Chapter 19

Genome-Scale ^{13}C Fluxomics Modeling for Metabolic Engineering of *Saccharomyces cerevisiae*

David Ando and Héctor García Martín

Abstract

Synthetic biology is a rapidly developing field that pursues the application of engineering principles and development approaches to biological engineering. Synthetic biology is poised to change the way biology is practiced, and has important practical applications: for example, building genetically engineered organisms to produce biofuels, medicines, and other chemicals. Traditionally, synthetic biology has focused on manipulating a few genes (e.g., in a single pathway or genetic circuit), but its combination with systems biology holds the promise of creating new cellular architectures and constructing complex biological systems from the ground up. Enabling this merge of synthetic and systems biology will require greater predictive capability for modeling the behavior of cellular systems, and more comprehensive data sets for building and calibrating these models. The so-called "-omics" data sets can now be generated via high throughput techniques in the form of genomic, proteomic, transcriptomic, and metabolomic information on the engineered biological system. Of particular interest with respect to the engineering of microbes capable of producing biofuels and other chemicals economically and at scale are metabolomic datasets, and their insights into intracellular metabolic fluxes. Metabolic fluxes provide a rapid and easy to understand picture of how carbon and energy flow throughout the cell. Here, we present a detailed guide to performing metabolic flux analysis and modeling using the open source JBEI Quantitative Metabolic Modeling (jQMM) library. This library allows the user to transform metabolomics data in the form of isotope labeling data from a ^{13}C labeling experiment into a determination of cellular fluxes that can be used to develop genetic engineering strategies for metabolic engineering.

The jQMM library presents a complete toolbox for performing a range of different tasks of interest in metabolic engineering. Various different types of flux analysis and modeling can be performed such as flux balance analysis, ^{13}C metabolic flux analysis, and two-scale ^{13}C metabolic flux analysis (2S-^{13}C MFA). 2S-^{13}C MFA is a novel method that determines genome-scale fluxes without the need of every single carbon transition in the metabolic network. In addition to several other capabilities, the jQMM library can make model based predictions for how various genetic engineering strategies can be incorporated toward bioengineering goals: it can predict the effects of reaction knockouts on metabolism using both the MoMA and ROOM methodologies. In this chapter, we will illustrate the use of the jQMM library through a step-by-step demonstration of flux determination and knockout prediction in a complex eukaryotic model organism: *Saccharomyces cerevisiae* (*S. cerevisiae*). Included with this chapter is a digital Jupyter Notebook file that provides a computable appendix showing a self-contained example of jQMM usage, which can be changed to fit the user's specific needs. As an open source software project, users can modify and extend the

Electronic supplementary material: The online version of this chapter (https://doi.org/10.1007/978-1-4939-8757-3_19) contains supplementary material, which is available to authorized users.

Edward E.K. Baidoo (ed.), *Microbial Metabolomics: Methods and Protocols*, Methods in Molecular Biology, vol. 1859, https://doi.org/10.1007/978-1-4939-8757-3_19, © Springer Science+Business Media, LLC, part of Springer Nature 2019

code base to make improvements at will, allowing them to share their development work and contribute back to the jQMM modeling community.

Key words Flux analysis, Two-scale ^{13}C metabolic flux analysis, -Omics data, Predictive biology, Engineering biology

1 Introduction

The ability to genetically engineer and modify an organism's DNA has radically changed the nature of biology over the last few decades. Synthetic biology is an emerging field which applies genetic engineering techniques toward the systematic design of biological organisms toward specific design and engineering goals [1] using traditional engineering principles such as the design–build–test–learn (DBTL) cycle [2]. Initial efforts in the realm of repurposing biological systems toward the production of biofuels and other chemicals using synthetic and systems biology have succeeded in producing a number of target molecules, although typically at low titers of μg/L to mg/L [3]. However, only the biological production of a few compounds, such as 1,3-propanediol [4], 1–4-butanediol [5], and artemisinin [6], has been produced at high titers and at a commercial scale. This relative scarcity of commercially successful examples showcases the difficulties that the synthetic biology field still faces, among them a lack of predictive tools and models to guide the bioengineering process [7]. Improving the yield, titer, and productivity of engineered bioprocesses to enable commercialization requires detailed manipulation of many processes including microbial physiology, stress response, and metabolism. This chapter focuses exclusively on the particular field of metabolic engineering, and will show how to use metabolomics data to calculate how carbon and energy flow through the cell, which is an effective piece of information to optimize [8] bioproduct formation [3].

The study of metabolic fluxes is of particular interest for engineering microbes that can produce biofuels and other chemicals as fluxes provide a mechanistic understanding for how carbon and energy flow throughout the cell. While often a heuristic and non-systematic approach is taken toward designing and genetically engineering new organisms [3], metabolic fluxes offer the potential for adding a powerful predictive capability to synthetic biology. Metabolic fluxes can be readily transformed into actionable predictions for improving bioproduct formation via the use of metabolic modeling [8]. Metabolic models include the stoichiometry of an organism's metabolic network and, using a variety of mathematical methods (e.g., COBRA [9]), can predict how fluxes will change through modifications of the metabolic reaction network arising from specific genetic engineering strategies.

The DBTL cycle is one of the common practices of engineering transplanted into synthetic biology. This cycle starts with the design (D) of a biological system in the form of a best guess, given initial data and biological understanding, on how to achieve a desired particular outcome (e.g., fuel production at a given yield). This biological design is then built (B) in the next phase of the cycle from DNA parts and an appropriate microbial chassis using the tools of synthetic biology. The next phase involves testing (T) whether the biological system performs as desired in the original design using a variety of assays (e.g., production measurement or/and omics profiling). Given the complex nature of biological systems, it is extremely unlikely that the first design would behave as desired, so further attempts will most likely be needed to meet the desired specification. It would be desirable to use a validated predictive model based on data generated in previous rounds to systematically converge toward engineering goals more quickly. This task is referred to as the learn (L) phase of the DBLT cycle and is, arguably, the hardest and least supported step in current metabolic engineering practice [2]. However, given recent increasingly available high-throughput workflows for data generation, there is an opportunity to use these copious amounts of data to validate and create models which can provide *actionable* items for metabolic engineers: that is, suggestions that can be acted upon with the available synthetic biology tools (e.g., to change a particular gene's ribozyme binding site (RBS) or knock out a particular gene in order to reroute metabolic fluxes).

In this chapter we will show how to use metabolomic data obtained from ^{13}C labeling experiments to generate actionable items to increase ethanol production in *S. cerevisiae*. We will demonstrate how to use the Joint BioEnergy Institute (JBEI) Quantitative Metabolic Modeling library (jQMM) [10] to calculate cellular fluxes and make predictions. The jQMM library is currently capable of measuring and predicting internal metabolic fluxes using a variety of techniques: ^{13}C metabolic flux analysis (^{13}C MFA) [11], flux balance analysis (FBA) [9], and two-scale ^{13}C metabolic flux analysis (2S-^{13}C MFA) [12]. First, we will provide a brief background on metabolomics and more specifically ^{13}C labeling experiments, which consist of cellular cultures in which the feed source (e.g., glucose) is labeled with carbon atoms that have an extra neutron (i.e., carbon isotopes) at selected and specific positions. We will then succinctly describe how to measure the ensuing carbon isotope labeling in the metabolites of the studied. Next, we will describe in detail how to use the labeling data from different metabolites in the cell to determine the cell's internal metabolic fluxes through a technique called two-scale ^{13}C metabolic flux analysis (2S-^{13}C MFA). 2S-^{13}C MFA is a novel technique which introduces the capability to use experimental ^{13}C labeling data to constrain comprehensive *genome-scale* models (rather than with small models of central metabolism as done with traditional ^{13}C MFA) by taking into account the system-wide balances of metabolites [8]. This is possible by assuming that flux flows from core to peripheral metabolism and does not flow back. This assumption stems from the bow tie structure of cellular metabolism [2], and is further justified by the fact that traditional ^{13}C MFA, using only core metabolism models, can convincingly explain labeling patterns for amino acids and intracellular metabolites for model organisms [13–16]. We then perform an External Labeling Variability Analysis (ELVA), a self-consistency test of this modeling approach which gauges the effect of the ignored noncore reactions [12]. ELVA finds the maximum impact that the unknown inflow metabolite labeling can have on the measured labeling pattern, and provides evidence of biophysical consistency of the calculated metabolic fluxes. Finally, we will show how to use the COBRA (Constraint Based Reconstruction and Analysis) [17] methods of MoMA (Minimization of Metabolic Adjustment) [18] and ROOM (Regulatory On/Off Minimization) [19] to predict, based on the measured flux profiles, which gene knock outs will increase acetate production in *S. cerevisiae*.

