1 Metabolic Networks : Dynamics

Recall from Lecture 11 that we can reconstruct (most of) a species' metabolic network from its genome by cross-referencing the genes in the genome with the enzymes they code for and then the corresponding biochemical reactions. This process is sometimes called **genome-scale network reconstruction**, and gives us a systems-level view of an organism's entire metabolic system.[1]

Today, metabolic network reconstruction is fairly well automated. For instance, the Python package `CarveMe` https://carveme.readthedocs.io/ provides a single tool for downloading a publicly indexed prokaryotic genome, identifying its enzymes, cross-referencing them with known reactions in the BiGG database, and returning the corresponding *bipartite network* of metabolites and the reactions the participate in.

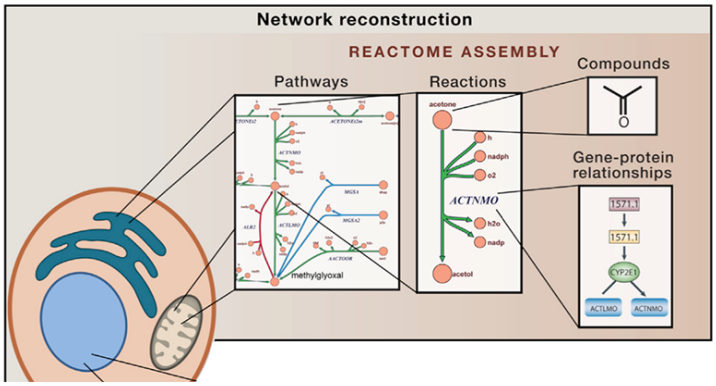


Figure 1. Network Reconstruction
An organism's reactome can be assembled in a way that is analogous to DNA-sequencing assembly. (Right to left) First the interacting compounds must be identified. Then, the reactions acting on these compounds are tabulated and the protein that catalyzes the reaction and the corresponding open reading frame is identified in the organism of interest. These reactions are assembled into pathways that can be laid out graphically to visualize a cell's metabolic map at the genome scale. Several tools for reactome assembly and curation exist, including the COBRA Toolbox (Ebrahim et al., 2013; Schellenberger et al., 2011b), KEGG (Kanehisa et al., 2014), EcoCyc (Keseler et al., 2013), ModelSeed (Henry et al., 2010b), BiGG (Schellenberger et al., 2010), Rbionet (Thorleifsson and Thiele, 2011), Subliminal (Swainston et al., 2011), Raven toolbox (Agren et al., 2013), and others.

Given a reconstructed metabolic network, we can use a computational technique called **flux balance analysis** (FBA) to investigate any number of computational questions about how an organism's metabolism is likely to behave under different conditions. FBA is most well-developed for prokaryotic model organisms like *E. coli*, where (1) our reference sets of metabolic reactions are the most complete, and (2) complexities from regulation and cell-type specialization are absent.

Examples of questions we can explore with FBA include:

- Given certain inputs to a metabolic network G, what is the maximum biomass growth rate we can expect?
- How fast will a given network produce a particular metabolite of interest?
- How robust is the system (e.g., growth) to removal of specific reactions?
- How do production rates differ between aerobic (with oxygen) and anaerobic (without oxygen) conditions?

[1] Figure reproduced from O'Brien, Monk, Palsson, *Cell* **161**, 971 (2015).

1.1 Network dynamics (a brief aside)

These questions are all examples of **network dynamics**, and FBA is a formal model of dynamics that run across the edges of a network. In our unit on protein interaction networks, the computations that signaling networks perform are another kind of network dynamics (because the signals run across the edges), although we did not formalize it to the same extent as we will do here for FBA. This kind of *dynamics **on** a network* is distinct from *dynamics **of** a network*, in which the edges themselves are changing—we saw that kind of model when we discussed both how protein networks get their shape (the DMC model), and how metabolic networks get their shape (the toolbox model).

