E.coli Core Model for Beginners

Author: H. Scott Hinton, Utah State University

Reviewer:

INTRODUCTION

The purpose of this tutorial is to show a beginner how the COBRA Toolbox can be used to explore the physiology of a cell. To illustrate the capabilities of the COBRA Toolbox [1, 2], this tutorial will focus on exploring the attributes of the *E.coli* core model that was developed by Orth, Fleming, and Palsson [3] and is included in the standard COBRA toolbox installation. This tutorial **will not** focus on the detailed physiology represented by the core model since that has been published in detail elsewhere [3], but **will** focus on how to use the COBRA Toolbox to explore the *E.coli* core or any other COBRA-based model.

This tutorial will include pragmatic discussions of the following topics:

1. The limitations of constraint-based modeling.
2. Basic components of a COBRA model including: A) genes, B) reactions, C) metabolites, D) gene-protein-reaction associations, E) constraints, and F) objective functions.
3. An overview of flux balance analysis.
4. The subsystems of the *E.coli* core model including sections on A) energy management of the cell, B) glycolysis pathway, C) pentose phosphate pathway, D) tricarbonoxylic acid cycle, E) glycoxylate cycle, gluconeogenesis, and anapleurotic reactions, F) fermentation, and G) the nitrogen metabolism.

MATERIALS

This tutorial is based on the *Constraint-Based Reconstruction and Analysis* (COBRA) Toolbox [2,3] that is currently under development. To use this tutorial will require the 2016a or newer version of Matlab (https://www.mathworks.com/) and the COBRA toolbox software that can be downloaded from https://opencobra.github.io/cobratoolbox/latest/index.html. The installation instructions and troubleshooting tips are also available on this website.

EQUIPMENT SETUP

To use the COBRA toolbox you first have to initalize the Matlab environment to include all the COBRA Toolbox functions. Before initializing the COBRA Toolbox, start Matlab and move to the Matlab directory that you want to be your work directory. The COBRA Toolbox initialization is accomplished with the "initCobraToolbox" function as shown below. *[Timing: < Minute]*

```
%initCobraToolbox

% change to the directory of the tutorial
cd(fileparts(which('tutorial_ecoliCoreModel.mlx')));
```

TROUBLESHOOTING

One of the biggest problems that users of this tutorial face, is that they have not setup the solver correctly before they start the tutorial. This is necessary for the network optimizations required by this tutorial. This can be done by selecting the approrpriate solver for the machine you are using by removing the "%" (comment) sign for only the desired solver. *[Timing: Seconds]*

```
changeCobraSolver('gurobi','all');
```

```
> Gurobi interface added to MATLAB path.
> Solver for LP problems has been set to gurobi.

> Gurobi interface added to MATLAB path.
> Solver for MILP problems has been set to gurobi.

> Gurobi interface added to MATLAB path.
> Solver for QP problems has been set to gurobi.

> Gurobi interface added to MATLAB path.
> Solver for MIQP problems has been set to gurobi.
> Solver gurobi not supported for problems of type NLP. Currently used: matlab
```

PROCEDURE

1. Constraint-based modeling

Both genome-scale metabolic network reconstructions [4] and constraint-based modeling [5,6,7] can be used to model steady-state phenotypes during the exponential growth phase.This can be useful in exploring and understanding the capabilities of each phenotype. It can also be used to identify and modify cellular pathways to favor specific bioproduct producing phenotypes. It is important to understand that most constraint-based models do not

- model transitions between phenotypes,
- include the genes required for the stationary phase (proteases, etc.),
- include the complete transcription and translation pathways.

These constraint-based models are based on a biomass function that represents the average metabolic load required during exponential cell growth. It represents the average percentages of the component parts (amino acids, nucleotides, energy, etc.) that are included in 1 gm dry weight per hour of cell biomass.

Through the use of genome-scale metabolic network reconstructions, Flux Balance Analysis (FBA) [8] can be used to calculate the flow of metabolites through a metabolic network. This capability makes it possible to predict the growth-rate of an organism and/or the rate of production of a given metabolite. It is important that it is understood that FBA has limitations! It does not use kinetic parameters, thus it cannot predict metabolite concentrations. It is also only capable of determining fluxes at steady state. Finally, traditional FBA does not account for regulatory effects such as the activation of enzymes by protein kinases or regulation of gene expression. Therefore, it's predictions may not always be accurate.

In this tutorial, we will show some simple examples using the COnstraint-Based Reconstruction and Analysis (COBRA) toolbox [4,5], a software package that operates in the Matlab (https://www.mathworks.com/) programming environment. As you will see, the COBRA toolbox allows users to explore the operation of a cell model with just a few lines of code. These results can then be used to predict cellular behavior that, in some cases, have been experimentally verified.

2. Basic Components of a COBRA model

The COBRA Toolbox is based on metabolic network reconstructions that are biochemically, genetically, and genomically (BiGG) structured databases composed of biochemical reactions and metabolites [9,10]. They store cellular organism metabolic information such as the reaction stoichiometry, reaction reversibility, and the relationships between genes, reactions, and proteins (enzymes). Although many organisms have similar central metabolic networks, there can be significant differences even between two closely related organisms, thus metabolic network reconstructions are therefore organism specific

[4]. A simplified model of *E.coli,* referred to as the *E.coli* core model, is a great model to explore the analysis and exploration tools available through the COBRA toolbox which is the purpose of this tutorial. The metabolic map of the *E.coli* core model is shown below in Figure 1. In this figure, the larger letters on this map (Glyc, PPP, etc.) refer to the major subsystems included in this simple model which includes; (OxP) oxidative phosphorylation or energy management of the cell, (Glyc) glycolysis pathway, (PPP) pentose phosphate pathway, (TCA) tricarboxylic acid cycle, (Ana) glycoxylate cycle, gluconeogenesis, and anapleurotic reactions, (Ferm) fermentation, and (N) nitrogen metabolism. These subsystems will all be discussed in more detail later in this tutorial.

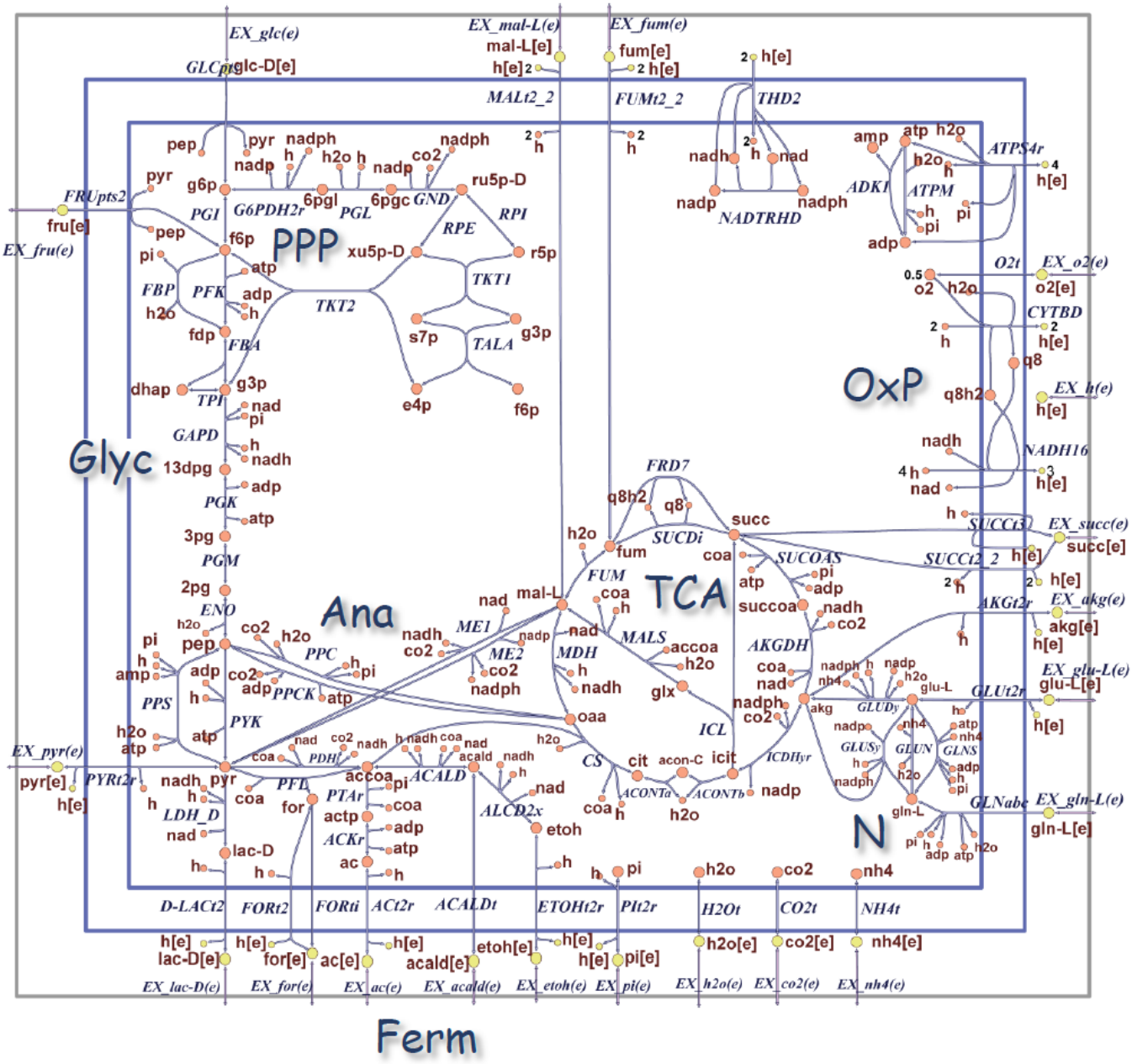


Figure 1. Metabolic map of the *E.coli* core model [3].

A metabolic reconstruction consists of a collection of genes, reactions which represent proteins (enzymes), metabolites, a stochiometric matrix that defines the relationship between the reactions and metabolites, reaction constraints, and a biomass function. The genes in the model are represented by gene name and genomic locus. Metabolites are represented with lowercase characters and typically have a suffix that represents the compartment they operate in. For the simplified *E.coli* core model there are only two compartments represented; cytosolic metabolites with the suffix "[c]" and extracellular metabolites with the suffix "[e]." To keep this model simple, the *E.coli* core model does not distinguish between the periplasmic space and the extracellular medium. The COBRA model's representation of

metabolites includes an abbreviation, the official name, the chemical fomula, and the charge of the metabolite.

Reactions in the COBRA models correspond to the enzymes in a cell and are represented by uppercase abbreviations, an offical name, and a stochiometric formula.The suffixes used in the reaction abbreviations, include; 'i' (irreversible), 'r' (reversible), 'abc' (ATP-Binding Cassette transporter), and 't' (transport). Most of the reactions used in the model are named after the enzymes that catalyze them. As an example, ENO represents the enzyme "enolase" and includes the formula "2pg[c] <=> h2o[c] + pep[c] ." A special type of reaction found in COBRA models are exchange reactions which have the form of "EX_xxx(e)" and are used to secrete/uptake metabolites to/from the extracellular space. As an example, the exchange reaction for glucose is "EX_glc(e)."

The COBRA models also include Boolean rules for each reaction describing the gene-reaction relationship. For example, ‘gene1 and gene2’ indicate that the two gene products are part of an enzyme whereas ‘gene1 or gene2’ indicate that the two gene products are isozymes that catalyze the same reaction.The gene-protein-reaction associations (GPRA) for a few reactions are shown in Figure 2. Each GPRA is composed of a gene locus, a translated peptide (mRNA), and functional proteins that work together to make a single reaction (enzyme). At the top of each GPRA is a portion of the genomic context that is highlighted. Genes in this figure are designated by their locus name and represented by light blue boxes. The translated peptides are represented by the purple boxes, the functional proteins are represented by red ovals, while the reactions are labeled dark blue boxes. As can be seen in the figure, isozymes include two different proteins that are connected to the same reaction. For the case of proteins with multiple peptide subunits, the peptides are connected with an '&' sign above the protein. For complexes of many functional proteins, the proteins are also connected with an '&' sign above the reaction. Certain genes that are responsibible for the creation of a given reaction, such as pykF and pykA, can be encoded by genes in operons that are widely separated on the genome. In this figure operons are represented by shaded rectangles around one or more genes. Genes are represented by rectangles with one side pointed to denote the direction of the sense strand. Other operons can contain multiple genes which encode protein subunits into larger proteins. As an example, the same sdhCDAB-sucABCD operon that codes for the SUCDi proteins also codes for two proteins of the 2-oxoglutarate dehydrogenase enzyme complex, AKGDH.

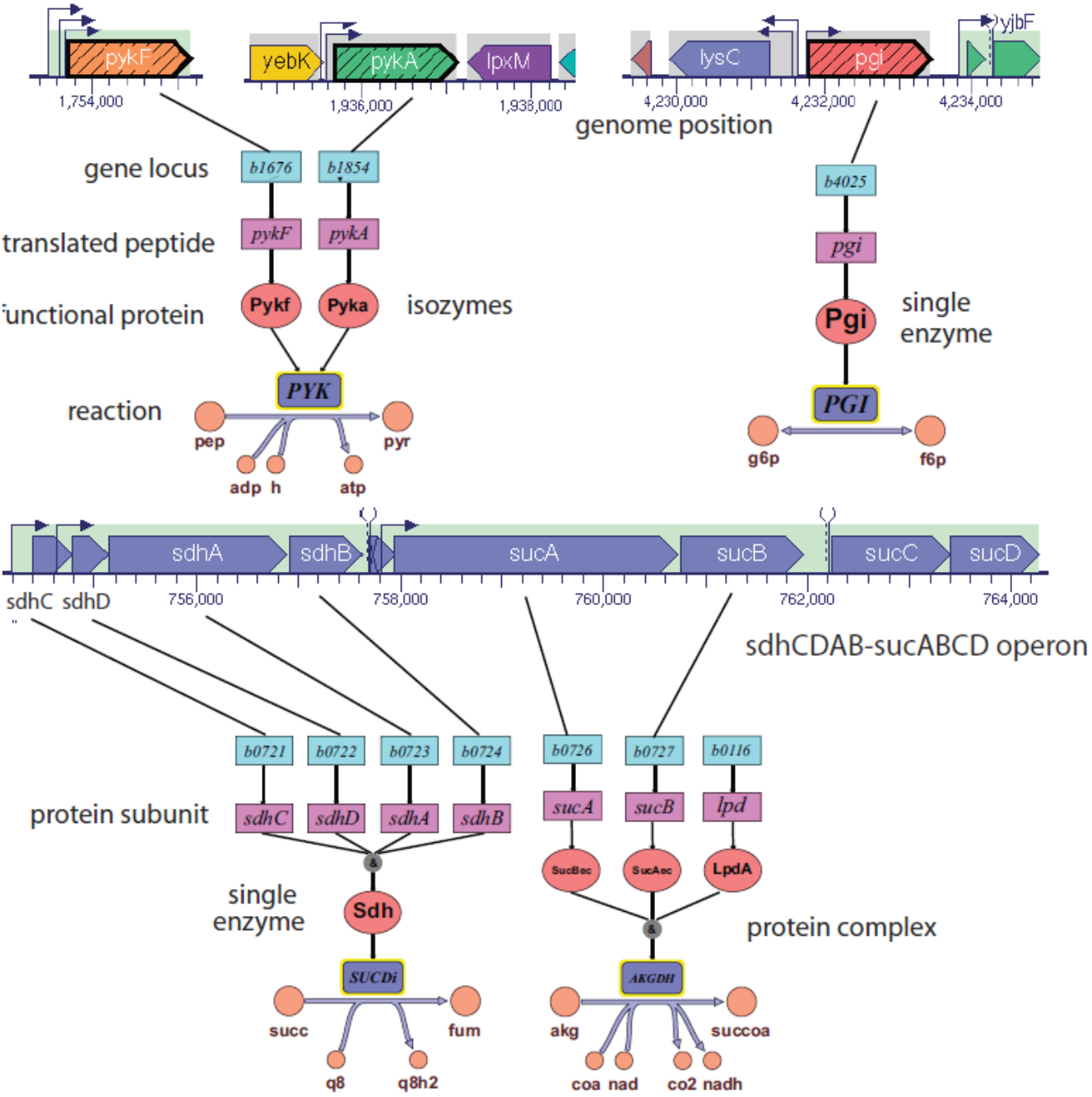


Figure 2. Examples of the gene-protein-reaction associations from the *E. coli* core model [3].

Now let's start to use the COBRA toolbox to begin exploring the *E.coli* core model. The first thing that needs to be done is to load the model into the Matlab work environment. This can be achieved by loading the Matlab version of the model (.mat) into Matlab. This model is available in the downloaded COBRA toolbox software. *[Timing: Seconds]*

```
global CBTDIR
readCbModel([CBTDIR filesep 'test' filesep 'models' filesep 'ecoli_core_model.mat']);
e_coli_core = model; % Save the original model for later use
```

After you load the *E.coli* core model into the Matlab, you should be able to look at the MATLAB workplace and see that the model is loaded as shown in Figure 3.

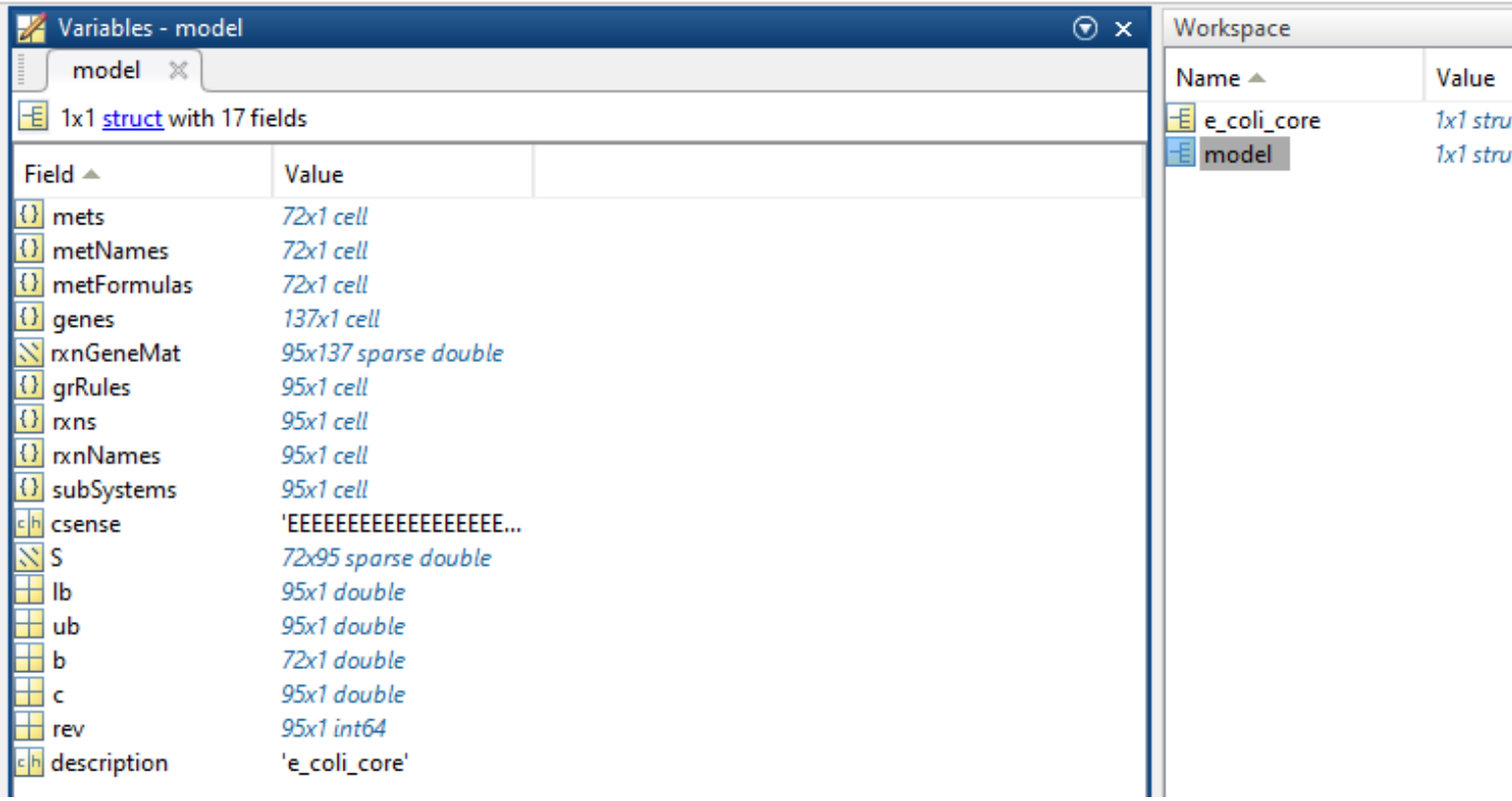


Figure 3. Matlab workspace after the *E.coli* core model has been loaded into Matlab.

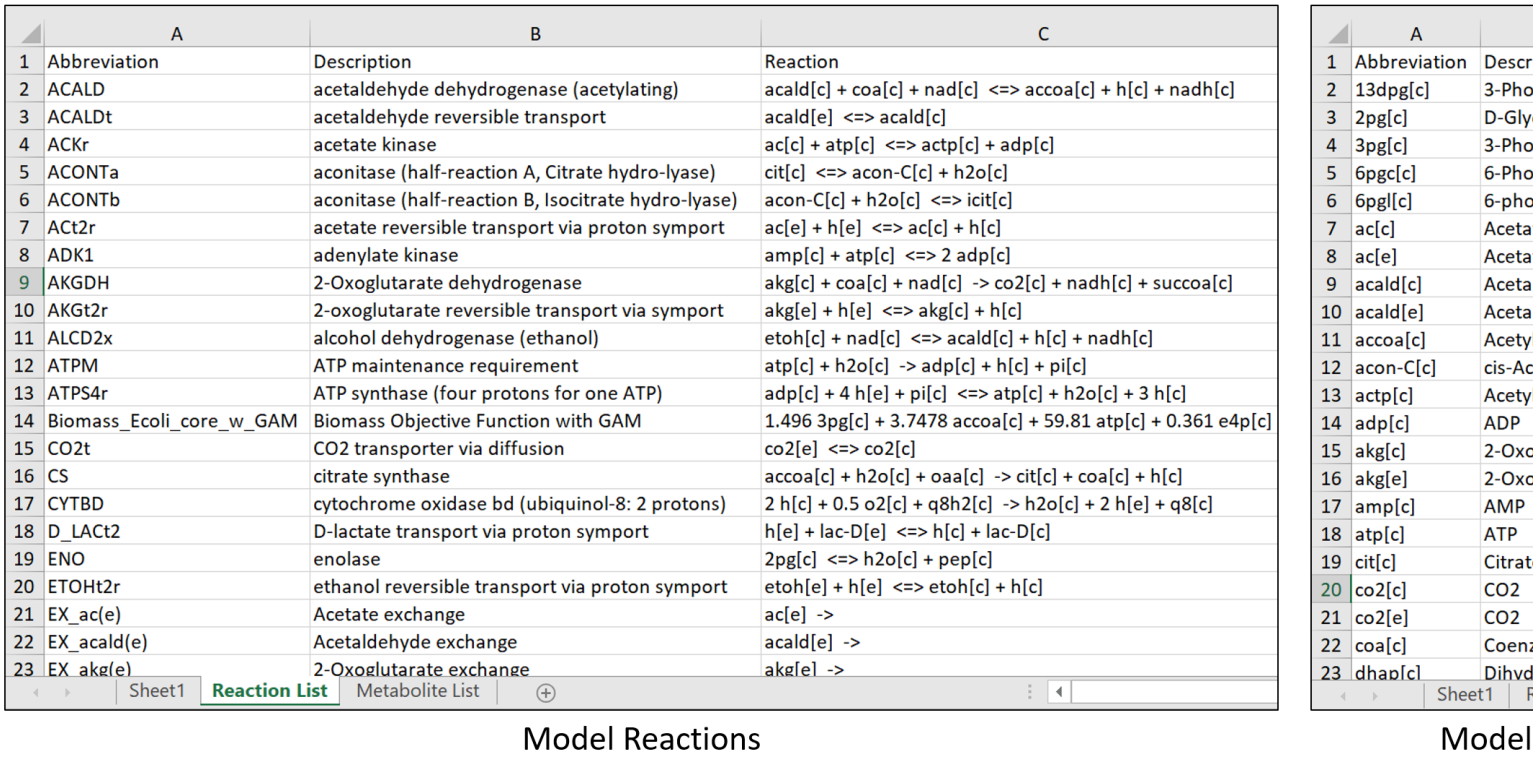
Perhaps the easiest way to access all the information in the COBRA model is to print out a spreadsheet that contains all the information stored in the model. This can be accomplished using the "writeCbModel" function. *[Timing: Seconds]*

```
outmodel = writeCbModel(model, 'xls', 'core_model.xls')
```

```
The following problems have been encountered in the model structure
missingFields:
csense
osense
```

```
outmodel =
           rxns: {95×1 cell}
           mets: {72×1 cell}
              S: [72×95 double]
             lb: [95×1 double]
             ub: [95×1 double]
              c: [95×1 double]
          rules: {95×1 cell}
          genes: {137×1 cell}
     rxnGeneMat: [95×137 double]
        grRules: {95×1 cell}
     subSystems: {95×1 cell}
       rxnNames: {95×1 cell}
      metNames: {72×1 cell}
    metFormulas: {72×1 cell}
              b: [72×1 double]
    description: 'Ecoli_core_model'
         osense: -1
         csense: [72×1 char]
```

This function will write the model to an Excel spreadsheet named "core_model.xls" and allow you to explore all the details associated with both the model reactions and metabolites. This is illustrated in Figure 4. You can also gain specific information on genes, reactions, metabolites and GPRA's using COBRA Toolbox functions. This will be shown later in this tutorial.



	A	B	C
1	Abbreviation	Description	Reaction
2	ACALD	acetaldehyde dehydrogenase (acetylating)	acald[c] + coa[c] + nad[c] <=> accoa[c] + h[c] + nadh[c]
3	ACALDt	acetaldehyde reversible transport	acald[e] <=> acald[c]
4	ACKr	acetate kinase	ac[c] + atp[c] <=> actp[c] + adp[c]
5	ACONTa	aconitase (half-reaction A, Citrate hydro-lyase)	cit[c] <=> acon-C[c] + h2o[c]
6	ACONTb	aconitase (half-reaction B, Isocitrate hydro-lyase)	acon-C[c] + h2o[c] <=> icit[c]
7	ACt2r	acetate reversible transport via proton symport	ac[e] + h[e] <=> ac[c] + h[c]
8	ADK1	adenylate kinase	amp[c] + atp[c] <=> 2 adp[c]
9	AKGDH	2-Oxoglutarate dehydrogenase	akg[c] + coa[c] + nad[c] -> co2[c] + nadh[c] + succoa[c]
10	AKGt2r	2-oxoglutarate reversible transport via symport	akg[e] + h[e] <=> akg[c] + h[c]
11	ALCD2x	alcohol dehydrogenase (ethanol)	etoh[c] + nad[c] <=> acald[c] + h[c] + nadh[c]
12	ATPM	ATP maintenance requirement	atp[c] + h2o[c] -> adp[c] + h[c] + pi[c]
13	ATPS4r	ATP synthase (four protons for one ATP)	adp[c] + 4 h[e] + pi[c] <=> atp[c] + h2o[c] + 3 h[c]
14	Biomass_Ecoli_core_w_GAM	Biomass Objective Function with GAM	1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c]
15	CO2t	CO2 transporter via diffusion	co2[e] <=> co2[c]
16	CS	citrate synthase	accoa[c] + h2o[c] + oaa[c] -> cit[c] + coa[c] + h[c]
17	CYTBD	cytochrome oxidase bd (ubiquinol-8: 2 protons)	2 h[c] + 0.5 o2[c] + q8h2[c] -> h2o[c] + 2 h[e] + q8[c]
18	D_LACt2	D-lactate transport via proton symport	h[e] + lac-D[e] <=> h[c] + lac-D[c]
19	ENO	enolase	2pg[c] <=> h2o[c] + pep[c]
20	ETOHt2r	ethanol reversible transport via proton symport	etoh[e] + h[e] <=> etoh[c] + h[c]
21	EX_ac(e)	Acetate exchange	ac[e] ->
22	EX_acald(e)	Acetaldehyde exchange	acald[e] ->
23	EX_akg(e)	2-Oxoglutarate exchange	akg[e] ->

Sheet1 | Reaction List | Metabolite List

Model Reactions

	A
1	Abbreviation
2	13dpg[c]
3	2pg[c]
4	3pg[c]
5	6pgc[c]
6	6pgl[c]
7	ac[c]
8	ac[e]
9	acald[c]
10	acald[e]
11	accoa[c]
12	acon-C[c]
13	actp[c]
14	adp[c]
15	akg[c]
16	akg[e]
17	amp[c]
18	atp[c]
19	cit[c]
20	co2[c]
21	co2[e]
22	coa[c]
23	dhap[c]

Sheet1

Model

Figure 4. Screenshot of "core_model.xls" showing a portion of the spreadsheet with both reactions and metabolites of the model.

