

SOFTWARE

Open Access

TIGER: Toolbox for integrating genome-scale metabolic models, expression data, and transcriptional regulatory networks

Paul A Jensen, Kyla A Lutz and Jason A Papin*

Abstract

Background: Several methods have been developed for analyzing genome-scale models of metabolism and transcriptional regulation. Many of these methods, such as Flux Balance Analysis, use constrained optimization to predict relationships between metabolic flux and the genes that encode and regulate enzyme activity. Recently, mixed integer programming has been used to encode these gene-protein-reaction (GPR) relationships into a single optimization problem, but these techniques are often of limited generality and lack a tool for automating the conversion of rules to a coupled regulatory/metabolic model.

Results: We present TIGER, a Toolbox for Integrating Genome-scale Metabolism, Expression, and Regulation. TIGER converts a series of generalized, Boolean or multilevel rules into a set of mixed integer inequalities. The package also includes implementations of existing algorithms to integrate high-throughput expression data with genome-scale models of metabolism and transcriptional regulation. We demonstrate how TIGER automates the coupling of a genome-scale metabolic model with GPR logic and models of transcriptional regulation, thereby serving as a platform for algorithm development and large-scale metabolic analysis. Additionally, we demonstrate how TIGER's algorithms can be used to identify inconsistencies and improve existing models of transcriptional regulation with examples from the reconstructed transcriptional regulatory network of *Saccharomyces cerevisiae*.

Conclusion: The TIGER package provides a consistent platform for algorithm development and extending existing genome-scale metabolic models with regulatory networks and high-throughput data.

Background

Constraint-Based Reconstruction and Analysis (COBRA) methods have allowed the study of metabolism on a genome-wide scale [1]. These models have been used to understand the interplay between environmental and genetic perturbations and the metabolic capabilities of an organism. Applications of COBRA methods have led to increased understanding in the fields of bioprocess optimization [2], pathogenicity [3], symbiosis [4], biofuel production [5], and human disease [6].

The Gene-Protein-Reaction relationship

COBRA models often contain two sets of biological information, a matrix of stoichiometric data for metabolic reactions, and a mapping between gene-encoding

enzymes and the reactions they catalyze (the gene-protein-reaction, or GPR, relationship). Predicting the metabolic capabilities of a COBRA model is possible with Flux Balance Analysis (FBA), a two-stage mathematical technique based on the observation that metabolic networks often display optimal dynamics [7]. In the first stage of FBA, genes in the modeled organism are classified as either "on" or "off" to create an *in silico* genetic state. Turning genes "off" can be used to simulate significant reductions in expression levels or complete knock-outs. The GPR for each reaction, represented as a binary rule, determines if a sufficient collection of proteins (isozymes, enzymatic subunits, etc.) is present for the reaction to carry flux. All reactions with satisfied GPR rules are collected into a stoichiometric matrix. The second stage of FBA uses linear programming to calculate a thermodynamically-feasible, mass-balanced flux distribution that maximizes the flux through an

* Correspondence: papin@virginia.edu
Department of Biomedical Engineering, University of Virginia, Charlottesville, VA 22908, USA

objective reaction. The objective reactions used in FBA vary among organisms, ranging from ATP maintenance to biomass production [8]. By assuming that the fluxes through a metabolic network have evolved to maximize an objective, FBA eliminates the need for detailed kinetic information for each of the thousands of reactions in a complete metabolic reconstruction.

The GPR rules do not always describe a one-to-one mapping between genes and reactions (where one gene encodes a complete enzyme that independently catalyzes one reaction). For example, a COBRA reconstruction for the yeast *Saccharomyces cerevisiae* [9] contains 1266 metabolic reactions; 231 (18.3%) of these reactions have complex GPR associations. The most complex GPR in this model involves the products of 18 open reading frames. The entire set of GPR rules contains 340 instances of isozyme-like behavior (two proteins both able to fully catalyze a reaction) and 279 different complexes of protein subunits.

Because of the complexity of the GPR mappings, early extensions to FBA were reaction, rather than gene, centric. For example, the OptKnock [2] algorithm removed reactions from a FBA model to design a strain of *E. coli* with optimal production of a metabolic byproduct. Ideally, OptKnock would operate by removing genes, not reactions, since it is not straightforward to independently remove reactions from a biological system without genetic manipulations. An optimization using genes as decision variables would require a method for encoding the GPR logic into a set of linear inequalities. This encoding was developed as SR-FBA [10] using a mixed integer optimization approach for GPR logic and other Boolean regulatory rules. An SR-FBA-based approach was later used to develop OptORF, a method to design microbial strains through gene knockouts and overexpression [11]. Other gene-centric, FBA-related algorithms have been developed, each using a variation of the SR-FBA method [12-14]. However, a general software platform for coupling GPR rules of arbitrary complexity with a COBRA model using mixed integer programming was not available. Such a tool would speed the development of new algorithms by removing the need for researchers to re-implement this complex process.

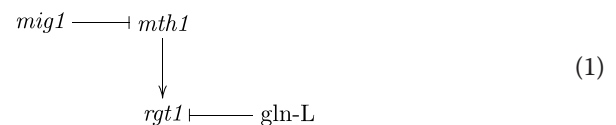
Transcriptional regulatory networks

The accuracy of COBRA models has been improved through the addition of transcriptional regulatory networks (TRNs) [15,16]. The TRN is a set of rules that relate the expression states of metabolic genes to various genetic and environmental cues. Because of the paucity of kinetic details available to describe these relationships, genome-scale models often represent gene expression and environmental cues in a binary, “on” or “off”

format. This approach allows TRNs to be described with Boolean logic.

The first genome-scale TRNs were applied to models of *Escherichia coli* [15] and *Saccharomyces cerevisiae* [16] metabolism. The rules were written in standard Boolean format, where each Boolean variable is given by an explicit function of the other variables. This method creates two significant problems. First, the TRN uses the absence or presence of metabolites in the extracellular environment to calculate which genes (and, subsequently, reactions) will be active. However, certain metabolic pathways secrete byproducts into the extracellular space, thereby changing the environment. Studies with the *E. coli* and *S. cerevisiae* TRNs used an iterative approach [17] - applying the TRN to the metabolic network in a starting environment, determining which metabolites would be secreted, and then repeating the process in the new environment until the environment no longer changes between iterations. A more straightforward approach would be to solve the TRN and metabolic networks simultaneously by formulating both problems in a single optimization.

A second obstacle with TRN integration is that the explicit rule formulation used by previous studies [17] can over-constrain the metabolic model. (In explicit rules, each gene’s state can be calculated unambiguously from the state of all other genes and metabolites.) Consider the following subnetwork of the iMH805 TRN for *S. cerevisiae* [16]:



Transcription factor *mth1* is repressed by *mig1* and promotes expression of *rgt1*. Extracellular L-glutamine (gln-L) represses *rgt1* expression. The original iMH805 study required this set of constraints be described with the following set of explicit rules [16]:

$$\begin{aligned}
 & \text{not } MIG1 \Leftrightarrow mth1 \\
 & MTH1 \text{ and } (\text{not } gln - L) \Leftrightarrow rgt1
 \end{aligned}$$

An implicit representation of (1) is

$$\begin{aligned}
 & MIG1 \Rightarrow \text{not } mth1 \\
 & MTH1 \Rightarrow rgt1 \\
 & gln - L \Rightarrow \text{not } rgt1
 \end{aligned}$$

The three transcription factors and one metabolite in these rules can be arranged in $2^4 = 16$ possible states. As shown in Table 1, only four of the sixteen states are feasible for the explicit rules. The implicit formulation of the same system allows four new states and makes one of the explicit states infeasible. This example

Table 1 Feasible states for explicit and implicit rules

State				Feasibility	
<i>mig1</i>	<i>mth1</i>	<i>rgt1</i>	<i>gln-L</i>	Explicit	Implicit
0	0	0	0		.
0	0	0	1		.
0	0	1	0		.
0	0	1	1		.
0	1	0	0		.
0	1	0	1	.	
0	1	1	0	.	.
0	1	1	1	.	.
1	0	0	0	.	.
1	0	0	1	.	.
1	0	1	0	.	.
1	0	1	1	.	.
1	1	0	0	.	.
1	1	0	1	.	.
1	1	1	0	.	.
1	1	1	1	.	.