In the first half of the course, we focused on building intuition about and developing tools for analyzing the **structure** of networks because structure constrains dynamics. The structure of a metabolic network strongly influences the fluxes that are allowed across its edges. If we change the structure, e.g., by making the degree distribution more or less heavy tailed or by increasing or decreasing the number of triangles, then we will likely also change its dynamical properties. The combination of structure and dynamics is what allows us to investigate biological function. FBA is a tool that lets us really make this leap to function, because we're going to build a given function (producing biomass) into the dynamics and then probe how it varies as a function of structure and dynamics.

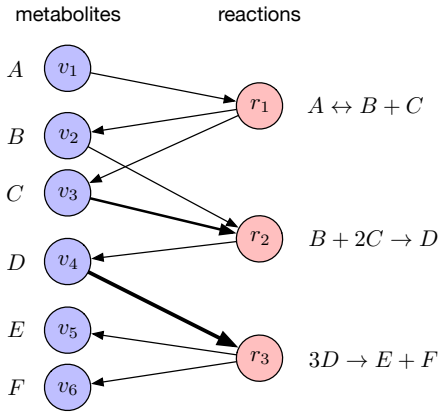
1.2 The stoichiometric matrix

The key component of FBA is a model of metabolism as an $n_m \times n_r$ (bipartite) stoichiometric matrix $\mathbf{S}$, with n_m metabolites and n_r reactions. A matrix entry S_{ij} thus gives the rate at which reaction j consumes or produces metabolite i. If $S_{ij} < 0$, then the metabolite is consumed, while if $S_{ij} > 0$, it's produced. Converting the extracted metabolic network into an appropriate stoichiometric matrix requires two things: (1) we need the metabolism to be represented as a bipartite network of metabolites and the reactions the participate in, and (2) we need to know which metabolites are inputs and which are outputs, for each reaction (e.g., by using directed edges).

For example, consider the following three reactions:

$$\begin{aligned} A &\leftrightarrow B + C \\ B + 2C &\rightarrow D \\ 3D &\rightarrow E + F \end{aligned}$$

These six metabolites and three reactions can be represented as the following directed bipartite metabolite-reaction graph G or stoichiometric matrix $\mathbf{S}$:

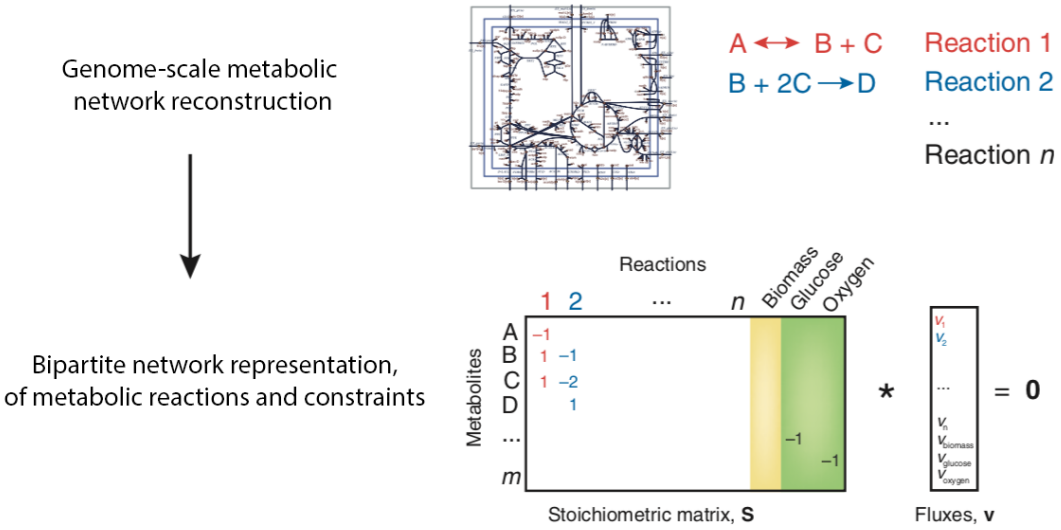


stoichiometric matrix

v_i	r_1	r_2	r_3
A	-1	0	0
B	1	-1	0
C	1	-2	0
D	0	1	-3
E	0	0	1
F	0	0	1

As described above, compounds on the left (substrates) of a reaction equation have negative values in the corresponding reaction column of **S**, while compounds on the right (products) have positive values. In the above example, you can see that the input to this metabolic pathway is compound A, and the outputs are compounds E and F.