One way to understand how the cell is operating is to visiualize the cell operation through a metabolic map. There are maps available in the COBRA toolbox installation for several different organisms that can be used to visualize the models and also overlay calculated flux values on the map (we will discuss this in the flux balance analysis section). To create a map requires a special file referred to as an "exportmap." For the case of the *E.coli* core model the exportmap is called the "ecoli_core_map.txt" and is included with the default COBRA toolbox installation. The following steps can be used to create a map of the *E.coli* core in an "SVG" file called "target.svg." This map file should be located in your working directory. *[Timing: Seconds]*

```
map=readCbMap('ecoli_core_map.txt');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawCbMap(map);
```

```
Document Written
```

Figure 5 is a screenshot of the *E.coli* core map produced by "drawflux".

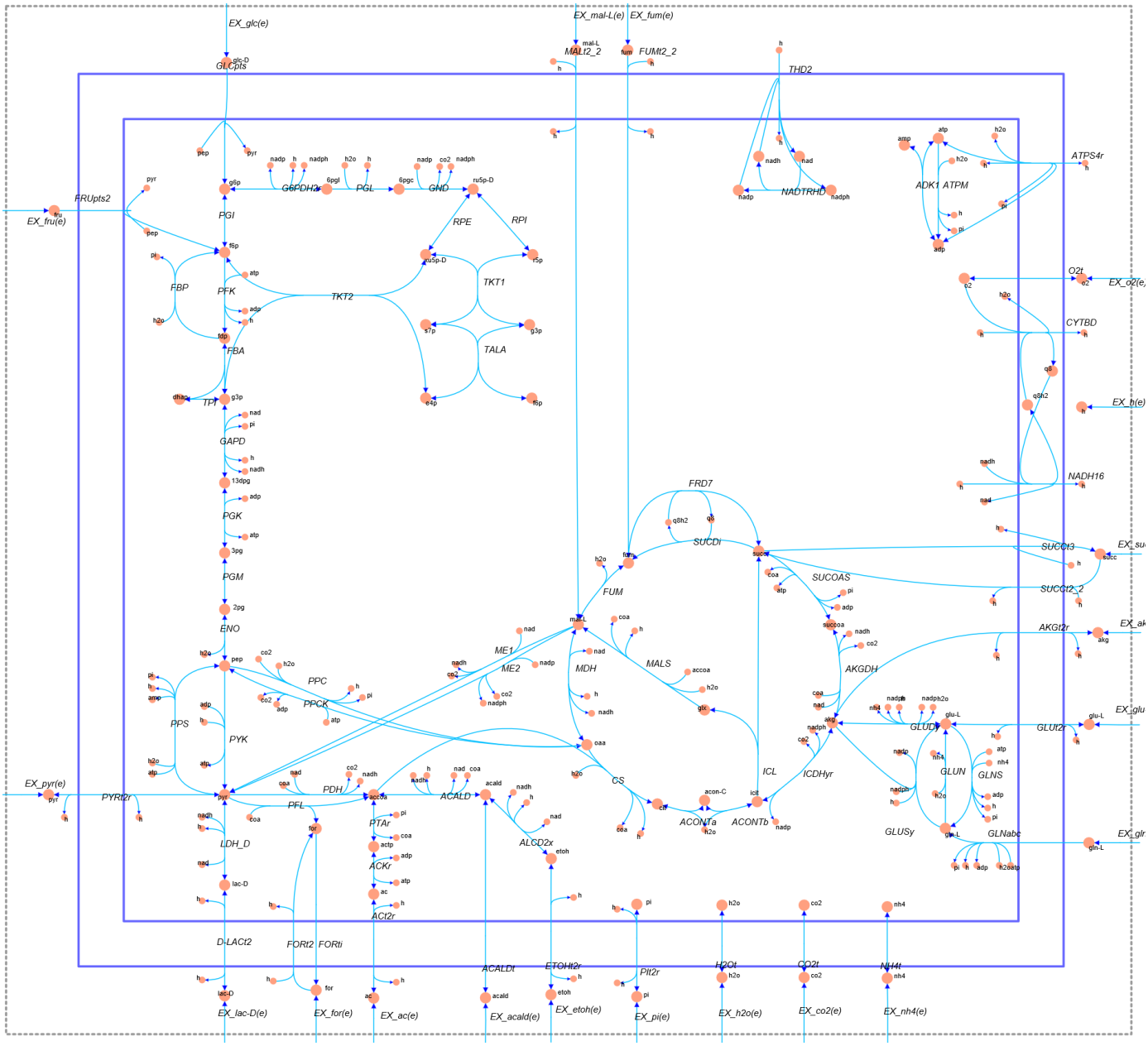


Figure 5. The screenshot of a "drawflux" produced *E.coli* core model map.

This SVG map can be read with most browsers and is very easy to use to search and explore the *E.coli* network.

2.A. Genes

Now let's begin by exploring some of the Matlab code and COBRA toolbox functions that can be used to extract information about the genes from the model. Below are a collection of COBRA Toolbox functions that retrieve key information from the genes in the model. The genes are only represented by their gene locus number (e.g. b2097). The gene name can be achieved with a quick search of the EcoCyc website (https://www.ecocyc.org/). To start with, the genes included in the model and their geneIDs are stored in the "model.genes" structure. The "findGeneIDs" COBRA Toolbox function can be used to pull the geneID from the model structure. The first 10 genes in the model, and their geneIDs, can be printed out as follows. *[Timing: Seconds]*

```
genes = cellstr(model.genes(1:10));
geneIDs = findGeneIDs(model, model.genes(1:10));
```

```
printLabeledData(model.genes(1:10),geneIDs)
```

```
b0008 1
b0114 2
b0115 3
b0116 4
b0118 5
b0351 6
b0356 7
b0451 8
b0474 9
b0485 10
```

For the case of finding a single geneID from the model, the gene locus number needs to be included in single quotes, such as 'b_number '. *[Timing: Seconds]*

```
findGeneIDs(model, 'b0116')
```

```
ans = 4
```

Now, to find the reactions that are associated with a given gene, you can use the "findRxnsFromGenes" function as shown below. *[Timing: Seconds]*

```
[results ListResults]=findRxnsFromGenes(model,'b0116',0,1)
```

```
Warning: 3rd argument is numericFlag, currently redundant, will be depreciated
results =
    b0116: {2×4 cell}

ListResults =
    'AKGDH'    'akg[c] + coa[c] + nad[c]  -> co2[c] + nadh[c] + succoa[c] '    'Citric Acid Cycle'
    'PDH'      'coa[c] + nad[c] + pyr[c]  -> accoa[c] + co2[c] + nadh[c] '     'Glycolysis/Gluconeogenes
```

This result shows that the gene "b0116" is associated with the two reactions AKGDH and PDH.

2.B. Reactions

Reactions and their rxnIDs are stored in the "model.rxns" structure with the reaction names being stored in "model.rxnNames." The COBRA Toolbox function "findRxnIDs" can be used to extract the rxnID from the model structure. Note that the biomass function "Biomass_Ecoli_core_w_GAM" is listed as one of the reactions. *[Timing: Seconds]*

```
rxnIDs = findRxnIDs(model, model.rxns(1:15));
printLabeledData(model.rxns(1:15),rxnIDs)
```

```
ACALD 1
ACALDt 2
ACKr 3
ACONTa 4
ACONTb 5
ACt2r 6
ADK1 7
AKGDH 8
AKGt2r 9
```

```
ALCD2x 10
ATPM 11
ATPS4r 12
Biomass_Ecoli_core_w_GAM 13
CO2t 14
CS 15
```

To find a single rxnID from the model use the "findRxnIDs" with the desired reaction abbreviation included in single quotes, e.g. 'ENO'. *[Timing: Seconds]*

```
rxnIDs = findRxnIDs(model, 'ENO')
```

```
rxnIDs = 18
```

Finding the name of the 'ENO' reaction can be recovered using the "model.rxnNames" structure with the desired reaction rxnID. *[Timing: Seconds]*

```
model.rxnNames(rxnIDs)
```

```
ans =
    'enolase'
```

To find the formula of the reaction, use the "printRxnFormula" function. *[Timing: Seconds]*

```
printRxnFormula(model,'ENO');
```

```
ENO 2pg[c]  <=> h2o[c] + pep[c]
```

To find the genes that are associated with a given reaction, you can use the "findGenesRxns" function. *[Timing: Seconds]*

```
[geneList]=findGenesFromRxns(model,'ENO');
geneList{1:1}
```

```
ans =
    'b2779'
```

Finally, there are times when it is ncessary to find all the "reactant" metabolites that feed a reaction as well as all the "product" metabolites that are produced by the reaction. This can be achieved using the "surfNet" function as shown below (there is a COBRA tutorial on this by Siu Hung Joshua Chan called "Browse Networks in the Matlab Command Window Using surfNet"). This functions output includes a listing of the reactants and products based on the reaction formulas. It should be pointed out that in situations where a reaction becomes reversible, a metabolite that is a reactant could become a product and a metabolite that is a product could become a reactant. *[Timing: Seconds]*

```
surfNet(model, 'GAPD')
```

```
Rxn #49  GAPD, Bd: -1000 / 1000, glyceraldehyde-3-phosphate dehydrogenase
g3p[c] + nad[c] + pi[c] <=> 13dpg[c] + h[c] + nadh[c]
  id   Met       Stoich     metNames, metFormulas
Reactant:
 #33     g3p[c]  -1         Glyceraldehyde-3-phosphate, C3H5O6P
 #50     nad[c]  -1         Nicotinamide-adenine-dinucleotide, C21H26N7O14P2
 #60      pi[c]  -1         Phosphate, HO4P
```

```
Product:
 #1    13dpg[c]  1          3-Phospho-D-glyceroyl-phosphate, C3H4O10P2
 #43       h[c]  1          H, H
 #51    nadh[c]  1          Nicotinamide-adenine-dinucleotide-reduced, C21H27N7O14P2

Show previous steps...
```

In this case, the "reactant" metabolites for the GAPD reaction are g3p[c], nad[c] and pi[c] while the "product" metabolites are 13dpg[c], h[c], and nadh[c].

2.C. Metabolites

The metabolites included in the model and their metabolite IDs (metIDs) are stored in the "model.mets" structure with the metabolite names being stored in model.metNames. The COBRA Toolbox function "findMetIDs" can be used to extract the metID's from the model structure as shown in the following example. *[Timing: Seconds]*

```
metIDs = findMetIDs(model, model.mets(1:15));
printLabeledData(model.mets(1:15),metIDs)
```

```
13dpg[c] 1
2pg[c] 2
3pg[c] 3
6pgc[c] 4
6pgl[c] 5
ac[c] 6
ac[e] 7
acald[c] 8
acald[e] 9
accoa[c] 10
acon-C[c] 11
actp[c] 12
adp[c] 13
akg[c] 14
akg[e] 15
```

To find a single metID from the model use the "findMetIDs" with the desired metabolite abbreviation included in single quotes, e.g. 'akg[c]'. *[Timing: Seconds]*

```
metIDs = findMetIDs(model, 'akg[c]')
```

```
metIDs = 14
```

Finding the name of the 'akg[c]' reaction yeilds *[Timing: Seconds]*

```
model.metNames(metIDs)
```

```
ans =
    '2-Oxoglutarate'
```

To find the chemical formula of the metabolite you can use the "model.metFormulas" structure with a metID. *[Timing: Seconds]*

```
model.metFormulas(metIDs)
```

```
ans =
    'C5H4O5'
```

Finally, to find reactions that both produce and consume a desired metabolite you can again use the "surfNet" function. This includes a listing of the consuming and producing reactions based on the reaction formulas. In some situations, if a reaction is reversible, the producing/consuming reactions could be switched. *[Timing: Seconds]*

```
surfNet(model, 'atp[c]')
```

```
Met #17  atp[c], ATP, C10H12N5O13P3
Consuming reactions:
  #3  ACKr, Bd: -1000 / 1000, acetate kinase
  ac[c] + atp[c] <=> actp[c] + adp[c]
  #7  ADK1, Bd: -1000 / 1000, adenylate kinase
  amp[c] + atp[c] <=> 2 adp[c]
  #11  ATPM, Bd: 8.39 / 1000, ATP maintenance requirement
  atp[c] + h2o[c] -> adp[c] + h[c] + pi[c]
  #13  Biomass_Ecoli_core_w_GAM, Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #51  GLNS, Bd: 0 / 1000, glutamine synthetase
  atp[c] + glu-L[c] + nh4[c] -> adp[c] + gln-L[c] + h[c] + pi[c]
  #52  GLNabc, Bd: 0 / 1000, L-glutamine transport via ABC system
  atp[c] + gln-L[e] + h2o[c] -> adp[c] + gln-L[c] + h[c] + pi[c]
  #72  PFK, Bd: 0 / 1000, phosphofructokinase
  atp[c] + f6p[c] -> adp[c] + fdp[c] + h[c]
  #75  PGK, Bd: -1000 / 1000, phosphoglycerate kinase
  3pg[c] + atp[c] <=> 13dpg[c] + adp[c]
  #80  PPCK, Bd: 0 / 1000, phosphoenolpyruvate carboxykinase
  atp[c] + oaa[c] -> adp[c] + co2[c] + pep[c]
  #81  PPS, Bd: 0 / 1000, phosphoenolpyruvate synthase
  atp[c] + h2o[c] + pyr[c] -> amp[c] + 2 h[c] + pep[c] + pi[c]
  #90  SUCOAS, Bd: -1000 / 1000, succinyl-CoA synthetase (ADP-forming)
  atp[c] + coa[c] + succ[c] <=> adp[c] + pi[c] + succoa[c]
Producing reactions:
  #12  ATPS4r, Bd: -1000 / 1000, ATP synthase (four protons for one ATP)
  adp[c] + 4 h[e] + pi[c] <=> atp[c] + h2o[c] + 3 h[c]
  #83  PYK, Bd: 0 / 1000, pyruvate kinase
  adp[c] + h[c] + pep[c] -> atp[c] + pyr[c]

Show previous steps...
```

As you would expect, there should be a large number of consumers of atp[c] but only a small number of producers.

2.D. Gene-Protein-Reaction Associations

A gene-protein-reaction association (GPRA) shows the Boolean relationship between the genes that are required to produce a specific reaction (see Figure 2). The Boolean relationship between the genes and a given reaction can be found using the "model.grRules" structure of the model. This is shown below. *[Timing: Seconds]*

```
rxnIDs = findRxnIDs(model, 'PYK');
model.grRules(rxnIDs)
```

```
ans =
```

```
    '(b1854 or b1676)'
```

Here is an example of a more complicated gene-reaction relationship. *[Timing: Seconds]*

```
rxnIDs = findRxnIDs(model, 'ATPS4r');
model.grRules(rxnIDs)
```

```
ans =
    '(((b3736 and b3737 and b3738) and (b3731 and b3732 and b3733 and b3734 and b3735)) or ((b3736 and b
```

2.E. Model Constraints

In constraint-based simulations, the system constraints are implemented in two ways in COBRA models: 1) as reaction formulas that balance reaction input and output metabolites (mass balance), and 2) as inequalities that impose upper and lower bounds on the flux rates of every reaction in the model. To set these constraints, every reaction is given both upper and lower bounds, which define the maximum and minimum allowable fluxes through the reactions.

To find the constraints for all the reactions in the model, the COBRA Toolbox provides the "printConstraints(model,lb, ub)" function where the "lb" value is the lower bound of the reactions potential flux while "ub" refers to the upper bound of the reactions flux. It is important to understand the default constraint settings for the *E.coli* core model. In this model all reversible reactions in the cytoplasm are initally set so that their lower bound is -1000 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$with an upper bound of +1000 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. On the other hand, irreversible reactions, except ATPM, are set with a lower bound of 0 and an upper bound of +1000 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. The ATP maintenance reaction (ATPM) is set with a lower bound of 0 and an upper bound of +8.39 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$(this will be discussed later). All the exchange reactions, except EX_glc(e), are set to allow secretion but not uptake, thus a lower bound of 0 and an upper bound of +1000 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. To avoid confusion, it should be understood that all exchange reactions, which are reactions that interface between the extracellular and cytoplasmic space, assume that secretion is positive while uptake is labeled negative. Finally, the glucose exchange reaction, EX_glc(e), is set with a lower bound of -10 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$ and an upper bound of +1000 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$.

To see the constraints for the reactions that are not set at the minimum/maximum (-100\100) values, then "lb" and "ub" can be adjusted. *[Timing: Seconds]*

```
printConstraints(model,-100, +100)
```

```
MinConstraints:
ATPM 8.39
EX_glc(e) -10
maxConstraints:
```

Note that the exchange reaction that controls the uptake of glucose, 'EX_glc(e) is automatically set to -10 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$.

The results of the upper or lower bounds for a particular reaction can be found by using the COBRA model structure, "model.lb" for the lower bound and "model.ub" for the upper bound. *[Timing: Seconds]*

```
rxnIDs = findRxnIDs(model,'EX_glc(e)');
```

```
model.lb(rxnIDs)
```

```
ans = -10
```

```
model.ub(rxnIDs)
```

```
ans = 1000
```

Altering the constraints for a reaction can be accomplished with the "model = changeRxnBounds(model,rxnNameList,value,boundType)" function. For this function the second parameter is the reaction(s) that need to be constrained, the third parameter is the desired flux rate in $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$, and the fourth parameter can be 'u' - upper limit, 'l' - lower limit, or 'b' - both (Default = 'b'). *[Timing: Seconds]*

```
model = changeRxnBounds(model,'EX_glc(e)',-5,'l');
printConstraints(model,-100, +100); % Showing the result of the change
```

```
MinConstraints:
ATPM 8.39
EX_glc(e) -5
maxConstraints:
```

You can now see that the lower bound for glucose has been changed to -5 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$.

2.F. Objective Functions

In order to perform flux balance analysis it is necessary to define a biological objective or objective function. For the case of predicting growth, the biological objective is the biomass production or the rate at which metabolic compounds are converted into biomass constituents. This biomass production is mathematically represented by the addition to the model of an artificial ‘biomass reaction’ (Biomass_Ecoli_core_w_GAM) which consumes precursor metabolites at stoichiometries that simulate biomass production. The precursors for the *E.coli* core model are shown in Figure 6.

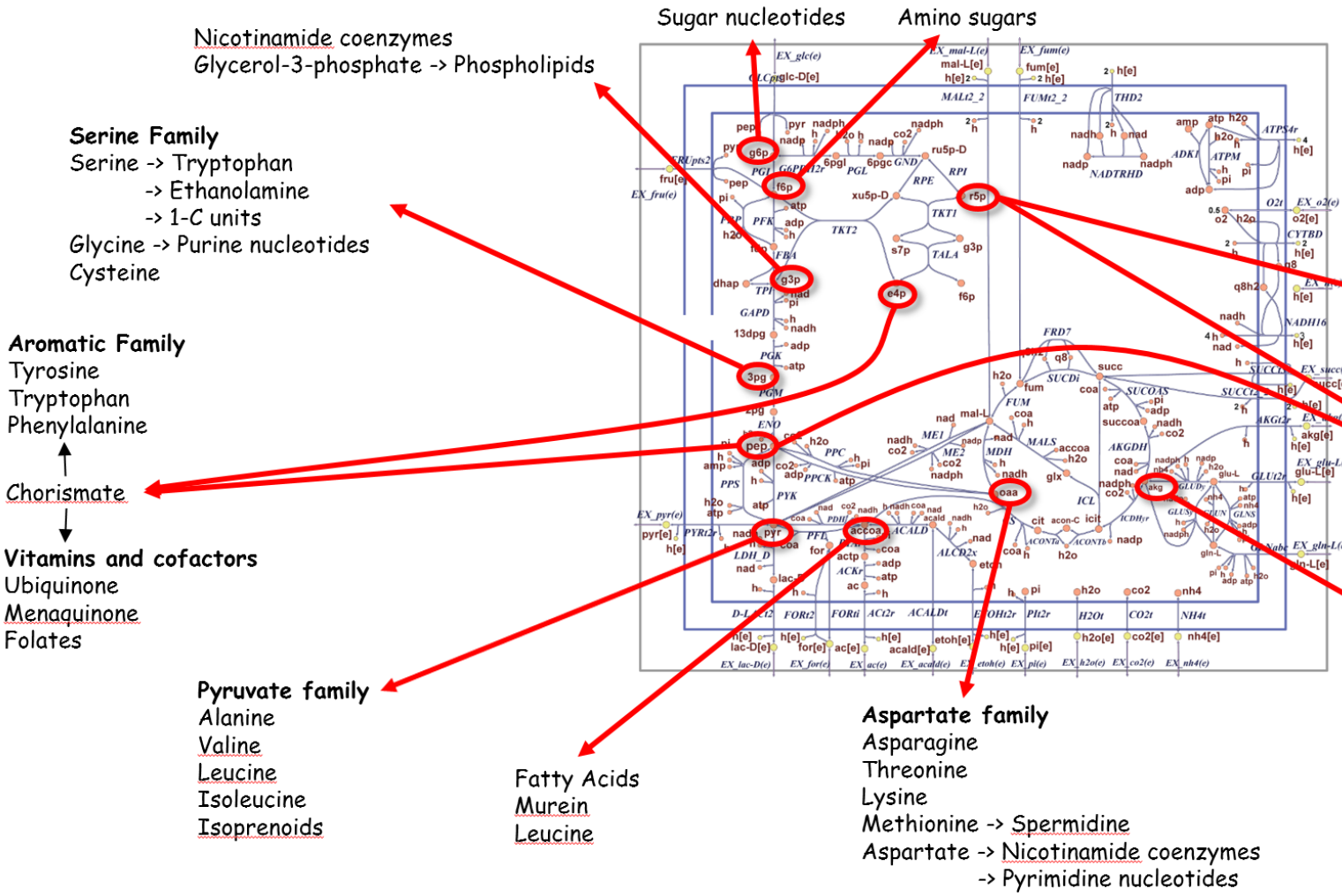


Figure 6. *E.coli* core model precursors [11].

The biomass reaction also includes key cofactors (atp[c], adp[c], nad[c], nadh[c], nadph[c], and pi[c]) that are required for cell growth and operation. The biomass reaction is based on experimental measurements of biomass components allowing the reaction to be scaled so that its flux is equal to the exponential growth-rate (μ) of the organism. With the biomass now represented in the model, the maximum growth rate can be predicted by calculating the conditions that maximize the flux through the biomass reaction.

The biomass reaction and the weighted precursor metaboites can be inspected by printing out the formula for the biomass function using the "printRxnFormula" COBRA Toolbox function. *[Timing: Seconds]*

```
printRxnFormula(model,'Biomass_Ecoli_core_w_GAM')
```

```
Biomass_Ecoli_core_w_GAM 1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] +
ans =
    '1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205
```

The objective function can also be checked using the "checkObjective(model)" function which will print out the stoichiometric coefficients for each metabolite along with the name of the objective. *[Timing: Seconds]*

```
checkObjective(model)
```

```
summaryT =
    Coefficient    Metabolite    metID            Reaction            RxnID
    ___________    __________    _____    ________________________    _____

     -1.496        3pg[c]         3       Biomass_Ecoli_core_w_GAM    13
    -3.7478        accoa[c]      10       Biomass_Ecoli_core_w_GAM    13
      59.81        adp[c]        13       Biomass_Ecoli_core_w_GAM    13
     4.1182        akg[c]        14       Biomass_Ecoli_core_w_GAM    13
     -59.81        atp[c]        17       Biomass_Ecoli_core_w_GAM    13
     3.7478        coa[c]        21       Biomass_Ecoli_core_w_GAM    13
     -0.361        e4p[c]        23       Biomass_Ecoli_core_w_GAM    13
    -0.0709        f6p[c]        26       Biomass_Ecoli_core_w_GAM    13
     -0.129        g3p[c]        33       Biomass_Ecoli_core_w_GAM    13
     -0.205        g6p[c]        34       Biomass_Ecoli_core_w_GAM    13
    -0.2557        gln-L[c]      36       Biomass_Ecoli_core_w_GAM    13
    -4.9414        glu-L[c]      38       Biomass_Ecoli_core_w_GAM    13
     -59.81        h2o[c]        41       Biomass_Ecoli_core_w_GAM    13
      59.81        h[c]          43       Biomass_Ecoli_core_w_GAM    13
     -3.547        nad[c]        50       Biomass_Ecoli_core_w_GAM    13
      3.547        nadh[c]       51       Biomass_Ecoli_core_w_GAM    13
     13.028        nadp[c]       52       Biomass_Ecoli_core_w_GAM    13
    -13.028        nadph[c]      53       Biomass_Ecoli_core_w_GAM    13
    -1.7867        oaa[c]        58       Biomass_Ecoli_core_w_GAM    13
    -0.5191        pep[c]        59       Biomass_Ecoli_core_w_GAM    13
      59.81        pi[c]         60       Biomass_Ecoli_core_w_GAM    13
    -2.8328        pyr[c]        62       Biomass_Ecoli_core_w_GAM    13
    -0.8977        r5p[c]        66       Biomass_Ecoli_core_w_GAM    13
```

```
ans =
    'Biomass_Ecoli_core_w_GAM'
```

The objective function for the *E.coli* core model is automatically set to be the biomass reaction. Setting the biomass function to be the objective function can also be done using the model = changeObjective(model,'reaction name'') as shown below. *[Timing: Seconds]*

```
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');
```

3. Flux Balance Analysis

Flux balance analysis (FBA) is used to calculate the flow of metabolites through a metabolic network making it possible to predict an organism's growth-rate or the production-rate of a bioproduct. Combining the stoichiometric matrix and the objective function can create a system of linear equations that can be used to calculate the fluxes through all the reactions in the network. In flux balance analysis, these equations are solved using linear programming algorithms that can quickly identify optimal solutions to large systems of equations.

Once the external conditions have been set, which include 1) defining the allowed carbon sources, 2) defining the oxygen uptake level, and 3) setting the objective function, then the simulation conditions are setup to perform FBA. This is accomplished through the use of the "optimizeCbModel(model,osenseStr)", a COBRA toolbox function where the first argument is the model name and the second argument determines if the optimization algorithm maximizes ('max') or minimizes ('min') the objective function. Below is an example for an aerobic environment with glucose as the carbon source optimizing for maximum growth-rate. *[Timing: Seconds]*

```
model = e_coli_core; % Starting with the original model
model = changeRxnBounds(model,'EX_glc(e)',-10,'l'); % Set maximum glucose uptake
```

```
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set maximum oxygen uptake
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM'); % Set the objective function
FBAsolution = optimizeCbModel(model,'max') % FBA analysis
```

```
FBAsolution =
         full: [95×1 double]
          obj: 0.8739
        rcost: [95×1 double]
         dual: [72×1 double]
       solver: 'gurobi'
    algorithm: 'default'
         stat: 1
     origStat: 'OPTIMAL'
         time: 0.0201
        basis: [1×1 struct]
            x: [95×1 double]
            f: 0.8739
            y: [72×1 double]
            w: [95×1 double]
            v: [95×1 double]
```

“FBAsolution” is a Matlab structure that contains the following outputs. “FBAsolution.f “ is the value of objective function as calculated by FBA, thus if the biomass reaction is the objective function then “FBAsolution.f" corresponds to the growth-rate of the cell. In the example above, it can be seen that the growth-rate "FBAsolution.f" is listed as 0.8739 hr^{-1}. “FBAsolution.x” is a vector listing the calculated fluxes flowing through the network. “FBAsolution.y” and “FBAsolution.w” contain vectors representing the shadow prices and reduced costs for each metabolite or reaction, respectively.

The flux values found in the structure "FBAsolution.x" can be printed out using the "printFluxVector(model,fluxData,nonZeroFlag,excFlag)" where the second argument is a vector of the flux values, the nonZeroFlag only prints nonzero rows (Default = false), and excFlag only prints exchange reaction fluxes (Default = false). Examples of printing non-zero fluxes and exchange reaction only fluxes are shown below. *[Timing: Seconds]*

```
printFluxVector(model,FBAsolution.x,true) % only prints nonzero rows
```

```
ACONTa 6.00725
ACONTb 6.00725
AKGDH 5.06438
ATPM 8.39
ATPS4r 45.514
Biomass_Ecoli_core_w_GAM 0.873922
CO2t -22.8098
CS 6.00725
CYTBD 43.599
ENO 14.7161
EX_co2(e) 22.8098
EX_glc(e) -10
EX_h(e) 17.5309
EX_h2o(e) 29.1758
EX_nh4(e) -4.76532
EX_o2(e) -21.7995
EX_pi(e) -3.2149
FBA 7.47738
FRD7 994.936
FUM 5.06438
G6PDH2r 4.95998
GAPD 16.0235
GLCpts 10
GLNS 0.223462
GLUDy -4.54186
```

```
GND 4.95998
H2Ot -29.1758
ICDHyr 6.00725
MDH 5.06438
NADH16 38.5346
NH4t 4.76532
O2t 21.7995
PDH 9.28253
PFK 7.47738
PGI 4.86086
PGK -16.0235
PGL 4.95998
PGM -14.7161
PIt2r 3.2149
PPC 2.50431
PYK 1.75818
RPE 2.67848
RPI -2.2815
SUCDi 1000
SUCOAS -5.06438
TALA 1.49698
TKT1 1.49698
TKT2 1.1815
TPI 7.47738
```

```
printFluxVector(model,FBAsolution.x,true,true) % only print exchange reaction fluxes
```

```
Biomass_Ecoli_core_w_GAM 0.873922
EX_co2(e) 22.8098
EX_glc(e) -10
EX_h(e) 17.5309
EX_h2o(e) 29.1758
EX_nh4(e) -4.76532
EX_o2(e) -21.7995
EX_pi(e) -3.2149
```

Printing all the zero and nonzero fluxes can be achieved using "printFluxVector(model,FBAsolution.x)."