2 Materials

The general workflow for determining and visualizing metabolic fluxes of cellular systems is shown in Fig. 1. First, a microbial culture is grown with ^{13}C labeled glucose feed. Next, mass spectroscopy is used for the analysis of the distribution of ^{13}C in metabolites extracted from cell culture to obtain the Mass Distributions Vectors (MDVs) [20] for each measured metabolite, which give the relative frequency of isotopologs. Each isotopolog of a metabolite has a different number of carbons with an extra neutron, and each isotopolog can exhibit the heavy carbon(s) substituting any of the carbon(s) located within the metabolite. For example, a metabolite with five carbon atoms could have five different isotopologs. The different labeling patterns for the measured metabolites ultimately provides enough information to determine the internal flux profile [12] via knowledge of how reactions in the metabolic network rearrange carbon atoms within metabolites (so called carbon transitions). MDVs then need to be then formatted for input into the jQMM, so that metabolic fluxes can be calculated at the genome scale using the 2S-^{13}C methodology. For genome scale models, there are thousands of reactions, so we recommend that fluxes be visualized and more easily understood via a plotting environment called “Arrowland” which is located at the following website: http://public-arrowland.jbei.org. Fluxes can also be more simply plotted within the jQMM.

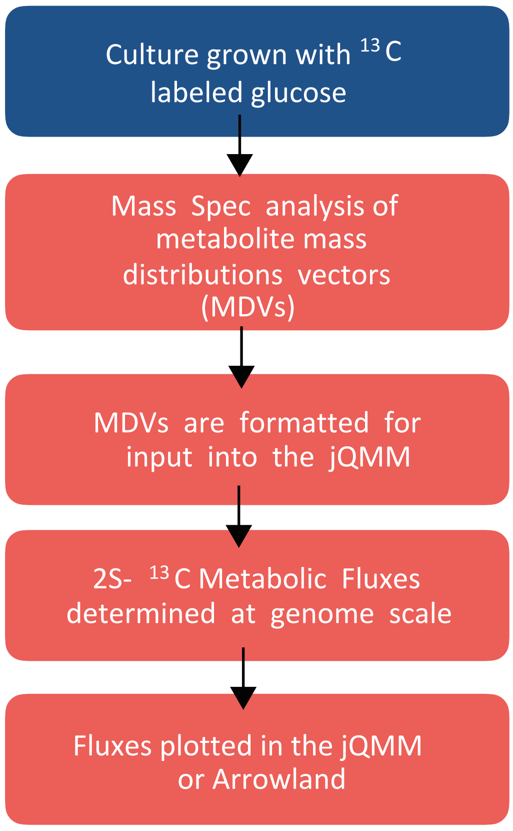
Culture grown with ^{13}C labeled glucose
Mass Spec analysis of metabolite mass distributions vectors (MDVs)
MDVs are formatted for input into the jQMM
2S- ^{13}C Metabolic Fluxes determined at genome scale
Fluxes plotted in the jQMM or Arrowland

Fig. 1 Overview of the workflow for ^{13}C two-scale metabolic flux analysis. The test phase of the workflow is in blue while the learn phase steps are in light red

2.1 Computational Requirements

To calculate fluxes using the jQMM, a desktop computer or server system that is capable of running heavy computational loads for extended periods of time is necessary. The more available cores/CPUs in the system the better, as calculations within the jQMM library often take many hours to days (for a desktop computer), although the library is parallelized and can leverage additional cores/CPUs to reduce computation time. Windows, Mac, and Linux operating systems are all compatible with running python code, which the jQMM is written in. Error-correcting code RAM (ECC RAM) reduces incorrect memory read and write operations to system RAM and is recommended to reduce the probability of faulty calculations over lengthy computations due to memory access errors. Xeon processors from Intel, for example, are specially designed for long continuous computations at high CPU load, unlike the consumer series of desktop processors sold by Intel such as the i7 and i5 series (circa 2016), and therefore we also recommend the use of Xeon like processors.

2S-^{13}C MFA computations require the following:

1. At least 32 GB of RAM and 500 GB of free disk space.
2. Ownership of both the GAMS and CONOPT solver licenses.
3. Python version 2.7 installed and configured.

2.2 Software Installation and Configuration

Use of the jQMM requires installation of accessory libraries (such as numpy, as described below), which can be done in two different ways: the traditional way, which includes a self-installation on the hardware and operating system at hand, or via the use of a preconfigured and preinstalled Docker container (http://www.docker.com). We recommend using our preconfigured Docker container for use of the jQMM as the flux modeling environment will be immediately usable once a GAMS and CONOPT license are purchased. However, for expert users or those who wish to use a custom installation, a self-installation is readily achieved as follows.

2.2.1 Self-Installation

First install the following jQMM dependencies to the Python 2.7 environment:

1. libSBML: available at http://sbml.org/Software/libSBML.
2. matplotlib: available at http://matplotlib.org/users/installing.html.
3. numpy: available at http://www.scipy.org/scipylib/download.html.
4. jupyter: available at http://jupyter.org/.

   For self-installation *see* **Note 1**.

2.2.2 Preconfigured and Preinstalled jQMM Library

Docker (https://docs.docker.com/) is a technology that is based on Linux containers that allows for the building, running, testing, and deploying applications such as the jQMM library into a complete file system that contains everything it needs to run: code, runtime, system tools, system libraries, and supporting data files. This guarantees that it will always run correctly and in the same way, regardless of the system environment it is running in. The jQMM docker container format is standardized and can be run on virtually any cloud computing service such as AWS (Amazon Web Services), Google Cloud Platform, Microsoft Azure, etc. Additionally, the jQMM Docker container can be conveniently run on a personal computer running either Microsoft Windows or the Mac operating system, although we discourage this practice for anything other than training purposes given how slowly the jQMM will run on personal computers. For further information on web-based platforms please *see* **Note 2**.

The prebuilt jQMM docker container, which is available for download at https://github.com/JBEI/jqmm, has all of the software needed to run the jQMM library preinstalled and preconfigured to work out of the box. The GAMS and CONOPT solvers (http://www.gams.com/, http://www.conopt.com/), are required to do flux analysis, but their usage within the Docker container will require purchase of GAMS and CONOPT licenses separately. Once these licenses are included with the GAMS and CONOPT solver installations the jQMM docker container will be fully functional.

2.3 ^{13}C Labeling Experiments and Mass Spectrometry Data Analysis

The initial step for any ^{13}C based metabolic flux experiment is performing a ^{13}C labeling experiment with the organism of interest. We will only give a brief description here, the reader can refer to previously published protocols for more information [21, 22]. Cultures should use minimal media to avoid nonlabeled carbon sources, and can be grown using different types of labeling for the feed (*see* **Note 3**). A common choice is to use 20% U-^{13}C glucose (universal labeling of carbon atoms) and 80% 1-^{13}C glucose (first carbon atom only is labeled) (for example *see* **Note 4**). Briefly, we recommend that culture samples are prepared for measurement on an LC-MS system by a rapid quenching step to halt cellular metabolism immediately after removing an aliquot of cells from culture, which is then followed by extraction of intracellular metabolites and LCMS sample preparation [21, 22].

The quantification of relative cellular metabolite isotopolog concentration, consisting of MDVs for metabolites in cellular metabolism (*see* Fig. 2 for an example), is done by determining the relative concentration of each metabolite's isotopologs via mass spectrometry. Once chromatograms from a mass spectrometry instrument have been analyzed and integrated such that the relative frequency of each metabolite's mass distribution has been determined, these data need to be prepared and formatted for input into the jQMM library so that metabolic fluxes can be determined.