Just like most real-world networks, **S** is a sparse matrix because, on average, compounds participate in a small number of reactions each. Notably, this representation of metabolism does not capture the energetics of these reactions, only the mass flow from substrates to products.[2]



[2]Figure adapted from Orth, Thiele, Palsson, *Nature Biotechnology* **28**, 245-248 (2010) `https://www.nature.com/articles/nbt.1614`

Performing FBA is equivalent to solving a kind of constraint satisfaction problem, and there are two kinds of constraints in the analysis:

1. (**fluxes**) The stoichiometric matrix **S** imposes a set of **flux constraints** on how metabolites flow through the network. These constraints require that the total amount of any particular metabolite or compound being produced in the system must equal the total amount being consumed, **at steady state**. The assumption of a steady state is one about time scales: it assumes that reactions happen quickly, relative to changes in the stoichiometric matrix. (Changes in the stoichiometric matrix can be generated by regulation, e.g., by turning on or off some metabolic pathway.)
2. (**input / output**) In addition to the regular reactions round in the genome, we typically add to **S** three special reactions, representing typical **inputs and outputs** of metabolism: (1) *glucose consumption*, (2) *oxygen consumption*, and (3) *biomass production*. We can also impose **bounding constraints** on reactions, which set the upper limit on allowable fluxes. Bounding constraints are how we do our experiments, e.g., by setting oxygen consumption to 0, thereby simulating anaerobic metabolism.

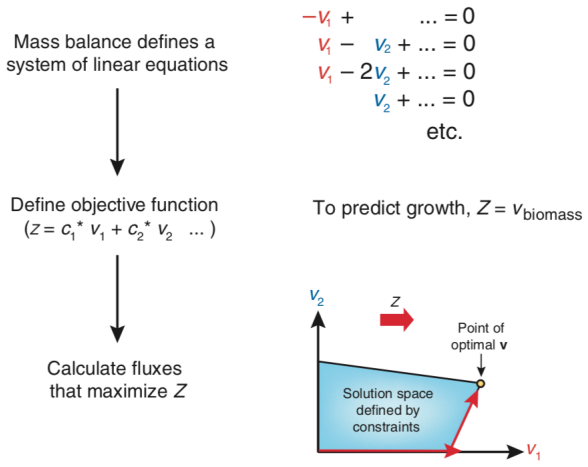
Biomass output is a special, synthetic reaction, which converts selected intermediate metabolites into a generic "biomass," e.g., nucleic acids, proteins and lipids, that we use as a proxy for organismal growth. The stoichiometric coefficients for the biomass reaction are typically based on experimental measurements, and are scaled so that the flux through the biomass reaction represents the standardized growth rate of the organism.

1.3 Solving for flux

If we define a flux vector $\mathbf{v}$, which is a $n_r \times 1$ vector, with each element v_j giving the flux for a particular reaction j, then we can quantify the flows across the entire network. That is, each choice of a flux vector $\mathbf{v}$ specifies a flow of resources from input to outputs, across the various metabolic pathways defined by **S**. At **steady state**, all fluxes, except the inputs and outputs, must be **balanced**, which implies a kind of matrix equation $\mathbf{S}\,\mathbf{v} = 0$.

Under this constraint, not all choices of $\mathbf{v}$ are allowed, because they fail to balance some fluxes. But, a key feature of the FBA model is that there are many more reactions than metabolites, i.e., $n_r > n_m$, which implies that there are many choices of flux vectors $\mathbf{v}$ that *are* allowed. Some of these produce more biomass production than others, and we can imagine searching for the $\mathbf{v}$ that **maximizes** flux through the biomass reaction. We find this by casting the problem as a **linear program**, and then solving it using something like the **simplex algorithm** (typically via a standard linear program solver).[3]

[3]Figure adapted from Orth, Thiele, Palsson, *Nature Biotechnology* **28**, 245-248 (2010) `https://www.nature.com/articles/nbt.1614`



Predictions: The optimal flux solution **v** can be interpreted as a set of predictions, e.g., a prediction of a growth rate given certain input conditions, which can then be tested experimentally. Similarly, knocking out some reactions (forcing their fluxes to 0), produces testable predictions. Incorrect predictions, e.g., a prediction of growth under a given circumstances but an experimental observation of no growth, or vice versa, can lead to biological discovery.