These fluxes can also be overlayed on a map of the model as shown below, *[Timing: Seconds]*

```
map=readCbMap('ecoli_core_map');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options); % Draw the flux values on the map "target.svg"
```

```
Document Written
```

This overlayed map will be written to a file named "target.svg" that should be located in your working directory. Figure 7 is a screenshot of that map.

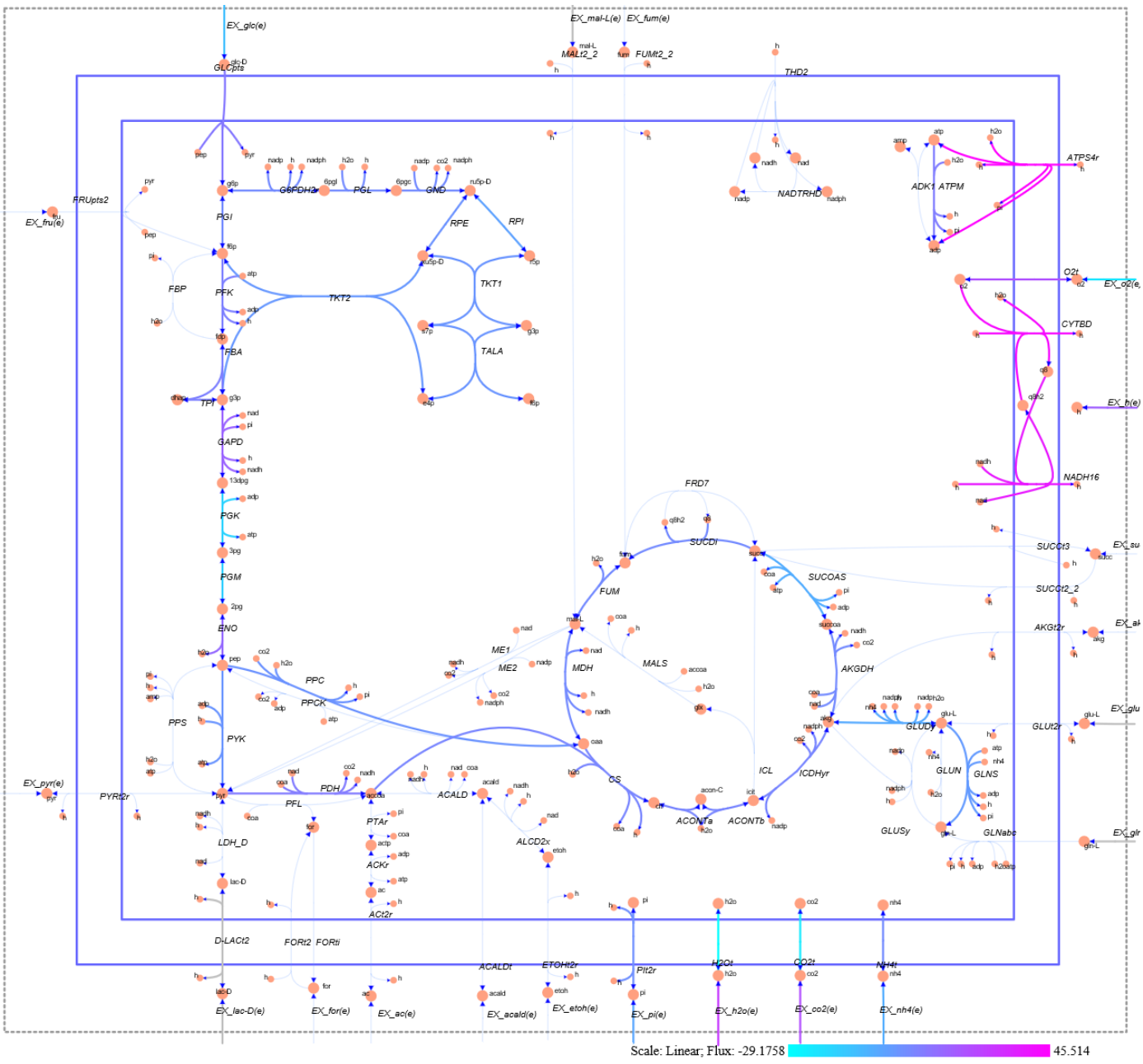


Figure 7. Screenshot of the network map of the *E.coli* core model with EX_glc(e) ≥ -10 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$ and EX_o2(e) ≥ -30 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$.

As a cautionary note, the default condition for the *E.coli* core model sets the carbon source as glucose with an uptake rate of -10 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$, the oxygen uptake is -1000 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$ which implies an aerobic environment with the objectve function defined as 'Biomass_Ecoli_core_w_GAM'. It is a good practice to define the conditions of your simulation explicity to avoid unexpected results and long troubleshooting times.

4. The Subsystems of the *E.coli* Core Model

Now with these basic Matlab and COBRA toolbox skills behind us, it is time to start exploring the subsytems that make up the *E.coli* core model. We will start by looking at the "energy production and management" section of the model that is referred to as the "oxidative phosphorylation" subsystem in

this core model. This subsystem is located in the upper right corner of the *E.coli* core map as shown below in Figure 8.

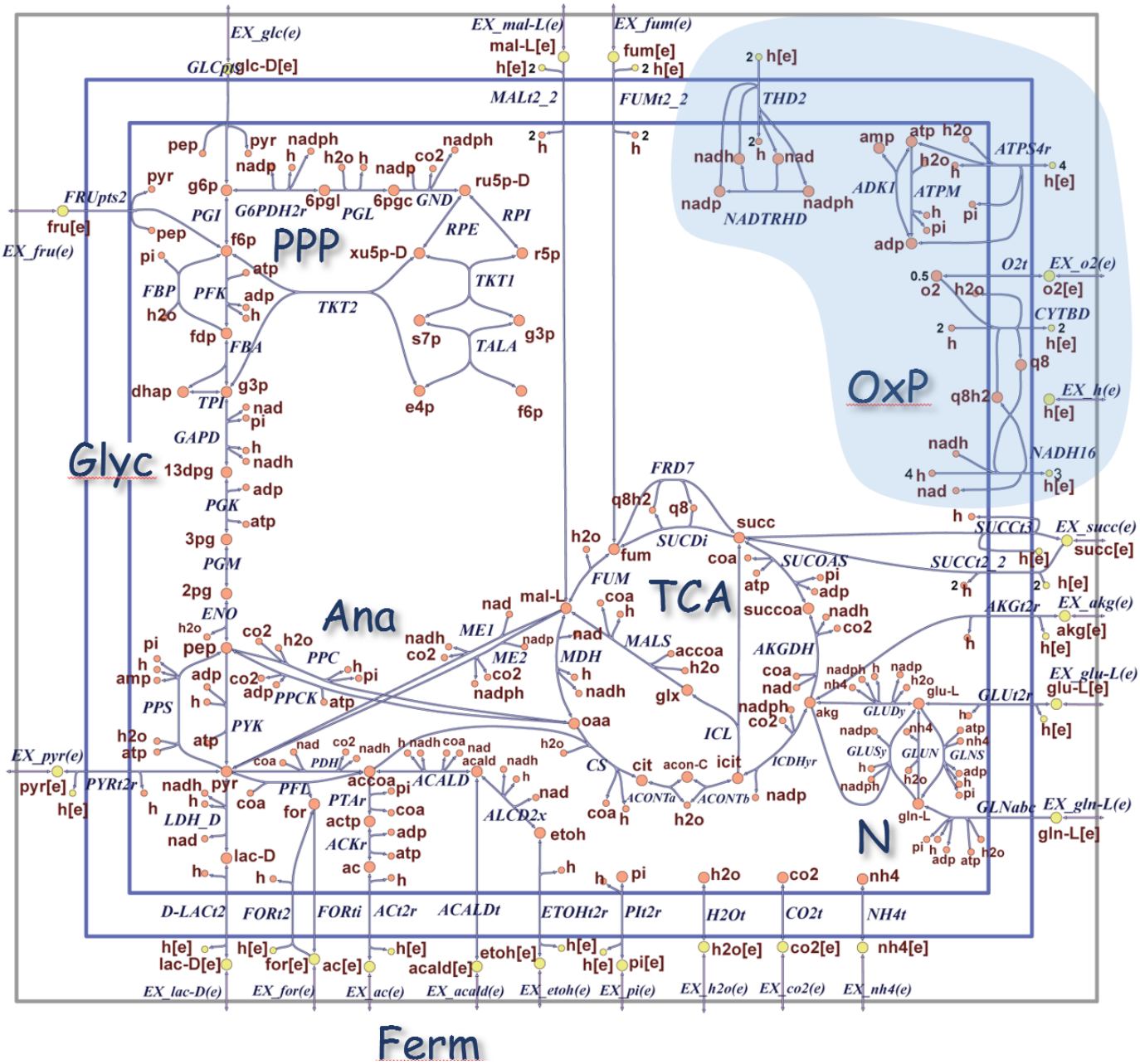


Figure 8. The location of the energy management subsystem and it's reactions highlighted in blue on the *E.coli* core map [3].

As you will see in this section, this subsystem not only includes the reactions for oxydative phosphorylation, it also includes reactions that are required for managing the reducing power needed in the cell. This subsystem will be followed by an exploration of the glycolysis pathway, the pentose phosphate pathway, the tricarboxylic acid cycle, the glycoxylate cycle, gluconeogenesis, and anapleurotic reactions, fermentation pathways, and the nitrogen metabolism.

4.A. Energy Production & Management

Perhaps the most important requirement of an operational cell is the production and management of energy and reducing power. There are two main mechanisms available within the *E.coli* core model for the production of ATP (atp[c]) energy: 1) substrate level phosphorylation, and 2) oxidative

phosphorylation through the use of the electron transport chain. Substrate level phosphorylation occurs when specific metabolic pathways within the cell are net producers of energy. In these cases, atp[c] is formed by a reaction between ADP (adp[c]) and a phosphorylated intermediate within the pathway. In the core model this occurs in the glycolysis pathway with both phosphoglycerate kinase (PGK), and pyruvate kinase (PYK), and in the tricarboxylic acid cycle with succinyl-CoA synthetase (SUCOAS). Through these substrate level phosphorylation enzymes each molecule of glucose can potentially add four molecules to the total cellular flux of atp[c].

The second mechanism for energy generation is oxidative phosphorylation through the electron transport chain, which under aerobic conditions, produces the bulk of the cell's atp[c]. In the simple core model, the electron transport chain is used to transport protons (h[c]) from the cytoplasm across the cytoplasmic membrane into the extracellular space (periplasmic space in actual cells) to create a proton-motive force which drives ATP synthase (ATPS4r) to produce atp[c].

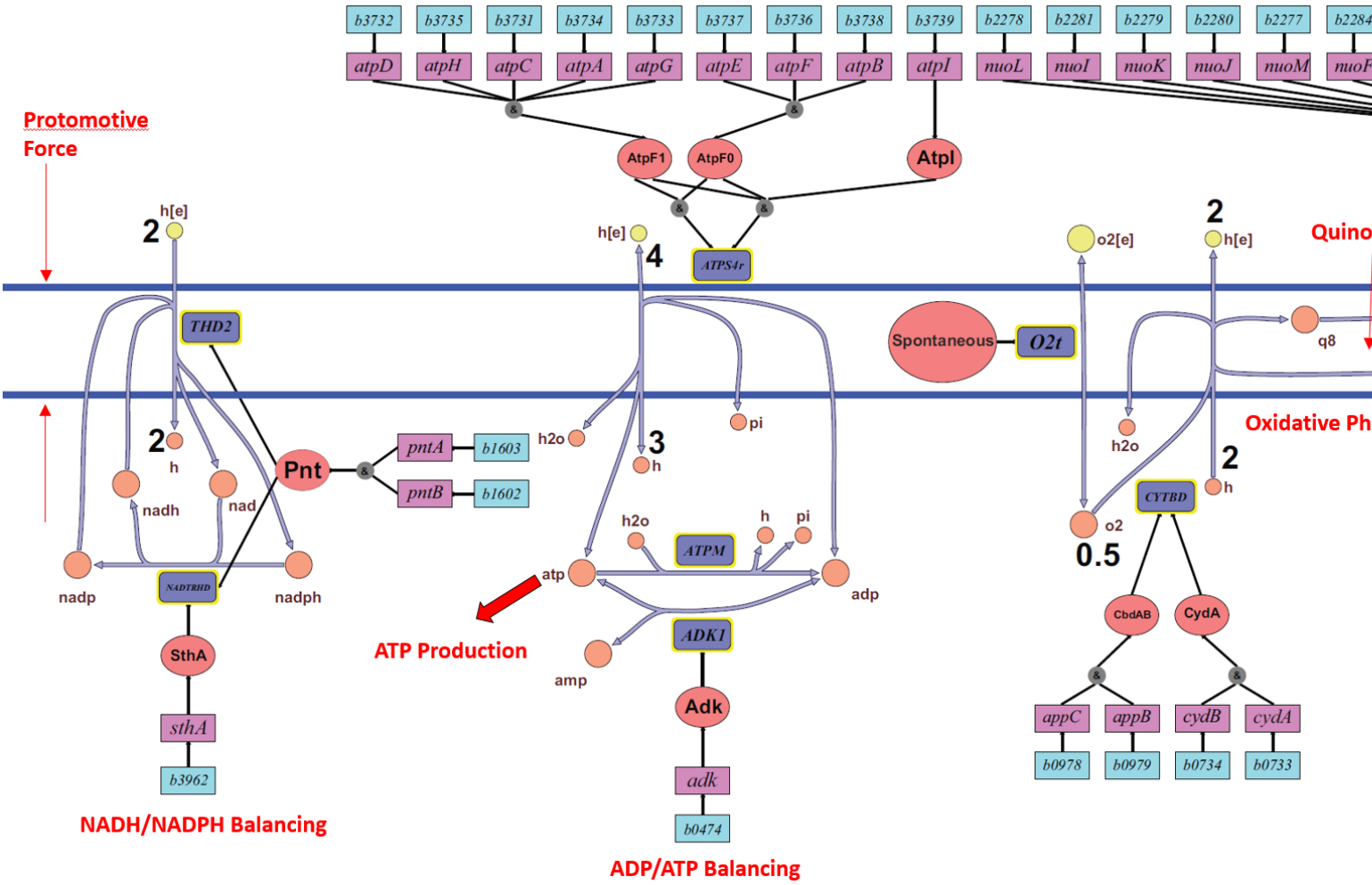


Figure 9. Oxidative Phosphorylation and Transfer of Reducing Equivalents [3].

Aerobic Respiration

For aerobic respiration, the primary source of atp[c] is produced through oxidative phosphorylation. This is illustrated in Figure 9 where NADH (nadh[c]), acting as a substrate for NADH dehydrogenase (NADH16), provides the reducing power necessary to trigger the electron transport chain. The *E. coli* core model combines the electron transport chain into two reactions. In the first of these two reactions, NADH16 catalyzes the oxidation of nadh[c] to form NAD+ (nad[c]) while extracting four protons (h[c]) from the cytoplasm. It then transports three protons to the extracellular space while combining the fourth proton with a proton and two electrons from NADH to transform ubiquinone-8) (q8[c]) to its reduced form ubiquinol-8 (q8h2[c]). Both q8[c] and q8h2[c] are oil soluble coenzymes that can diffuse freely within the lipid environment of the cytoplasmic membrane allows q8h2[c] to eventually transfer its two electrons

and two protons to cytochrome oxidase (CYTBD). The two protons (h[e]) are then transferred into the extracellular space where they add to the proton-motive force. The two electrons from q8h2[c] are then combined with two cytoplasmic protons and an oxygen atom, the terminal electron acceptor, to form water. In this model, oxygen (o2[c]) spontaneously diffuses from the environment into the cell through the spontaneous 02t reaction.

With a proton-motive force now created by the pumping of protons from the cytoplasm to the extracellular space, the reaction ATPS4r can synthesize atp[c] from adp[c]. For this simple model the *P*/0 ratio is stoichiometrically set to 1.25. Another reaction included in the energy management suite is adenylate kinase (ADK1), a phosphotransferase enzyme that catalyzes the interconversion of adenine nucleotides, and plays an important role in the adp[c]/atp[c] balance or cellular energy homeostasis.

Finally, the ATP maintenance function (ATPM), which is set at 8.39 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$ accounts for the energy (in form of atp[c]) necessary to replicate a cell, including for macromolecular synthesis (e.g., proteins, DNA, and RNA). Thus, for growth to occur in the *E.coli* model, the flux rate through ATPM must be greater than 8.39 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. If the model detects that ATPM has not reached its minimum value it will not produce FBA results.

Another part of the energy management of a cell is the reducing power that is required for both cellular catabolism and anabolism. Catabolism refers to a set of metabolic pathways that break down molecules into smaller units and release energy. For this core model, nadh[c] provides the reducing power necessary for the catabolic activities of the cell.

Anabolism, on the other hand, is the set of metabolic pathways that construct molecules from smaller units. These anabolic reactions are endergonic and therefore require an input of energy. In this case, NADPH (nadph[c]) is the reducing power required for biosynthesis using the cell's precursor metabolites.

Maintaining the proper balance between anabolic reduction charge, nadph[c]/ nadp[c], and catabolic reduction charge, nadh[c]/ nad[c], is achieved by reactions catalyzed by transhydrogenase enzymes, as shown in Figure 9. Using the proton-motive force, NAD(P) transhydrogenase (THD2) catalyzes the transfer of a hydride ion, a negative ion of hydrogen, from nadh[c] to create nadph[c]. The opposite transfer, of a hydride ion from nadph[c], to create nadh[c], is catalyzed by another enzyme, NAD+ transhydrogenase (NADTRHD), but it is not coupled to the translocation of protons. These pair of reactions effectively allow transfer of reducing equivalents between anabolic and catabolic reduction charge.

Now let's use the COBRA Toolbox to explore the details of the energy managing elements of the *E.coli* core model. In this tutorial, we will focus on exploring the role of cofactors in a core model that is optimized for growth-rate. There is a good discussion of how to find the maximum cofactor fluxes possible in a COBRA-based model in Chapter 19 of Palsson's book [1]. To start with let's print out a table that includes all the reaction abbreviations, names, and their formulas for the reactions invovled in oxidative phosphorylation and the cell's energy and reducing power management (see Figure 9). *[Timing: Seconds]*

```
model = e_coli_core; % Starting this section with the original model
energySubSystems = {'Oxidative Phosphorylation'};
energyReactions = model.rxns(ismember(model.subSystems,energySubSystems));
[tmp,energy_rxnID] = ismember(energyReactions,model.rxns);
reactionNames = model.rxnNames(energy_rxnID);
reactionFormulas = printRxnFormula(model,energyReactions,0);
T = table(reactionNames,reactionFormulas,'RowNames',energyReactions)
```

```
T =

                                reactionNames                                                reactionFormula
    ____________________________________________    ____________________________________
```

```
ADK1       'adenylate kinase'                                  'amp[c] + atp[c]  <=> 2 adp[c] '
ATPM       'ATP maintenance requirement'                       'atp[c] + h2o[c]  -> adp[c] + h[c] +
ATPS4r     'ATP synthase (four protons for one ATP)'           'adp[c] + 4 h[e] + pi[c]  <=> atp[c]
CYTBD      'cytochrome oxidase bd (ubiquinol-8: 2 protons)'    '2 h[c] + 0.5 o2[c] + q8h2[c]  -> h2o
FRD7       'fumarate reductase'                                'fum[c] + q8h2[c]  -> q8[c] + succ[c]
NADH16     'NADH dehydrogenase (ubiquinone-8 & 3 protons)'     '4 h[c] + nadh[c] + q8[c]  -> 3 h[e]
NADTRHD    'NAD transhydrogenase'                              'nad[c] + nadph[c]  -> nadh[c] + nadp
SUCDi      'succinate dehydrogenase (irreversible)'            'q8[c] + succ[c]  -> fum[c] + q8h2[c]
THD2       'NAD(P) transhydrogenase'                           '2 h[e] + nadh[c] + nadp[c]  -> 2 h[c]
```

Although this is a specifc table for the reactions associated with energy management, it illusttrates how you can pull up the full reaction (enzyme) name and formula for any subsystem in the core model. It should be pointed out that although the reactions succinate dehydrogenase (SUCDi) and fumarate reductase (FRD7) are included in the oxidative phosphorylation subsystem because they are membrane-bound enzymes that interact with the quinone pool, they are a better fit functionally in the TCA cycle, as will be seen later.

Now lets explore the flux through these reactions in aerobic conditons with the glucose uptake set at -10 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$ and the oxygen uptake at -30 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. *[Timing: Seconds]*

```
model = changeRxnBounds(model,'EX_glc(e)',-10,'l'); % Set maximum glucose uptake
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set oxygen uptake
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM'); % Set the objective function
FBAsolution = optimizeCbModel(model,'max'); % Perform FBA
printLabeledData(energyReactions,FBAsolution.x(energy_rxnID))
```

```
ADK1 0
ATPM 8.39
ATPS4r 45.514
CYTBD 43.599
FRD7 994.936
NADH16 38.5346
NADTRHD 0
SUCDi 1000
THD2 0
```

Below in Figure 10 is screenshot showing these fluxes flowing through the oxidative phosphorylation section of the core map (upper right corner). In this figure we can see the electrons from nadh[c] entering the electron transport change at NADH16, flowing through the quinone pool, and then finding their way to reduce oxygen through CYTBD and O2t. With the proton-motive force in place, ATPS4r can now use that energy to convert adp[c] to atp[c]. We can also see the flux flowing through the dummy reaction ATPM that is used to model the atp[c] load required for cell growth. Finally, THD2, NADTRHD or ADK1 are not required to recycle any of the key energy cofactors.

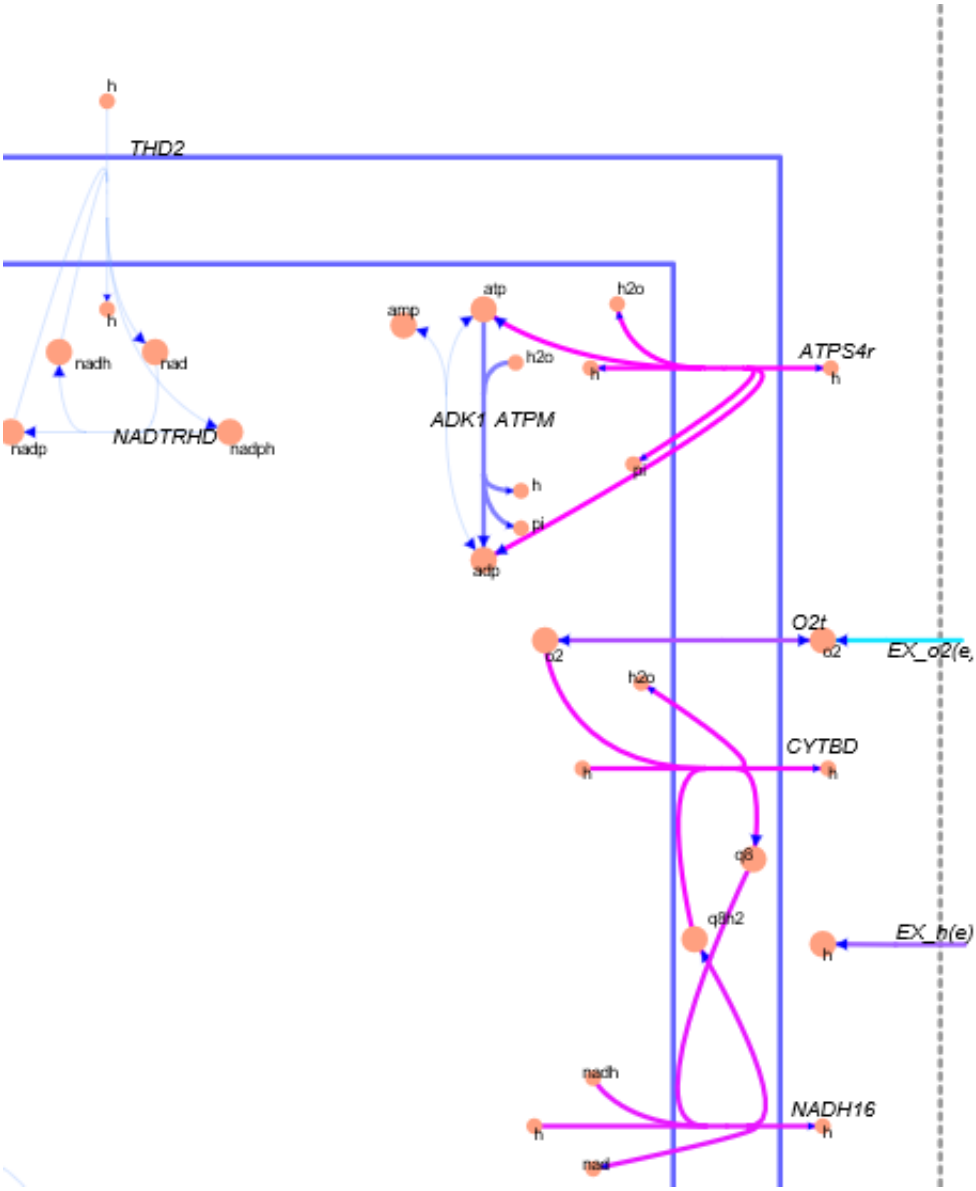


Figure 10: Close-up of the oxidative phosphorylation section of the *E.coli* core map in aerobic conditions with glucose as the sole carbon source (see Figure 7).

ATP Production

Now let's explore in more detail the production and consumption of atp[c] in the core model. The atp[c] produced by ATPS4r is added to the total cellular atp[c] flux that provides the cell's energy. Remember that in aerobic conditions, atp[c] is produced by both substrate phosphorylation and oxidative phosphorylation. All of the reactions that either produce or consume atp[c] can be found using the "surfNet" COBRA toolbox function. *[Timing: Seconds]*

```
surfNet(model, 'atp[c]', 0,FBAsolution.x,1,1)
```

```
Met #17  atp[c], ATP, C10H12N5O13P3
Consuming reactions with non-zero fluxes :
  #11  ATPM (8.39), Bd: 8.39 / 1000, ATP maintenance requirement
  atp[c] + h2o[c] -> adp[c] + h[c] + pi[c]
  #13  Biomass_Ecoli_core_w_GAM (0.87392), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
```

```
  #51  GLNS (0.22346), Bd: 0 / 1000, glutamine synthetase
  atp[c] + glu-L[c] + nh4[c] -> adp[c] + gln-L[c] + h[c] + pi[c]
  #72  PFK (7.47738), Bd: 0 / 1000, phosphofructokinase
  atp[c] + f6p[c] -> adp[c] + fdp[c] + h[c]
Producing reactions with non-zero fluxes :
  #12  ATPS4r (45.514), Bd: -1000 / 1000, ATP synthase (four protons for one ATP)
  adp[c] + 4 h[e] + pi[c] <=> atp[c] + h2o[c] + 3 h[c]
  #75  PGK (-16.0235), Bd: -1000 / 1000, phosphoglycerate kinase
  3pg[c] + atp[c] <=> 13dpg[c] + adp[c]
  #83  PYK (1.75818), Bd: 0 / 1000, pyruvate kinase
  adp[c] + h[c] + pep[c] -> atp[c] + pyr[c]
  #90  SUCOAS (-5.06438), Bd: -1000 / 1000, succinyl-CoA synthetase (ADP-forming)
  atp[c] + coa[c] + succ[c] <=> adp[c] + pi[c] + succoa[c]

Show previous steps...
```

These results show that under aerobic conditions with glucose as the sole carbon source there are four producers of atp[c] within the core model. These include ATPS4r (oxidative phosphorylation) as the primary contributor and PGK, PYK, and SOCAS (substrate phosphoylation) as secondary sources. This also shows the consumers to be GLNS, PFK, ATPM and the biomass function. As we will see later, the atp[c] associated with PFK is required by the glycolysis pathway. The atp[c] used by ATPM must be greater than or equal to 8.39 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$ to allow the cell to grow. Finally the biomass function shows that 52.27 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$(0.873922 x 59.81) is used for the cell's biosynthesis needs.