2.4 Input Data Preparation

2.4.1 ^{13}C Labeling Data

Once the measured MDVs have been determined and normalized such that their sum is equal to one for each metabolite, they have to be entered into a text document with the following format so that the data can be read by the jQMM:

```
Amino acid Mass distribution
 m0  m1  m2  m3  m4  m5  m6  m7  m8
Gly M-0 0.7426 0.0592 0.1982 - - - - -
Ala M-0 0.3903 0.3884 0.0221 0.1992 - - - -
Thr M-0 0.3260 0.3416 0.1670 0.1513 0.0000 - - -
Asp M-0 0.3192 0.3564 0.2003 0.1035 0.0206 - - -
```

The first two lines should remain fixed and are ignored by the jQMM library, while the following lines need to include the MDV information for every metabolite for which isotopolog data exists in the following format:

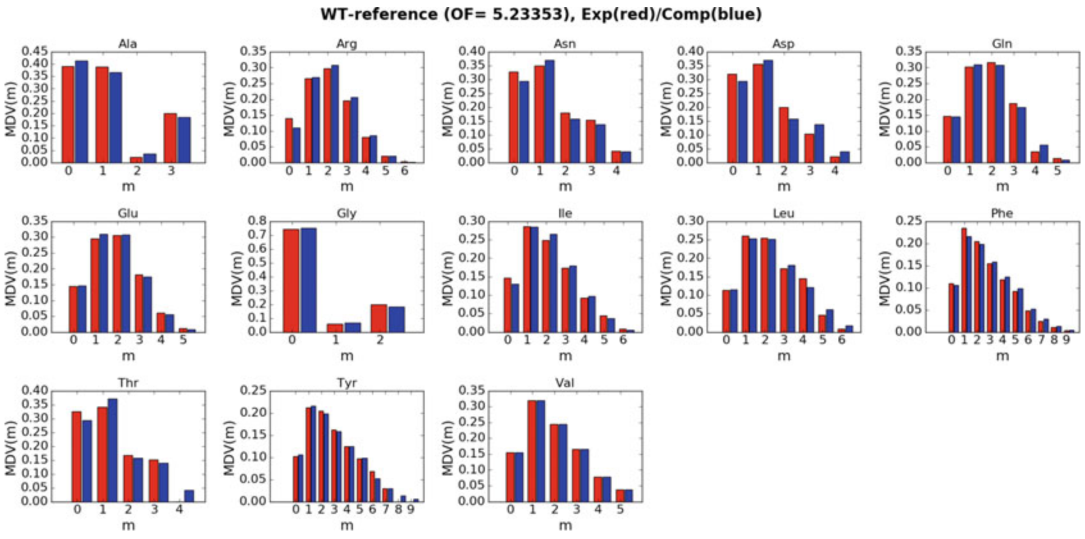

Fig. 2 Plot of wild-type *S. cerevisiae* MDVs from Ghosh et al. [8]. Experimentally measured MDVs are plotted in red bars for each metabolite, while the MDVs implied by the predicted fluxes are shown in the blue bars. Computational and experimental data match closely implying that the model is quantitatively correct

1. First the metabolite name using the metabolite abbreviation used in the BIGG database (http://bigg.ucsd.edu) is specified and followed by a tab character or space(s).
2. Next tab or space separated relative frequencies of each metabolite isotopolog are specified. The sum of all isotopolog frequencies needs to add to 1 for each individual metabolite.
3. Isotopologs which do not exist for a particular metabolite, should be represented by a "-" (dash), and must not be entered as a zero value, which would indicate a different meaning in the jQMM. Zero values indicate that an isotopolog can exist but is not present in the sample.

2.4.2 Choose, Edit, and Define the Metabolic Model

To define a metabolic model for the jQMM to use in the modeling of the particular organism being studied, we recommend using the BIGG database (http://bigg.ucsd.edu) and downloading a SBML version of the metabolic model which is appropriate to the problem being studied. Smaller models run much faster in the jQMM, while more comprehensive genome scale models are necessary for problems involving peripheral metabolism or which utilize the 2S-^{13}C MFA analysis methodology. Retooling of the jQMM library code may be required for the library to understand metabolite names which do not follow the naming convention used in the iMM904 metabolic model [23] format, although sample code is already included in the jQMM for using metabolite names from the iJO1366 [24] and iAF1260 [25] models, which is located in the "sbmlio.py" python code file located in the jQMM "code/core" directory. Network reactions which come from a heterologous pathway that has been engineered into a microbe should be manually copied into the downloaded SBML model.

2.4.3 Defining Exchange Reaction Fluxes

Both measured fluxes through exchange reactions and the biomass flux that correspond to the time that a sample was taken for ^{13}C isotopolog analysis are detailed in a FLUX.txt file. A sample exchange flux file should be formatted as follows for input into the jQMM library:

```
biomass_SC5_notrace: 0.24 [==] 0.24
GLCt1: 8.5 [==] 8.5
EX_glc(e): -8.5 [==] -8.5
EX_ac(e): 0.45 [==] 0.45
```

The Biomass flux is in units of 1/h and is normalized to equal the growth rate while all other exchange fluxes are in units of mMol/gdw/h (millimoles/grams of dry weight/hour). The glucose uptake rate can be measured for example by a HPLC determination of glucose concentration at two different times around the ^{13}C analysis sample time. If the glucose concentration is measured as g_1 and g_2 (in millimoles/volume) at times t_1 and t_2, the glucose flux can be approximated as $(g_2 - g_1)/gdw_1/(t_2 - t_1)$, where gdw_1 is the grams of dry weight of cells per unit volume at time t_1. Similarly, HPLC measurements can be used to determine fluxes of acetate, lactate, and other organic acids excreted by the cell.

2.4.4 Defining Carbon Transitions in the Metabolic Model

Atom transitions can be used to represent the fate of each carbon in a reaction [11]. In the example reaction:

```
A (abc) -> B (ab) + C (c)
```

Uppercase letters in a carbon transition reaction represent the metabolites present in a reaction, while the lowercase letters in parentheses represent the carbon atom transitions, *see* **Note 5**. Irreversible reactions are denoted by a ’->’ or ’=>’ arrow and reversible reactions are denoted by a ’<->’ or ’<=>’ arrow. In the jQMM, multiple reactions are separated by carriage returns or by placing a semicolon at the end of each reaction equation.

For example, in the following reaction from the iMM904 metabolic model for *S. cerevisiae*:

```
AKGDam akg_m --> sdhlam_m + co2_m abcde : bcde + a
```

akg (2-Oxoglutarate) gets split into sdhlam (Succinyldihydrolipoamide) and co2 (carbon dioxide) in reaction AKGDam, with the last four carbons going to sdhlam and the first carbon going to co2.

For 2S-^{13}C based metabolic flux analysis in the jQMM, one needs to create a REACTIONS.txt file which contains the carbon transition information for core reactions in the metabolic network and which has the following format:

```
# Metabolic Exchange fluxes
&MLIST etoh 0
&MLIST glucon 0
&MLIST lac-L 0

# Amino acid metabolite fluxes
&MLIST ala-L[c] 0
&MLIST asp-L[c] 0
&MLIST asn-L[c] 0
&MLIST gly[c] 0
&MLIST val-L[c] 0
&MLIST arg-L[c] 0

# Carbon source
&SOURCE glcD[e]

# Input Reactions
EX_glc(e) glc-D[e] <==> glcDEx abcdef : abcdef
GLCt1 glc-D[e] --> glc-D abcdef : abcdef
HEX1 glc-D --> g6p abcdef : abcdef

# Carbon Transitions
PGI g6p <==> f6p abcdef : abcdef
PFK f6p --> fdp abcdef : abcdef
FBA fdp <==> g3p + dhap abcdef : cba + def
```

See **Note 6** for metabolite abbreviations and **Note 7** for additional REACTIONS.txt file specifications.

2.4.5 Defining Feed Labeling

The type of labeled glucose used in the experiment, together with its concentration relative to the amount of unlabeled glucose, is detailed in a FEED.txt file. A sample feed specification file contains the following line:

```
1.5% Glucose: 80% 1-C 20% U 0% UN
```

See **Note 8** for file specification details.

3 Methods

An example Jupyter notebook which demonstrates metabolic flux analysis on *S. cerevisiae* is included in the Supplementary Information and can be run directly within the jQMM (*see* Electronic

Supplementary Material, *see* **Note 9**). This notebook is also reproduced in this section as an example of how to use the FluxModels module in the jQMM to do 2S-^{13}C MFA and then predict the outcomes on ethanol production of different reaction knockouts. The Jupyter notebook provides a convenient way to reproduce results and is easily modified to fit the user's specific needs, and it is divided into seven different steps for turning experimental data into actionable predictions for increasing a targeted biochemical via genetic engineering:

1. Setting up python environment.
2. Gathering input data.
3. Creating the Reaction Network.
4. Creating the two-scale metabolic model.
5. Calculating internal metabolic fluxes through 2S-^{13}C MFA.
6. Performing an ELVA consistency check.
7. Visualizing flux profiles.
8. Predicting which genes to knockout using MoMA and ROOM.

```
-- Jupyter Notebook Start --
```

^{13}C-2S Metabolic Flux Analysis for *S.cerevisiae* with ELVA methods: This Jupyter notebook presents a computable step-by-step description of how to use metabolite data from ^{13}C labeling experiments to determine fluxes for *S. cerevisiae* with quality control of flux results via ELVA methods. We also use MOMA and ROOM methodologies to produce actionable insights to improve ethanol production.