Limitations: FBA is a powerful tool for modeling how metabolites flow across the metabolic network, under the assumption of a steady state. But, it has three key limitations:

1. Although fluxes are guaranteed to be balanced under the model, FBA cannot estimate metabolite concentrations.
2. Most FBA cannot account for the effects of regulation, e.g., activation or inhibition of enzymes by things like protein kinases or regulation by gene expression.
3. FBA focuses mainly on generic biomass production, which is not the only metabolic output that may be of interest.

1.4 Simulating a FBA knockout experiment

There are several Python packages available for running flux balance analysis:

- **CobraPy**, which provides an interface around using the commonly used COBRA optimization methods in the context of flux balance analysis, flux variability analysis, and gene deletion analyses. `https://opencobra.github.io/cobrapy/`
- **PyFBA**, which allows a user to build network models from genomes, gapfill them to estimate a more complete network, and then run flux-balance-analysis on that model. `http://linsalrob.github.io/PyFBA/`

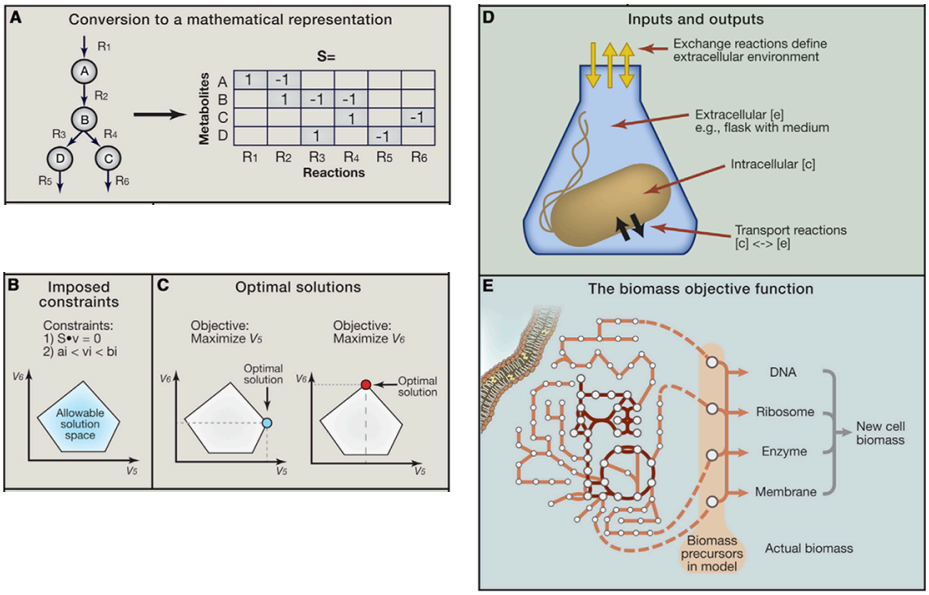


Figure 1: Adapted from O'Brien, Monk, Palsson, *Cell* **161**, 971 (2015).

1.4.1 An example using *E. coli*'s metabolism

To illustrate how FBA can work in practice, we'll use the built-in genome scale model *E. coli iJO1366* within `CobraPy` to simulate the impact of a "knock out" experiment on *E. coli*'s core metabolism and hence on its growth rate.[4] [5]

The model contains 1367 genes, $n_r = 2583$ reactions, and $n_m = 1805$ metabolites. It also contains several "output" nodes that represent different kinds of biomass production – these are groups of intermediate molecules that collectively are necessary for cell division. Hence, maximizing the flux into a biomass node, given levels for input nodes like oxygen and glucose, and the constraints given by the metabolic network itself, can be interpreted as making a prediction about how an organism will respond in a laboratory setting. Crucially, the solver will search for a solution $\mathbf{v}$ so that all the "internal" nodes of the metabolic network will be "flux balanced," meaning the input flux equals their output flux.