One of the important concepts associated with these constraint-based steady state models is that the total cell fluxes for key cofactors like atp[c] and adp[c] must be equal. This means that for every atp[c] metabolite that is produced, one adp[c] metabolite will be consumed, but to maintain the mass balance throughout the cell somewhere else in the cell an adp[c] molecule will be created from another atp[c] molecule. Thus, the total cellular atp[c] flux must equal the total cellular adp[c] flux. This can be observed using the COBRA Toolbox function called "computeFluxSplits" as shown below. *[Timing: Seconds]*

```
[P, C, vP, vC] = computeFluxSplits(model, {'adp[c]'}, FBAsolution.x);
total_adp_flux = sum(vP)
```

```
total_adp_flux = 68.3601
```


```
[P, C, vP, vC] = computeFluxSplits(model, {'adp[c]'}, FBAsolution.x);
total_adp_flux = sum(vP)
```

```
total_adp_flux = 68.3601
```

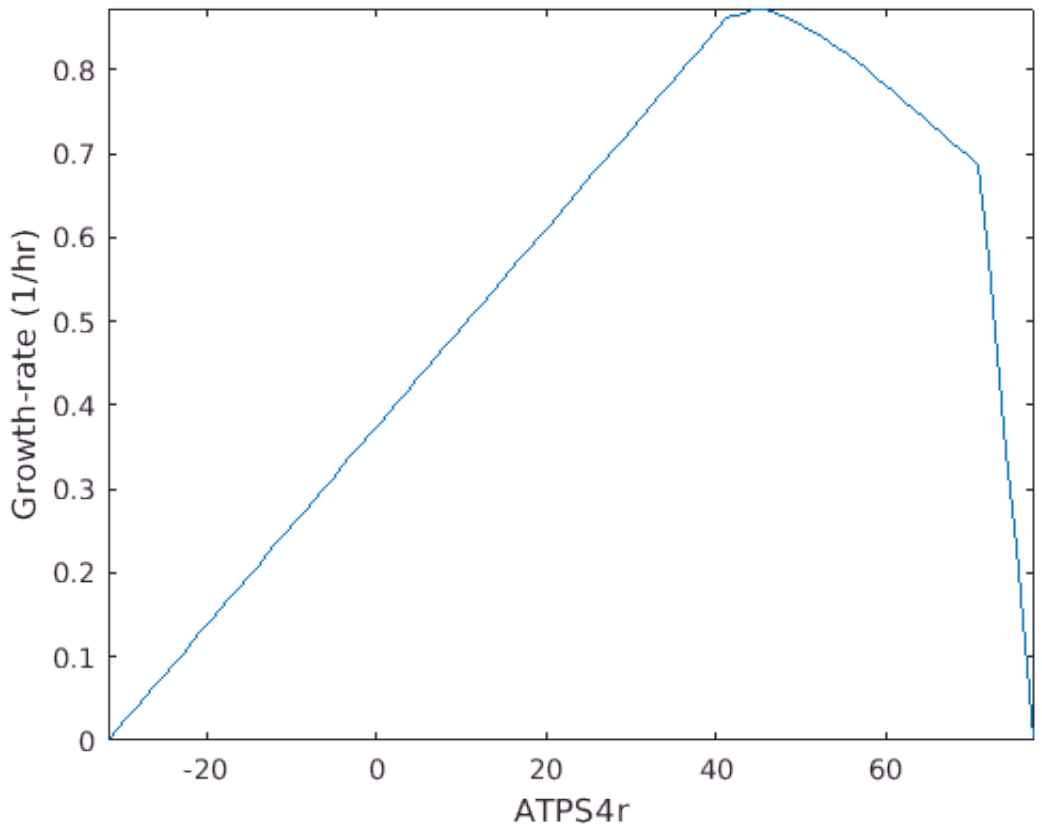

These results show that the amount of atp[c] flux in the cell equals the amount of adp[c] flux. Thus, the adp[c]/atp[c] flux ratio is 1. This is also true for nadp[c]/nadph[c] and the nad[c]/nadh[c] flux ratios.

Another way to explore the ATPS4r's ability to produce atp[c] is through the use of robustness analysis [12]. Assuming that the objective function is the biomass function (growth-rate), then the following simulation illustrates that the maximum atp[c] flux that can be supported by ATPS4r under aerobic conditions with glucose as the sole carbon source. *[Timing: Minutes]*

```
model = changeRxnBounds(model,'EX_glc(e)',-10,'l'); % Set maximum glusose uptake
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set oxygen uptake
[controlFlux, objFlux] = robustnessAnalysis(model,'ATPS4r',100);
```

```
Robustness analysis in progress ...
1%      [                                        ]2%      [                                        ]3%
```

```
ylabel('Growth-rate (1/hr)');
```



This graph shows the entire capability of ATPS4r when the carbon source glucose has a maximum uptake rate greater than or equal to -10 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. If we start at the left of this of this figure, it can be seen that ATPS4r takes on negative values which implies that instead of producing atp[c] through the proton-motive force, it has become an energy-dependent proton pump removing protons from the cytoplasm and transporting them to the extracellular space. Note that the growth-rate under these anaerobic conditions is small. As the flux through ATPS4r becomes positive it starts producing atp[c] providing the majority of the atp[c] required for aerobic operation. At the beginning of aerobic operation there is a nice linear relationship between the produced atp[c] and the growth-rate.

Eventually the growth-rate reaches a maximum of 0.8738 hr^{-1} when the ATPS4r flux level reaches 45.54 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. After the maximum growth-rate has been achieved the cell then needs to find ways to recycle the extra ATP. This can be seen below by fixing the flux through ATPS4r to a value greater than 45.54 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. *[Timing: Seconds]*

```
model = e_coli_core; % Starting the original model
model = changeRxnBounds(model,'EX_glc(e)',-10,'l'); % Set maximum glucose uptake
model = changeRxnBounds(model,'ATPS4r',60,'b'); % Fix ATPS4r flux rate
FBAsolution = optimizeCbModel(model,'max'); % Perform FBA
surfNet(model, 'atp[c]', 0,FBAsolution.x,1,1)
```

```
Met #17  atp[c], ATP, C10H12N5O13P3
Consuming reactions with non-zero fluxes :
  #7  ADK1 (4.01673), Bd: -1000 / 1000, adenylate kinase
  amp[c] + atp[c] <=> 2 adp[c]
  #11  ATPM (13.7463), Bd: 8.39 / 1000, ATP maintenance requirement
  atp[c] + h2o[c] -> adp[c] + h[c] + pi[c]
```

```
  #13  Biomass_Ecoli_core_w_GAM (0.78116), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #51  GLNS (0.19974), Bd: 0 / 1000, glutamine synthetase
  atp[c] + glu-L[c] + nh4[c] -> adp[c] + gln-L[c] + h[c] + pi[c]
  #72  PFK (1.0954), Bd: 0 / 1000, phosphofructokinase
  atp[c] + f6p[c] -> adp[c] + fdp[c] + h[c]
  #81  PPS (4.01673), Bd: 0 / 1000, phosphoenolpyruvate synthase
  atp[c] + h2o[c] + pyr[c] -> amp[c] + 2 h[c] + pep[c] + pi[c]
Producing reactions with non-zero fluxes :
  #12  ATPS4r (60), Bd: 60 / 60, ATP synthase (four protons for one ATP)
  adp[c] + 4 h[e] + pi[c] -> atp[c] + h2o[c] + 3 h[c]
  #75  PGK (-9.79587), Bd: -1000 / 1000, phosphoglycerate kinase
  3pg[c] + atp[c] <=> 13dpg[c] + adp[c]

Show previous steps...
```

```
map=readCbMap('ecoli_core_map');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options); % Draw the flux values on the map "target.svg"
```

```
Document Written
```

If we compare these results with the previous fluxes calculated for the optimized cell performance under aerobic conditions with a similar glucose carbon source uptake, we can see the differences in atp[c] flux distribution.To start with it can be seen that the flux through ATPM increases (13.74 > 8.39). Notice that ADK1 has been activated to recycle atp[c] to adp[c]. Since the growth-rate decreases, we would also expect the flux used by the biomass function to decrease along with other parts of the cell by selecting alternate pathways to help absorb the extra atp[c]. This is illustrated in the core metabolic map shown below.

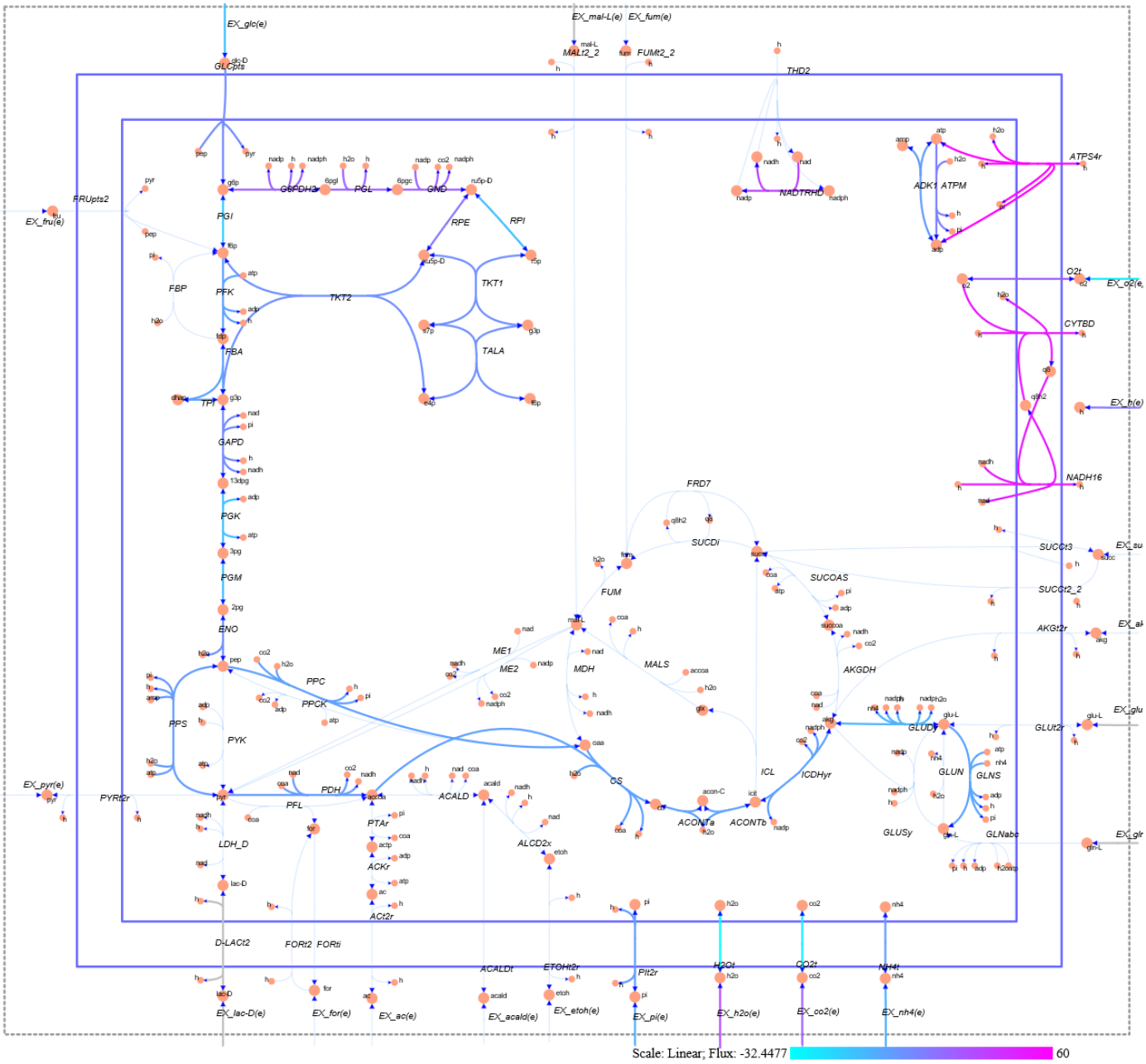


Figure 11. A screenshot of the core map with ATPS4r fixed at 60 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$.

NADH Production

Now that we have explored the production and consumption of atp[c], let's look at the producers and consumers of nadh[c]. *[Timing: Seconds]*

```
model = e_coli_core; % Starting with the original model
model = changeRxnBounds(model,'EX_glc(e)',-10,'l'); % Set maximum glucose uptake
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set oxygen uptake
FBAsolution = optimizeCbModel(model,'max'); % Perform FBA
surfNet(model,'nadh[c]',0,FBAsolution.x,1,1)
```

```
Met #51  nadh[c], Nicotinamide-adenine-dinucleotide-reduced, C21H27N7O14P2
Consuming reactions with non-zero fluxes :
  #67  NADH16 (38.5346), Bd: 0 / 1000, NADH dehydrogenase (ubiquinone-8 & 3 protons)
  4 h[c] + nadh[c] + q8[c] -> 3 h[e] + nad[c] + q8h2[c]
```

```
Producing reactions with non-zero fluxes :
  #8  AKGDH (5.06438), Bd: 0 / 1000, 2-Oxoglutarate dehydrogenase
  akg[c] + coa[c] + nad[c] -> co2[c] + nadh[c] + succoa[c]
  #13  Biomass_Ecoli_core_w_GAM (0.87392), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #49  GAPD (16.0235), Bd: -1000 / 1000, glyceraldehyde-3-phosphate dehydrogenase
  g3p[c] + nad[c] + pi[c] <=> 13dpg[c] + h[c] + nadh[c]
  #64  MDH (5.06438), Bd: -1000 / 1000, malate dehydrogenase
  mal-L[c] + nad[c] <=> h[c] + nadh[c] + oaa[c]
  #71  PDH (9.28253), Bd: 0 / 1000, pyruvate dehydrogenase
  coa[c] + nad[c] + pyr[c] -> accoa[c] + co2[c] + nadh[c]

Show previous steps...
```

Note that in this case, the only consumer of nadh[c] is NAD16 which is the beginning of the electron transport chain. The producing reactions, as we will discuss later, are primarily located in the glycolysis and TCA pathways. Note that for this core model, the biomass function is also listed as a producer. Since the biomass function represents all the functionality not included in the core model (e.g. biosynthesis pathways), this implies that NADH would be produced in other parts of the cell that are not included in this simple core model. The flux supplied through the biomass function is calculated by multiplying the total biomass flux (0.873922) times the nadh[c] biomass function coefficient (3.547) to yielding a total nadh[c] biomass flux of 3.0998 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. This can also be calculated using the COBRA Toolbox function "computeFluxSplits" as follows. *[Timing: Seconds]*

```
[nadh_P, nadh_C, nadh_vP, nadh_vC] = computeFluxSplits(model, {'nadh[c]'}, FBAsolution.x);
[tmp,nadh_rxnID] = ismember('Biomass_Ecoli_core_w_GAM',model.rxns);
nadhBiomassFlux = nadh_vP(nadh_rxnID)
```

```
nadhBiomassFlux = 3.0998
```

NADPH production

Finally, we can also obtain this same information for nadph[c], the reducing power for cellular biosynthesis. *[Timing: Seconds]*

```
surfNet(model,'nadph[c]',0,FBAsolution.x,1,1)
```

```
Met #53  nadph[c], Nicotinamide-adenine-dinucleotide-phosphate-reduced, C21H26N7O17P3
Consuming reactions with non-zero fluxes :
  #13  Biomass_Ecoli_core_w_GAM (0.87392), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #53  GLUDy (-4.54186), Bd: -1000 / 1000, glutamate dehydrogenase (NADP)
  glu-L[c] + h2o[c] + nadp[c] <=> akg[c] + h[c] + nadph[c] + nh4[c]
Producing reactions with non-zero fluxes :
  #48  G6PDH2r (4.95998), Bd: -1000 / 1000, glucose 6-phosphate dehydrogenase
  g6p[c] + nadp[c] <=> 6pgl[c] + h[c] + nadph[c]
  #57  GND (4.95998), Bd: 0 / 1000, phosphogluconate dehydrogenase
  6pgc[c] + nadp[c] -> co2[c] + nadph[c] + ru5p-D[c]
  #59  ICDHyr (6.00725), Bd: -1000 / 1000, isocitrate dehydrogenase (NADP)
  icit[c] + nadp[c] <=> akg[c] + co2[c] + nadph[c]

Show previous steps...
```

Due to the simplicity of the *E.coli* core model, most of the nadph[c] is consumed by the biomass function (0.873922 x 13.0279 = 11.385) to support the cell's biosynthesis needs. The other consumer is the nitrogen metabolism (GLUDy). On the other hand, nadph[c] is produced by reactions in the oxidative phosphorylation pathways, pentose phosphate pathway, and the TCA cycle. It is worth pointing out that

in the larger models, that incorporate most of the cells biosynthesis pathways, the number of reactions consuming nadph[c] could be very large. *[Timing: Seconds]*

Anaerobic Respiration

Now let's turn our attention to anaerobic cell operation. During aerobic respiration, oxygen is the terminal electron acceptor for the electron transport chain, which yields the bulk of atp[c] required for biosynthesis. Anaerobic respiration refers to respiration without molecular oxygen. For anaerobic respiration, *E. coli* only generates atp[c] by substrate level phosphorylation. Glycolysis results in the net production of two atp[c] per glucose by substrate level phosphorylation, but this is low compared to the total atp[c] production of 17.5 atp[c] per glucose for aerobic respiration [1].

The substrates of fermentation are typically sugars, so during fermentative growth, it is necessary for each cell to support large flux values through glycolysis to generate sufficient atp[c] to drive cell growth. Glycolysis also produces two molecules of nadh[c] for each molecule of glucose [1]. As a result, nadh[c] must be reoxidized by fermentation in order to regenerate nad[c] necessary to maintain the oxidation-reduction balance of the cell.

Figure 12 is a map of anaerobic operation using glucose as the only carbon source.

```
model = e_coli_core; % Starting with the original model
model = changeRxnBounds(model,'EX_glc(e)',-10,'l'); % Set maximum glusose uptake
model = changeRxnBounds(model,'EX_o2(e)',-0,'l'); % Set maximum oxygen uptake
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM'); % Set the objective function
FBAsolution = optimizeCbModel(model,'max'); % Perform FBA
map=readCbMap('ecoli_core_map');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options); % Draw the flux values on the map "target.svg"
```

```
Document Written
```

A screenshot of the produced map of anaerobic operation is shown below.

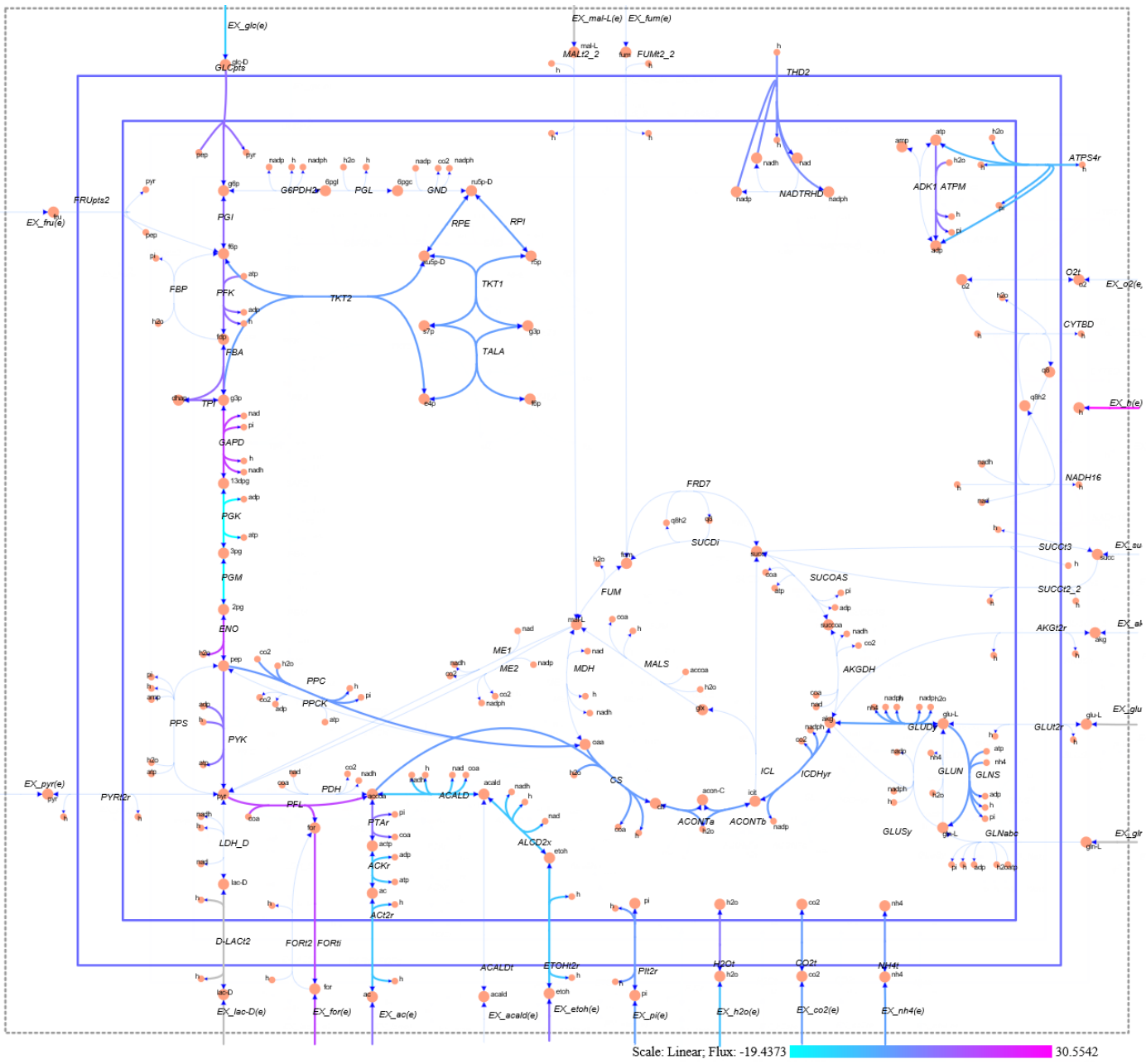


Figure 12. Network map of the *E.coli* core model with glucose as the carbon source (EX_glc(e) ≥ -10 $\mathrm{mmol \cdot gDW^{-1} \cdot hr^{-1}}$) in an anaerobic environment (EX_o2(e) ≥ 0 $\mathrm{mmol \cdot gDW^{-1} \cdot hr^{-1}}$).

Note that for anaerobic operation the flux through oxidative phosphorylation pathways (electron transport chain) is zero. Let's look at the nonzero fluxes associated with anaerobic operation to understand the role of THD2 and ATPS4r. *[Timing: Seconds]*

```
Reactions = transpose({'ATPS4r','THD2'});
[tmp,rxnID] = ismember(Reactions,model.rxns);
printLabeledData(Reactions,FBAsolution.x(rxnID))
```

```
ATPS4r -5.45205
THD2 3.62919
```

Now let's look at the fomulas for these reactions to understand what is happening in this condition. *[Timing: Seconds]*

```
printRxnFormula(model,Reactions)
```

```
ATPS4r adp[c] + 4 h[e] + pi[c]  <=> atp[c] + h2o[c] + 3 h[c]
THD2 2 h[e] + nadh[c] + nadp[c]  -> 2 h[c] + nad[c] + nadph[c]
ans =
    'adp[c] + 4 h[e] + pi[c]  <=> atp[c] + h2o[c] + 3 h[c] '
    '2 h[e] + nadh[c] + nadp[c]  -> 2 h[c] + nad[c] + nadph[c] '
```

Since the flux for ATPS4r is negative, we can assume that ATPS4r is operating in reverse and pumping protons from the cytoplasm into the extracellular space. Some of these protons can now be used by THD2 to convert nadh[c], which is not needed for the electron transport chain, into nadph[c] where they can be used for cellular biosynthesis.

All the nonzero fluxes for this anaerobic example are printed below. *[Timing: Seconds]*

```
printFluxVector(model,FBAsolution.x,true) % only print nonzero reaction fluxes
```

```
ACALD -8.27946
ACKr -8.50359
ACONTa 0.228363
ACONTb 0.228363
ACt2r -8.50359
ALCD2x -8.27946
ATPM 8.39
ATPS4r -5.45205
Biomass_Ecoli_core_w_GAM 0.211663
CO2t 0.378178
CS 0.228363
ENO 19.1207
ETOHt2r -8.27946
EX_ac(e) 8.50359
EX_co2(e) -0.378178
EX_etoh(e) 8.27946
EX_for(e) 17.8047
EX_glc(e) -10
EX_h(e) 30.5542
EX_h2o(e) -7.1158
EX_nh4(e) -1.15416
EX_pi(e) -0.778644
FBA 9.78946
FORti 17.8047
GAPD 19.4373
GLCpts 10
GLNS 0.0541222
GLUDy -1.10003
H2Ot 7.1158
ICDHyr 0.228363
NH4t 1.15416
PFK 9.78946
PFL 17.8047
PGI 9.95661
PGK -19.4373
PGM -19.1207
PIt2r 0.778644
PPC 0.606541
PTAr 8.50359
PYK 8.40427
RPE -0.152143
RPI -0.152143
TALA -0.0378665
THD2 3.62919
TKT1 -0.0378665
```

```
TKT2 -0.114277
TPI 9.78946
```

So one question that could be asked is this anaerobic environment is, where is the nadh[c] produced and where is it consumed. Using "surfNet" we can find out. *[Timing: Seconds]*

```
surfNet(model, 'nadh[c]',0,FBAsolution.x,1,1)
```

```
Met #51  nadh[c], Nicotinamide-adenine-dinucleotide-reduced, C21H27N7O14P2
Consuming reactions with non-zero fluxes :
  #1  ACALD (-8.27946), Bd: -1000 / 1000, acetaldehyde dehydrogenase (acetylating)
  acald[c] + coa[c] + nad[c] <=> accoa[c] + h[c] + nadh[c]
  #10  ALCD2x (-8.27946), Bd: -1000 / 1000, alcohol dehydrogenase (ethanol)
  etoh[c] + nad[c] <=> acald[c] + h[c] + nadh[c]
  #92  THD2 (3.62919), Bd: 0 / 1000, NAD(P) transhydrogenase
  2 h[e] + nadh[c] + nadp[c] -> 2 h[c] + nad[c] + nadph[c]
Producing reactions with non-zero fluxes :
  #13  Biomass_Ecoli_core_w_GAM (0.21166), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #49  GAPD (19.4373), Bd: -1000 / 1000, glyceraldehyde-3-phosphate dehydrogenase
  g3p[c] + nad[c] + pi[c] <=> 13dpg[c] + h[c] + nadh[c]

Show previous steps...
```

In this case, the nadh[c] is primarily used to support mixed fermentation through the ethanol pathway. This will be described in the fermentation section.

Now let's explore the production of atp[c] in an anaerobic environment. *[Timing: Seconds]*

```
surfNet(model, 'atp[c]',0,FBAsolution.x,1,1)
```

```
Met #17  atp[c], ATP, C10H12N5O13P3
Consuming reactions with non-zero fluxes :
  #11  ATPM (8.39), Bd: 8.39 / 1000, ATP maintenance requirement
  atp[c] + h2o[c] -> adp[c] + h[c] + pi[c]
  #12  ATPS4r (-5.45205), Bd: -1000 / 1000, ATP synthase (four protons for one ATP)
  adp[c] + 4 h[e] + pi[c] <=> atp[c] + h2o[c] + 3 h[c]
  #13  Biomass_Ecoli_core_w_GAM (0.21166), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #51  GLNS (0.05412), Bd: 0 / 1000, glutamine synthetase
  atp[c] + glu-L[c] + nh4[c] -> adp[c] + gln-L[c] + h[c] + pi[c]
  #72  PFK (9.78946), Bd: 0 / 1000, phosphofructokinase
  atp[c] + f6p[c] -> adp[c] + fdp[c] + h[c]
Producing reactions with non-zero fluxes :
  #3  ACKr (-8.50359), Bd: -1000 / 1000, acetate kinase
  ac[c] + atp[c] <=> actp[c] + adp[c]
  #75  PGK (-19.4373), Bd: -1000 / 1000, phosphoglycerate kinase
  3pg[c] + atp[c] <=> 13dpg[c] + adp[c]
  #83  PYK (8.40427), Bd: 0 / 1000, pyruvate kinase
  adp[c] + h[c] + pep[c] -> atp[c] + pyr[c]

Show previous steps...
```

As can be seen above, the production of atp[c] is exclusively through substrate phosphorylation (ACKr, PGK, PYK).