3.1 Setup

The first step involves specifying the correct path for the library to be loaded:

```
In [1]:
%matplotlib inline
import sys, os
path = "/scratch/user"
pythonPath = path + "/quantmodel/code/core"
if pythonPath not in sys.path:
 sys.path.append(path + '/quantmodel/code/core')
os.environ["QUANTMODELPATH"] = path +'/quantmodel'
```

We then need to import the needed classes for the notebook:

```
In [2]:
from IPython.display import SVG
import FluxModels as FM
import enhancedLists, ReactionNetworks, predictions, copy,
core, os
```

and then move to a defined working directory where output and intermediate files will be kept:

```
In [3]:
cd ~/tests
```

Test that GAMS and CONOPT solvers are installed and working:

```
In [5]:
status = os.system('gams')
if status == 0:
 print "GAMS call working."
else:
 print "GAMS is not installed."
GAMS call working.
```

3.2 Gathering Input Data

As part of the test (T) phase of the DBTL cycle, we gather all the relevant experimental data from the ^{13}C labeling experiments (*see* "Materials" Subheading 2). These data involve:

- A base *genome-scale model* that will act as the reference for all other data types.
- *Exchange fluxes* containing the measured fluxes of metabolites being exchange by cells with the environment.
- *Transition information* on the fate of each carbon in the core reaction network.
- *Metabolite labeling* information in the form of Mass Distribution Vectors (MDVs).
- *Metabolite labeling error* information.
- *Feed labeling* information on the type of labeled glucose the cell culture was fed.

Discussion of these data types can be seen in Subheading 2.

For this demonstration, we will use the data from Ghosh et al. [8] for the wild type strain grown using a glucose feed:

```
In [6]:
datadir = os.environ['QUANTMODELPATH']+'/data/tests/yeast2S/
wyr1/'
strain ='wyr1Glc'
#A base genome-scale model:
BASEfilename = datadir + 'SciMM904wyr1GlcTS.xml'
#Exchange fluxes
```

```
FLUXESfilename = datadir + 'FLUX'+strain+'.txt'
#Transition information
TRANSITIONSfilename = datadir + 'REACTIONS'+strain+'-core-too-
small.txt'
#Metabolite labeling
MSfilename = datadir + 'LCMS'+strain+'.txt'
#Metabolite labeling error
MSSTDfilename = datadir + 'LCMSerr'+strain+'.txt'
#Feed labeling
FEEDfilename = datadir + 'FEED'+strain+'.txt'
```

3.3 Creating the Reaction Network

Once we have gathered all the needed input files, we can condense all this information into a single sbml file. We will do this using a reaction network from the ReactionNetworks module in the jQMM library. A reaction network contains all information related to the metabolic reaction network used for the simulation:

```
In [7]:
# Load initial SBML file
reacNet = ReactionNetworks.TSReactionNetwork(BASEfilename)
# Add Measured fluxes
reacNet.loadFluxBounds(FLUXESfilename)
# Add carbon transitions
reacNet.addTransitions(TRANSITIONSfilename,translate2SBML=-
True)
# Add measured labeling information
reacNet.addLabeling(MSfilename,'LCMS',MSSTDfilename,
minSTD=0.001)
# Add feed labeling information
reacNet.addFeed(FEEDfilename)
# Limit fluxes to 500
reacNet.capFluxBounds(500)
# Create sbml file to store the two-scale model.
# All input files are combined in a tuple of the type:
(fileName, string of contents)
SBMLfile = ('SciMM904'+strain+'TS.xml',reacNet.write('to-
String'))
```

3.4 Creating the Two-Scale Metabolic Model

The next step is to use the SBML file we just created to create a two-scale model [12] that we will use to calculate fluxes through 2S–^{13}C MFA:

```
In [9]:
TSmodel = FM.TwoSC13Model(('SciMM904'+strain+'TS.xml',reac-
Net.write('toString')))
```

TSmodel now contains all the information needed to calculate fluxes along with the methods to do this calculation and other analysis [12].

3.5 Calculating Internal Metabolic Fluxes through 2S-^{13}C MFA

We can now use the *findFluxesRanges* method in *TSmodel* to find the fluxes that best fit the experimentally obtained metabolite labeling data (MDVs) and find the ranges of fluxes compatible with this labeling data and the corresponding experimental error:

```
In [10]:
%%time
fluxNames =
TSmodel.reactionNetwork.C13ReacNet.reactionList.getReaction-
NameList(level=1)
TSresult =
TSmodel.findFluxesRanges(Nrep=30,fluxNames=fluxNames,proc-
String='proc')Wall time: 1.75 hrs
```

Nrep represents the number of replicates used for the calculation. Since the problem to be solved is a nonconvex problem there is no guarantee that a single run will find the best global fit. Hence, we run 30 independent processes and keep the one that best fits the data. *fluxNames* indicates the fluxes for which full flux confidence intervals will be calculated. *procString* indicates that the data (for this case) needs no derivatization correction. For large models, especially with eukaryotic organisms, these flux calculations can be lengthy (on the order of several hours).

We can check how accurate the model is by comparing the measured labeling distribution (MDVs, red) with the one predicted through the computational model (blue) by using the `plotExpvsCompLabelFragment` method:

```
In [11]:
TSresult.plotExpvsCompLabelFragment(titleFig='WT-reference')
```

See Fig. 2.

or by using `plotExpvsCompLabelXvsY`, if we prefer to see these fits as an X vs. Y plot:

```
In [12]:

TSresult.plotExpvsCompLabelXvsY(titleFig='WT-
reference')
```

See Fig. 3.

The closeness of the fit data and the experimental data validate the use of this model.

Results are stored in a reaction network inside *TSresult* and can be explored through the reactionList methods. For example, we can print the fluxes that best fit the data:

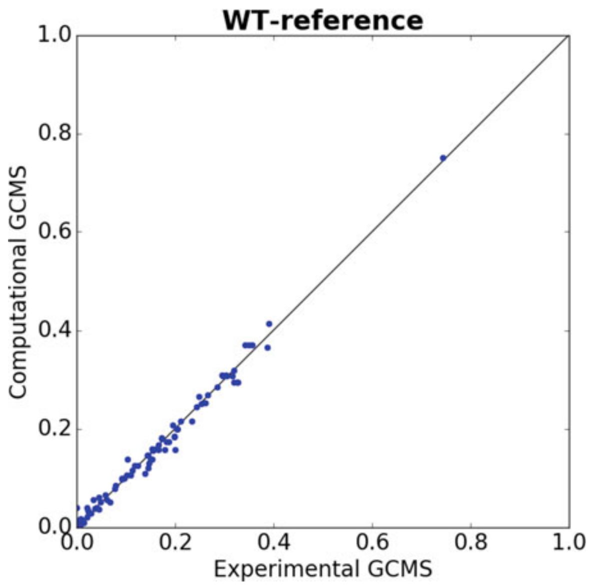

Fig. 3 Plot of experimentally measured MDVs versus the computationally predicted MDVs in an x-y plot (different way to plot data in Fig. 2), which demonstrates that experimental and predicted MDVs are comparable

```
In [13]:
TSresult.reactionNetwork.reactionList.printFluxes(brief="-
True",names="exchange")
EX_co2_e_: 14.2007023028
EX_glc_e_: -8.5
EX_etoh_e_: 8.5
EX_h2o_e_: 7.11362355138 EX_o2_e_: -4.90795595138
EX_h_e_: 2.72395044862
EX_glyc_e_: 2.45 EX_nh4_e_: -1.3431504
EX_glx_e_: 0.999214848625
EX_ac_e_: 0.45 biomass_SC5_notrace: 0.24
EX_pi_e_: -0.0474479999997
EX_ttdca_e_: 0.03
EX_ddca_e_: 0.02
EX_so4_e_: -0.018552
EX_hdca_e_: 0.0143
EX_ocdca_e_: 0.00193
EX_for_e_: 0.0012408
```

3.6 Performing an ELVA Consistency Check

The 2S-^{13}C methods assume flux flows from core metabolism to reactions in the periphery, and does not flow back. This assumption of one way flow out of central metabolism stems from the bow-tie structure of cellular metabolism [2], and is further justified by the fact that traditional ^{13}C MFA, using only core metabolism models, can convincingly explain labeling patterns for amino acids and intracellular metabolites for model organisms [13–16].

ELVA [12] is a test that checks that the error incurred by ignoring noncore reactions is negligible. If the ELVA indicates that the noncore contributions are large, then the core set of reactions which include carbon transition information is expanded. This is repeated until a self-consistent result is found that produces ELVA flux ranges which are small and compatible with the experimental data.