Feasible states for explicit and implicit rules describing the transcriptional regulatory network (1). Rules are taken from the *S. cerevisiae* regulatory network model iMH805 [16]

illustrates that two mathematical descriptions of the same biological process can lead to distinct model predictions. The model developer should be free to choose the rule formulation that best encompasses the underlying biology. However, implicit rules require simultaneous solution with a metabolic model and are often more difficult to parse into a mixed integer linear program. As a result, previous TRN integration studies have relied solely on explicit rules to describe regulatory interactions [17]. A software platform that can correctly parse both explicit and implicit rules would ease the development of large TRN models.

Objectives

Software suites have been developed to enable COBRA analyses. Packages such as CellNetAnalyzer [18], the BioMet Toolbox [19], and the COBRA Toolbox [20] implement several useful algorithms for studying COBRA models and TRNs. However, to date, no single software platform has been developed to 1.) convert COBRA models and TRNs into integrated optimization problems, 2.) analyze these integrated models with existing algorithms to incorporate high-throughput expression data, and 3.) allow users to easily develop new algorithms for the integrated models. To overcome these limitations, we present a Toolbox for Integrating Genome-scale metabolism, Expression, and Regulation (TIGER). TIGER automatically converts a list of implicit or explicit GPR and TRN rules into a set of linear inequalities; these equations are integrated with an existing COBRA model. The software allows rules to be

written in a generalized Boolean format, enabling TRN logic to more accurately reflect the underlying biology. We demonstrate how this increased expressivity can overcome inconsistencies in existing TRN models. We will also show how TIGER simplifies the development of gene-centric extensions to FBA by improving three algorithms for integrating high-throughput expression data with a COBRA model.

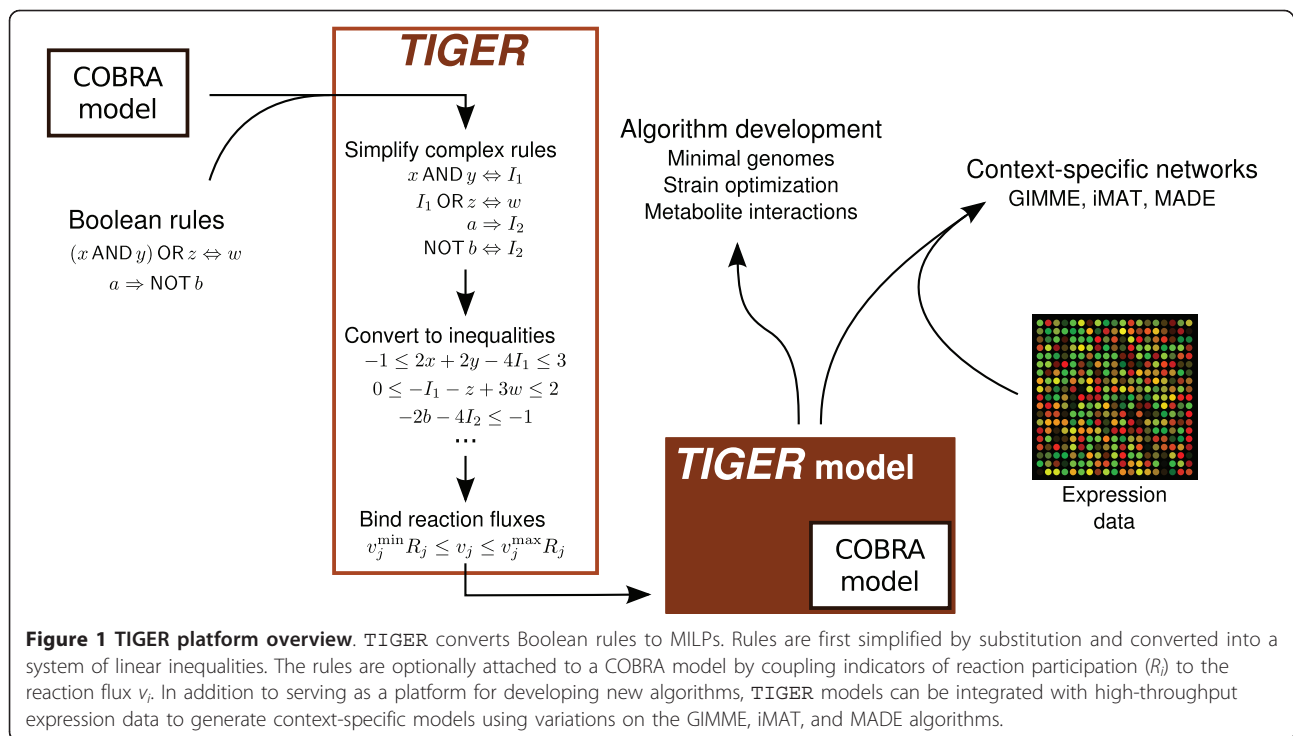
Implementation

The primary functions of TIGER are shown in Figure 1. TIGER converts a GPR and additional regulatory rules into an equivalent mixed integer linear program (MILP). The MILP constraints are added to a COBRA metabolic model to create a TIGER model that combines metabolism, GPR associations, and transcriptional regulation. This integrated model serves as a platform for applying many gene-centric extensions to FBA, including algorithms that incorporate “omics” data for model refinement. In this section, we describe how the rules parsed by TIGER are constructed, and how they are converted to an MILP. Sample files depicting a COBRA model, GPR, and TRN are provided in the “test/samples” directory of the TIGER distribution.

Creating rules

The GPR uses Boolean logic to describe the nonlinear relationship among genes, their protein products, and the reactions they catalyze. Examples of GPR relationships appear in Figure 2. The majority of metabolic reactions adhere to a one gene, one enzyme, one reaction relationship, as demonstrated by *gene a*, enzyme A, and reaction 1. If *gene a* is expressed, FBA assumes that reaction 1 can carry any physiologically feasible flux. Some enzymes, such as enzyme B, can catalyze two or more separate reactions. Other reactions, such as reaction 4, require two enzymes (C and D) to carry any flux. Enzymes C and D are commonly protein subunits of an enzymatic complex; if either is absent, no substrate conversion is possible. Genes *e* and *f* encode isozymes; either protein can independently catalyze reaction 5.