Finally, the nadph[c] producers and consumers are shown below. *[Timing: Seconds]*

```
surfNet(model, 'nadph[c]',0,FBAsolution.x,1,1)
```

```
Met #53  nadph[c], Nicotinamide-adenine-dinucleotide-phosphate-reduced, C21H26N7O17P3
Consuming reactions with non-zero fluxes :
  #13  Biomass_Ecoli_core_w_GAM (0.21166), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #53  GLUDy (-1.10003), Bd: -1000 / 1000, glutamate dehydrogenase (NADP)
  glu-L[c] + h2o[c] + nadp[c] <=> akg[c] + h[c] + nadph[c] + nh4[c]
Producing reactions with non-zero fluxes :
  #59  ICDHyr (0.22836), Bd: -1000 / 1000, isocitrate dehydrogenase (NADP)
  icit[c] + nadp[c] <=> akg[c] + co2[c] + nadph[c]
  #92  THD2 (3.62919), Bd: 0 / 1000, NAD(P) transhydrogenase
  2 h[e] + nadh[c] + nadp[c] -> 2 h[c] + nad[c] + nadph[c]

Show previous steps...
```

Note that the primary producer of nadph[c] in this anaerobic environment is THD2, which converts the surplus nadh[c] to nadph[c].

4.B. Glycolysis Pathway

Now that we have completed the exploration of the energy management subsystem of the core model, it is time to start looking at the other included subsytems. Glycolysis is the metabolic pathway in the *E.coli* core model that converts glucose and fructose into pyruvate. The free energy released in this process is used to form the high-energy compounds of atp[c] and nadh[c]. The location of the glycolysis pathway on the *E.coli* core map is highlighted in the Figure 13.

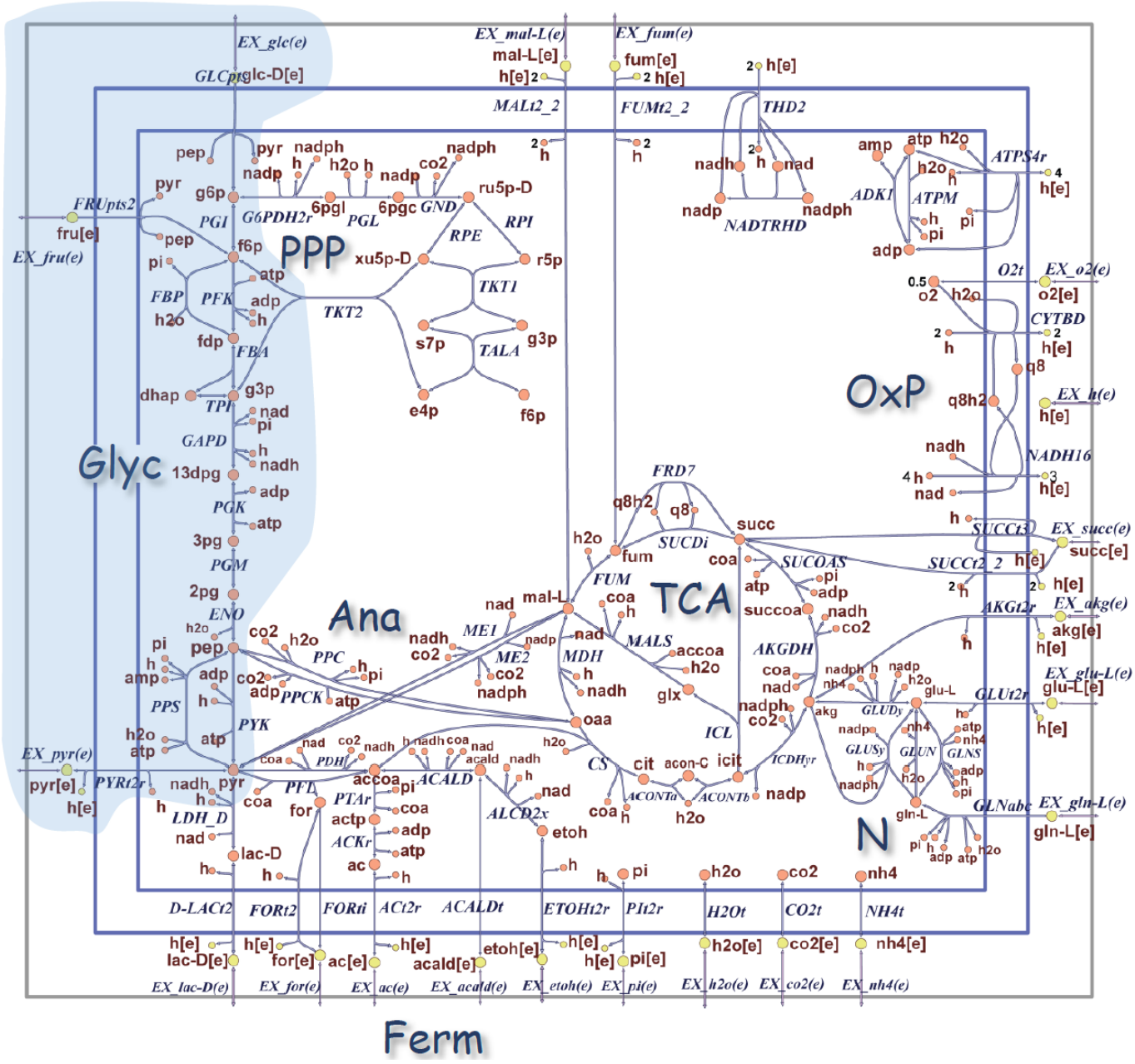


Figure 13. The location of the glycolysis pathway subsystem reactions are highlighted in blue on the *E.coli* core map [3].

A table showing the reactions associated with the glycolysis pathway can be extracted from the core model as follows: *[Timing: Seconds]*

```
model = e_coli_core; % Starting with the original model
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-30,'l');
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');
glycolysisSubystem = {'Glycolysis/Gluconeogenesis'};
glycolysisReactions = model.rxns(ismember(model.subSystems,glycolysisSubystem));
[tmp,glycolysis_rxnID] = ismember(glycolysisReactions,model.rxns);
Reaction_Names = model.rxnNames(glycolysis_rxnID);
Reaction_Formulas = printRxnFormula(model,glycolysisReactions,0);
T = table(Reaction_Names,Reaction_Formulas,'RowNames',glycolysisReactions)
```

```
T =
```

	Reaction_Names	Reaction_Formulas
ENO	'enolase'	'2pg[c] <=> h2o[c] + pep[c] '
FBA	'fructose-bisphosphate aldolase'	'fdp[c] <=> dhap[c] + g3p[c] '
FBP	'fructose-bisphosphatase'	'fdp[c] + h2o[c] -> f6p[c] + pi[c] '
GAPD	'glyceraldehyde-3-phosphate dehydrogenase'	'g3p[c] + nad[c] + pi[c] <=> 13dpg[c] + h[c]
PDH	'pyruvate dehydrogenase'	'coa[c] + nad[c] + pyr[c] -> accoa[c] + co2[c
PFK	'phosphofructokinase'	'atp[c] + f6p[c] -> adp[c] + fdp[c] + h[c] '
PGI	'glucose-6-phosphate isomerase'	'g6p[c] <=> f6p[c] '
PGK	'phosphoglycerate kinase'	'3pg[c] + atp[c] <=> 13dpg[c] + adp[c] '
PGM	'phosphoglycerate mutase'	'2pg[c] <=> 3pg[c] '
PPS	'phosphoenolpyruvate synthase'	'atp[c] + h2o[c] + pyr[c] -> amp[c] + 2 h[c]
PYK	'pyruvate kinase'	'adp[c] + h[c] + pep[c] -> atp[c] + pyr[c] '
TPI	'triose-phosphate isomerase'	'dhap[c] <=> g3p[c] '

It should be pointed out that although the reaction pyrvate dehydrogenase (PDH) is included in the glycolysis subsystem it is functionally a better fit in the "Glycoxylate Cycle, Gluconeogenesis, and Anapleurotic Reactions" subsystem, as described in section 4.E.

In addition to providing some atp[c] through substrate phosphorylation (PGK and PYK), the glycolysis pathway also proves a major source of nadh[c] (GAPD) that is used to power the electron transport chain. It also supplies several key precursors needed for the biosynthesis pathways. These precursors include: D-Glucose 6-phosphate (g6p[c]) a precursor for sugar nucleotides, D-Fructose 6-phosphate (f6p[c]) a precursor for amino sugars, glyceraldehyde 3-phosphate (g3p[c]) a precursor for phospholipids, 3-Phospho-D-glycerate (3pg[c]) a precursor for cysteine, glycine, and serine, phosphoenolpyruvate (pep[c]) a precursor for tyrosine, tryptophan and phenylalanine, and finally pyruvate (pyr[c]) the precursor for alanine, leucine, and valine [5]. These precursors and their location on the glycolysis pathway are illustrated in Figure 14.

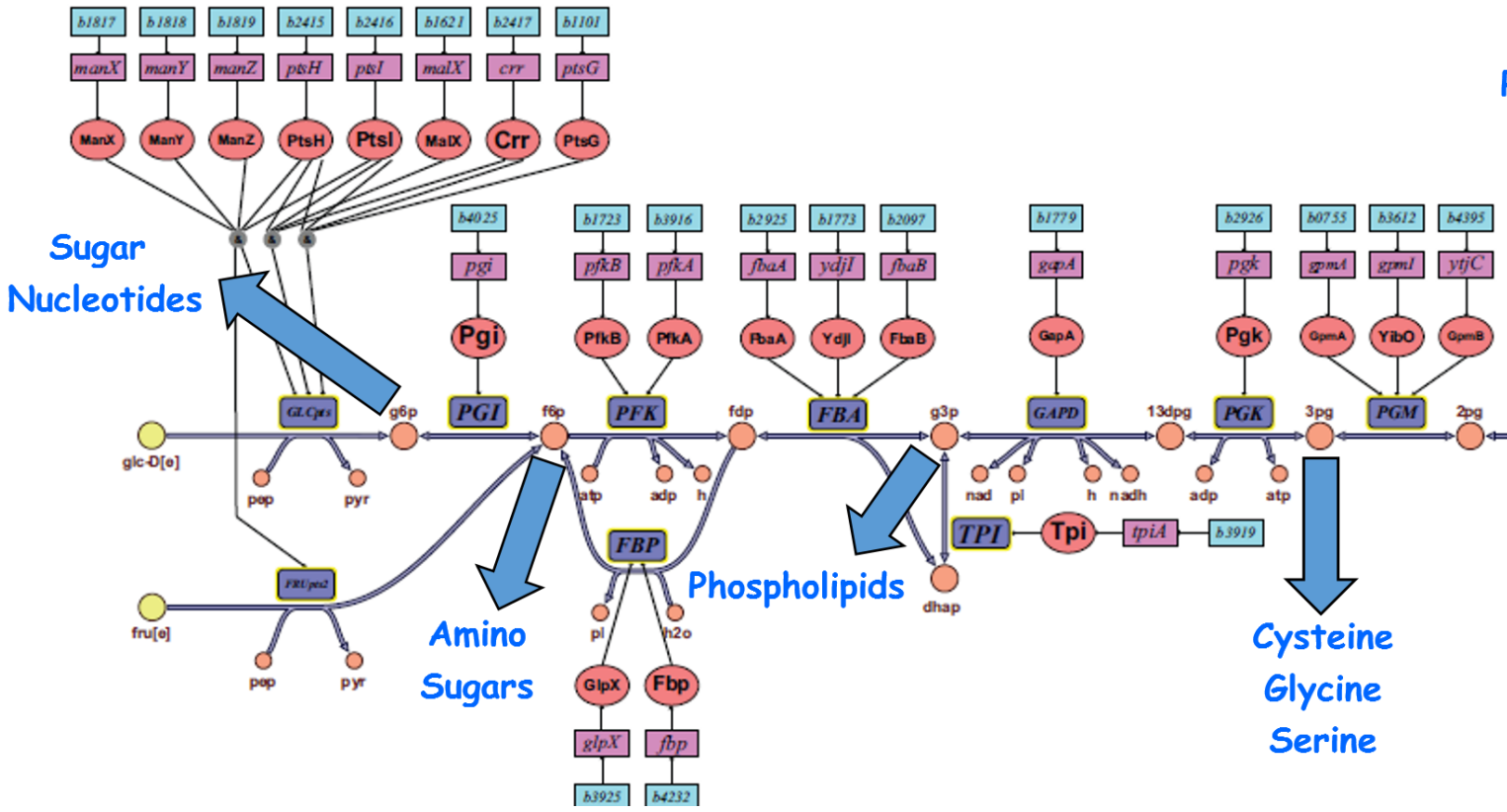


Figure 14. Precursors produced in the glycolysis pathway [3].

Visualizing the flux though the glycolysis pathways can be seen by using the draw package available with COBRA Toolbox. This is illustrated in the Matlab and COBRA toolbox code listed below for the case of anaerobic operation with fructose as the carbon source. *[Timing: Seconds]*

```
model = e_coli_core; % Starting with the original model
model = changeRxnBounds(model,'EX_glc(e)',0,'l');
model = changeRxnBounds(model,'EX_fru(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');
FBAsolution = optimizeCbModel(model,'max',0,0);

% Import E.coli core map and adjust parameters
map=readCbMap('ecoli_Textbook_ExportMap');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);
```

```
Document Written
```

A screenshot of the saved "target.svg" file is shown in the Figure 15.

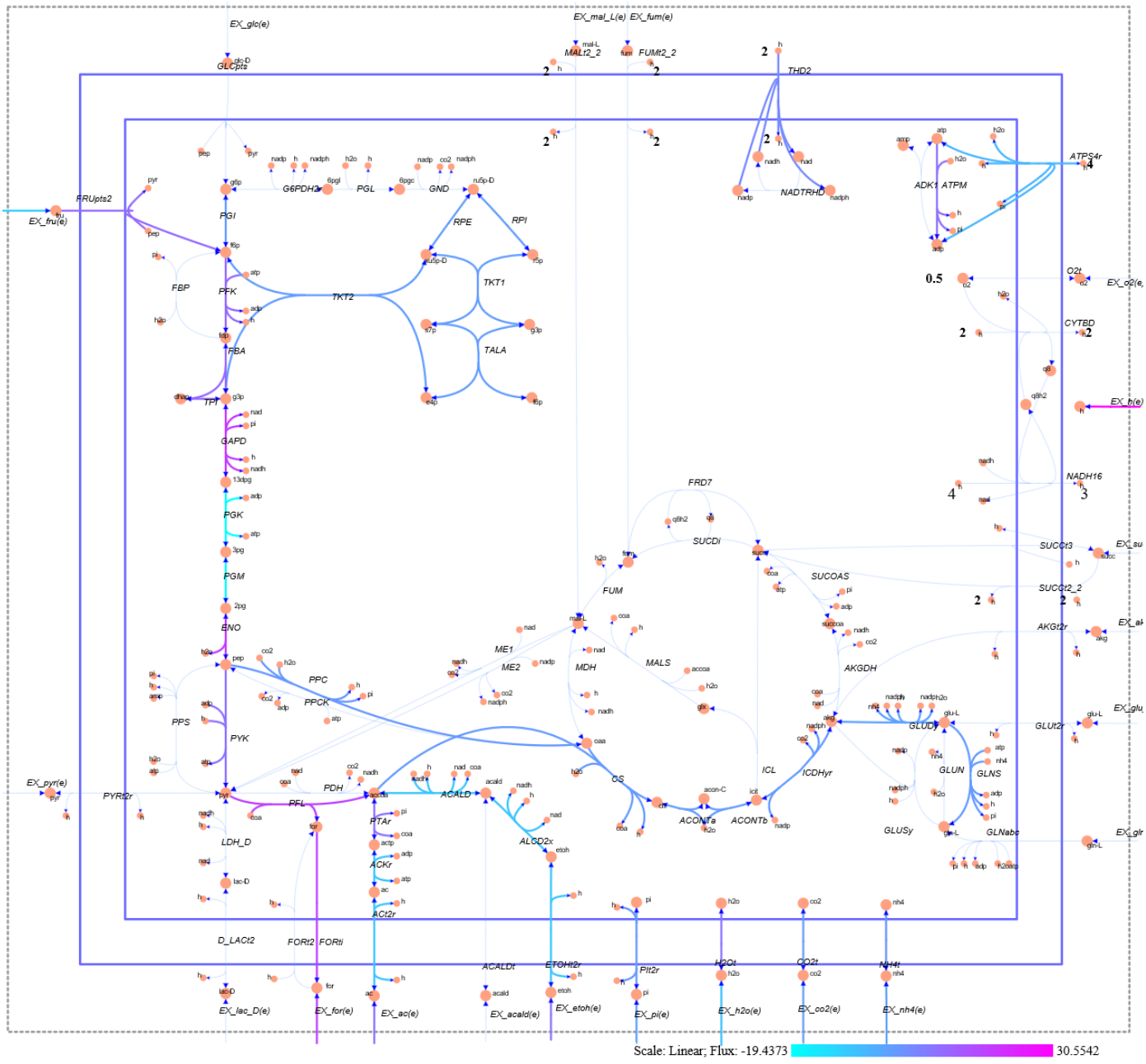


Figure 15. Network map of the *E.coli* core model using fructose as the carbon source (EX_fru(e) ≥ -10 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$) in an anaerobic environment (EX_o2(e) ≥ 0 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$).

Note that the fructose enters the network on the top left of the map. The detailed flux values for all the active reactions are shown below. *[Timing: Seconds]*

```
% Print the non-zero flux values
printFluxVector(model, FBAsolution.x, true)
```

```
ACALD -8.27946
ACKr -8.50359
ACONTa 0.228363
ACONTb 0.228363
ACt2r -8.50359
ALCD2x -8.27946
ATPM 8.39
ATPS4r -5.45205
Biomass_Ecoli_core_w_GAM 0.211663
```

```
CO2t 0.378178
CS 0.228363
ENO 19.1207
ETOHt2r -8.27946
EX_ac(e) 8.50359
EX_co2(e) -0.378178
EX_etoh(e) 8.27946
EX_for(e) 17.8047
EX_fru(e) -10
EX_h(e) 30.5542
EX_h2o(e) -7.1158
EX_nh4(e) -1.15416
EX_pi(e) -0.778644
FBA 9.78946
FORti 17.8047
FRUpts2 10
GAPD 19.4373
GLNS 0.0541222
GLUDy -1.10003
H2Ot 7.1158
ICDHyr 0.228363
NH4t 1.15416
PFK 9.78946
PFL 17.8047
PGI -0.0433909
PGK -19.4373
PGM -19.1207
PIt2r 0.778644
PPC 0.606541
PTAr 8.50359
PYK 8.40427
RPE -0.152143
RPI -0.152143
TALA -0.0378665
THD2 3.62919
TKT1 -0.0378665
TKT2 -0.114277
TPI 9.78946
```

The consumers of precursors formed in the glycolysis pathways can be found using the "surfNet" COBRA Toolbox function. An example looking for both the producers and consumers of "f6p[c]," a precursor for amino sugars is shown below. *[Timing: Seconds]*

```
surfNet(model, 'f6p[c]',0,FBAsolution.x,1,1)
```

```
Met #26  f6p[c], D-Fructose-6-phosphate, C6H11O9P
Consuming reactions with non-zero fluxes :
  #13  Biomass_Ecoli_core_w_GAM (0.21166), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #72  PFK (9.78946), Bd: 0 / 1000, phosphofructokinase
  atp[c] + f6p[c] -> adp[c] + fdp[c] + h[c]
  #74  PGI (-0.04339), Bd: -1000 / 1000, glucose-6-phosphate isomerase
  g6p[c] <=> f6p[c]
  #91  TALA (-0.03787), Bd: -1000 / 1000, transaldolase
  g3p[c] + s7p[c] <=> e4p[c] + f6p[c]
  #94  TKT2 (-0.11428), Bd: -1000 / 1000, transketolase
  e4p[c] + xu5p-D[c] <=> f6p[c] + g3p[c]
Producing reactions with non-zero fluxes :
  #45  FRUpts2 (10), Bd: 0 / 1000, Fructose transport via PEP:Pyr PTS (f6p generating)
  fru[e] + pep[c] -> f6p[c] + pyr[c]

Show previous steps...
```

Note that the majority of the f6p[c] flux is directed down the glycolysis pathway (PFK), a modest amount is directed to the pentose phosphate pathway (PGI, TALA, TKT2), with a small amount directed to the biomass function (0.211663 x 0.0709 = 0.015) which represents the biosynthesis load of the precursors. A similar approach can be used to understand the producer/consumer relationships with the other glycolytic precursors.

Using the COBRA Toolbox, it is possible to create a table of reactions and their flux values for both glycolysis supported carbon sources, glucose and fructose. This is illustrated below. *[Timing: Seconds]*

```
% Starting with the original model
model = e_coli_core;

% Obtain the rxnIDs for the glycolysis pathway reactions
[tmp,glycolysis_rxnID] = ismember(glycolysisReactions,model.rxns);

% Glucose aerobic flux
FBAsolution = optimizeCbModel(model,'max',0,0);
Glucose_Aerobic_Flux = FBAsolution.x(glycolysis_rxnID);

% Fructose aerobic flux
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_fru(e)',-10,'l');
FBAsolution = optimizeCbModel(model,'max',0,0);
Fructcose_Aerobic_Flux = FBAsolution.x(glycolysis_rxnID);

% Set anaerobic conditions
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');

% Glucose anaerobic flux
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
FBAsolution = optimizeCbModel(model,'max',0,0);
Glucose_Anaerobic_Flux = FBAsolution.x(glycolysis_rxnID);

% Fructose anaerobic flux
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_fru(e)',-10,'l');
FBAsolution = optimizeCbModel(model,'max',0,0);
Fructose_Anaerobic_Flux = FBAsolution.x(glycolysis_rxnID);

T = table(Glucose_Aerobic_Flux,Fructcose_Aerobic_Flux,Glucose_Anaerobic_Flux,...
    Fructose_Anaerobic_Flux,'RowNames',glycolysisReactions)
```

```
T =
```

	Glucose_Aerobic_Flux	Fructcose_Aerobic_Flux	Glucose_Anaerobic_Flux	Fructose_Anaerob
ENO	14.716	14.716	37.855	19.121
FBA	7.4774	7.4774	19.486	9.7895
FBP	0	0	0	0
GAPD	16.024	16.024	38.628	19.437
PDH	9.2825	9.2825	0	0
PFK	7.4774	7.4774	19.486	9.7895
PGI	4.8609	-5.1391	9.8942	-0.043391
PGK	-16.024	-16.024	-38.628	-19.437
PGM	-14.716	-14.716	-37.855	-19.121
PPS	0	0	0	0
PYK	1.7582	1.7582	16.108	8.4043
TPI	7.4774	7.4774	19.486	9.7895

From this table, it can be seen that in all four situations, the flux flows from the carbon source at the top left of the metabolic maps down the glycolysis pathway to form pyruvate in the lower right. In aerobic conditions, part of the flux is diverted to the G6PDH2r entrance to the pentose phosphate pathways. For the anaerobic case, the flux is only diverted to the lower half of the pentose phosphate pathway (TKT2) to produce the pentose phosphate pathway precursors. Also note that the flux through GAPD has almost doubled since the number of g3p[c] metabolites leaving the FBA and TPI reaction are double the number of fdp[c] metabolites entering FBA. This is possible since the output of FBA provides both a molecule of g3p[c] and a molecule of dhap[c]. The dhap[c] is rapidly converted to g3p[c] thus creating the effect of doubling the g3p[c] entering GAPD. A more detailed understanding of the fluxes through glycolysis using the COBRA toolbox is left as an exploration opportunity for the reader.

4.C. Pentose Phosphate Pathway

The primary purpose of the pentose phosphate pathway (PPP) is to provide the 4-, 5- and 7-carbon precursors for the cell and produce nadph[c]. The 4-, 5- and 7-carbon precursors include D-erythrose-4-phosphate (e4p[c]), alpha-D-ribose-5-phosphate, (r5p[c]), and sedoheptulose-7-phosphate (s7p[c]), respectively. The nadph[c] is produced in the oxidative pathway by glucose-6-phosphate dehydrogenase (G6PDH2r) and phosphogluconate dehydrogenase (GND).

The location of the reactions associated with the PPP are shown below on the *E.coli* core map in Figure 16.

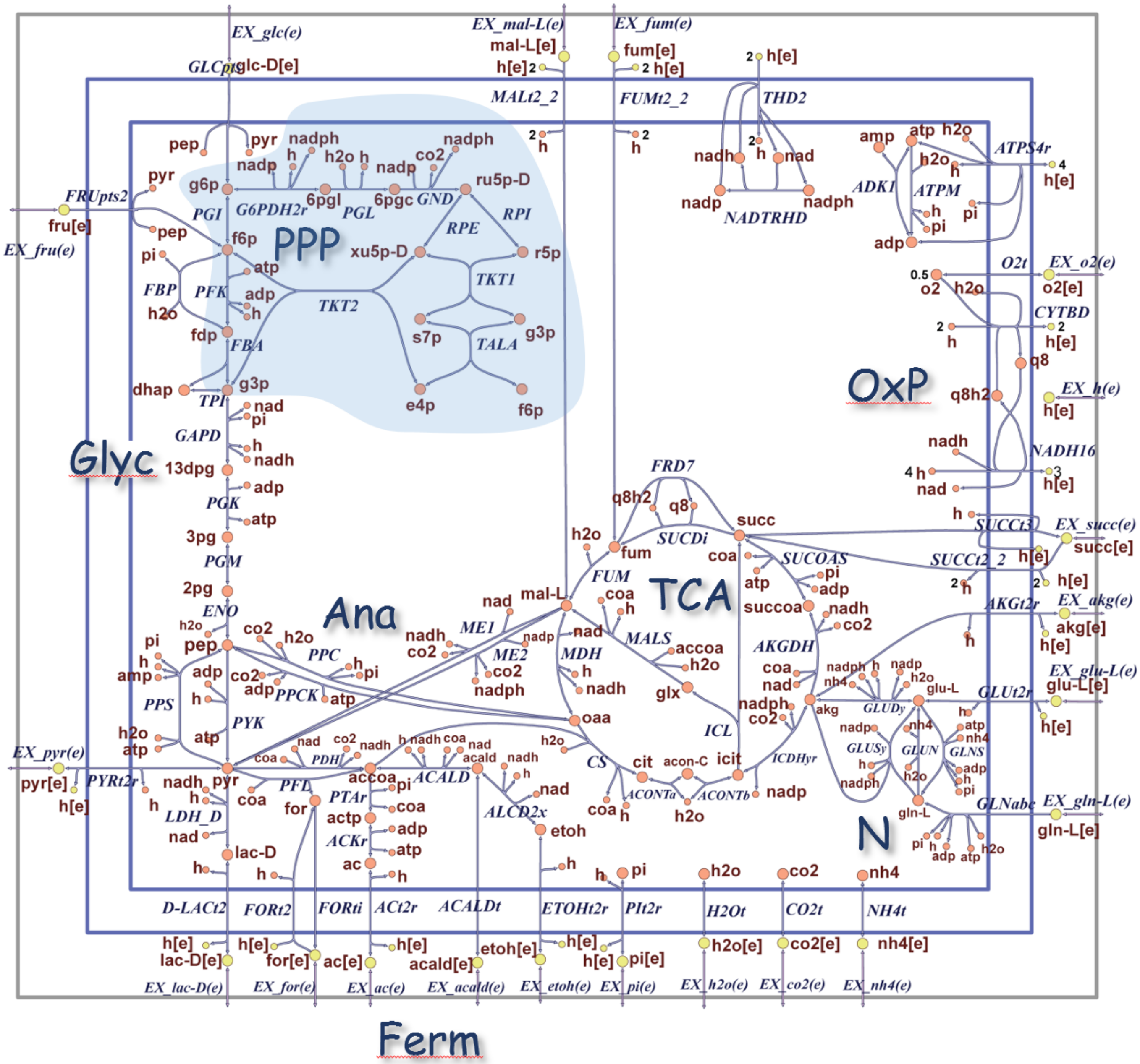


Figure 16. Pentose phosphate pathway subsystem reactions highlighted in blue on the *E.coli* core map [3].

The pentose phosphate pathway subsystem includes the following reactions derived from the core model. *[Timing: Seconds]*

```
model = e_coli_core; % Starting with the original model
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-30,'l');
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');
pppSubsystem = {'Pentose Phosphate Pathway'};
pppReactions = model.rxns(ismember(model.subSystems,pppSubsystem));
[tmp,ppp_rxnID] = ismember(pppReactions,model.rxns);
Reaction_Names = model.rxnNames(ppp_rxnID);
Reaction_Formulas = printRxnFormula(model,pppReactions,0);
T = table(Reaction_Names,Reaction_Formulas,'RowNames',pppReactions)
```

```
T =
                          Reaction_Names                                        Reaction_Formulas
```

```
            -----------------------------------   --------------------------------------------------

G6PDH2r     'glucose 6-phosphate dehydrogenase'   'g6p[c] + nadp[c]  <=> 6pgl[c] + h[c] + nadph[c] '
GND         'phosphogluconate dehydrogenase'      '6pgc[c] + nadp[c]  -> co2[c] + nadph[c] + ru5p-D[
PGL         '6-phosphogluconolactonase'           '6pgl[c] + h2o[c]  -> 6pgc[c] + h[c] '
RPE         'ribulose 5-phosphate 3-epimerase'    'ru5p-D[c]  <=> xu5p-D[c] '
RPI         'ribose-5-phosphate isomerase'        'r5p[c]  <=> ru5p-D[c] '
TALA        'transaldolase'                       'g3p[c] + s7p[c]  <=> e4p[c] + f6p[c] '
TKT1        'transketolase'                       'r5p[c] + xu5p-D[c]  <=> g3p[c] + s7p[c] '
TKT2        'transketolase'                       'e4p[c] + xu5p-D[c]  <=> f6p[c] + g3p[c] '
```

There are two distinct phases of the pentose phosphate pathway. The first is the "oxidative phase," in which nadph[c] is generated. Note that the pentose phosphate pathway is not the only source of nadph[c] in aerobic conditions. This was explored using "surftNet" in the energy management section (Section 4.A). The second phase of the pentose phosphate pathway is referred to as the "non-oxidative" phase that provides a pathway for the synthesis of 4-, 5-, and 7-carbon precursors in anaerobic conditions. The pentose phosphate pathway reactions and supported precursors are shown in the Figure17 below.