In general, the ELVA optimization problem finds the maximum impact that the unknown inflow metabolite labeling can have on the measured labeling pattern. ELVA determines if the fluxes obtained from the fit are consistent with the two-scale approximation [12]. This extra step sets the method apart from standard ^{13}C metabolic flux analysis (MFA), which does not check the effect of ignored reactions. ELVA considers only core metabolism with the impact of noncore metabolism being represented through inflow metabolites, dummy metabolites with unfixed labeling since their labeling is, by definition, unknown. Essentially, using the previously obtained genome-scale flux solution as a constraint, ELVA allows the labeling for the inflow metabolites to vary and, for each metabolite with measured labeling, uses each element of the mass distribution vector (MDV) as the objective function to be maximized or minimized. By this method, we obtain a confidence interval that represents the maximum possible difference in labeling that could be attributed to noncore reactions for the current solution. The reactions that contribute an unacceptable amount of uncertainty are then added to the core set and the procedure can be repeated as necessary, until a core set of reactions is found which fully justifies the two-scale approximation. ELVA results can be calculated as follows:

```
In [14]:
resultELVA = TSmodel.ELVA(TSresult)
```

ELVA results can be plotted in an x-y graph showing the experimentally determined isotope labeling which defines a confidence interval that represents the maximum possible difference in labeling that could be attributed to noncore reactions for the current solution. If there are reactions that contribute an unacceptable amount of uncertainty they can be added to the core set and the procedure can be repeated as necessary, until a core set of reactions is found which fully justifies the two-scale approximation. In this example, 13 metabolites have large fluctuations as predicted by computational labeling (y-axis error bars).

```
In [15]:
resultELVA.plotExpvsCompLabelxvsy(titleFig="WT",outputFileNa-
```

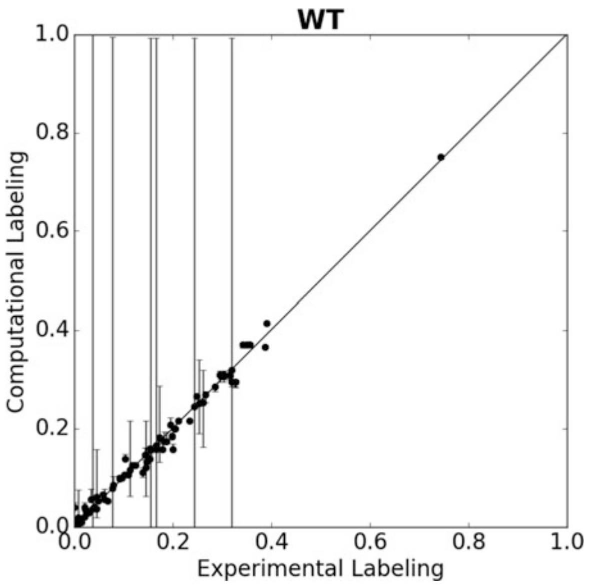

Fig. 4 ELVA plot that shows an x-y graph of the experimentally determined isotope labeling versus the computationally predicted labeling. The vertical error bars define the computational error that represent the maximum possible difference in labeling that could be attributed to noncore reactions for the current solution. In this example, several reactions display large computational errors, indicating that noncore reactions do significantly contribute to core metabolite labeling [12]. More reactions with carbon transition information need to be added to the core reactions list to remedy this problem and satisfy the two-scale assumptions

```
me="ELVAComparisonWT.txt",save="ELVA-W.eps")
```

See Fig. 4.

The error bars in the y axis (computational error) are NOT of the same order of magnitude as the experimental error, and therefore we need to rework the set of core reactions until we can satisfy the two-scale assumptions of no flux back into the core network [12]. We therefore add additional reactions to the REACTIONS file, and repeat the two-scale modeling process. We can see from Fig. 4 that there are 13 metabolites for which the fluctuation in computational labeling was large, which are described in more detail in an ELVAfluxReport that is generated by the jQMM.

The file "*ELVAfluxReport.txt*" is automatically saved to our current working directory /~tests and indicates which metabolites had large fluctuations. The format of the ELVAfluxREPORT file is as follows: first a metabolite is listed, which is then followed by the reactions that flow into it from noncore metabolism, then the reactions which flow into that metabolite from core metabolism.

The following ELVA report has been truncated for space, please see the full Jupyter notebook for complete details.

```
In [16]:
cat ELVAfluxReport.txt
RI3mob_c(0.0104):
       ----> Non Core:
       3MOBtm [ 0.0881793]
       ----> Core:
       IPPS [-0.0246753]
       VALTA [-0.063504]
RIakg_c(0.0019):
       ----> Non Core:
       HSTPT [ 0.015912]
       ----> Core:
       AATA [ 0.068688]
       ALATA_L [ 0.11842353]
       ASPTA [-0.69541853]
       GLUDyi [-1.1814192]
       ICDHy [ 1.230851]
       ILETA [ 0.046248]
       LEUTA [ 0.071136]
       PHETA1 [ 0.032136]
       PSERT [ 0.1367712]
       SACCD2 [ 0.068688]
       TYRTAi [ 0.02448]
       VALTA [ 0.063504]
RIhdca_c(0.0002):
       ----> Non Core:
       FACOAL160 [ 0.01623]
       HDCAt [-0.0143]
       ----> Core:
       FAS180 [-0.00193]
RIpyr_c(0.0008):
       ----> Non Core:
       ANS [ 0.006816]
       ----> Core:
       AGTi [ 0.00831153]
       ALATA_L [-0.11842353]
       PC [-2.9406824]
       PYK [ 12.82902984]
       PYRDC [-9.54111443]
       PYRt2m [-0.24393701]
RO3mob_m(0.0104):
       ----> Non Core:
       3MOBtm [-0.0881793]
       ----> Core:
       DHAD1m [ 0.13464]
       IPPSm [-0.0464607]
```

See **Note 6** for metabolite abbreviations. As a result of the ELVA report file which indicates that metabolites akg_c, co2_c, co2_m, fum_c, g3p_c, pyr_c, accoa_m, asp_L_c, f6p_c, g6p_c, glu_L_c, glx_c, and glyc3p_c produce large computational errors, we have added the reactions for Valine and Isoleucine synthesis and reactions 3C4MOPtm, 3MOBtm, OMCDCm to the REACTIONS file containing carbon transitions for core reactions. After expanding the core set of reactions we now need to rerun the two-scale ^{13}C MFA with the corrected core set of reactions.

```
In [17]:
datadir = os.environ['QUANTMODELPATH']+'/data/tests/yeast2S/
wyr1/'
strain ='wyr1Glc'
#A base genome-scale model:
BASEfilename = datadir + 'SciMM904wyr1GlcTS.xml'
#Exchange fluxes
FLUXESfilename = datadir + 'FLUX'+strain+'.txt'
#Corrected REACTIONS file with an expanded core set of reac-
tions
#Transition information
TRANSITIONSfilename = datadir + 'REACTIONS'+strain+'-fixed.
txt'
#Metabolite labeling
MSfilename = datadir + 'LCMS'+strain+'.txt'
#Metabolite labeling error
MSSTDfilename = datadir + 'LCMSerr'+strain+'.txt'
#Feed labeling
FEEDfilename = datadir + 'FEED'+strain+'.txt'
```

3.6.1 Re-Creating the New Reaction Network for Fixed Reactions File

Now that we have gathered all the needed input files for recalculating fluxes with the expanded set of core reactions, we again condense all this information into a single SBML file. We will do this using a reaction network from the ReactionNetworks module in the jQMM library. A reaction network contains all information related to the metabolic reaction network used for the simulation:

```
In [21]:
# Load initial SBML file
reacNet = ReactionNetworks.TSReactionNetwork(BASEfilename)
# Add Measured fluxes
reacNet.loadFluxBounds(FLUXESfilename)
# Add carbon transitions
reacNet.addTransitions(TRANSITIONSfilename,translate2SBML=-
True)
# Add measured labeling information
reacNet.addLabeling(MSfilename,'LCMS',MSSTDfilename,
minSTD=0.001)
# Add feed labeling information
reacNet.addFeed(FEEDfilename)
# Limit fluxes to 500
reacNet.capFluxBounds(500)
# Create sbml file to store the two-scale model.
# All input files are combined in a tuple of the type:
(fileName, string of contents)
SBMLfile = ('SciMM904'+strain+'TS.xml',reacNet.write('to-
String'))
```

3.6.2 Re-Creating the Two-Scale Metabolic Model

We again use the SBML file we just created to create a two-scale model [12] that we will use to calculate fluxes through 2S-^{13}C MFA:

```
In [22]:
TSmodel = FM.TwoSC13Model(('SciMM904'+strain+'TS.xml',reac-
Net.write('toString')))
```

TSmodel now contains all the information needed to calculate fluxes along with the methods to do this calculation and other analysis [12].