We begin by importing the model, and then (1) setting the objective function to point to a particular biomass node of interest, and then (2) setting minimum (typically negative) values for the flux into

[4] This model is just one of 2606 whole genome metabolism models available from the European Molecular Biology Laboratory (EMBL): `https://www.ebi.ac.uk/biomodels-main/path2models?cat=genome-scale`

[5] The example follows the one given in Keesha Erickson's "FBA with CobraPy" tutorial from June 2018, `https://bit.ly/3eiZUia` (points to `https://cnls.lanl.gov/`).

the input glucose and oxygen nutrients. These values represent the absolute maximum amount of nutrients available to metabolism. Finally, we apply the optimizer to find a set of fluxes $\mathbf{v}$ that maximize the flux into objective biomass reaction, and print out both the optimized growth rate and a summary of the fluxes.

```
import cobra
import cobra.test

model = cobra.test.create_test_model("ecoli") # import the model

# set the biomass objective
model.reactions.get_by_id("BIOMASS_Ec_iJO1366_core_53p95M").objective_coefficient = 0
model.reactions.get_by_id("BIOMASS_Ec_iJO1366_WT_53p95M").objective_coefficient   = 1.0

# set input constraints
model.reactions.get_by_id("EX_glc__D_e").lower_bound = -10  # input, glucose
model.reactions.get_by_id("EX_o2_e").lower_bound = -15      # input, oxygen

solution = model.optimize()

print(f'Growth Rate: '+str(solution.objective_value)+' 1/h')
model.summary()
```

This yields a growth rate value of `0.9009 1/h`, which provides a numerical reference point against which to compare growth rates under other experimental settings. The summary statement provides some insight into the particular fluxes the optimizer has chosen, but the details of the fluxes are given the `solution` data structure. We can visualize these using `CobraPy`'s companion tool `Escher` (`http://escher.github.io/`), which overlays the fluxes onto a visualization of *E. coli*'s core metabolism (shown below).

Now let's compare this solution to one in which we "knock out" a single key reaction, which will force the optimizer to route mass across other reactions in the network in order to maximize the biomass output. In practice, a knockout is accomplished by constraining the flux across a particular reaction to be 0. There are many choices of reactions to knock out, each of which will produce a different solution. To illustrate this behavior, we'll choose the central `FUM` reaction, which is part of the Krebs cycle.

```
model.reactions.FUM.knock_out() # knockout FUM

solution = model.optimize() # reoptimize the model

print(f'Growth Rate: '+str(solution.objective_value)+' 1/h')
model.summary()
```

This yields a growth rate value of `0.84776 1/h`, which is only slightly lower that the previous one. This modest difference to the knockout illustrates the structural robustness of *E. coli*'s metabolism to such a loss. But, it is important to remember that these models are highly simplified, and we are