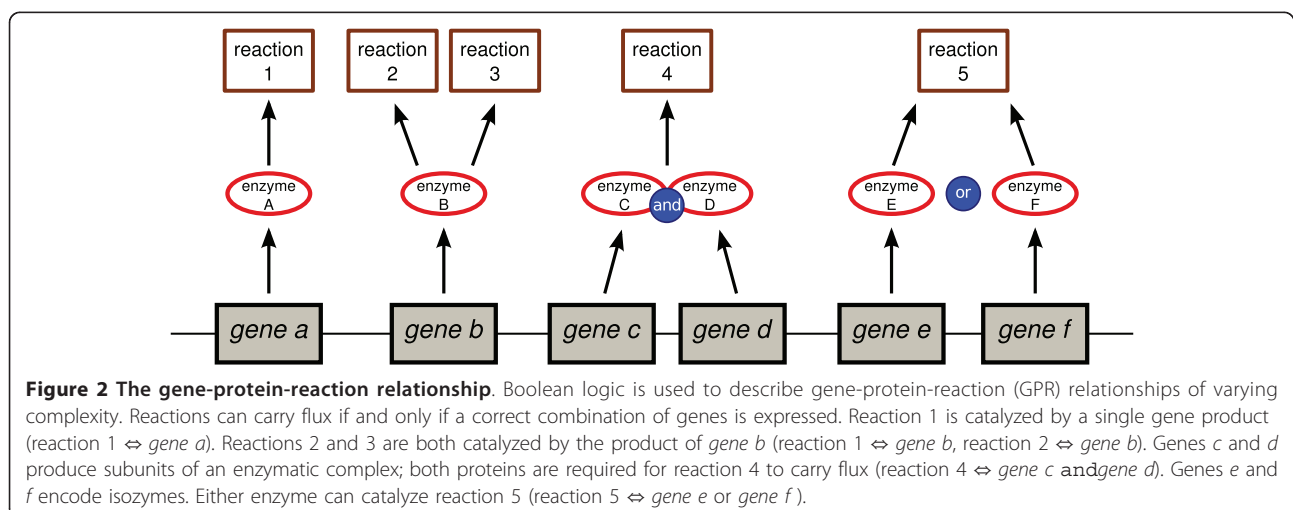
These GPR relationships can be described as a Boolean *expression* using the standard operators and and or. For example, a reaction that is catalyzed by either of two isozymes, the second of which is composed of two subunits, would have a GPR of the form “ $isozyme_1 \text{ or } (isozyme_{2a} \text{ and } isozyme_{2b})$ ”. Two expressions are joined with an implication operator (\Rightarrow or \Leftrightarrow corresponding to “if” and “if and only if”), to form a *rule*. For GPR associations, rules are formed as “GPR \Leftrightarrow reaction”, where *reaction* is an indicator variable that constrains the flux through a reaction to be zero when the GPR expression is false.



TIGER expressions allow additional features to describe logical relationships that are more complex than those typically found in GPRs. The `not` operator allows logical negation, which is often used to construct rules for transcriptional repression. Expressions can also contain *conditionals* that compare the numerical values of individual variables. If a gene g was known to be expressed when glucose uptake is greater than 10 flux units, this relationship could be represented by the rule “ $\text{glc_ex} > 10 \Rightarrow g$ ”, where “ glc_ex ” is the glucose exchange reaction in the metabolic model. Any two expressions of

arbitrary complexity can be combined as a rule and parsed by TIGER. The grammar used by TIGER for rules was designed to resemble logical operations in common programming languages and to be compatible with the GPRs of widely-used COBRA models. A complete description of the TIGER syntax appears in Additional File 1.

Some transcriptional regulators, such as the response of *crp* to *cAMP* in *E. coli*, display multiple levels of activity and cannot be easily described with Boolean logic [15]. Rather than require users to create several



variables describing each state of activation, TIGER allows multilevel variables. If a transcription factor t activates target genes g_{low} at low levels of expression and g_{high} at high levels of expression, then this relationship could be described with the rules

$$(t = 1) \Rightarrow g_{\text{low}} \quad (2)$$

$$(t = 2) \Rightarrow g_{\text{high}} \quad (3)$$

where $t = 0, 1,$ and 2 corresponds to no, low, and high expression. Logical operators have a different interpretation when applied to multilevel variables. If proteins x and y form a promoter complex for expression of gene z , then the corresponding rule for z expression would be “ x and $y \Rightarrow z$ ”, since both x and y are required for z transcription. If x and y were multilevel, one would assume that z expression would be proportional to the promoter subunit in lower abundance, since this species would limit the amount of complete promoter complex that could be formed. Thus, the and operation often corresponds to a minimization:

$$x \text{ and } y \equiv \min\{x, y\} \quad (4)$$

The or operation would be used in situations where either factor can independently induce expression. In this situation, the species in higher abundance determines the target gene’s transcription level. TIGER implements the multilevel or as a maximization:

$$x \text{ or } y \equiv \max\{x, y\} \quad (5)$$

The not operator can have two interpretations when applied to multilevel variables:

$$\text{not } x \equiv \begin{cases} x > 0 & \text{pseudo - binary} \\ \bar{x} - x & \text{inversion, } x \in \{0, \dots, \bar{x}\} \end{cases} \quad (6)$$

where $x \in \{0, \dots, \bar{x}\}$. The first case (pseudo-binary) regards any nonzero value as true, regardless of the number of levels the variable may occupy. The second case (inversion) requires that the value of x and the quantity $\text{not } x$ always sum to the maximum value that x can occupy. In this case, $\text{not } x$ is a measure of how far x is from its upper bound. Users are able to select the pseudo-binary or inversion representation depending on which interpretation is a better approximation of the biological context. For example, consider a gene/repressor relationship $R \Rightarrow \text{not } G$, where the repressor R can take on three biologically distinct levels - “off,” “low,” and “high”. If both “low” and “high” levels of R prevent any expression of G , then the pseudo-binary not operator would be appropriate as G is off whenever R is not “off”. However, if G also has the same three levels of expression, then the

inversion interpretation of the not operator is more appropriate. This choice implies that

$$R = \text{off} \rightarrow G = \text{high} \quad (7)$$

$$R = \text{low} \rightarrow G = \text{low} \quad (8)$$

$$R = \text{high} \rightarrow G = \text{off} \quad (9)$$

Any variable in a TIGER model can be declared with multiple levels. Such declarations are made when rules are added to a model using the `add_rule` function in two ways: 1.) setting the default upper bound to all variables to any integer greater than one, or 2.) providing a list of variable names and a set of upper and lower bounds.

Rule simplification

Simple Boolean rules can be represented by systems of linear inequalities of integer variables [21]. A general Boolean rule can be converted by the following procedure to a set of simple rules before conversion to an MILP.