3.6.3 Recalculating Internal Metabolic Fluxes Through 2S-^{13}C MFA

We can now use the *findFluxesRanges* method in *TSmodel* to find the fluxes that best fit the experimentally obtained metabolite labeling data (MDVs) and find the ranges of fluxes compatible with this labeling data and the corresponding experimental error:

```
In [23]:
%%time
fluxNames =
TSmodel.reactionNetwork.C13ReacNet.reactionList.getReaction-
NameList(level=1)
TSresult =
TSmodel.findFluxesRanges(Nrep=30,fluxNames=fluxNames,proc-
String='proc')
```

Again, *Nrep* represents the number of replicates used for the calculation. Since the problem to be solved is a nonconvex problem there is no guarantee that a single run will find the best global fit. Hence, we run 30 independent processes and keep the one that best fits the data.

We can now explore the new results through the reactionList methods. For example, we can print some of the new fluxes:

```
In [24]:
TSresult.reactionNetwork.reactionList.printFluxes(brief="-
True",names="exchange")
EX_co2_e_: 13.7769121853
EX_glc_e_: -8.5
EX_etoh_e_: 8.5
EX_h2o_e_: 6.90234889266
EX_o2_e_: -4.69606089266
EX_h_e_: 2.93522510734
EX_glyc_e_: 2.45
EX_nh4_e_: -1.3431504
EX_glx_e_: 1.21173030734
EX_ac_e_: 0.45
biomass_SC5_notrace: 0.24
EX_pi_e_: -0.0474479999999
EX_ttdca_e_: 0.03
EX_ddca_e_: 0.02
EX_so4_e_: -0.018552
EX_hdca_e_: 0.0143
EX_ocdca_e_: 0.00193
```

Again, we test to make sure that the assumptions used in 2S-^{13}C MFA hold using ELVA.

```
In [25]:
resultELVA = TSmodel.ELVA(TSresult)
```

We now plot ELVA results as an x-y graph.

```
In [26]:
resultELVA.plotExpvsCompLabelxvsy(titleFig="WT",outputFileNa-
me="ELVAComparisonWT.txt",save="ELVA-W.eps")
```

See Fig. 5.

The error bars in the y-axis (computational error) are now of the same order of magnitude as the experimental error (x-axis), which implies that our carbon reactions for the core of metabolism now satisfy the two-scale model assumptions [12]. We now have biophysical consistency in the calculated fluxes and can move on to analyzing and visualizing fluxes. If a metabolite still had large fluctuations at this point in the analysis we would need to add more reactions to the core set and repeat until fluctuations in the ELVA plot were small.

3.7 Visualizing Flux Profiles

Once the metabolic fluxes have been calculated they can be understood visually via their plotting on a flux map. In the jQMM library fluxes can be plotted via the commands:

```
In [27]:
TSresult.drawFluxes('wt.svg',svgInFileName='SciMM904TKs.svg',
```

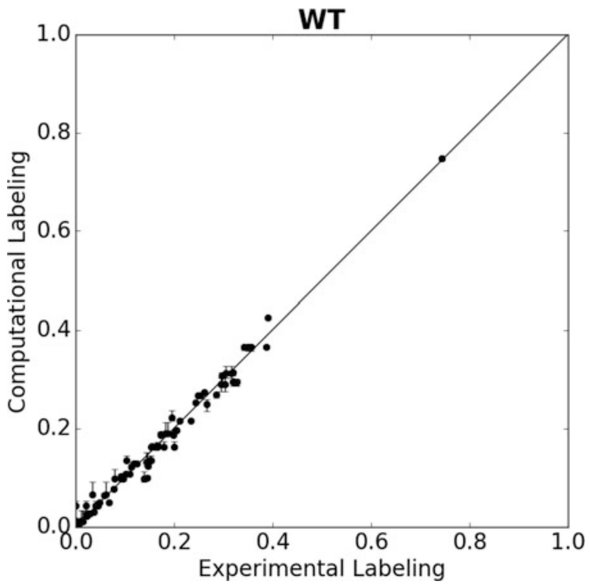

Fig. 5 ELVA plot which shows an x-y graph of the experimentally determined isotope labeling versus the computationally predicted labeling. Since more reactions have been added to the core, all reactions now have only small computational errors in predicted computational labeling, confirming that noncore reactions do not significantly contribute to core metabolite labeling

```
norm='EX_glc_e_')
svgin: /scratch/david.ando/quantmodel/code/core/SciMM904TKs.
svg
```

where 'SciMM904TKs.svg' is the base flux map contained in the jQMM library [10]. The *drawFluxes*( ) method will indicate the flux magnitude on the base flux map in two ways: visually by changing the flux arrow width according to the flux magnitude through a reaction, and also numerically by showing the net flux value (with confidence intervals) next to the reaction:

The command 'SVG' displays the flux map in the Jupyter notebook which is contained in the svg file which was saved locally.

```
In [28]:
SVG(filename='wt.svg')
Out[28]:
```

See Figs. 6a and b.

Fluxes can also be displayed in a more sophisticated manner using the web browser-based flux plotting library Arrowland (http://public-arrowland.jbei.org).

3.8 Predicting Which Genes to Knockout Using MoMA and ROOM

So far, we have used targeted metabolomic data from ^{13}C labeling experiments to infer the underlying internal metabolic fluxes in the cell. We will now use these inferred fluxes along with two COBRA methods to predict which genes to knock out in order to increase the production of ethanol. These methods are MoMA (Minimization of Metabolic Adjustment [18]) and ROOM (Regulatory On/Off Minimization [19]). MoMA provides an approximate solution for a sub-optimal growth flux state after a knockout has been made to an organism, which is nearest in flux distribution to the unperturbed state. On the other hand, ROOM aims to

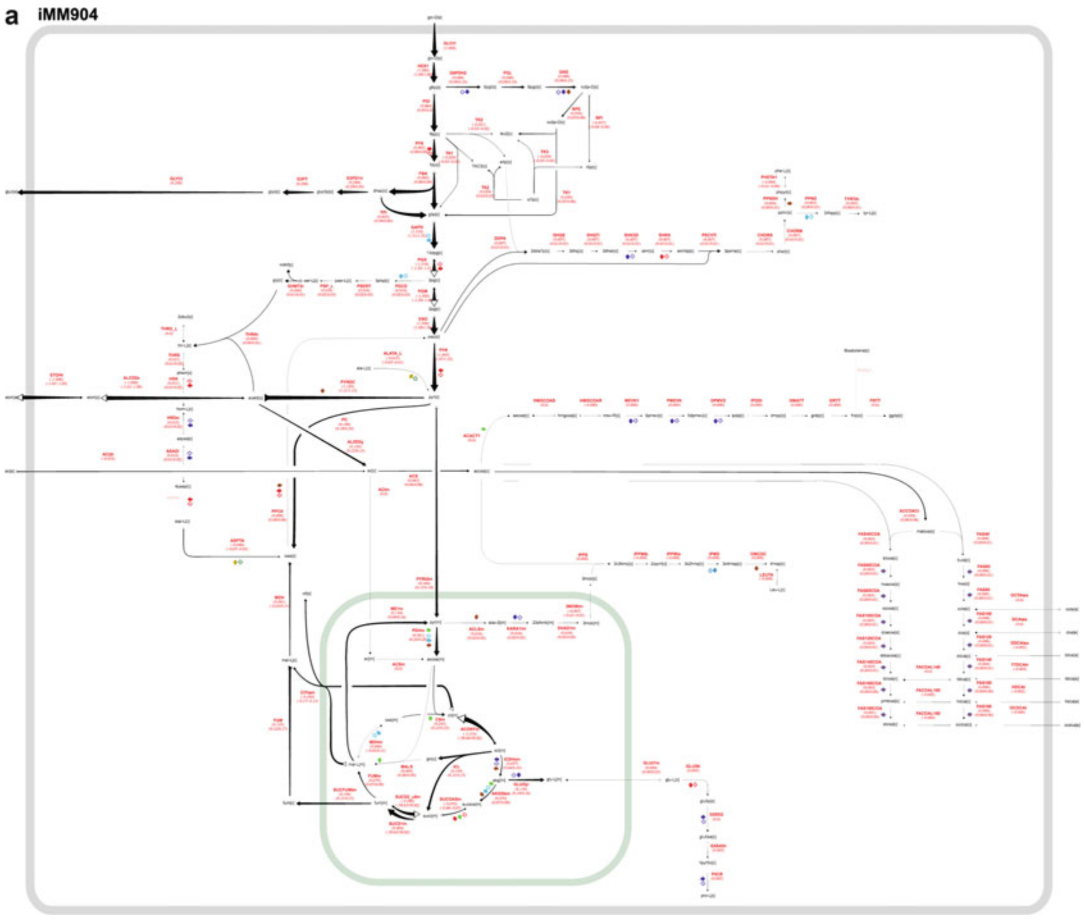

Fig. 6 a) Entire flux map of SciMM904TKs.svg, which displays some of the key 2S-^{13}C metabolic fluxes that are calculated from the iMM904 model for S. cerevisiae. Reactions in the green box are those which are located in the mitochondria. **b**) A zoomed-in plot of these fluxes focused on the upper glycolysis as calculated by the jQMM library using the SciMM904TKs.svg base flux map for a wild type strain of *S. cerevisiae* from Ghosh et al. [8]. This map can be interactively studied using Arrowland (https:/publicarrowland.jbei.org/, wry1Glc). Below each reaction is a single number in parenthesis which indicates the flux value which best fits the experimental data, while the two numbers below the best fit flux represent minimum and maximum fluxes compatible with the MDVs provided by the user

