)+1:cnf(remove(t))) = [];
113        F2(:,cnf(remove(t)-1)+1:cnf(remove(t))) = [];
114    end
115
116    cij_t{t}(:,remove(t)) = [];
117    cij2 = cij_t{t};
118    x2 = x{t};
119    x2(remove(t)) = [];
120
121    for i = remove+1:n(t)
122        varF2(varF2==i) = i-1;
123    end
124
125    tmp = find(nv2==0);
126    cnf2 = [0 cumsum(nf2)];
127    if ~isempty(tmp)
128        for i = 1:length(cnf2)-1
129            if any(tmp>cnf2(i)+1 & tmp<cnf2(i+1))
130                nf2(i) = nf2(i)-sum(tmp>cnf2(i)+1 ...
131                    & tmp<cnf2(i+1));
132            end
133        end
134    end

```

```

135     [r1 c1] = find(cij2 $\neq$ -1);
136     for i = 1:length(tmp)
137         cij2(:,c1(tmp(i))) = cij2(:,c1(tmp(i))) + ...
138             cij2(r1(tmp(i)),c1(tmp(i)))/ ...
139             (sum(cij2(:,c1(tmp(i))) $\neq$ -1)-1);
140         cij2(r1(tmp(i)),c1(tmp(i))) = -1;
141         cij2(cij2(:,c1(tmp(i)))<0,c1(tmp(i))) = -1;
142     end
143     varF2(:,tmp) = [];
144     F2(:,tmp) = [];
145     nv2(nv2==0) = [];
146     for i = 1:length(nv2)
147         varF2(:,i) = [sort(varF2(varF2(:,i)>0,i), 'ascend'); ...
148             varF2(varF2(:,i)<0,i)];
149     end
150     clear tmp
151     if size(varF2,1)  $\neq$  max(nv2)
152         for i = 1:size(varF2,1)
153             tmp(i) = all(varF2(i,:)<0);
154         end
155     end
156     varF2(tmp,:) = [];
157 end
158
159 %% Can check here that
160
161 % sum(nf) = length(nv)
162 % size(varF,2) = length(nv)
163 % size(varF,1) = max(nv)
164 % sum(varF(:,i) $\neq$ -1) = nv(i)
165 % size(F,2) = length(nv)
166 % size(F,1) = 2^(max(nv))
167 % sum(cij(:,i) $\neq$ -1) = nf(i)
168
169 %% Part B. Assess fitness of candidate state
170
171 x_dash = pbnNextState(x2,F2,varF2,nf2,nv2,cij2,0);
172 S_t_dash = MN(x_dash,gamma,S_t{t},remove);
173 v_star{t+1} = FBA(S_t_dash,a,b,c);
174 if v_star{t+1}  $\geq$  v_star{t}
175     x{t+1} = x_dash;
176 elseif v_star{t+1}/v_star{t}  $\geq$  rand
177     x{t+1} = x_dash;
178 else
179     n(t+1) = n(t); % overwrite n(t+1) with old n(t)
180     x{t+1} = pbnNextState(x{t},F{t},varF{t}, ...
181         nf{t},nv{t},cij_t{t},0);
182     S_t_dash = MN(x{t+1},gamma,S_t{t},remove);
183 end
184 S_t{t+1} = S_t_dash;
185 end

```

7 Evaluation and Remaining Work

This report has thus far covered the theoretical and computational aspects of the presented model. Much work remains. The first task is to complete implementing the algorithm to have

a working model and to evaluate the results of running the algorithm on the toy network. Like many mathematical models, it is expected that not all biological details will be encompassed. However, it is crucial that the model in one way or another answers the question initially posed. Therefore, once the algorithm has been fully brought about and optimised on the toy network, it is essential that real data is also obtained in order to compare the results from both the theoretical and experimental approaches. This step will be by no means singular and is expected to be an iterative process.

To start to this model-verification step, the internal reactions from the galactose metabolism pathway (Figure 5, map00052 of KEGG [9, 10]) have been manually curated in Tables 2 and 3. This must be continued to account for the extracellular metabolites, such as ATP or H_2O , for a complete stoichiometric matrix for FBA. Naturally, the GRN governing the galactose metabolism is also required. This would involve searching the literature (or carrying out experiments) to find a Boolean representation of the necessary genes.

A crucial inspection yet to be implemented is the check for *stoichiometric consistency*, where metabolic reactions inserted or deleted must be done so only if mass conservation holds. Mathematically, this involves the search for a vector $\mathbf{m} > \mathbf{0}$ such that $\mathbf{S}^T \mathbf{m} = \mathbf{0}$ is satisfied. These conditions essentially state that the weighted sum of molecular masses of metabolites with stoichiometric constants as weights should equal zero, i.e. mass conservation is maintained. The MN is deemed *consistent* if there is at least one vector \mathbf{m} which satisfies this condition and *inconsistent* otherwise [7].