We define an “atomic” expression as either a variable (x) or a negated variable ($\text{not } x$). If a not operator appears before an expression that is not atomic, TIGER applies DeMorgan’s laws to move the negation onto atomic expressions (e.g., $\text{not } (x \text{ and } y)$ becomes the equivalent expression $(\text{not } x) \text{ or } (\text{not } y)$). A simple rule then conforms to one of the following patterns

$$x \quad (\Rightarrow \mid \Leftrightarrow) \quad z \quad (10)$$

$$x \text{ and } y \quad (\Rightarrow \mid \Leftrightarrow) \quad z \quad (11)$$

$$x \text{ or } y \quad (\Rightarrow \mid \Leftrightarrow) \quad z \quad (12)$$

$$x \langle op \rangle y \quad (\Rightarrow \mid \Leftrightarrow) \quad z \quad (13)$$

where $x, y,$ and z are atomic, and $\langle op \rangle$ is a conditional operator ($\leq, \geq,$ etc.). Non-simple rules are converted to simple rules through a series of recursive substitutions. For example, the rule

$$(x \text{ or } (\text{not } y)) \text{ and } z \Rightarrow w \quad (14)$$

is not simple, since the expression $x \text{ or } (\text{not } y)$ is not atomic. By defining an indicator variable I , which is true if and only if the expression $x \text{ or } (\text{not } y)$ is true, equation (14) can be written as two simple rules:

$$x \text{ or } (\text{not } y) \Leftrightarrow I \quad (15)$$

$$I \text{ and } z \Rightarrow w \quad (16)$$

The bounds of I are determined by the bounds of x and y . If $x \in \{0, \dots, \bar{x}\}$ and $y \in \{0, \dots, \bar{y}\}$, then $I \in \{0, \dots, \bar{I}\}$, where

$$\bar{I} = \begin{cases} \max\{\bar{x}, \bar{y}\} & \text{for } x \text{ or } y \\ \min\{\bar{x}, \bar{y}\} & \text{for } x \text{ and } y \end{cases} \quad (17)$$

Thus, if x and y are binary, $\bar{x} = \bar{y} = 1$, so I is binary as well.

TIGER applies the above substitutions recursively, creating indicator variables as necessary until all rules are simple. Each simple rule is converted to a set of linear inequalities that are added as constraints to the model structure. If a variable name already appears in the model, TIGER assumes that these variables represent the same quantity and thus allows new rules to be added to an existing model without recompiling previous rules. At the same time, TIGER creates variables to substitute for negated variables. For efficiency, TIGER ensures that only one negated variable is created for each original variable, regardless of the number of times the negated expression appears in the set of simple rules. Details of the conversion between simple rules and inequalities, along with methods for handling conditionals, are provided in Additional File 1.

Reaction coupling

If the GPR expression for a reaction is not satisfied, the reaction is not allowed to carry flux. To enforce this relationship during an optimization, a set of discrete variables R_i are defined, where $R_i = 0$ if the GPR for reaction i is not satisfied, and $R_i > 0$ otherwise. To enforce the GPR's effect on flux, TIGER adds the constraint

$$v_i^{\min} R_i \leq v_i \leq v_i^{\max} R_i \quad (18)$$

where v_i is the flux through the i th reaction, with lower and upper bounds v_i^{\min} and v_i^{\max} .

Model structure

TIGER models are represented as Matlab structures. The layout of this structure is shown in Figure 3. The structure contains fields `obj`, `A`, `b`, `lb`, and `ub` that correspond to the values in the following MILP problem:

$$\begin{aligned} & \min \text{obj}'x \\ & \text{subject to} \\ & Ax(\leq | = | \geq)b \\ & lb \leq x \leq ub \end{aligned}$$

The type of (in)equality for each constraint in `A` is determined by the character vector `ctype`. The type of variable for each entry in `x` is specified by the field

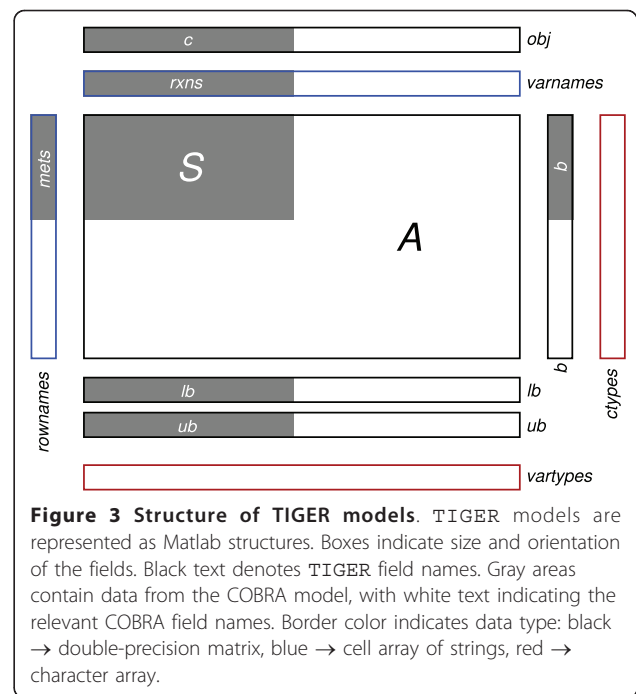


Figure 3 Structure of TIGER models. TIGER models are represented as Matlab structures. Boxes indicate size and orientation of the fields. Black text denotes TIGER field names. Gray areas contain data from the COBRA model, with white text indicating the relevant COBRA field names. Border color indicates data type: black → double-precision matrix, blue → cell array of strings, red → character array.

`vartype`, where 'c', 'b', and 'i' denote continuous, binary, and general integer variables. Reaction fluxes are continuous variables, while all other variables are either binary or integer depending on the corresponding upper bound. The fields `rownames` and `vartypes` contain descriptive names of the constraints and variables, stored as cell arrays of strings. Functions in TIGER allow variables to be interchangeably referenced by their name, column index, or through Matlab's logical indexing features.

The format for TIGER models is designed for compatibility with the model structure for the COBRA Toolbox [20]. TIGER can use a COBRA Toolbox model as a starting point for converting a genome-scale reconstruction; therefore, any model in a file format supported by the COBRA Toolbox (SBML, Simpheny, etc.) can be converted to a TIGER model.

Accessing the MILP solver

TIGER uses a custom Matlab class `CMPI` (Common Mathematical Programming Interface) to create and solve mathematical programming problems. `CMPI` defines a consistent structure for MILP (and mixed-integer quadratic programming, MIQP) problems, providing independence from the underlying MILP solver software. TIGER currently supports the CPLEX, Gurobi (via Gurobi MEX), and GLPK (via GLPK Mex) software packages, all of which are freely available for academic use. Porting TIGER to use a new solver requires modifying only the `CMPI` method `solve_mip` to specify the new interface.

CMPI also provides a standardized method for configuring common solver parameters (maximum solution time, optimality and feasibility tolerances, etc.).

Previous work has indicated that the computation time of some FBA-related algorithms, such as Flux Variability Analysis [22], can be reduced by saving information about the problem structure between calls to the MILP solver [23]. CMPI provides a method, `solve_multiple_milps`, to preserve the solver state between successive calls to the CPLEX optimizer and reduce runtime in this manner. (Gurobi and GLPK currently do not support this feature in their Matlab interfaces.) If the CPLEX optimizer is not installed, CMPI will automatically make successive calls to the installed optimizer. While this removes the potential speed increase from using solver restarts, it allows TIGER code to remain solver independent and portable.

Using TIGER

TIGER source code and installation instructions are available online at <http://bme.virginia.edu/csbl/downloads/> or <http://csbl.bitbucket.org/tiger>. The version of TIGER used for the examples in this study is included as Additional File 2. All functions in the toolbox are documented using Matlab's "help" facilities. Complete documentation and a step-by-step tutorial are also available on the TIGER website. The software includes a testing suite to verify the installation. These tests contain examples that build a TIGER structure from a simple COBRA model, add a set of TRN rules, call a MILP solver, and display the solution.