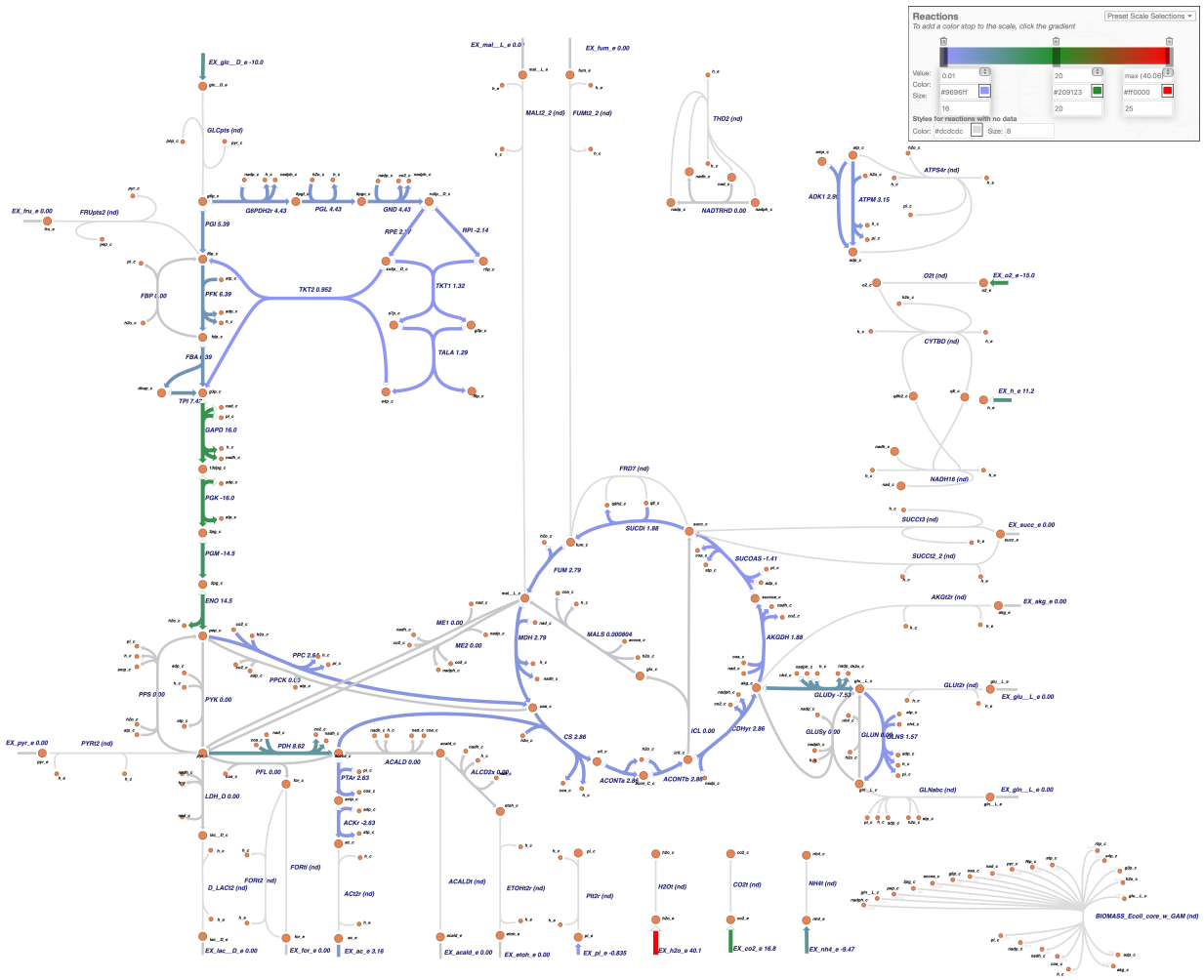


Figure 2: *Escher* visualization of a simple aerobic optimization of *E. coli*'s core metabolism.

only optimizing the biomass node, i.e., the growth rate. That is not to say that an actual *E. coli* without the `FUM` reaction would be able to do *other* key cellular functions. Visualizing the resulting fluxes illustrates how the optimizer reroutes the fluxes. Can you see the differences?

Now imagine knocking out other reactions. Can we find one that drops the growth rate to very close to 0? Can we find one that increases the growth rate? Or, does increasing the glucose supply increase the flux that goes through to BIOMASS? How high a growth rate can the network sustain? How many reactions can we knock out and still get some growth? How much do "core"

vs. "periphery" reactions matter for altering the growth rate? Flux balance analysis provides a computational platform for exploring such questions, to identify the most interesting predictions to test in the laboratory.