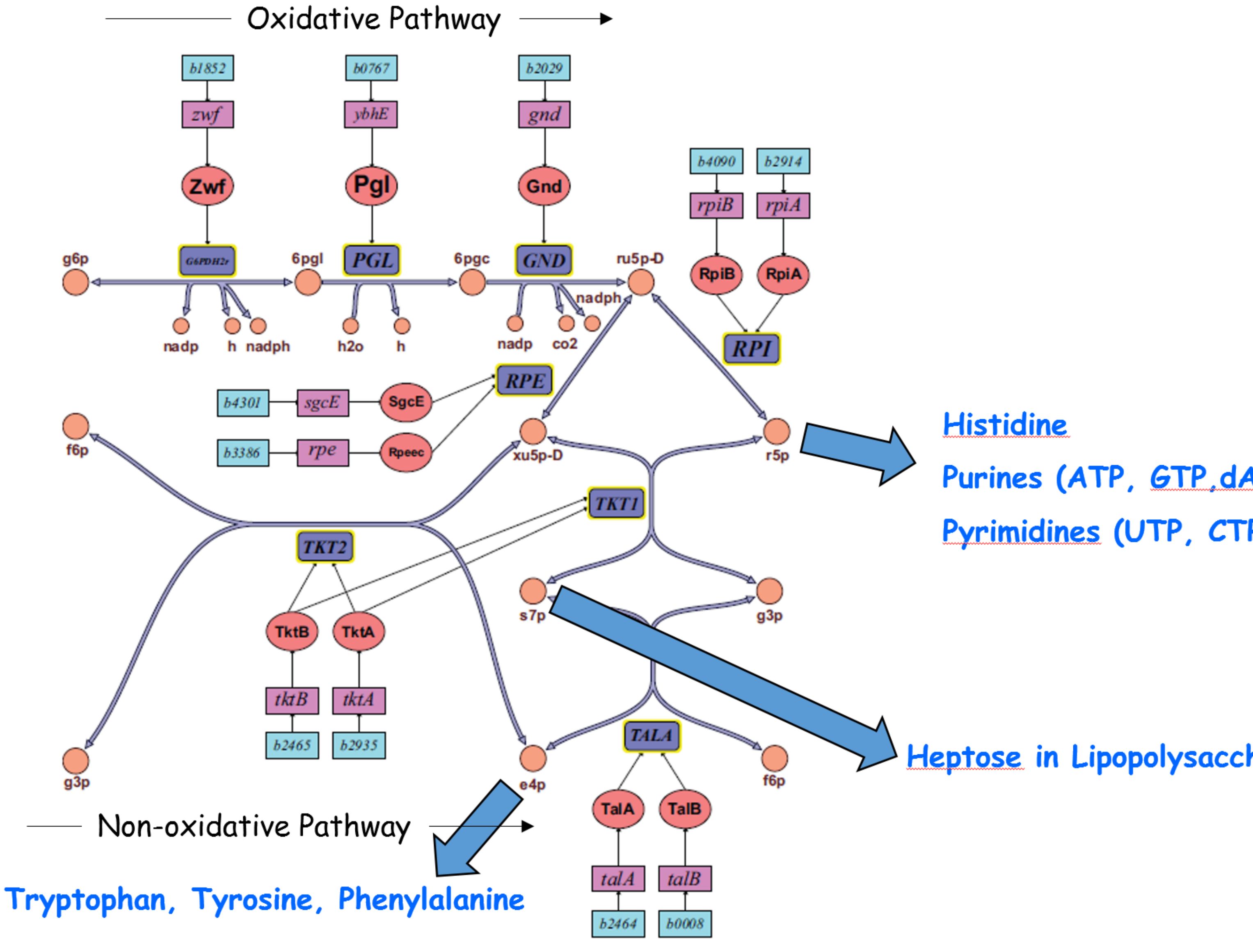


Figure 17. Pentose phosphate pathway reactions and precursors [3].

The direction of the flux flowing through the non-oxidative part of the pentose phosphate pathway changes based on aerobic versus anaerobic conditions. This variation in flux direction is shown below in Figure 18.

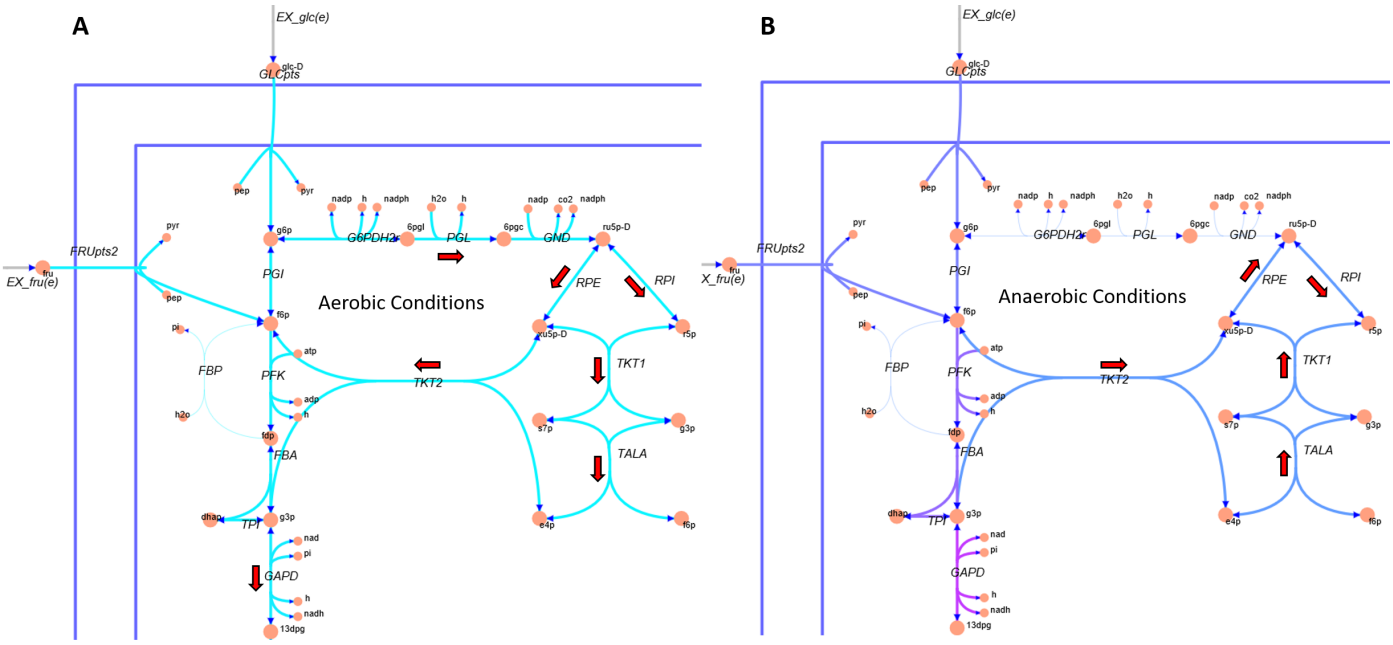


Figure 18. The flow of flux through the pentose phosphate pathway based on A) aerobic or B) anaerobic conditions.

In this figure it can be seen that under (A) aerobic conditions the flux flows through the oxidative phase of the pentose phosphate pathway and then is directed downward through the non-oxidative phase and then works its way back to the glycolysis cycle. On the other hand, under (B) anaerobic conditions the flux enters the left side of reaction TKT2 of the pentose phosphate pathway from the glycolysis pathway operating under the condition of gluconeogenesis. The flux then splits to feed the needs of the three major precursors e4p[c], r5p[c], and s7p[c]. These specific flux values can be calculated using the COBRA Toolbox as follows. *[Timing: Seconds]*

```
% Obtain the rxnIDs for the pentose phosphate pathway reactions
[tmp,glycolysis_rxnID] = ismember(glycolysisReactions,model.rxns);

% Glucose aerobic flux
FBAsolution = optimizeCbModel(model,'max',0,0);
Glucose_Aerobic_Flux = round(FBAsolution.x(ppp_rxnID),3);

% Fructose aerobic flux
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_fru(e)',-10,'l');
FBAsolution = optimizeCbModel(model,'max',0,0);
Fructcose_Aerobic_Flux = round(FBAsolution.x(ppp_rxnID),3);

% Set anaerobic conditions
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');

% Glucose anaerobic flux
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
FBAsolution = optimizeCbModel(model,'max',0,0);
Glucose_Anaerobic_Flux = round(FBAsolution.x(ppp_rxnID),3);

% Fructose anaerobic flux
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_fru(e)',-10,'l');
```

```
FBAsolution = optimizeCbModel(model,'max',0,0);
Fructose_Anaerobic_Flux = round(FBAsolution.x(ppp_rxnID),3);

T = table(Glucose_Aerobic_Flux,Fructcose_Aerobic_Flux,Glucose_Anaerobic_Flux,...
    Fructose_Anaerobic_Flux,'RowNames',pppReactions)
```

```
T =
              Glucose_Aerobic_Flux    Fructcose_Aerobic_Flux    Glucose_Anaerobic_Flux    Fructose_Anae
              ____________________    ______________________    ______________________    _____________

    G6PDH2r      4.96                    4.96                         0                         0
    GND          4.96                    4.96                         0                         0
    PGL          4.96                    4.96                         0                         0
    RPE         2.678                   2.678                    -0.371                    -0.152
    RPI        -2.282                  -2.282                    -0.371                    -0.152
    TALA        1.497                   1.497                    -0.092                    -0.038
    TKT1        1.497                   1.497                    -0.092                    -0.038
    TKT2        1.181                   1.181                    -0.279                    -0.114
```

4.D. Tricarboxylic Acid Cycle

The tricarboxylic acid (TCA) cycle or the citric acid cycle supports a variety of cellular functions depending on the environment. Under aerobic conditions the TCA cycle operates in a counter-clockwise direction using acetyl-CoA as a substrate to produce three cellular precursors, reducing power nadh[c] and nadph[c], cellular energy atp[c] through substrate phosphorylation, and carbon dioxide (co2[c]). While in the anaerobic condition, only part of the TCA cycle will be used to produce two of the three precursors and the reducing power nadph[c]. The location of the TCA cycle subsystem is shown on the following *E.coli* core map (Figure 19).

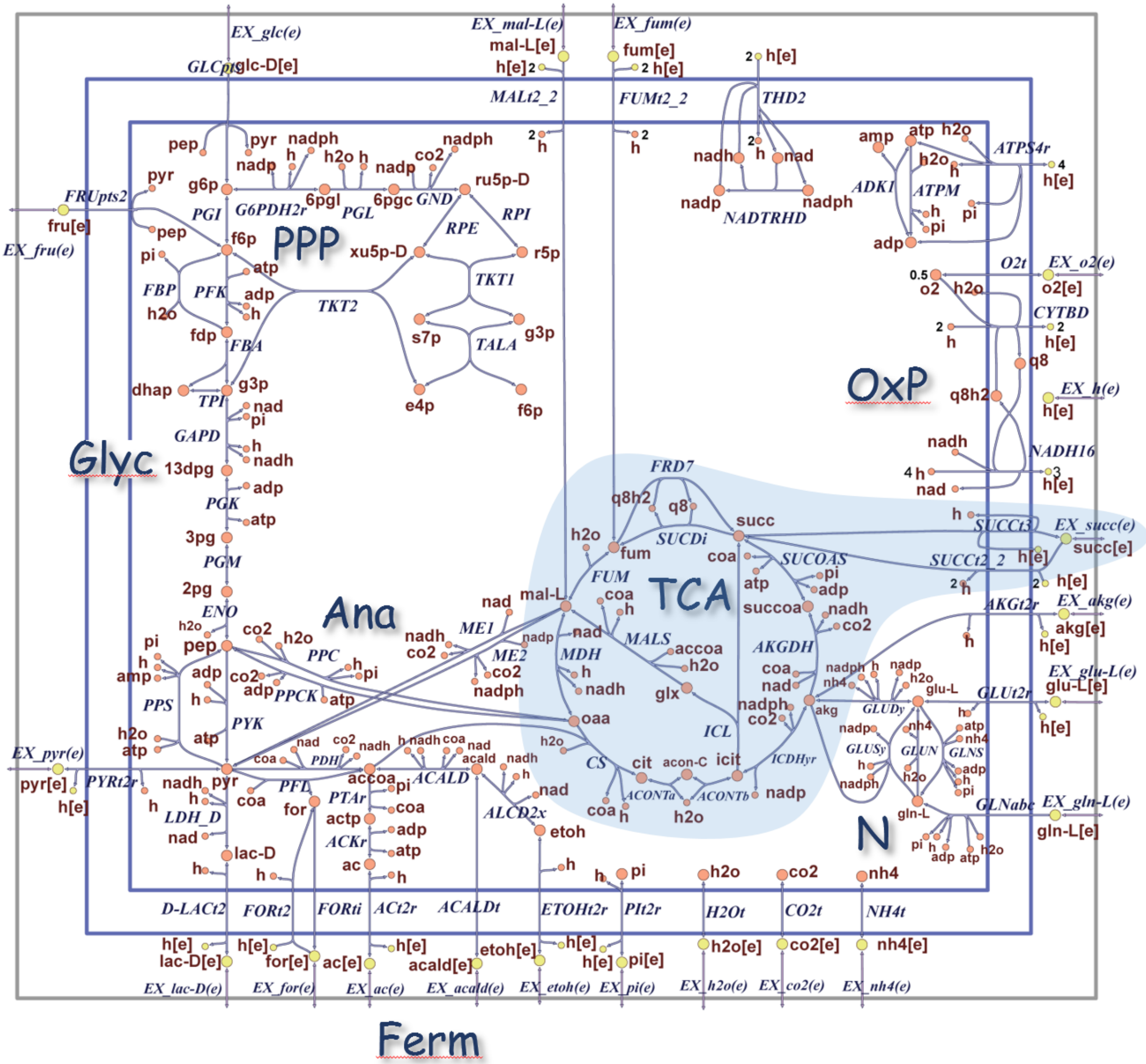


Figure 19. TCA pathway subsystem reactions highlighted in blue on *E.coli* core map [3].

The reactions associated with the TCA cycle can be retrieved from the *E.coli* core model as shown below. *[Timing: Seconds]*

```
model = e_coli_core;
TCA_Reactions = transpose({'CS','ACONTa','ACONTb','ICDHyr','AKGDH','SUCOAS',...
    'FRD7','SUCDi','FUM','MDH'});
[tmp,TCA_rxnID] = ismember(TCA_Reactions,model.rxns);
Reaction_Names = model.rxnNames(TCA_rxnID);
Reaction_Formulas = printRxnFormula(model,TCA_Reactions,0);
T = table(Reaction_Names,Reaction_Formulas,'RowNames',TCA_Reactions)
```

```
T =
                                  Reaction_Names                                     Reaction_For
                   ____________________________________________________     ________________________________

    CS        'citrate synthase'                                            'accoa[c] + h2o[c] + oaa[c]  -> c
    ACONTa    'aconitase (half-reaction A, Citrate hydro-lyase)'            'cit[c]  <=> acon-C[c] + h2o[c]  '
```

```
ACONTb    'aconitase (half-reaction B, Isocitrate hydro-lyase)'    'acon-C[c] + h2o[c]  <=> icit[c]
ICDHyr    'isocitrate dehydrogenase (NADP)'                        'icit[c] + nadp[c]  <=> akg[c] +
AKGDH     '2-Oxoglutarate dehydrogenase'                           'akg[c] + coa[c] + nad[c]  -> co2
SUCOAS    'succinyl-CoA synthetase (ADP-forming)'                  'atp[c] + coa[c] + succ[c]  <=> a
FRD7      'fumarate reductase'                                     'fum[c] + q8h2[c]  -> q8[c] + suc
SUCDi     'succinate dehydrogenase (irreversible)'                 'q8[c] + succ[c]  -> fum[c] + q8h
FUM       'fumarase'                                               'fum[c] + h2o[c]  <=> mal-L[c] '
MDH       'malate dehydrogenase'                                   'mal-L[c] + nad[c]  <=> h[c] + na
```

The *E.coli* core model does not include the membrane reactions (FRD7 and SUCDi) in the TCA cycle (Citric Acid Cycle) subsystem. They have been added to this discussion since they close the TCA loop and allow complete TCA operation.

The precursors associated with the TCA cyle are shown below in Figure 20. The precursors include; 1) oxaloacetate (oaa[c]) for the biosynthesis of asparagine, aspartic acid, isoleucine, lysine, methionine, and threonine, 2) 2-oxoglutarate or alpha-ketoglutarate (akg[c]) for the biosynthesis of arginine, glutamine, glutamic acid, and proline and finally 3) succinyl-CoA (succoa[c]) for heme biosynthesis.

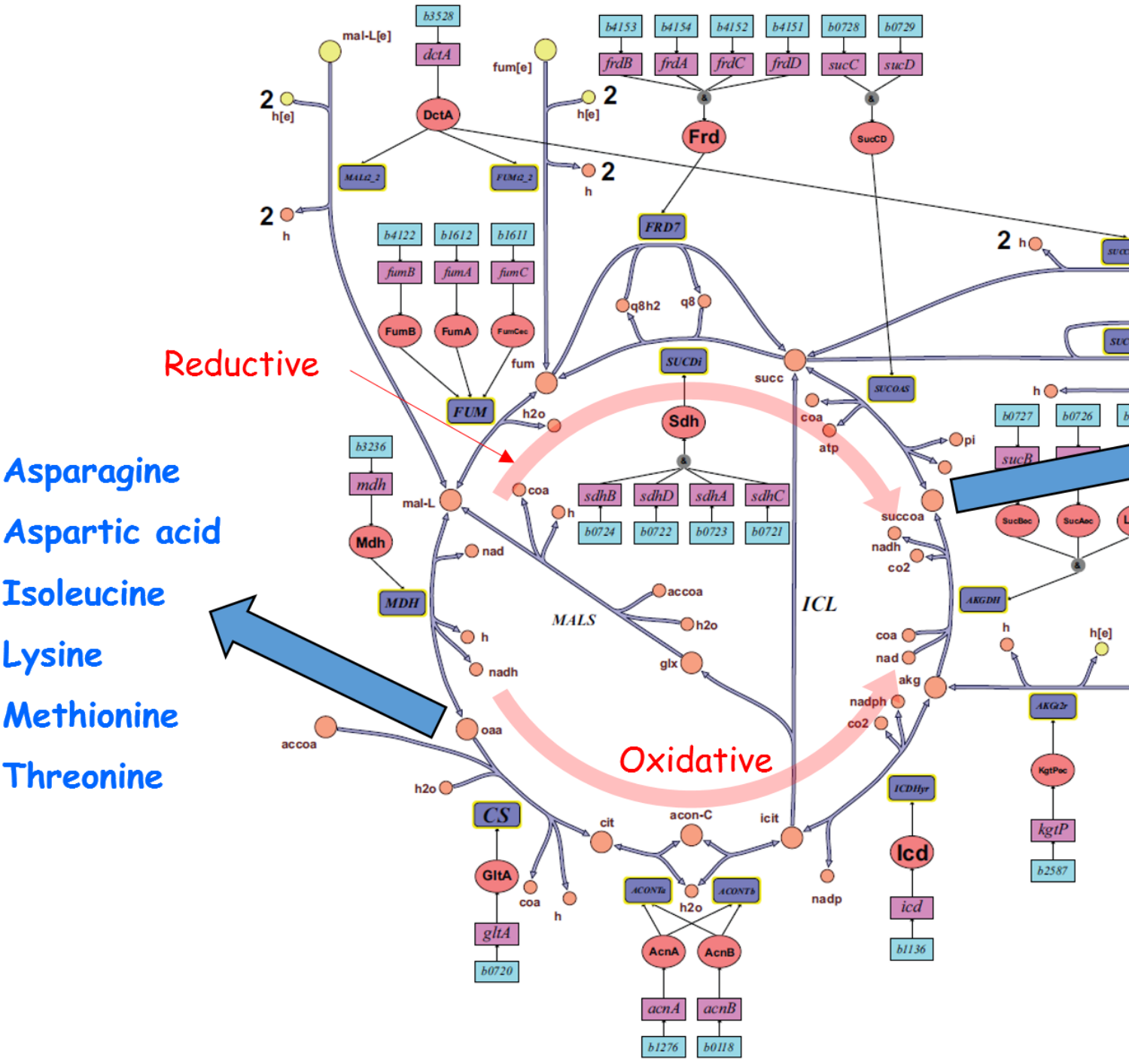


Figure 20. TCA pathway reactions and precursors [3].

The TCA cycle can be divided into an oxidative pathway and a reductive pathway as illustrated in Figure 19. The oxidative pathway of the TCA cycle runs counterclockwise in the lower part of the cycle, from oxaloacetate (oaa[c]), through 2-oxoglutarate (akg[c]). Under aerobic conditons the oxidative pathway can continue counterclockwise from 2-oxoglutarate (akg[c]) full circle to oxaloacetate (oaa[c]). The full TCA cycle can totally oxidize acetyl-CoA (accoa[c]), but only during aerobic growth on acetate or fatty acids.

Under anaerobic conditions, the TCA cycle functions not as a cycle, but as two separate pathways. The oxidative pathway, the counterclockwise lower part of the cycle, still forms the precursor 2-oxoglutarate. The reductive pathway, the clockwise upper part of the cycle, can form the precursor succinyl-CoA.

Let's begin this exploration by visualizing the fluxes through the core model when pyruvate is used as the carbon source for both aerobic and anaerobic conditions. *[Timing: Seconds]*

```
% Key parameters for TCA pathway section
model = e_coli_core;
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_pyr(e)',-20,'l');
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set at -30 for aerobic
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');
FBAsolution = optimizeCbModel(model,'max',0,0);

% Import E.coli core map and adjust parameters
map=readCbMap('ecoli_core_map.txt');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);
```

```
Document Written
```

A close-up on the TCA cycle for both the aerobic and anaerobic cases are shown in Figure 21.

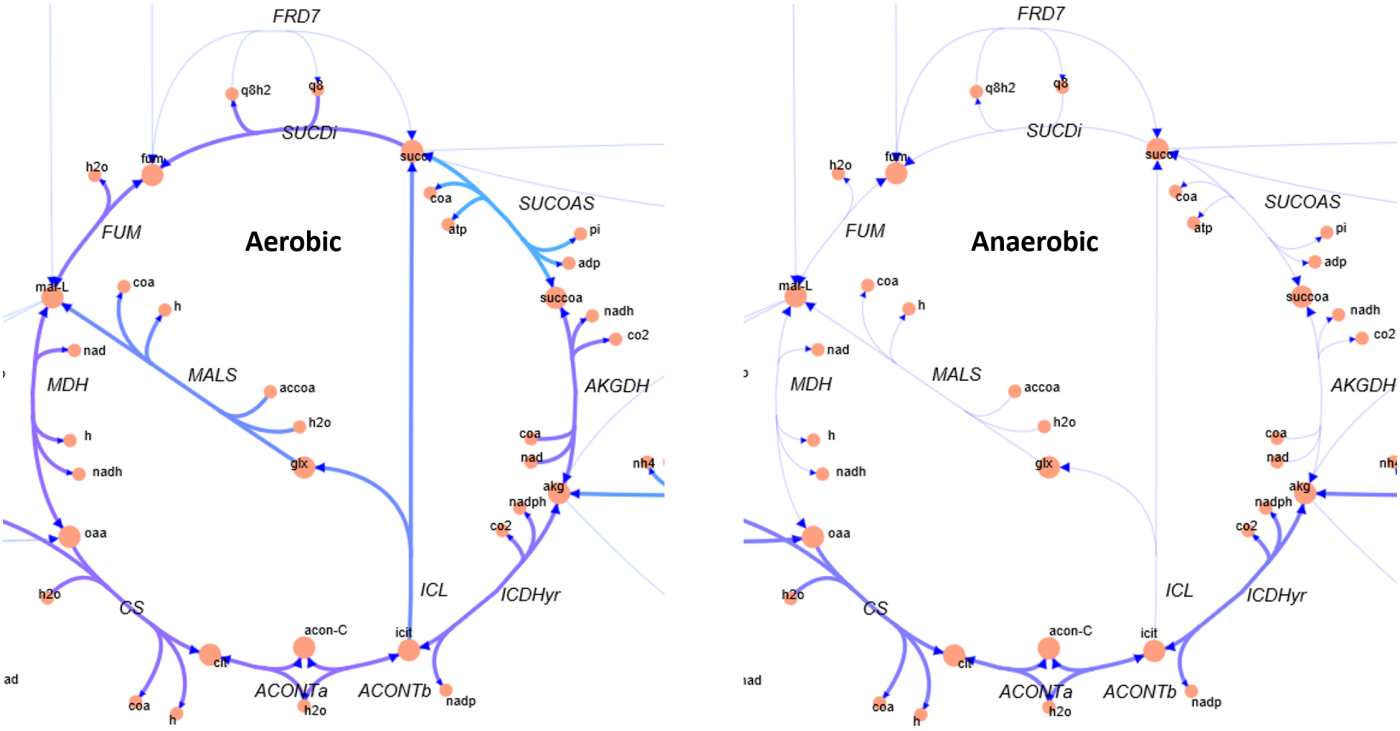


Figure 21. A close-up of the TCA cycle with pyruvate as the carbon source for both aerobic and anaerobic conditions.

The specific flux values for each of these conditions is calculated below. *[Timing: Seconds]*

```
model = e_coli_core;
% Pyruvate aerobic flux
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_pyr(e)',-20,'l');
model = changeRxnBounds(model,'EX_o2(e)',-30,'l');
FBAsolution = optimizeCbModel(model,'max',0,0);
Pyrvate_Aerobic_Flux = round(FBAsolution.x(TCA_rxnID),3);

% Pyruvate anaerobic flux
```

```
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
FBAsolution = optimizeCbModel(model,'max',0,0);
Pyrvate_Anaerobic_Flux = round(FBAsolution.x(TCA_rxnID),3);

T = table(Pyrvate_Aerobic_Flux,Pyrvate_Anaerobic_Flux,...
    'RowNames',TCA_Reactions)
```

```
T =
              Pyrvate_Aerobic_Flux    Pyrvate_Anaerobic_Flux
              ____________________    ______________________

    CS        11.216                  0.071
    ACONTa    11.216                  0.071
    ACONTb    11.216                  0.071
    ICDHyr     9.433                  0.071
    AKGDH      8.762                      0
    SUCOAS    -8.762                      0
    FRD7           0                      0
    SUCDi     10.545                      0
    FUM       10.545                      0
    MDH       12.328                      0
```

These fluxes show that under aerobic conditions the full TCA cyle is operational while under anaerobic conditions only the lower part of the TCA cycle (CS, ACONTa, ACONTb and ICDHyr), the oxidative pathway, is used.

4.E. Glycoxylate Cycle, Gluconeogenesis, and Anapleurotic Reactions

The glycoxylate cycle and gluconeogenic reactions are necessary to allow *E. coli* to grow on 3-carbon (pyruvate) and 4-carbon compounds (malate, fumarate, and succinate). This occurs by avoiding the loss of carbon to carbon dioxide in the TCA cycle (glycoxylate cycle), providing a pathway for generation of glycolytic intermediates from TCA intermediates (anapleurotic reactions), and reversing the carbon flux through glycolysis (gluconeogenesis) to produce essential precursors for biosynthesis.

The location of the glycoxylate cycle, gluconeogenesis, and anapleurotic reactions on the *E.coli* core map is shown in Figure 22 below.