Results and Discussion

Refining integrated models for *Saccharomyces cerevisiae*

TIGER was used to couple the 1266 reactions in iND750 [9], a genome-scale model of *Saccharomyces cerevisiae* metabolism, with 750 metabolic genes. The resulting TIGER model contained 4498 constraints in 3214 variables. A model of *S. cerevisiae* transcriptional regulation, iMH805 [16], was added. The additional 805 rules contributed 1057 constraints and 562 variables to the TIGER model. The conversion took 53.66 s for iND750 and 20.31 s to add the TRN using an Intel 3.2 GHz i7-quad core processor running Linux.

As mentioned above, previous methods for integrating TRNs involve an iterative process, alternating between calculating gene states from a given environment and determining an environment based on metabolic byproducts [17]. However, the multiple layers of transcriptional regulation may require several iterations of this method to reach a stable gene state. The number of iterations to reach a stable state varies by environment and cannot easily be determined *a priori* [24]. In fact, some feedback mechanisms in TRNs may lead

to a stable cycle of gene activation/inactivation rather than a single gene state. TIGER solves the TRN and FBA problems simultaneously, so the resulting gene state is always stable (or an optimal state inside a stable cycle).

Applying large-scale TRNs to COBRA models may result in infeasible models, i.e., models unable to produce any biomass. This is often due to a small number of rules that turn off reactions that are essential for biomass production. Previous work has developed techniques for finding which rules create the model infeasibility [13]. TIGER includes the function `find_infeasible_rules` to identify rules that prevent feasible solutions to the resulting MILP. Given a model and a set of rules that prevent a feasible solution, `find_infeasible_rules` creates a MILP that preserves the logic of the rules but allows each rule to be artificially satisfied. The objective of this MILP is to minimize the number of rules that must be artificially satisfied while finding a feasible solution for the model. (Details of this process are available in Additional File 1.)

Analysis by TIGER reported that the combined iND750/iMH805 metabolic and TRN network was unable to produce biomass under aerobic conditions in a glucose minimal media. Since *S. cerevisiae* is well-known to grow in this environment, we used TIGER's `find_infeasible_rules` function to identify the following three rules that prevented growth:

$$\text{O}_2[\text{e}] \text{ or } (\text{not ROX1}) \Leftrightarrow \text{hap1} \quad (19)$$

$$\text{O}_2[\text{e}] \text{ and HAP1} \Leftrightarrow \text{rox1} \quad (20)$$

$$\text{glucose}[\text{e}] \text{ and HAP1 and } (\text{not ROX1}) \Leftrightarrow \text{erg11} \quad (21)$$

The product of gene *erg11*, lanosterol 14 α -demethylase, is an essential enzyme for the production of ergosterol, a main sterol in *S. cerevisiae* [25]. This gene is essential in the iND750 metabolic model and must remain "on" during aerobic growth on glucose. However, the stable gene state for *erg11* in the above rules is always "off" after three iterations, as described in Table 2. Because the iMH805 study used the results of the second iteration as the final gene state, this inaccuracy was unnoticed.

Rule (21) was originally derived from Turi & Loper [25]. Re-examination of this manuscript revealed that while ROX1 represses *erg11*, complete repression is only observed under low oxygen conditions. To incorporate these findings, we create an indicator "high o₂" that is true if and only if the cell uptakes more than 10% of the maximum oxygen consumption rate. This relationship is expressed as

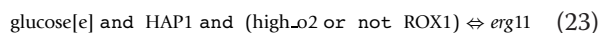
$$\text{high_o2} \Leftrightarrow \text{EX_O2}(\text{e}) < -0.244 \quad (22)$$

Table 2 Gene states for *erg11* regulation

	Start	Iteration				Rule
		1	2	3	4	
O ₂ [e]	1	1	1	1	1	
glucose [e]	1	1	1	1	1	
<i>hap1</i>	0	1	1	1	1	O ₂ [e] or not ROX1
<i>rox1</i>	0	0	1	1	1	O ₂ [e] and HAP1
<i>erg11</i>	1	1	1	0	0	O ₂ [e] and HAP1 and (not ROX1)
<i>erg11</i>	1	1	1	1	1	O ₂ [e] and HAP1 and (high_o2 or not ROX1)

To start, all transcription factors are assumed to be “off”, and all metabolic genes are “on”. The extracellular environment contains only glucose (glucose [e]), oxygen (O₂[e]), and essential salts and minerals. Each iteration calculates the next gene state based on the current state of metabolites in the environment and transcription factors. After three iterations, expression of *erg11* turns off, unless a modified rule is used that account for high oxygen uptake. The modified rule (shown below the double line) is more consistent with the results in Turi & Loper [25] Rules are taken from the *S. cerevisiae* regulatory network model iMH805 [16]

where “EX_o2(e)” is the iND750 name for the oxygen exchange reaction, and the maximum oxygen uptake rate for growth on glucose is 2.44 mol/(g dry cell weight)/h (negative flux through exchange reactions indicate uptake by the cell). Rule (21) for *erg11* expression was re-written to only exhibit ROX1 repression under low oxygen conditions:



The set of refined rules (20,21,23) reproduces the correct growth phenotype in aerobic glucose conditions. This example demonstrates a three-step procedure for refining existing TRN models using TIGER: 1.) apply the existing TRN to a COBRA model, 2.) use the `find_infeasible_rules` function to identify rules that cause the model to differ from a known phenotype, and 3.) re-examine the evidence for these rules and make appropriate modifications. As shown in the previous example, new biological information can often be incorporated into existing rules using TIGER’s support for complex logical expressions.

Improved methods for expression data

Coupling the GPR with a metabolic model is a starting point for several algorithms designed to refine metabolic models by integrating high-throughput gene or protein expression data. The TIGER package contains implementations of three of these methods.

GIMME

GIMME was designed to generate context-specific metabolic networks by designating each reaction as “on” or “off” [26]. Given expression data and a minimum expression threshold, GIMME calculates a normalized gene “score” for each reaction by averaging the

expression values of all genes that appear in the GPR for the reaction. Reactions with a score below a cutoff are turned off; only reactions scoring above the cutoff are allowed to carry flux. Because this thresholding does not guarantee a functioning model, an optimization problem is formed to minimize the number of “off” reactions that must carry flux when the model produces a minimum objective flux.