Table 3: Design key table for metabolic reactions from the reference network shown in Figure 5. Reaction numbers in bold show reactions deemed to increase biomass.

Metabolic Reactions		
Reaction No.	KEGG EC/Reaction Number	Chemical Equation
R_1	1.1.3.9/R01098	$M_1 + O_2 + H_2O \rightarrow M_3 + H_2O_2$
R_2	1.1.1.48/R01094	$M_1 + NAD^+ \rightarrow M_2 + NADH + H^+$
R_3	1.1.1.120/R01097	$M_1 + NADP^+ \rightarrow M_2 + NADPH + H^+$
R_4	1.1.1.21/R01093	$M_1 + NADH + H^+ \leftrightarrow M_{40} + NAD^+$
R_5	1.1.1.16/R02928	$M_{40} + NAD^+ \rightarrow M_{41} + NADH + H^+$
R_6	1.1.1.251/R05571	$M_{42} + NAD^+ \rightarrow M_{39} + NADH + H^+$
R_7	2.7.1.6/R01092	$ATP + M_1 \rightarrow ADP + M_9$
R_8	2.7.7.12/R00955	$M_8 + M_9 \leftrightarrow M_{10} + M_{12}$
R_9	2.7.7.10/R00502	$UTP + M_9 \rightarrow P_2H_4O_7 + M_{12}$
R_{10}	2.4.1.22/R00503	$M_{12} + C_6H_{12}O_6 \rightarrow UDP + M_{14}$
R_{11}	2.7.1.69/ $\left\{ \begin{array}{l} R08366 \\ R04393 \\ R05570 \\ R08367 \end{array} \right\}$	$\left\{ \begin{array}{l} C_7H_9N_4O_5PR_2 + M_{14} \rightarrow C_7H_8N_4O_2R_2 + M_{23} \\ C_7H_9N_4O_5PR_2 + M_{14} \rightarrow C_7H_8N_4O_2R_2 + M_{23} \\ M_{40} + C_7H_9N_4O_5PR_2 \rightarrow M_{42} + C_7H_8N_4O_2R_2 \\ M_{25} + C_7H_9N_4O_5PR_2 \rightarrow M_{35} + C_7H_8N_4O_2R_2 \end{array} \right\}$
R_{12}	2.7.1.1/R01786	$ATP + M_{19} \leftrightarrow ADP + M_{20}$
R_{13}	2.7.1.2/R01786	$ATP + M_{19} \leftrightarrow ADP + M_{20}$
R_{14}	2.7.7.9/R00289	$UTP + M_{10} \leftrightarrow P_2H_4O_7 + M_8$
R_{15}	2.7.1.58/R03387	$ATP + M_4 \rightarrow ADP + M_5$
R_{16}	2.4.1.123/R06214	$M_{12} + C_6H_{12}O_6 \leftrightarrow UDP + M_{13}$
R_{17}	2.4.1.82/R02411	$M_{13} + M_{16} \rightarrow C_6H_{12}O_6 + M_{15}$
Continued ...		

Table 3: (continued)

Metabolic Reactions (continued)		
Reaction No.	KEGG EC/Reaction Number	Chemical Equation
R_{18}	2.4.1.67/R03418	$C_{12}H_{22}O_{11} + M_{15} \rightarrow C_6H_{12}O_6 + M_{21}$
R_{19}	2.7.1.69/R08366	$M_{17} + C_7H_9N_4O_5PR_2 \rightarrow M_{26} + C_7H_8N_4O_2R_2$
R_{20}	2.7.1.101/R02927	$ATP + M_{41} \leftrightarrow ADP + M_{39}$
R_{21}	2.7.1.11/R03236	$M_{39} + ATP \leftrightarrow M_{18} + ADP$
R_{22}	2.7.1.144/R03236	$M_{39} + ATP \leftrightarrow M_{18} + ADP$
R_{23}	3.2.1.22/ $\left\{ \begin{array}{l} R01194 \\ R02926 \\ R01329 \\ R01104 \\ R03634 \\ R05549 \\ R01103 \end{array} \right\}$	$\left\{ \begin{array}{l} M_{13} + H_2O \leftrightarrow M_{22} + C_6H_{12}O_6 \\ M_{28} + H_2O \leftrightarrow M_{27} + C_6H_{12}O_6 \\ M_{31} + H_2O \leftrightarrow M_{30} + C_6H_{12}O_6 \\ M_{37} + H_2O \leftrightarrow M_{36} + C_6H_{12}O_6 \\ M_{21} + H_2O \rightarrow M_{15} + M_{32} \\ M_{24} + H_2O \rightarrow M_{32} + M_{29} \\ M_{15} + H_2O \rightarrow C_6H_{12}O_6 + M_{16} \\ M_{29} + H_2O \rightarrow M_{32} + M_{33} \end{array} \right\}$
R_{24}	3.2.1.23/ $\left\{ \begin{array}{l} R01105 \\ R01678 \end{array} \right\}$	$\left\{ \begin{array}{l} M_{11} + H_2O \rightarrow M_1 + C_{12}H_{20}O_{11} \\ M_{14} + H_2O \rightarrow C_6H_{12}O_6 + M_1 \end{array} \right\}$