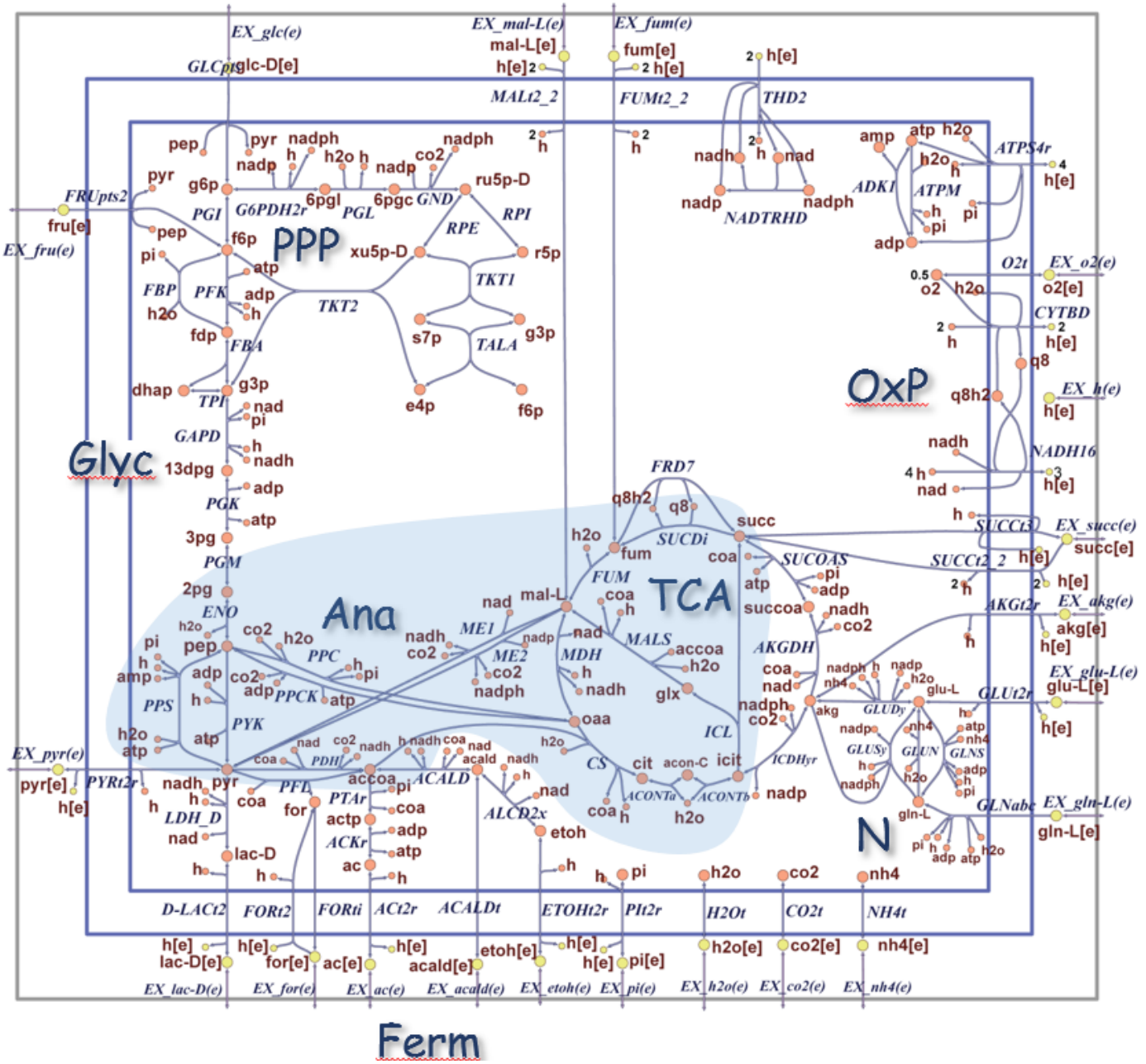


Figure 22. Glycoxylate cycle, gluconeogenesis, and anapleurotic reactions highlighted in blue on the *E.coli* core map [3].

The reactions included in this section on the glycoxylate cycle, gluconeogenesis, and anapleurotic reactions are shown below. This subsystem is referred to in the core model as the "anapleurotic reactions" subsystem. *[Timing: Seconds]*

```
% Set initial constraints for glycoxylate cycle, gluconeogenesis, and anapleurotic reactions s
model = e_coli_core;
ANA_Reactions = transpose({'ICL','MALS','ME1','ME2','PPS','PPCK',...
    'PPC'});
[tmp,ANA_rxnID] = ismember(ANA_Reactions,model.rxns);
Reaction_Names = model.rxnNames(ANA_rxnID);
Reaction_Formulas = printRxnFormula(model,ANA_Reactions,0);
T = table(Reaction_Names,Reaction_Formulas,'RowNames',ANA_Reactions)
```

```
T =
                                Reaction_Names                                                  Reaction_Formulas
```

ICL	'Isocitrate lyase'	'icit[c] -> glx[c] + succ[c] '
MALS	'malate synthase'	'accoa[c] + glx[c] + h2o[c] -> coa[c] + h[c] + mal-L
ME1	'malic enzyme (NAD)'	'mal-L[c] + nad[c] -> co2[c] + nadh[c] + pyr[c] '
ME2	'malic enzyme (NADP)'	'mal-L[c] + nadp[c] -> co2[c] + nadph[c] + pyr[c] '
PPS	'phosphoenolpyruvate synthase'	'atp[c] + h2o[c] + pyr[c] -> amp[c] + 2 h[c] + pep[c
PPCK	'phosphoenolpyruvate carboxykinase'	'atp[c] + oaa[c] -> adp[c] + co2[c] + pep[c] '
PPC	'phosphoenolpyruvate carboxylase'	'co2[c] + h2o[c] + pep[c] -> h[c] + oaa[c] + pi[c] '

These individual reactions associated with the glycoxylate cycle, gluconeogenesis, and anapleurotic reactions are graphically shown in Figure 23.

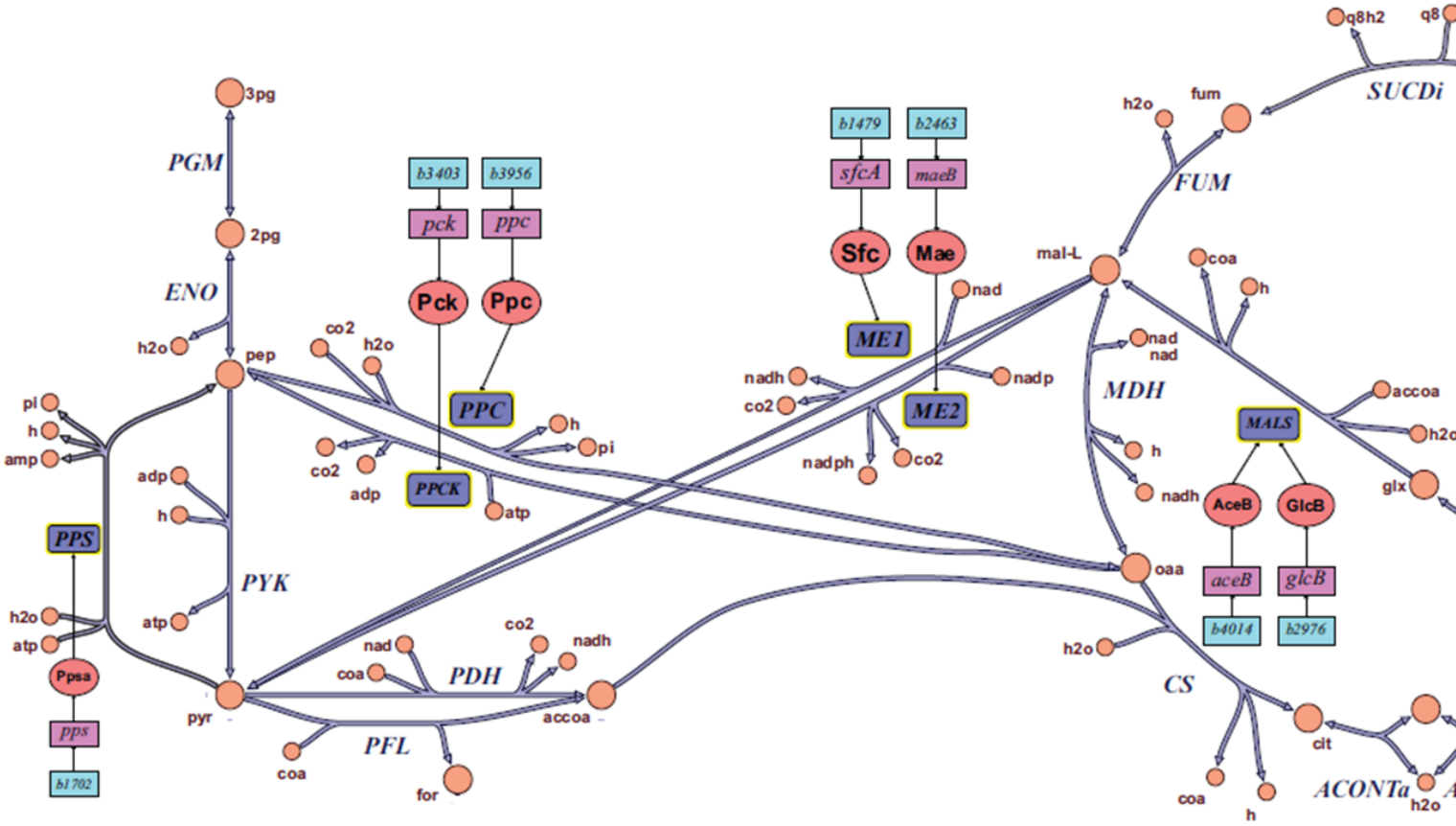


Figure 23. Reactions associated with the glycoxylate cycle, gluconeogenesis, and anapleurotic reactions [3].

The anapleurotic reactions (PPC, PPS, PPCK, PPC, ME1, and ME2) are interconnecting, reversing and bypassing reactions that replenish TCA cycle intermediates. The glycoxylate cycle (CS, ACONTa, ACONTb, ICL, MALS, MDH, SUCDi and FUM), which includes some TCA cycle reactions, is essential for growth on 3-carbon (pyruvate) and 4-carbon compounds since it can convert the precursor acetyl-CoA into glycolytic intermediates without loss of carbon to carbon dioxide (ICDHyr & AKGDH). Finally, growth on 4-carbon intermediates of the TCA cycle, such as malate, requires that the cell be able to produce phosphoenolpyruvate (pep[c]) for gluconeogenesis. Gluconeogenesis refers to the reversal of flux through the glycolytic pathway. There are two pathways able to fulfill these pep[c] demands. The first pathway involves the conversion of malate (mal[c]) to pyruvate (pyr[c]) by a malic enzyme (ME1 or ME2). This is followed by the synthesis of pep[c] from pyr[c] by phosphoenolpyruvate synthase (PPS). Malic enzyme (ME1) reduces one molecule of nad[c] to nadh[c] while converting mal[c] to pyr[c]. A second parallel reaction, ME2 reduces one molecule of nadp[c] to nadph[c].

Now it is time to explore the the impact on the cell of these pathways for different carbon sources. Let's begin by looking at the aerobic operation of the cell growing on acetate. *[Timing: Seconds]*

```
% Key parameters for TCA pathway section
model = e_coli_core;
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_ac(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set at -30 for aerobic
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');

% Perform FBA with Biomass_Ecoli_core_N(w/GAM)_Nmet2 as the objective,
FBAsolution = optimizeCbModel(model,'max',0,0);

% Import E.coli core map and adjust parameters
map=readCbMap('ecoli_Textbook_ExportMap');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
% Draw the flux values on the map "target.svg" which can be opened in FireFox
drawFlux(map, model, FBAsolution.x, options);
```

```
Document Written
```

A copy of the figure stored in "target.svg" is shown in Figure 24.

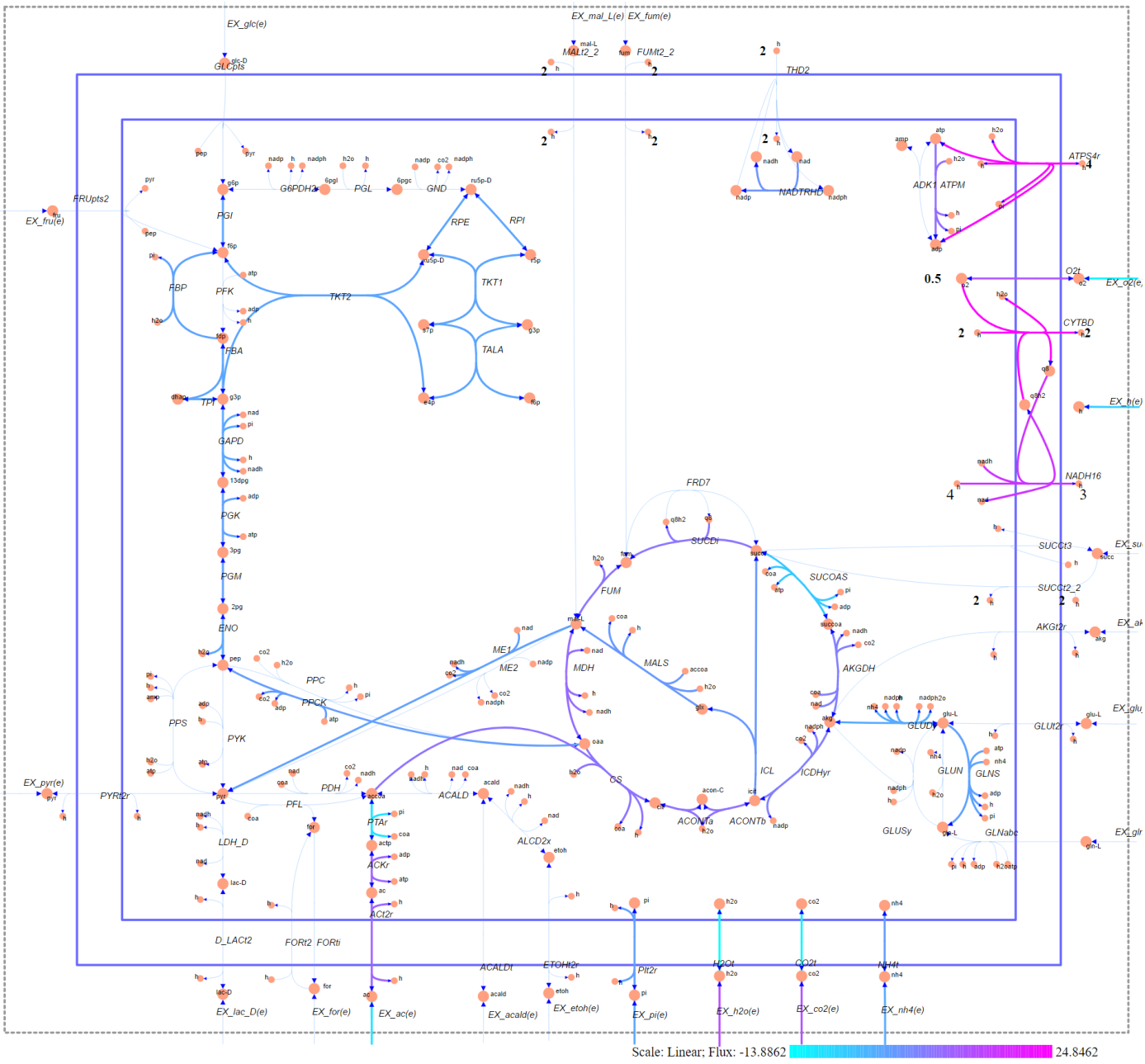


Figure 24. Screenshot of the core model with acetate as the carbon source under aerobic conditions.

The active fluxes for this simulaton are given below. *[Timing: Seconds]*

```
printFluxVector(model,FBAsolution.x,true) % only prints nonzero fluxes
```

```
ACKr 10
ACONTa 7.55253
ACONTb 7.55253
ACt2r 10
AKGDH 5.56768
ATPM 8.39
ATPS4r 24.5049
Biomass_Ecoli_core_w_GAM 0.173339
CO2t -12.6235
CS 7.55253
CYTBD 24.8462
ENO -0.7201
EX_ac(e) -10
EX_co2(e) 12.6235
```

```
EX_h(e) -6.52283
EX_h2o(e) 13.8862
EX_nh4(e) -0.945181
EX_o2(e) -12.4231
EX_pi(e) -0.637661
FBA -0.17242
FBP 0.17242
FUM 7.36551
GAPD -0.460786
GLNS 0.0443227
GLUDy -0.900858
H2Ot -13.8862
ICDHyr 5.7547
ICL 1.79783
MALS 1.79783
MDH 8.67231
ME2 0.491034
NADH16 17.4807
NADTRHD 3.08663
NH4t 0.945181
O2t 12.4231
PGI -0.0355344
PGK 0.460786
PGM 0.7201
PIt2r 0.637661
PPCK 0.810081
PTAr -10
RPE -0.124596
RPI -0.124596
SUCDi 7.36551
SUCOAS -5.56768
TALA -0.0310103
TKT1 -0.0310103
TKT2 -0.0935855
TPI -0.17242
```

It can be seen, using the map and the fluxes listed above, that acetate enters the network at the bottom and flows into the TCA cycle. From there it can be observed that not only is the full TCA cycle operational but so is the glycoxolate cycle. Part of the oaa[c] metabolite flux is then directed through the glycolysis pathway (gluconeogenesis) to the pentose phosphate pathway to create the 4-, 5- and 7-carbon precursors precursors.

Using malate as a carbon source under aerobic conditions is another good example of the role of the glycoxylate cycle, gluconeogenesis, and anapleurotic reactions. The Matlab/COBRA Toolbox code for this example is shown below. *[Timing: Seconds]*

```
model = e_coli_core;
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_mal_L(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set at -30 for aerobic
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');

% Perform FBA with Biomass_Ecoli_core_N(w/GAM)_Nmet2 as the objective,
FBAsolution = optimizeCbModel(model,'max',0,0);

% Import E.coli core map and adjust parameters
map=readCbMap('ecoli_Textbook_ExportMap');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);
```

```
Document Written
```

A screenshot of the figure stored in "target.svg" is shown in Figure 25.

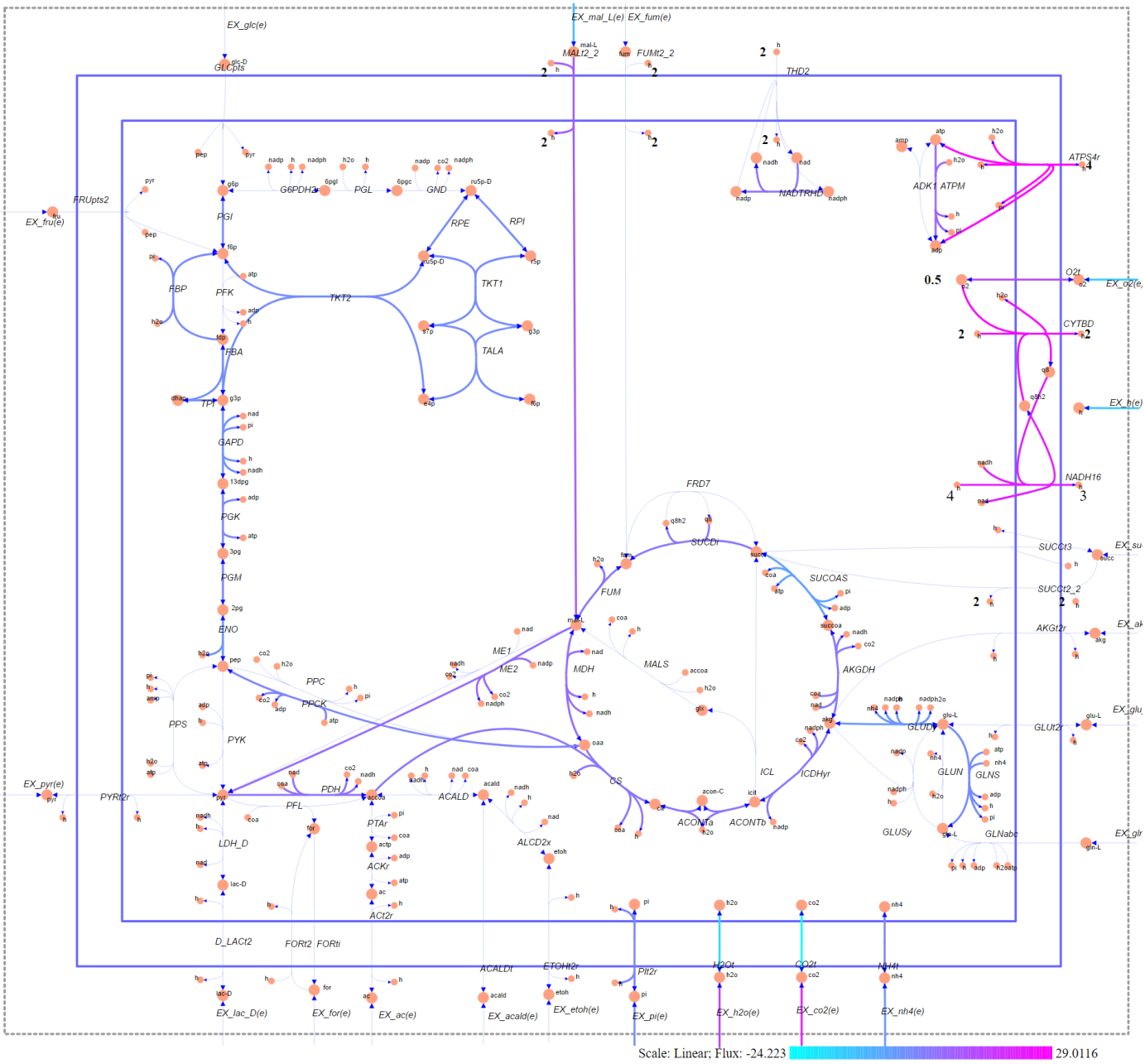


Figure 25. COBRA Toolbox produced map showing aerobic operation with malate as the carbon source.

The active fluxes for this simulaton are given below. *[Timing: Seconds]*

```
printFluxVector(model,FBAsolution.x,true) % only prints nonzero fluxes
```

```
ACONTa 4.76529
ACONTb 4.76529
AKGDH 4.3653
ATPM 8.39
ATPS4r 29.0116
Biomass_Ecoli_core_w_GAM 0.370741
CO2t -24.223
```

```
CS 4.76529
CYTBD 27.5887
ENO -1.54017
EX_co2(e) 24.223
EX_h(e) -12.5629
EX_h2o(e) 16.9236
EX_mal_L(e) -10
EX_nh4(e) -2.02157
EX_o2(e) -13.7943
EX_pi(e) -1.36384
FBA -0.368776
FBP 0.368776
FUM 4.3653
GAPD -0.98554
GLNS 0.0947984
GLUDy -1.92678
H2Ot -16.9236
ICDHyr 4.76529
MALt2_2 10
MDH 7.16031
ME1 5.21353
ME2 1.99145
NADH16 23.2234
NH4t 2.02157
O2t 13.7943
PDH 6.15475
PGI -0.0760018
PGK 0.98554
PGM 1.54017
PIt2r 1.36384
PPCK 1.73262
RPE -0.266488
RPI -0.266488
SUCDi 4.3653
SUCOAS -4.3653
TALA -0.0663255
TKT1 -0.0663255
TKT2 -0.200163
TPI -0.368776
```

In this situation, the malate enters the network from the top and flows to the TCA cycle. Part of the malate metabolite flux is converted to be used as the pyruvate precursor while the rest enters the fully operational TCA cycle. Note that the glycoxolate cycle is inactive. Part of the oaa[c] metabolite flux is then directed through the glycolysis pathway (gluconeogenesis), to the pentose phosphate pathway, to create the 4-, 5- and 7-carbon precursors.

4.F. Fermentation

Fermentation is the process of extracting energy from the oxidation of organic compounds without oxygen. The location of the fermentation reactions on the *E.coli* core map are shown in the Figure 26.

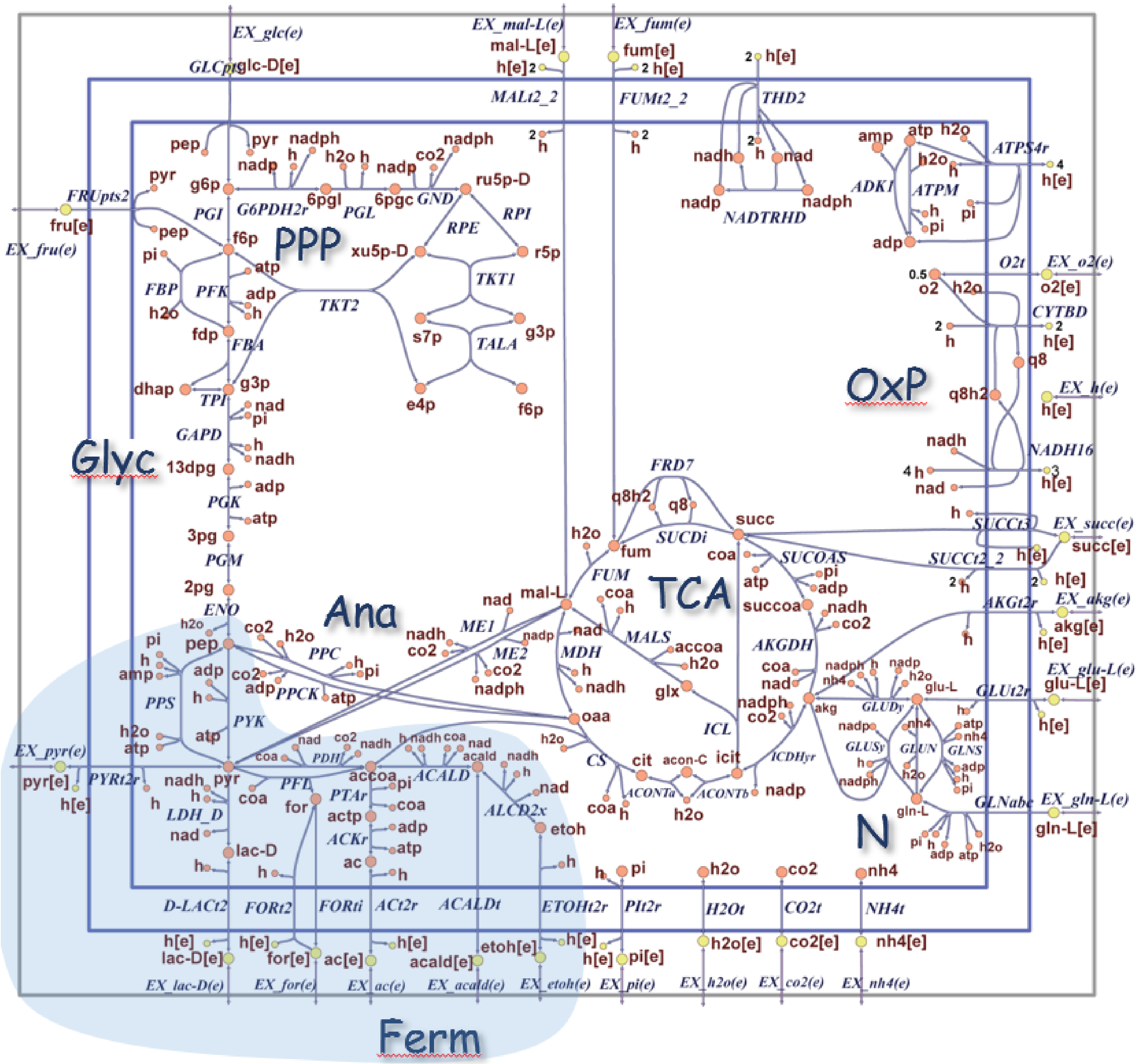


Figure 26. Fermentation reactions highlighted in blue on the *E.coli* core map [3].

The reactions associated with the fermentation pathways include: *[Timing: Seconds]*

```
% Set initial constraints for fermentation metabolism section
model = e_coli_core;
FERM_Reactions = transpose({'LDH_D','D_LACt2','PDH','PFL','FORti','FORt2',...
    'PTAr','ACKr','ACALD','ALCD2x','ACt2r','ACALDt','ETOHt2r'});
[tmp,FERM_rxnID] = ismember(FERM_Reactions,model.rxns);
Reaction_Names = model.rxnNames(FERM_rxnID);
Reaction_Formulas = printRxnFormula(model,FERM_Reactions,0);
T = table(Reaction_Names,Reaction_Formulas,'RowNames',FERM_Reactions)
```

```
T =
                                 Reaction_Names                                        Reaction_For
                  ____________________________________________________          ________________________

    LDH_D         'D lactate dehydrogenase'                                     'lac-D[c] + nad[c]  <=> h[c] + na
```

```
D_LACt2    'D-lactate transport via proton symport'                'h[e] + lac-D[e]  <=> h[c] + lac-
PDH        'pyruvate dehydrogenase'                                'coa[c] + nad[c] + pyr[c]  -> acc
PFL        'pyruvate formate lyase'                                'coa[c] + pyr[c]  -> accoa[c] + f
FORti      'formate transport via diffusion'                       'for[c]  -> for[e] '
FORt2      'formate transport via proton symport (uptake only)'    'for[e] + h[e]  -> for[c] + h[c]
PTAr       'phosphotransacetylase'                                 'accoa[c] + pi[c]  <=> actp[c] +
ACKr       'acetate kinase'                                        'ac[c] + atp[c]  <=> actp[c] + ac
ACALD      'acetaldehyde dehydrogenase (acetylating)'              'acald[c] + coa[c] + nad[c]  <=>
ALCD2x     'alcohol dehydrogenase (ethanol)'                       'etoh[c] + nad[c]  <=> acald[c] +
ACt2r      'acetate reversible transport via proton symport'        'ac[e] + h[e]  <=> ac[c] + h[c] '
ACALDt     'acetaldehyde reversible transport'                     'acald[e]  <=> acald[c] '
ETOHt2r    'ethanol reversible transport via proton symport'        'etoh[e] + h[e]  <=> etoh[c] + h[
```

The reactions, GRPA relationships, and precursors for this section on fermentation are shown in Figure 27 below.