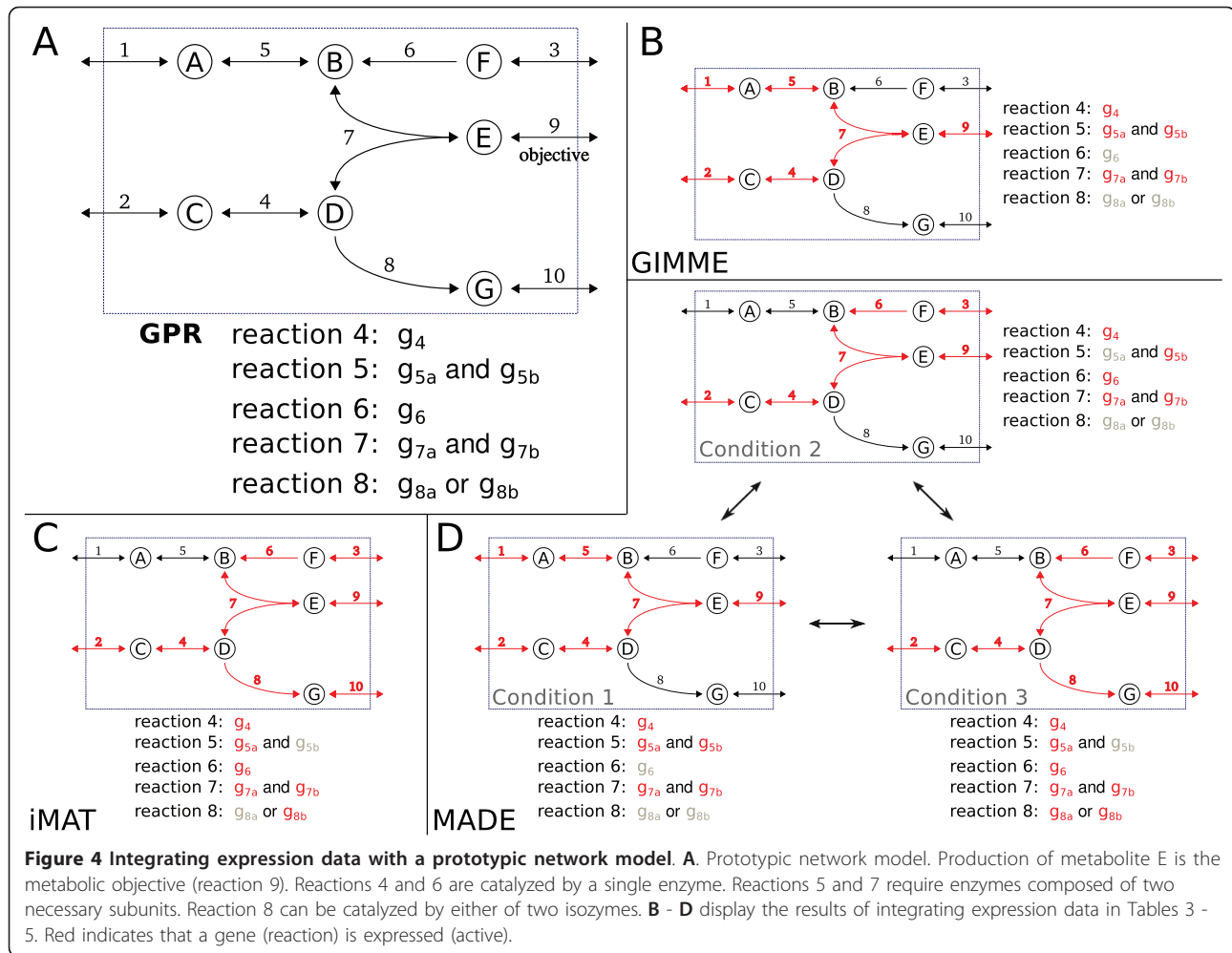
The GIMME algorithm is an excellent candidate for conversion to a gene-centric approach. Rather than average the gene expression values for each reaction, a simpler approach is to turn genes “on” or “off” if their expression is above or below a threshold. Using an integrated GPR/metabolic model, an optimization problem could re-activate “off” genes to allow the network to produce an objective flux. TIGER provides a `gimme` function that implements this gene-centric approach. Similar to the original algorithm, TIGER’s GIMME uses the distance below the expression threshold as a weight when selecting genes for re-activation. In addition to removing discrepancies caused by the averaging of gene expression values over each reaction, TIGER’s integrated approach allows GIMME to identify each gene as either “on” or “off” in the resulting network (GIMME originally reported only the state of each reaction).

We applied GIMME to the prototypic network shown in Figure 4a. In this network, three metabolites, A, C, and F, are transported into the system and undergo chemical transformation into products E and G. The metabolic objective for this system is the production of E. The GPR associations are also shown in Figure 4a. Reactions 4 through 8 are gene associated. Reactions 5 and 7 both require two enzymatic subunits to carry flux, and reaction 8 can carry flux through either of two isozymes.

The expression data applied to this network are shown in Table 3. Using simple thresholding at a threshold expression level of 10 does not produce a functional model. As shown in Table 3 and Figure 4b, GIMME reactivates the genes *g5a* and *g7b*. Because reaction 7 is the only method of producing metabolite E (the metabolic objective), this reaction must be active and both subunits (*g7a* and *g7b*) must be expressed. Production of E also requires metabolite B and either reaction 5 or reaction 6. The expression data showed that both genes *g5a* and *g6* were below the expression threshold; GIMME chose to activate *g5a* since its expression was closer to the threshold.

iMAT

Shlomi, et al. [27], introduced the iMAT algorithm to create tissue-specific models of mammalian metabolism. Briefly, reactions or genes were classified as having either low, regular, or high activity using information from several databases. An optimization problem was formulated such that



$$v_i \begin{cases} \geq \varepsilon & \text{if reaction } i \text{ has high activity} \\ \leq \varepsilon & \text{if reaction } i \text{ has low activity} \end{cases} \quad (24)$$

for each reaction flux v_i and some small flux ε . The optimization attempted to preserve the reaction classifications while enforcing that the resulting set of reaction fluxes be

feasible (mass balanced). The mass-balance approach attempts to yield functional models in multicellular organisms that lack a clearly defined metabolic objective.

TIGER includes a gene-centric version of the iMAT algorithm. Because TIGER allows multilevel variables, each gene in the GPR is allowed to occupy a state of low (0), medium (1), or high (2) activity. The multilevel

Table 3 Expression data for GIMME example

Gene	Expression	Simple thresholding	GIMME
g_4	13	1	1
g_{5a}	6	0	1
g_{5b}	12	1	1
g_6	2	0	0
g_{7a}	18	1	1
g_{7b}	6	0	1
g_{8a}	2	0	0
g_{8b}	4	0	0

Expression values for genes of the network shown in Figure 5a. "Simple thresholding" is the gene state after a simple threshold of 10 is applied to the expression values. "GIMME" is the gene state calculated by GIMME that produces a functioning model. All expression values are arbitrary units.

Table 4 Activity levels for iMAT example

Gene	Activity level
g_4	medium
g_{5a}	low
g_{5b}	low
g_6	high
g_{7a}	high
g_{7b}	high
g_{8a}	medium
g_{8b}	high

Classification of genes according to "low", "medium", or "high" activity for iMAT. Genes correspond to the network shown in Figure 5a. iMAT produces the integrated network shown in Figure 5c.

operators in the GPR then combine the gene expression values to create a reaction indicator with the same three levels. Rules are added to enforce the constraints in equation (24). Additionally, if a metabolic objective is available for the organism, the flux through the corresponding objective reaction can be constrained above a minimum value.

iMAT was used to integrate the activity data in Table 4 with the network model in Figure 4a. The results are shown in Figure 4c. Notice that the high activity of gene g_{sb} requires that reactions 8 and 10 carry flux. Because production of metabolite G is not a part of the system's metabolic objective, reaction 8 would most likely not carry flux in the FBA flux distribution for this network; the activity seen in this side pathway is enforced by the iMAT constraints.

MADE

Metabolic Adjustment by Differential Expression (MADE) removes the need for a pre-defined "on"/"off" threshold when integrating expression data [12]. Instead, MADE uses the differential expression between two or more conditions to determine which genes or proteins are likely to be "on" or "off". If a gene increases significantly between conditions 1 and 2, MADE attempts to turn the gene "off" in condition 1 and "on" in condition 2. The expression data are mapped while ensuring model functionality in all conditions, and the statistical significance of the expression changes are used to prioritize discrepancies.