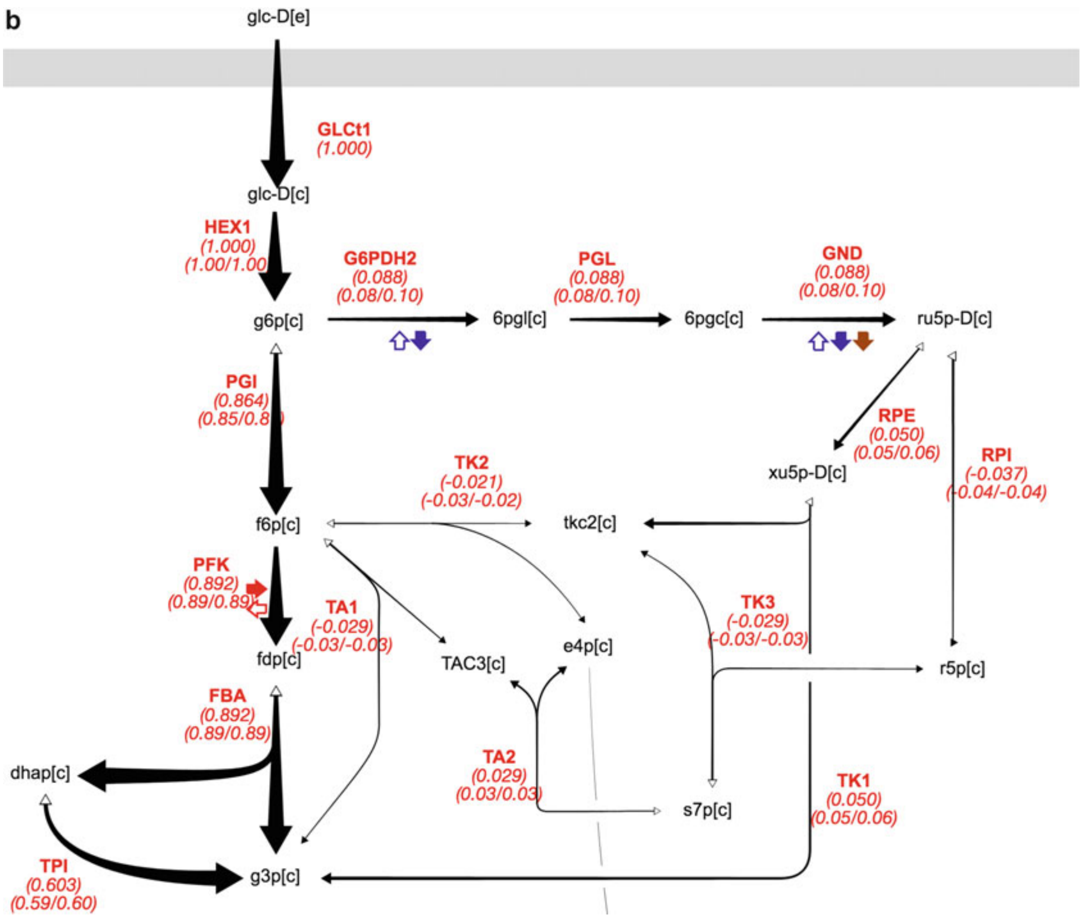

Fig. 6 (continued)

minimize the number of significant flux changes with respect to the wild type to predict resultant fluxes from a knockout of a reaction.

First, we need to specify flexible flux bounds for the final solution in order to avoid biasing the knockout predictions:

```
In [29]:
reactionNetwork = copy.deepcopy(TSmodel.reactionNetwork)
reactionNetwork.changeFluxBounds('GLCt1',core.fluxBounds
(0,10,False)[1])
reactionNetwork.changeFluxBounds('EX_glc_e_',core.fluxBounds
(-10,-0,True,True)[1])
reactionNetwork.changeFluxBounds('biomass_SC5_notrace',core.
fluxBounds( 0,0.25,False)[1])
reactionNetwork.changeFluxBounds('EX_ac_e_',core.fluxBounds
(0,0.5,True,True)[1])
reactionNetwork.changeFluxBounds('EX_etoh_e_',core.fluxBounds
(0,10,True,True)[1])
```

Then we can calculate the base flux profiles for MoMA and ROOM:

```
In [30]:
TSresult = TSmodel.findFluxesStds(Nrep=30,Nrand=10)
```

We then specify a list of reactions to knock out and determine resultant fluxes.

```
In [31]:
KOs = ['PYRt2m','RPI']
```

For reference, we determine the amount of ethanol production in the base WT strain:

```
In [32]:
fluxDict = TSresult.reactionNetwork.reactionList.getReaction-
Dictionary() print       'predicted ethanol flux = ',fluxDict
['EX_etoh_e_'].flux.net.best
predicted ethanol flux = 8.5
```

This implies that before any reaction knockouts that the base strain is releasing 8.5 mmol/gdw/h of ethanol into the environment.

Perform MOMA and ROOM predictions over the set of specified knockouts:

```
In [33]:
for KO in KOs:
 print KO,'knockout:'
 TS13CMOMA = predictions.predict(TSresult, KO, 'MOMA', reac-
tionNetwork.getSBMLString())
 TS13CROOM = predictions.predict(TSresult, KO, 'ROOM', reac-
tionNetwork.getSBMLString())
 fluxDict =
TS13CMOMA.reactionNetwork.reactionList.getReactionDictionary()
  print  ' MoMA predicted ethanol flux = ',fluxDict['EX_e-
toh_e_'].flux.net.best
 fluxDict =
TS13CROOM.reactionNetwork.reactionList.getReactionDictionary()
  print  ' ROOM predicted ethanol flux = ',fluxDict['EX_e-
toh_e_'].flux.net.best
 print '--------------------'
 print ''PYRt2m knockout:
 MoMA predicted ethanol flux = 8.48554566176
 ROOM predicted ethanol flux = 8.756
--------------------RPI knockout:
 MoMA predicted ethanol flux = 8.15179595791
 ROOM predicted ethanol flux = 8.244
--------------------
```

As can be observed, knocking out the PYRt2m reaction is predicted to increase ethanol production by 3.0% according to the ROOM methodology and decrease ethanol production by less than 1% when using the MoMA methodology. As can be seen with an RPI knockout, both MoMA and ROOM predict a decline in ethanol production.

```
-- Jupyter Notebook End --
```

4 Notes

1. Self-installation of the jQMM can be achieved by first downloading the jQMM library from https://github.com/JBEI/jqmm and unpacking the downloaded file. For reproducibility we recommend that the jQMM be used through an interactive python (iPython) notebook server called Jupyter Notebook (http://jupyter.org/) which is a web application that allows for the interactive execution, visualization, and documentation of python code. This integrated computing format greatly enhances repeatability and reproducibility of results within the computational modeling community. Next, usage of the jQMM library can begin by logging into the local Jupyter server using a web browser and navigating to the jQMM folder, and then running some of the example Jupyter notebooks contained within the jQMM. If desired, the Jupyter server can be run directly from the command line within a linux terminal via the command "jupyter notebook". Both GAMS and CONOPT licenses are needed for the jQMM to be fully functional and must be purchased separately.
2. If choosing to run the jQMM docker container on a web-based platform one will avoid the upfront costs of having to purchase an expensive high performance server system, but over the lifetime of a purchased server it may be more inexpensive than a web-based solution (depending on intended usage intensity). Pricing for cloud computing services is typically based on usage, which allows for the ability to automatically adjust the usage of computational services to as much or as little as needed, at any time. When choosing a type of instance to use on a cloud based system we recommend instances which focus on computational speed and not RAM size or disk drive access speed. On the AWS, this includes the instances of type M4 and C4, with the C4 instance currently featuring the highest performing processors and the lowest price/compute performance ratio offered by AWS (circa 2016).