R_{25}	3.1.1.25/R03034	$M_2 + H_2O \leftrightarrow M_3$
R_{26}	3.2.1.108/R01678	$M_{14} + H_2O \rightarrow M_{19} + C_6H_{12}O_6$
R_{27}	3.2.1.85/R03256	$M_{23} + H_2O \rightarrow C_6H_{12}O_6 + M_{38}$
R_{28}	3.1.3.9/R01788	$M_{20} + H_2O \leftrightarrow M_{19} + H_3PO_4$
R_{29}	3.2.1.26/ $\left\{ \begin{array}{l} R03635 \\ R02410 \end{array} \right\}$	$\left\{ \begin{array}{l} M_{21} + H_2O \rightarrow M_{24} + C_6H_{12}O_6 \\ M_{15} + H_2O \rightarrow M_{29} + C_6H_{12}O_6 \end{array} \right\}$
R_{30}	3.2.1.20/R00801	$M_{16} + H_2O \rightarrow M_{35} + M_{34}$
R_{31}	3.5.1.25/R05168	$M_{26} + H_2O \rightarrow M_{35} + C_2H_4O_2$
R_{32}	4.2.1.6/R03033	$M_3 \rightarrow M_4 + H_2O$
R_{33}	4.1.2.21/R01064	$M_5 \rightarrow C_3H_4O_3 + M_6$
R_{34}	4.1.2.40/R01069	$M_{18} \leftrightarrow M_7 + M_6$
R_{35}	5.4.2.2/R00959	$M_{10} \leftrightarrow M_{20}$
R_{36}	5.1.3.2/R00291	$M_8 \leftrightarrow M_{12}$
R_{37}	5.3.1.-/R08365	$M_{35} + H_2O \rightarrow M_{39} + NH_3$
R_{38}	5.3.1.26/R03240	$M_{38} \rightarrow M_{39}$

Table 2: Design key table for metabolites from the reference network shown in Figure 5.

Metabolites		
Metabolite No.	Metabolite	Chemical Equation
M_1	D-Galactose	$C_6H_{12}O_6$
M_2	D-Galactono-1,4-lactone	$C_6H_{10}O_6$
M_3	D-Galactonate	$C_6H_{12}O_7$
M_4	2-Dehydro-3-deoxy-D-galactonate	$C_6H_{10}O_6$
M_5	2-Dehydro-3-deoxy-D-galactonate-6P	$C_6H_{11}O_9P$
M_6	D-Glyceraldehyde-3P	$C_3H_7O_6P$
M_7	Glycerone-P	$C_3H_7O_6P$
M_8	UDP-glucose	$C_{15}H_{24}N_2O_{17}P_2$
M_9	α -D-Galactose-1P	$C_6H_{13}O_9P$
M_{10}	α -D-Glucose-1P	$C_6H_{13}O_9P$
M_{11}	Galactan	$(C_{12}H_{20}O_{11})_n$
M_{12}	UDP-galactose	$C_{15}H_{24}N_2O_{17}P_2$
M_{13}	Galactinol	$C_{12}H_{22}O_{11}$
M_{14}	Lactose	$C_{12}H_{22}O_{11}$
M_{15}	Raffinose	$C_{18}H_{32}O_{16}$
M_{16}	Sucrose	$C_{12}H_{22}O_{11}$
M_{17}	N-Acetyl-D-galactosamine	$C_8H_{15}NO_6$
M_{18}	D-Tagatose-1,6P ₂	$C_6H_{14}O_{12}P_2$
M_{19}	α -D-glucose	$C_6H_{12}O_6$
M_{20}	α -D-Glucose-6P	$C_6H_{13}O_9P$
M_{21}	Stachyose	$C_{24}H_{42}O_{21}$
M_{22}	D-myo-Inositol	$C_6H_{12}O_6$
M_{23}	Lactose-6'P	$C_{12}H_{23}O_{14}P$
M_{24}	Manninotriose	$C_{18}H_{32}O_{16}$
M_{25}	D-Galactosamine	$C_6H_{13}NO_5$
M_{26}	N-Acetyl-D-galactosamine-6P	$C_8H_{16}NO_9P$
M_{27}	D-sorbitol	$C_6H_{14}O_6$
M_{28}	Melibiitol	$C_{12}H_{24}O_{11}$
M_{29}	Melibiose	$C_{12}H_{22}O_{11}$
M_{30}	D-Mannose	$C_6H_{12}O_6$
M_{31}	Epimelibiose	$C_{12}H_{22}O_{11}$
M_{32}	D-Galactose	$C_6H_{12}O_6$
M_{33}	D-Glucose	$C_6H_{12}O_6$
M_{34}	D-Fructose	$C_6H_{12}O_6$
M_{35}	D-Galactosamine-6P	$C_6H_{14}NO_8P$
M_{36}	Glycerol	$C_3H_8O_3$
M_{37}	Galactosyl-glycerol	$C_9H_{18}O_8$
M_{38}	D-Galactose-6P	$C_6H_{13}O_9P$
M_{39}	D-Tagatose-6P	$C_6H_{13}O_9P$
M_{40}	Galactitol	$C_6H_{14}O_6$
M_{41}	D-Tagatose	$C_6H_{12}O_6$
M_{42}	Galactitol-1P	$C_6H_{15}O_9P$

References

- [1] Bruce Alberts, Dennis Bray, Julian Lewis, Martin Raff, Keith Roberts, and James D. Watson. *Molecular Biology of the Cell*. Garland Publishing, 3rd edition, 1994.