The TIGER implementation of MADE offers two improvements. First, genes are allowed to be multilevel

instead of binary. The user is allowed to define a mapping between the multiple levels of gene expression and the flux constraints for the corresponding reactions. Second, TIGER MADE allows comparisons among states that do not appear as a linear sequence. The original MADE algorithm used a series of n conditions C_1, C_2, \dots, C_n , and $n - 1$ sets of expression data describing the changes $C_1 \rightarrow C_2, C_2 \rightarrow C_3, \dots, C_{n-1} \rightarrow C_n$. TIGER MADE allows any number of connections between the n states.

A conceptual example of TIGER MADE's capabilities is presented in Figure 5. The goal in this example is to use expression data from five bacterial strains to develop strain-specific metabolic models. Strains A and B were evolved from the same parent strain P. Strain A later gave rise to two additional strains, A1 and A2. A matrix containing the differential expression between each strain and its predecessor is calculated. A transition matrix T is defined as follows: if $T(i, j) = k$, then the k th column of the differential expression matrix was calculated between strains i and j . Values greater than one in this column indicate that expression is higher in strain j than in strain i . These two matrices are used by TIGER MADE to create a single optimization problem. The result is a functional gene state for each condition that maximizes the significant changes in gene expression between each strain and its parent.

Figure 4d shows the results of applying MADE on three network models simultaneously. Each resulting model is functional and captures the most significant

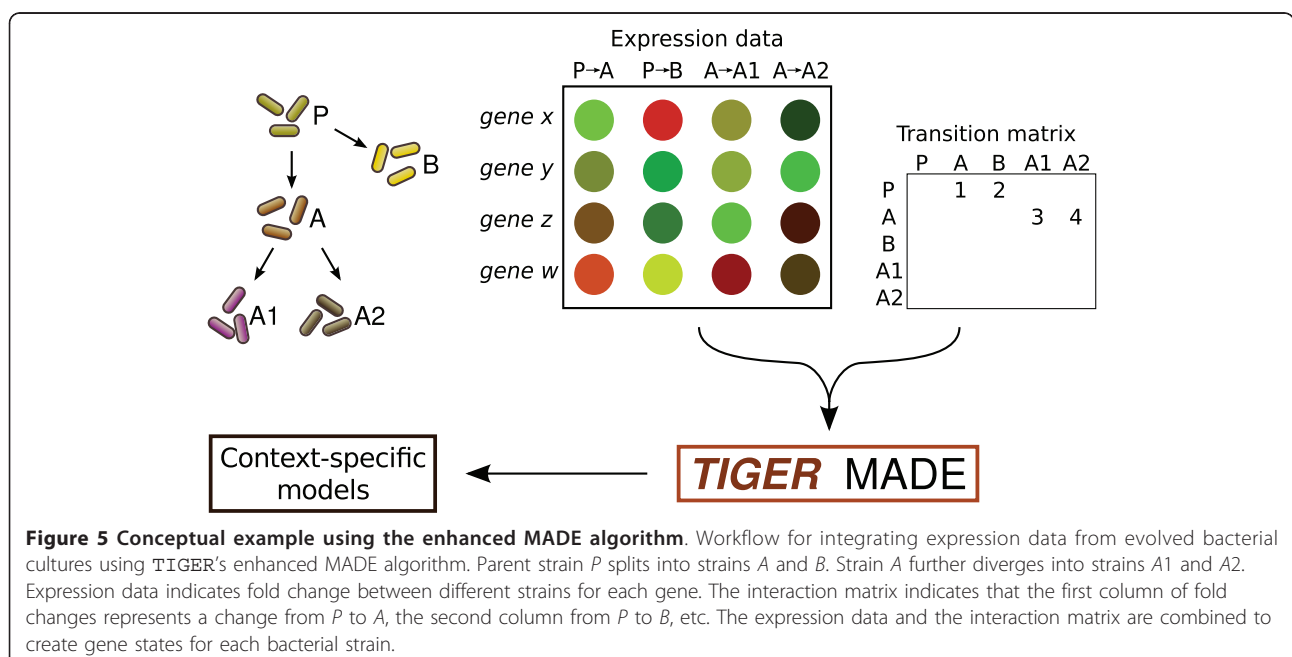


Table 5 Expression data for MADE example

Gene	Fold change						MADE states		
	1 → 2	P	2 → 3	P	3 → 1	P	1	2	3
<i>g₄</i>	2.1	0.68	3.0	0.08	0.7	0.15	1	1	1
<i>g_{5a}</i>	0.5	0.45	8.0	0.44	0.8	0.22	1	0	1
<i>g_{5b}</i>	1.1	0.07	0.1	0.49	2.2	0.03	1	1	0
<i>g₆</i>	3.6	0.05	1.6	0.22	0.4	0.48	0	1	1
<i>g_{7a}</i>	1.4	0.38	4.2	0.40	1.8	0.43	1	1	1
<i>g_{7b}</i>	2.3	0.04	0.4	0.19	0.3	0.12	1	1	1
<i>g_{8a}</i>	1.3	0.40	7.3	0.21	0.2	0.45	0	0	1
<i>g_{8b}</i>	0.9	0.89	4.1	0.83	0.1	0.28	0	0	1

Fold change and *P*-values for the MADE example. Column "1 → 2" and indicates the fold change in expression between conditions 1 and 2; *P* is the *P*-value for this change. "MADE states" are the binary gene states returned by the algorithm. The states maximize the transitions in the expression data while retaining model functionality in each condition. Gene names correspond to the network in Figure 5a. Results are also depicted in Figure 5d.

differences described the fold changes listed in Table 5. Notice that despite significant changes in enzyme expression for reactions 4 and 7, these reactions remain "on" in all models because they are required to produce metabolite E, the metabolic objective.

Source code for all of the examples in Figure 4 are available in the file `test/gimme_imat_made_examples.m` of the TIGER distribution.

Conclusions

We have presented TIGER, a software platform for converting generalized Boolean and multilevel rules to mixed-integer linear programs, and coupling these rules to genome-scale models of metabolism. The flexibility of TIGER's generalized rule format allows for a more accurate description of biological processes such as catalysis by isozymes and multi-meric proteins, metabolic flux control, and transcriptional regulation. These features were used to identify and correct inconsistencies within an existing TRN model of *Saccharomyces cerevisiae*. We have also demonstrated how TIGER can be used as a starting point for implementing and improving existing algorithms for genome-scale analysis.

In addition to adding implementations of other gene-centric algorithms to TIGER, we are exploring methods to improve the solution efficiency of the generated MILP. Possible strategies include exploiting indicator constraints, specially-ordered-sets (SOS), and other solver optimizations through CMPI.

Availability and requirements

Project name: TIGER

Project home page: <http://bme.virginia.edu/csbl/downloads> or <http://csbl.bitbucket.org/tiger>

Operating system: Platform independent

Programming language: Matlab

Other requirements: Matlab v7.0 or higher, a Mixed-Integer Linear Programming solver (e.g. CPLEX, Gurobi, or GLPK)

License: MIT

Non-academic use restrictions: None

Additional material

Additional file 1: Supplementary Methods. Detailed grammar for TIGER rules and specifics on the rule to MILP conversion process.

Additional file 2: TIGER source code. Source code, documentation, and tutorials are also available online at <http://bme.virginia.edu/csbl/downloads/> or <http://csbl.bitbucket.org/tiger>.