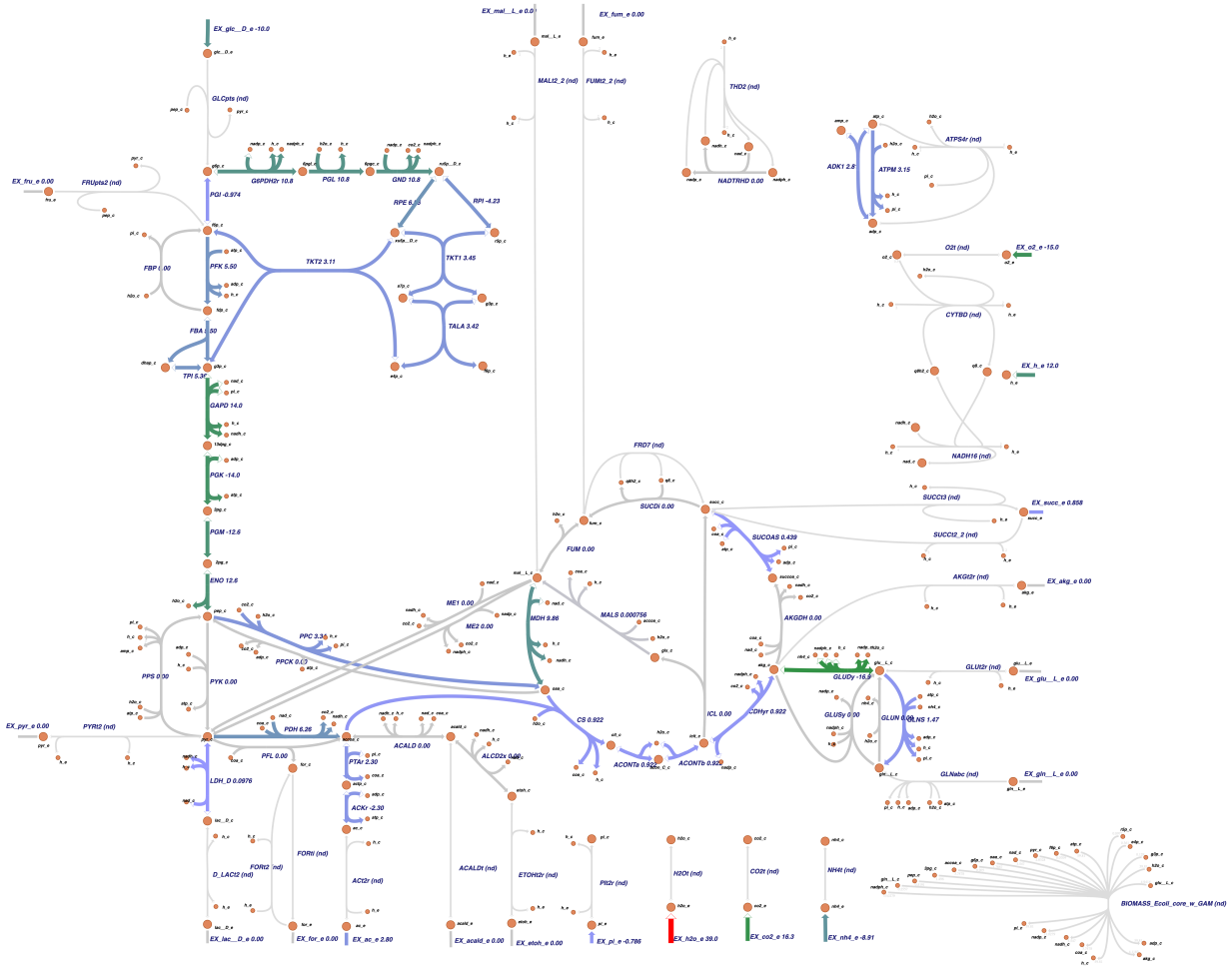


Figure 3: *Escher* visualization of a simple aerobic optimization of *E. coli*'s core metabolism, after knocking out the `FUM` reaction in the central Krebs cycle.

Supplemental readings

1. Orth, Thiele, & Palsson, "What is flux balance analysis?" *Nature Biotechnology* **28**, 245-248 (2010).
   `https://www.nature.com/articles/nbt.1614`

2. King et al., "BiGG models: A platform for integrating, standardizing and sharing genome-scale models." *Nucleic Acids Research* (2015)
   `https://academic.oup.com/nar/article/44/D1/D515/2502593`