3. Proper quality control of MDV data is crucial to proper determination of fluxes. Experiments should be designed to include internal controls, and should include several biological and technical replicates.
4. The use of labeled glucose of the form U-^{13}C glucose (universal labeling of carbon atoms) together with 1-^{13}C glucose (first carbon atom only is labeled) in an 20:80 ratio is convenient because the 1-^{13}C glucose provides resolution on the activity of the pentose phosphate cycle, while the U-^{13}C glucose provides insight into the activity of reactions which mix carbon backbones of substrates. The pentose phosphate pathway loses the first labeled carbon as CO_2, while other reactions within central metabolism that mix carbon backbones can be resolved by having a combination of U-^{13}C and 1-^{13}C glucose present.
5. It is not necessary to follow this naming convention of using uppercase for metabolites and lowercase for atom transitions. The parentheses delimit the start and end of each atom transition. Any alphabetic, numeric, or underscore character comprising the regular expression [a-zA-Z0-9] can be included in the metabolite names and atom transitions.
6. Metabolite abbreviations used:
   (a) akg, 2-Oxoglutarate,
   (b) co2, carbon dioxide,
   (c) fum, fumarate,
   (d) g3p, glyceraldehyde 3-phosphate,
   (e) pyr, pyruvate,
   (f) accoa, acetyl-CoA,
   (g) asp, aspartate,
   (h) f6p, D-fructose 6-phosphate,
   (i) g6p, D-glucose 6-phosphate,
   (j) glu, glutamate,
   (k) glx, glyoxylate,
   (l) glyc3p, glycerol 3-phosphate,
   (m) glc-D, glucose,
   (n) fdp, D-fructose 1,6-bisphosphate,
   (o) g3p, glyceraldehyde 3-phosphate,
   (p) 3mob, 3-methyl-2-oxobutanoate,
   (q) hdca, hexadecanoate.
7. Reactions that must be specified in the REACTIONS.txt file, in terms of carbon atom transition information, are those that utilize metabolites for which ^{13}C isotopolog data are input into the jQMM library and for reactions which are considered to be

at the core of the metabolic network. Finally, this file also specifies a metabolite which serves as the ^{13}C carbon source (typically glucose, i.e., glcD[e], via the &SOURCE command) and input reactions which bring this ^{13}C labeled metabolite into the cell.

8. The number (1.5% in our example FLUX.txt file) at the beginning of a feed definition specifies the total glucose percentage of the initial cell culture. The percentage of 1-C glucose, which has its first carbon atom labeled, is specified next, together with the percentage of U glucose, which is uniformly labeled among all the glucose carbon atoms, and finally the percentage of normal glucose which is completely unlabeled (UN).
9. Free open-source software tools, such as the jQMM, provide for universal accessibility and unlimited modification and customization. Overall, we wish that the community can also support the jQMM's further development by submitting bug fixes to the github repo (https://github.com/JBEI/jqmm) and including any functional extensions that different research groups have achieved. Community driven development will help in the adoption of fluxomics tools like the jQMM which are free, flexible, and cater to the needs of the metabolic engineering community.

Acknowledgments

This work was part of the DOE Joint BioEnergy Institute (http://www.jbei.org) supported by the US Department of Energy, Office of Science, Office of Biological and Environmental Research, through contract DE-AC02-05CH11231 between Lawrence Berkeley National Laboratory and the US Department of Energy. The US Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this chapter, or allow others to do so, for US Government purposes; and the publisher, by accepting the chapter for publication, acknowledges the same.

References

1. Kitney R, Freemont P (2012) Synthetic biology—the state of play. FEBS Lett 586:2029–2036. https://doi.org/10.1016/j.febslet.2012.06.002
2. Nielsen J, Keasling JD (2016) Engineering cellular metabolism. Cell 164:1185–1197. https://doi.org/10.1016/j.cell.2016.02.004
3. Chubukov V, Mukhopadhyay A, Petzold C, Keasling J (2016) Synthetic and systems biology for microbial production of commodity chemicals: from target selection to scale-up. npj Syst Biol Appl 16009:1–11. https://doi.org/10.1038/npjsba.2016.9
4. Nakamura CE, Whited GM (2003) Metabolic engineering for the microbial production of 1,3-propanediol. Curr Opin Biotechnol 14:454–459. https://doi.org/10.1016/j.copbio.2003.08.005

5. Yim H, Haselbeck R, Niu W et al (2011) Metabolic engineering of Escherichia coli for direct production of 1,4-butanediol. Nat Chem Biol 7:445–452. https://doi.org/10.1038/nchembio.580
6. Paddon CJ, Westfall PJ, Pitera DJ et al (2013) High-level semi-synthetic production of the potent antimalarial artemisinin. Nature 496:528–532. https://doi.org/10.1038/nature12051
7. Van Dien S (2013) From the first drop to the first truckload: commercialization of microbial processes for renewable chemicals. Curr Opin Biotechnol 24:1061–1068. https://doi.org/10.1016/j.copbio.2013.03.002
8. Ghosh A, Ando D, Gin J et al (2016) 13C metabolic flux analysis for systematic metabolic engineering of S. cerevisiae for overproduction of fatty acids. Front Bioeng Biotechnol 4:76. https://doi.org/10.3389/fbioe.2016.00076
9. Lewis NE, Nagarajan H, Palsson BO (2012) Constraining the metabolic genotype-phenotype relationship using a phylogeny of in silico methods. Nat Rev Microbiol 10:291–305. https://doi.org/10.1038/nrmicro2737
10. Birkel G, Ghosh A, Vinay K et al The JBEI quantitative metabolic modeling library (jQMM): a python library for modeling microbial metabolism. Microb Cell Factories In review
11. Wiechert W (2001) 13C metabolic flux analysis. Metab Eng 3:195–206. https://doi.org/10.1006/mben.2001.0187
12. Garcia Martin H, Kumar VS, Weaver D et al (2015) A method to constrain genome-scale models with 13C labeling data. PLoS Comput Biol. https://doi.org/10.1371/journal.pcbi.1004363
13. Antoniewicz MR, Kraynie DF, Laffend LA et al (2007) Metabolic flux analysis in a nonstationary system: fed-batch fermentation of a high yielding strain of E. coli producing 1,3-propanediol. Metab Eng 9:277–292. https://doi.org/10.1016/j.ymben.2007.01.003
14. Schaub J, Mauch K, Reuss M (2008) Metabolic flux analysis in Escherichia coli by integrating isotopic dynamic and isotopic stationary 13C labeling data. Biotechnol Bioeng 99:1170–1185. https://doi.org/10.1002/bit.21675
15. Moxley JF, Jewett MC, Antoniewicz MR et al (2009) Linking high-resolution metabolic flux phenotypes and transcriptional regulation in yeast modulated by the global regulator Gcn4p. Proc Natl Acad Sci 106:6477–6482. https://doi.org/10.1073/pnas.0811091106
16. Kajihata S, Matsuda F, Yoshimi M et al (2014) 13C-based metabolic flux analysis of Saccharomyces cerevisiae with a reduced Crabtree effect. J Biosci Bioeng 120:140–144. https://doi.org/10.1016/j.jbiosc.2014.12.014
17. Schellenberger J, Que R, Fleming RMT et al (2011) Quantitative prediction of cellular metabolism with constraint-based models: the COBRA toolbox v2.0. Nat Protoc 6:1290–1307. https://doi.org/10.1038/nprot.2011.308
18. Segrè D, Vitkup D, Church GM (2002) Analysis of optimality in natural and perturbed metabolic networks. Proc Natl Acad Sci U S A 99:15112–15117. https://doi.org/10.1073/pnas.232349399
19. Shlomi T, Berkman O, Ruppin E (2005) Regulatory on/off minimization of metabolic flux changes after genetic perturbations. Proc Natl Acad Sci U S A 102:7695–7700. https://doi.org/10.1073/pnas.0406346102
20. Suthers PF, Burgard AP, Dasika MS et al (2007) Metabolic flux elucidation for large-scale models using 13C labeled isotopes. Metab Eng 9:387–405. https://doi.org/10.1016/j.ymben.2007.05.005
21. Toya Y, Ishii N, Hirasawa T et al (2007) Direct measurement of isotopomer of intracellular metabolites using capillary electrophoresis time-of-flight mass spectrometry for efficient metabolic flux analysis. J Chromatogr A 1159:134–141. https://doi.org/10.1016/j.chroma.2007.04.011
22. Zamboni N, Fendt S-M, Rühl M, Sauer U (2009) 13C-based metabolic flux analysis. Nat Protoc 4:878–892. https://doi.org/10.1038/nprot.2009.58
23. Reed JL, Vo TD, Schilling CH, Palsson BO (2003) An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR). Genome Biol 4:R54. https://doi.org/10.1186/gb-2003-4-9-r54
24. Orth JD, Conrad TM, Na J et al (2011) A comprehensive genome-scale reconstruction of Escherichia coli metabolism--2011. Mol Syst Biol 7:535. https://doi.org/10.1038/msb.2011.65
25. Feist AM, Henry CS, Reed JL et al (2007) A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. Mol Syst Biol 3:121. https://doi.org/10.1038/msb4100155