List of abbreviations used

TIGER: Toolbox for Integrating Genome-scale metabolism, Expression, and Regulation; COBRA: COntstraint-Based Reconstruction and Analysis; FBA: Flux Balance Analysis; GPR: Gene-Protein-Reaction; TRN: Transcriptional Regulatory Network; MILP: Mixed-Integer Linear Program; SR-FBA: Steady-State Regulatory FBA; iND750: Genome-scale model of *Saccharomyces cerevisiae* metabolism; iMH805: Genome-scale transcriptional regulatory network for *Saccharomyces cerevisiae*; CMPI: Common Mathematical Programming Interface; GIMME: Gene Inactivity Moderated by Metabolism and Expression; iMAT: Integrative Metabolic Analysis Tool; MADE: Metabolic Adjustment by Differential Expression.

Acknowledgements and Funding

We thank the reviewers for their thoughtful suggestions. Charles Haggart tested many of the functions and offered helpful feedback. This work is supported by the NSF (GRF to PAJ, REU to KAL, and CAREER award 0643548 to JP) and the NIH (R01 GM088244 to JP).

Authors' contributions

PAJ designed and wrote the TIGER software and performed the simulations. KAL prepared the TRN validation study and contributed TIGER functions. The project was conceived by PAJ and JP, and advised by JP. All authors wrote and reviewed the manuscript.

Received: 12 April 2011 Accepted: 23 September 2011

Published: 23 September 2011

References

1. Oberhardt MA, Palsson BØ, Papin JA: **Applications of genome-scale metabolic reconstructions.** *Mol Syst Biol* 2009, **5**:320.
2. Burgard AP, Pharkya P, Maranas CD: **Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization.** *Biotechnol Bioeng* 2003, **84**(6):647-57.
3. Oberhardt MA, Goldberg JB, Hogardt M, Papin JA: **Metabolic network analysis of *Pseudomonas aeruginosa* during chronic cystic fibrosis lung infection.** *J Bacteriol* 2010, **192**(20):5534-48.
4. Klitgord N, Segrè D: **Environments that induce synthetic microbial ecosystems.** *PLoS Comput Biol* 2010, **6**(11):e1001002.
5. Schmidt BJ, Lin-Schmidt X, Chamberlin A, Salehi-Ashtiani K, Papin JA: **Metabolic systems analysis to advance algal biotechnology.** *Biotechnol J* 2010, **5**(7):660-70.
6. Lewis NE, Schramm G, Bordbar A, Schellenberger J, Andersen MP, Cheng JK, Patel N, Yee A, Lewis RA, Eils R, König R, Palsson BØ: **Large-scale in silico modeling of metabolic interactions between cell types in the human brain.** *Nat Biotechnol* 2010, **28**(12):1279-85.
7. Orth JD, Thiele I, Palsson BØ: **What is flux balance analysis?** *Nat Biotechnol* 2010, **28**(3):245-8.
8. Feist AM, Palsson BO: **The biomass objective function.** *Curr Opin Microbiol* 2010, **13**(3):344-349.

9. Duarte NC, Herrgård MJ, Palsson BØ: **Reconstruction and validation of *Saccharomyces cerevisiae* iND750, a fully compartmentalized genome-scale metabolic model.** *Genome Res* 2004, **14**(7):1298-309.
10. Shlomi T, Eisenberg Y, Sharan R, Ruppin E: **A genome-scale computational study of the interplay between transcriptional regulation and metabolism.** *Mol Syst Biol* 2007, **3**:101.
11. Kim J, Reed JL: **OptORF: Optimal metabolic and regulatory perturbations for metabolic engineering of microbial strains.** *BMC Syst Biol* 2010, **4**:53.
12. Jensen PA, Papin JA: **Functional integration of a metabolic network model and expression data without arbitrary thresholding.** *Bioinformatics* 2011, **27**(4):541-7.
13. Barua D, Kim J, Reed JL: **An automated phenotype-driven approach (GeneForce) for refining metabolic and regulatory models.** *PLoS Comput Biol* 2010, **6**(10):e1000970.
14. Suthers PF, Zomorodi A, Maranas CD: **Genome-scale gene/reaction essentiality and synthetic lethality analysis.** *Mol Syst Biol* 2009, **5**:301.
15. Covert MW, Knight EM, Reed JL, Herrgård MJ, Palsson BO: **Integrating high-throughput and computational data elucidates bacterial networks.** *Nature* 2004, **429**(6987):92-6.
16. Herrgård MJ, Lee BS, Portnoy V, Palsson BØ: **Integrated analysis of regulatory and metabolic networks reveals novel regulatory mechanisms in *Saccharomyces cerevisiae*.** *Genome Res* 2006, **16**(5):627-35.
17. Covert MW, Schilling CH, Palsson B: **Regulation of gene expression in flux balance models of metabolism.** *J Theor Biol* 2001, **213**:73-88.
18. Klamt S, Saez-Rodriguez J, Gilles ED: **Structural and functional analysis of cellular networks with CellNetAnalyzer.** *BMC Syst Biol* 2007, **1**:2.
19. Cvijovic M, Olivares-Hernández R, Agren R, Dahr N, Vongsangnak W, Nookaew I, Patil KR, Nielsen J: **BioMet Toolbox: genome-wide analysis of metabolism.** *Nucleic Acids Res* 2010, **38** Web Server: W144-W149.
20. Becker SA, Feist AM, Mo ML, Hannum G, Palsson BØ, Herrgård MJ: **Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox.** *Nat Protoc* 2007, **2**(3):727-38.
21. Kuhn HW, Tucker AW, Dantzig GB: In *Linear inequalities and related systems. Volume 38.* Princeton: Princeton University Press; 1956.
22. Mahadevan R, Schilling CH: **The effects of alternate optimal solutions in constraint-based genome-scale metabolic models.** *Metab Eng* 2003, **5**(4):264-276.
23. Gudmundsson S, Thiele I: **Computationally efficient flux variability analysis.** *BMC Bioinformatics* 2010, **11**:489.
24. Gianchandani EP, Papin JA, Price ND, Joyce AR, Palsson BO: **Matrix formalism to describe functional states of transcriptional regulatory systems.** *PLoS Comput Biol* 2006, **2**(8):e101.
25. Turi TG, Loper JC: **Multiple regulatory elements control expression of the gene encoding the *Saccharomyces cerevisiae* cytochrome P450, lanosterol 14 alpha-demethylase (ERG11).** *J Biol Chem* 1992, **267**(3):2046-2056.
26. Becker SA, Palsson BO: **Context-specific metabolic networks are consistent with experiments.** *PLoS Comput Biol* 2008, **4**(5):e1000082.
27. Shlomi T, Cabili MN, Herrgård MJ, Palsson BØ, Ruppin E: **Network-based prediction of human tissue-specific metabolism.** *Nat Biotechnol* 2008, **26**(9):1003-10.

doi:10.1186/1752-0509-5-147

Cite this article as: Jensen et al.: TIGER: Toolbox for integrating genome-scale metabolic models, expression data, and transcriptional regulatory networks. *BMC Systems Biology* 2011 **5**:147.

Submit your next manuscript to BioMed Central and take full advantage of:

- Convenient online submission
- Thorough peer review
- No space constraints or color figure charges
- Immediate publication on acceptance
- Inclusion in PubMed, CAS, Scopus and Google Scholar
- Research which is freely available for redistribution

Submit your manuscript at
www.biomedcentral.com/submit