- [2] Uri Alon. *An Introduction to Systems Biology - Design Principles of Biological Circuits*. Chapman & Hall, 2007.
- [3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509 – 512, 1999.
- [4] Jeremy S. Edwards, Rafael U. Ibarra, and Bernhard Ø Palsson. *In silico* Predictions of *Escherichia coli* Metabolic Capabilities are Consistent with Experimental Data. *Nature Biotechnology*, 19:125 – 130, 2001.
- [5] M Madan Babu et al. Structure and evolution of transcriptional regulatory networks. *Current Opinion in Structural Biology*, 14:283 – 291, 2004.
- [6] M. W. Covert et al. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429:92 – 96, 2004.
- [7] Albert Gevorgyan, Mark G. Poolman, and David A. Fell. Detection of stoichiometric inconsistencies in biomolecular models. *Bioinformatics*, 24(19):2245 – 2251, 2008.
- [8] H. Jeong, B. Tombor, R. Albert, Z. N. Oltval, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651 – 654, 2000.
- [9] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa. Kegg for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38:D355 – D360, 2009.
- [10] M. Kanehisa, S. Goto, and M. Hirakawa. From genomics to chemical genomics: New developments in kegg. *Nucleic Acids Research*, 34:D354 – D357, 2006.
- [11] Neil D. Lawrence, Mark Girolami, Magnus Rattray, and Guido Sanguinetti. *Learning and Inference in Computational Systems Biology*. MIT Press, 2010.
- [12] Aziz Mithani. Evolutionary modelling and analysis of metabolic networks. *DPhil Thesis*, 2009.
- [13] Aziz Mithani, Gail M. Preston, and Jotun Hein. A stochastic model for the evolution of metabolic networks with neighbor dependence. *Bioinformatics*, 2009.
- [14] Nancy A. Moran. Microbial minimalism: Genome reduction in bacterial pathogens. *Cell*, 108:583 – 586, 2002.
- [15] A. I. Nilsson, S. Koskineniemi, S. Eriksson, E. Kugelberg, J. C. D. Hinton, and D. I. Andersson. Bacterial genome size reduction by experimental evolution. *Proceedings of the National Academy of Sciences, USA*, 102:12112 – 12116, 2005.
- [16] Howard Ochman and Liliana M. Davalos. The nature and dynamics of bacterial genomes. *Science*, 311:1730 – 1733, 2006.

- [17] Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature Biotechnology*, 28:245 – 248, 2010.
- [18] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, New York, USA.
- [19] Shai S. Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of *escherichia coli*. *Nature Genetics*, 31:64 – 68, 2002.
- [20] Tomer Shlomi, Yariv Eisenberg, Roded Sharan, and Eytan Ruppin. A genome-scale computational study of the interplay between transcriptional regulation and metabolism. *Molecular Systems Biology*, 3(101), 2007.
- [21] Ilya Shmulevich, Edward R. Dougherty, Seungchan Kim, and Wei Zhang. Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261 – 274, 2002.
- [22] S. A. Teichmann and M. M. Babu. Gene regulatory network growth by duplication. *Nature Genetics*, 36:492 – 496, 2004.
- [23] Chen-Hsiang Yeang and Martin Vingron. A joint model of regulatory and metabolic networks. *BMC Bioinformatics*, 7(332), 2006.