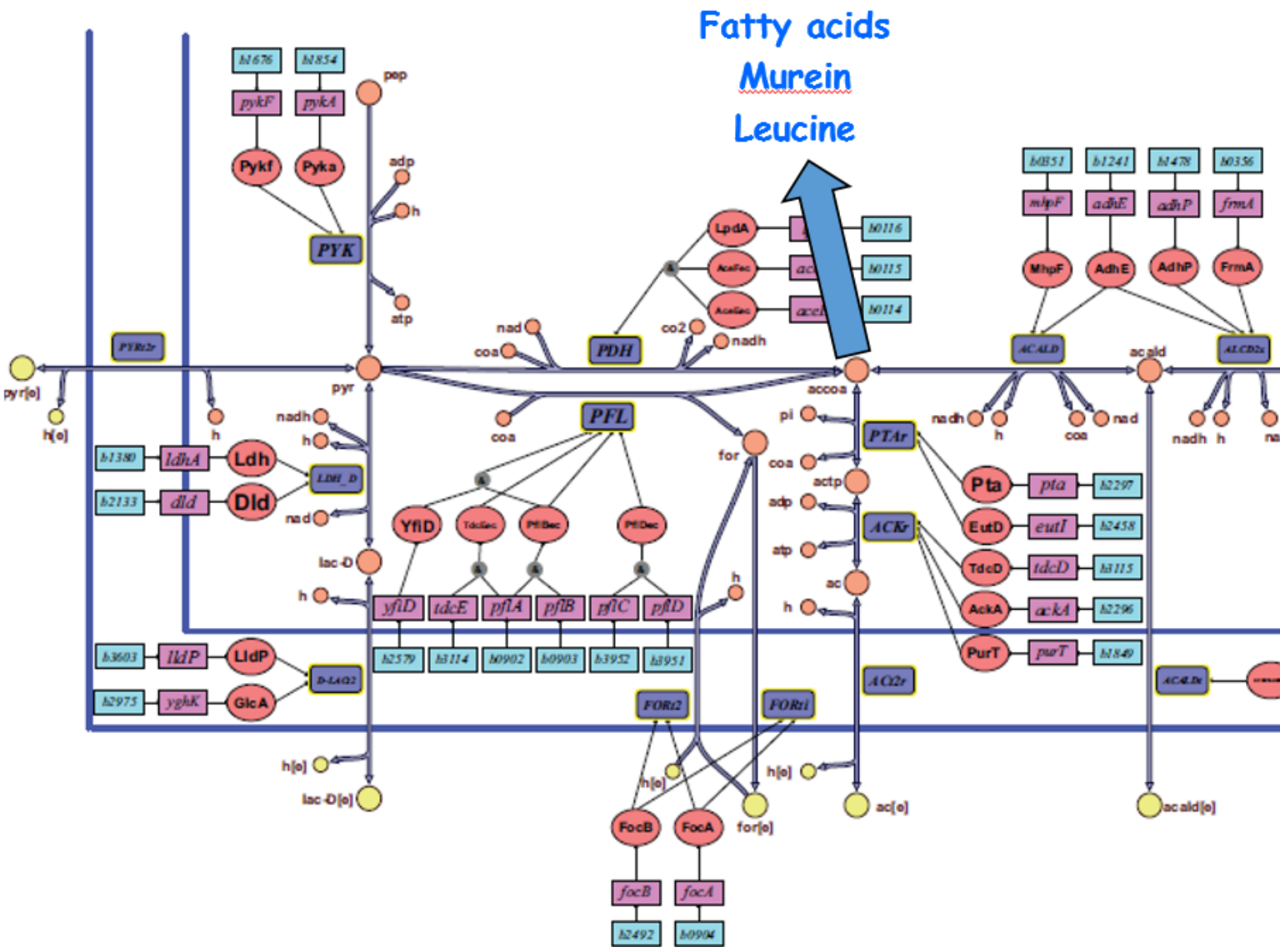


Figure 27. Reactions, GRPA relationships, and precursors for the fermentation metabolism [3].

During aerobic respiration, oxygen is used as the terminal electron acceptor for the oxidative phosphorylation process yielding the bulk of atp[c] required for cells biosynthesis. Anaerobic respiration, on the other hand, refers to respiration without molecular oxygen. In this case, *E. coli* can only generate atp[c] through substrate level phosphorylation which significantly reduces the amount of atp[c] that can be produced per molecule of glucose. In anaerobic conditions, glycolysis results in the net production

of 2 atp[c] per glucose by substrate level phosphorylation. This is compared to the total of 17.5 atp[c] per glucose molecule that can be produced for aerobic respiration [1]. To maintain the necessary energy needed for cellular operation during anaerobic growth, this forces each cell to maintain a large magnitude of flux through the glycolysis pathway to generate the necessary atp[c] to meet the cells growth requirements. This results in a large magnitude efflux of fermentative end products (lactate(lac-D[c]), formate (for[c]), acetate (ac[c]), acetaldehyde (acald[c]), and ethanol (etoh[c])) since there is insufficient atp[c] to assimilate all the carbon into biomass. It should be pointed out that only ~10% of carbon substrate is effectively assimilated into the cell due to the poor energy yield of fermentation.

There are two main fermentive processes included in the core model; homolactic fermentation and mixed acid fermentation. Homolactic fermentation refers to the conversion of pyruvate to lactate as shown on the bottom left of Figure 26 and includes the reactions LDH_D and D_LACt2 . Mixed acid fermentation is the process that converts pyrvate into a mixture of end products including lactate, acetate, succinate, formate, ethanol and includes the following reactions; PDH, PFL, FORti, FORt2, PTAr, ACKr, ACALD, ALCD2x, ACt2r, ACALDt, and ETOHt2r. It should also be pointed out that the end products of each fermentation pathway, with the exception of acetaldehyde, exit the cell along a concentration gradient and transport a proton from the cytoplasm into the extracellular space.

Let's begin our exploration of the fermentation metabolism by determining the secreted bioproducts produced in anaerobic conditions with a glucose carbon source. *[Timing: Seconds]*

```
model = e_coli_core;
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l'); % Anaerobic
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');
FBAsolution = optimizeCbModel(model,'max',0,0);
printFluxVector(model,FBAsolution.x,true, true) % only prints nonzero fluxes
```

```
Biomass_Ecoli_core_w_GAM 0.211663
EX_ac(e) 8.50359
EX_co2(e) -0.378178
EX_etoh(e) 8.27946
EX_for(e) 17.8047
EX_glc(e) -10
EX_h(e) 30.5542
EX_h2o(e) -7.1158
EX_nh4(e) -1.15416
EX_pi(e) -0.778644
```

With these results we can see that acetate, ethanol, and formate are the mixed fermentation products. Figure 12 shows the cell in this anaerobic condition. Note the flux flow in the paths of the secreted mixed acid fermentation products. Now let's explore the producers and consumers of atp[c] in anaerobic conditions with a glucose carbon source using "surfNet". *[Timing: Seconds]*

```
surfNet(model,'atp[c]',0,FBAsolution.x,1,1)
```

```
Met #17  atp[c], ATP, C10H12N5O13P3
Consuming reactions with non-zero fluxes :
  #11  ATPM (8.39), Bd: 8.39 / 1000, ATP maintenance requirement
  atp[c] + h2o[c] -> adp[c] + h[c] + pi[c]
  #12  ATPS4r (-5.45205), Bd: -1000 / 1000, ATP synthase (four protons for one ATP)
  adp[c] + 4 h[e] + pi[c] <=> atp[c] + h2o[c] + 3 h[c]
  #13  Biomass_Ecoli_core_w_GAM (0.21166), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #51  GLNS (0.05412), Bd: 0 / 1000, glutamine synthetase
  atp[c] + glu-L[c] + nh4[c] -> adp[c] + gln-L[c] + h[c] + pi[c]
  #72  PFK (9.78946), Bd: 0 / 1000, phosphofructokinase
  atp[c] + f6p[c] -> adp[c] + fdp[c] + h[c]
```

```
Producing reactions with non-zero fluxes :
  #3  ACKr (-8.50359), Bd: -1000 / 1000, acetate kinase
  ac[c] + atp[c] <=> actp[c] + adp[c]
  #75  PGK (-19.4373), Bd: -1000 / 1000, phosphoglycerate kinase
  3pg[c] + atp[c] <=> 13dpg[c] + adp[c]
  #83  PYK (8.40427), Bd: 0 / 1000, pyruvate kinase
  adp[c] + h[c] + pep[c] -> atp[c] + pyr[c]

Show previous steps...
```

Note that all the atp[c] is produced through substrate phosphorylation through PGK and PYK in the glycolysis pathway and ACKr in the fermentation pathway that produces acetate. Now let's check to see if the majority of the produced nadh[c] is reduced to nad[c] by the fermentation pathways. *[Timing: Seconds]*

```
surfNet(model,'nadh[c]',0,FBAsolution.x,1,1)
```

```
Met #51  nadh[c], Nicotinamide-adenine-dinucleotide-reduced, C21H27N7O14P2
Consuming reactions with non-zero fluxes :
  #1  ACALD (-8.27946), Bd: -1000 / 1000, acetaldehyde dehydrogenase (acetylating)
  acald[c] + coa[c] + nad[c] <=> accoa[c] + h[c] + nadh[c]
  #10  ALCD2x (-8.27946), Bd: -1000 / 1000, alcohol dehydrogenase (ethanol)
  etoh[c] + nad[c] <=> acald[c] + h[c] + nadh[c]
  #92  THD2 (3.62919), Bd: 0 / 1000, NAD(P) transhydrogenase
  2 h[e] + nadh[c] + nadp[c] -> 2 h[c] + nad[c] + nadph[c]
Producing reactions with non-zero fluxes :
  #13  Biomass_Ecoli_core_w_GAM (0.21166), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #49  GAPD (19.4373), Bd: -1000 / 1000, glyceraldehyde-3-phosphate dehydrogenase
  g3p[c] + nad[c] + pi[c] <=> 13dpg[c] + h[c] + nadh[c]

Show previous steps...
```

In this case we can see that the nadh[c] produced in the glycolysis pathway is either oxidized to nad[c] in the ethanol pathway (ACALD, ALCD2x) or converted to nadph[c] for cell biosynthesis through the energy management reactions (THD2).

Now let's expore the impact of pyruvate as the carbon sources in an anaerobic environment. *[Timing: Seconds]*

```
% Key parameters for fermentation section
model = e_coli_core;
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_pyr(e)',-20,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l'); % Set at -30 for aerobic
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');

% Perform FBA with Biomass_Ecoli_core_N(w/GAM)_Nmet2 as the objective,
FBAsolution = optimizeCbModel(model,'max',0,0);

% Import E.coli core map and adjust parameters
map=readCbMap('ecoli_core_map');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);
```

```
Document Written
```

A screenshot of that map is shown below (Figure 28).

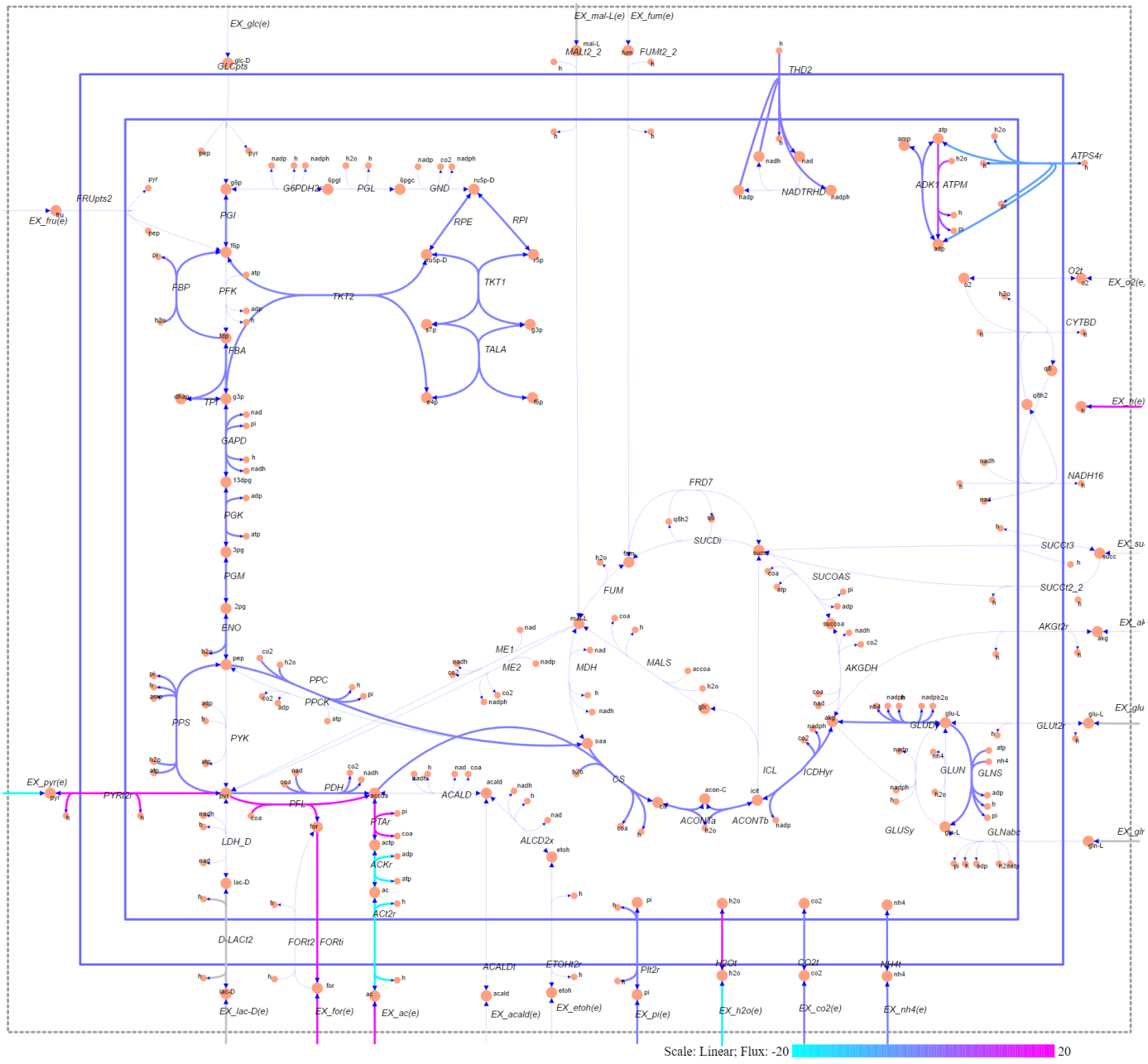


Figure 28. Screenshot of the core network with pyruate as the carbon source in an anaerobic environment.

From this map we can see that as the pyruvate enters the cell, part of the flux is directed upward through the glycolysis pathway (gluconeogenesis) to the pentose phosphate pathway to create the 4-, 5- and 7-carbon precursors. Part of the flux is also directed to the TCA cycle to feed the nitrogen metabolism, with the remaining flux being directed through the fermentation pathways to produce formate, acetate, and some atp[c] through substrate phosphorylation.

The flux values for this condition are calculated below. *[Timing: Seconds]*

```
printFluxVector(model,FBAsolution.x,true) % only prints nonzero reactions
```

```
ACKr -19.0039
ACONTa 0.0707136
ACONTb 0.0707136
```

```
ACt2r -19.0039
ADK1 0.494123
ATPM 8.39
ATPS4r -5.51453
Biomass_Ecoli_core_w_GAM 0.0655423
CO2t -0.948443
CS 0.0707136
ENO -0.272282
EX_ac(e) 19.0039
EX_co2(e) 0.948443
EX_for(e) 18.2547
EX_h(e) 18.5733
EX_h2o(e) -18.5741
EX_nh4(e) -0.357389
EX_pi(e) -0.24111
EX_pyr(e) -20
FBA -0.0651949
FBP 0.0651949
FORti 18.2547
GAPD -0.174231
GLNS 0.0167592
GLUDy -0.34063
H2Ot 18.5741
ICDHyr 0.0707136
NH4t 0.357389
PDH 1.06555
PFL 18.2547
PGI -0.0134362
PGK 0.174231
PGM 0.272282
PIt2r 0.24111
PPC 0.187818
PPS 0.494123
PTAr 19.0039
PYRt2r 20
RPE -0.0471118
RPI -0.0471118
TALA -0.0117255
THD2 1.12379
TKT1 -0.0117255
TKT2 -0.0353863
TPI -0.0651949
```

4.G. Nitrogen Metabolism

The final subsystem to be discussed in this tutorial is the nitrogen metabolism. Nitrogen enters the cell as either ammonium ion (nh4[c]), or as a moiety within glutamine (glu-L[c]) or glutamate (gln-L[c]). The *E.coli* core model covers the pathways between 2-oxoglutarate, L-glutamate, and L-glutamine. The location of the nitrogen metabolism reactions on the *E.coli* core map is shown in Figure 29.

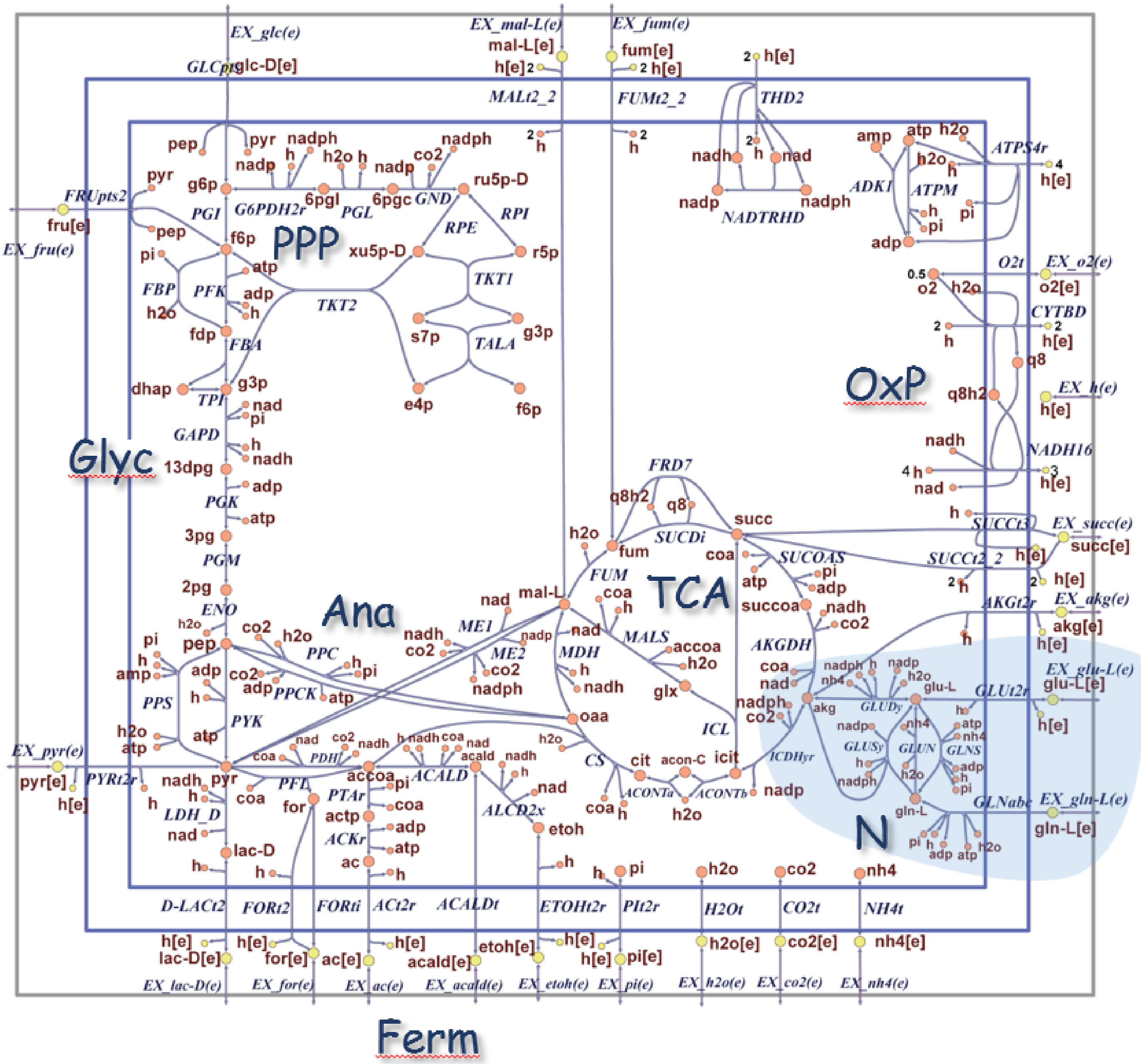


Figure 29. Nitrogen metabolism reactions highlighted in blue on the *E.coli* core map [3].

The reactions of the nitrogen metabolism include: *[Timing: Seconds]*

```
% Set initial constraints for nitrogen metabolism section
model = e_coli_core;
NIT_Reactions = transpose({'GLNabc','GLUt2r','GLUDy','GLNS','GLUSy','GLUN'});
[tmp,NIT_rxnID] = ismember(NIT_Reactions,model.rxns);
Reaction_Names = model.rxnNames(NIT_rxnID);
Reaction_Formulas = printRxnFormula(model,NIT_Reactions,0);
T = table(Reaction_Names,Reaction_Formulas,'RowNames',NIT_Reactions)
```

```
T =
                                     Reaction_Names
              _________________________________________________________________      ____________________

    GLNabc    'L-glutamine transport via ABC system'                                'atp[c] + gln-L[e] +
    GLUt2r    'L-glutamate transport via proton symport, reversible (periplasm)'    'glu-L[e] + h[e]  <=
    GLUDy     'glutamate dehydrogenase (NADP)'                                      'glu-L[c] + h2o[c] +
```

```
GLNS      'glutamine synthetase'                                    'atp[c] + glu-L[c] +
GLUSy     'glutamate synthase (NADPH)'                              'akg[c] + gln-L[c] +
GLUN      'glutaminase'                                             'gln-L[c] + h2o[c]
```

The reactions, GRPA relationships, and precursors for this section on the nitrogen metabolism are shown in the Figure 30 below.

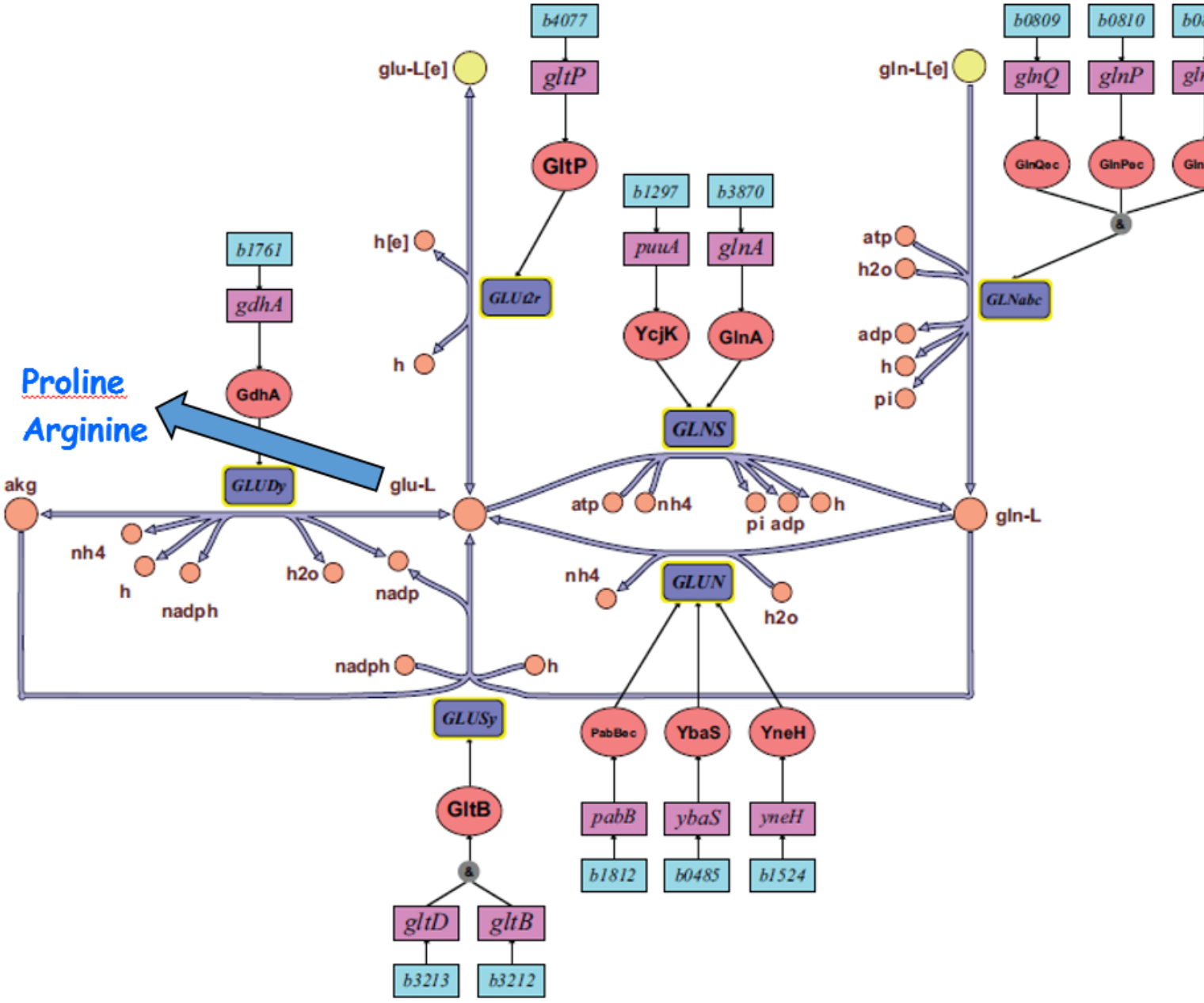


Figure 30. Reactions GRPA relationships, and precursors associated with the nitrogen metabolism [3].

Note that the precursors supported by nitrogen metaboism are proline and arginine.

In this simple model, one of the potential sources of nitrogen is through ammonium which is transported into the cell through a transporter (NH4t). Within the cell there are only two reactions (GLNS, GLUDy) that can also assimulate the needed nitrogen into the cell. This can be seen using the "surfNet" function. *[Timing: Seconds]*

```
surfNet(model,'nh4[c]',0,FBAsolution.x,1,1)
```

```
Met #54  nh4[c], Ammonium, H4N
Consuming reactions with non-zero fluxes :
  #51  GLNS (0.01676), Bd: 0 / 1000, glutamine synthetase
  atp[c] + glu-L[c] + nh4[c] -> adp[c] + gln-L[c] + h[c] + pi[c]
  #53  GLUDy (-0.34063), Bd: -1000 / 1000, glutamate dehydrogenase (NADP)
  glu-L[c] + h2o[c] + nadp[c] <=> akg[c] + h[c] + nadph[c] + nh4[c]
Producing reactions with non-zero fluxes :
  #69  NH4t (0.35739), Bd: -1000 / 1000, ammonia reversible transport
  nh4[e] <=> nh4[c]

Show previous steps...
```

Nitrogen can also enter the cell through the uptake of glutamate or glutamine. As a reminder, the default settings for the core model do not allow any amino acids to enter the core model. To change this you would need to use the "changeRxnBounds" COBRA Toolbox function to allow either glutamate or gluamine uptake capability.

Both glutamate and glutamine can serve as both carbon and nitrogen sources under aerobic conditions. An example of glutamate serving as both carbon and nitrogen source is shown in the COBRA code and Figure 31 below. *[Timing: Seconds]*

```
% Key parameters for fermentation section
model = e_coli_core;
model = changeRxnBounds(model,'EX_glc(e)',-0,'l');
model = changeRxnBounds(model,'EX_glu_L(e)',-20,'l');
model = changeRxnBounds(model,'EX_nh4(e)',-0,'l');
model = changeRxnBounds(model,'EX_o2(e)',-30,'l'); % Set at -30 for aerobic
model = changeObjective(model,'Biomass_Ecoli_core_w_GAM');

% Perform FBA with Biomass_Ecoli_core_N(w/GAM)_Nmet2 as the objective,
FBAsolution = optimizeCbModel(model,'max',0,0);

% Import E.coli core map and adjust parameters
map=readCbMap('ecoli_core_map');
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);
```

```
Document Written
```

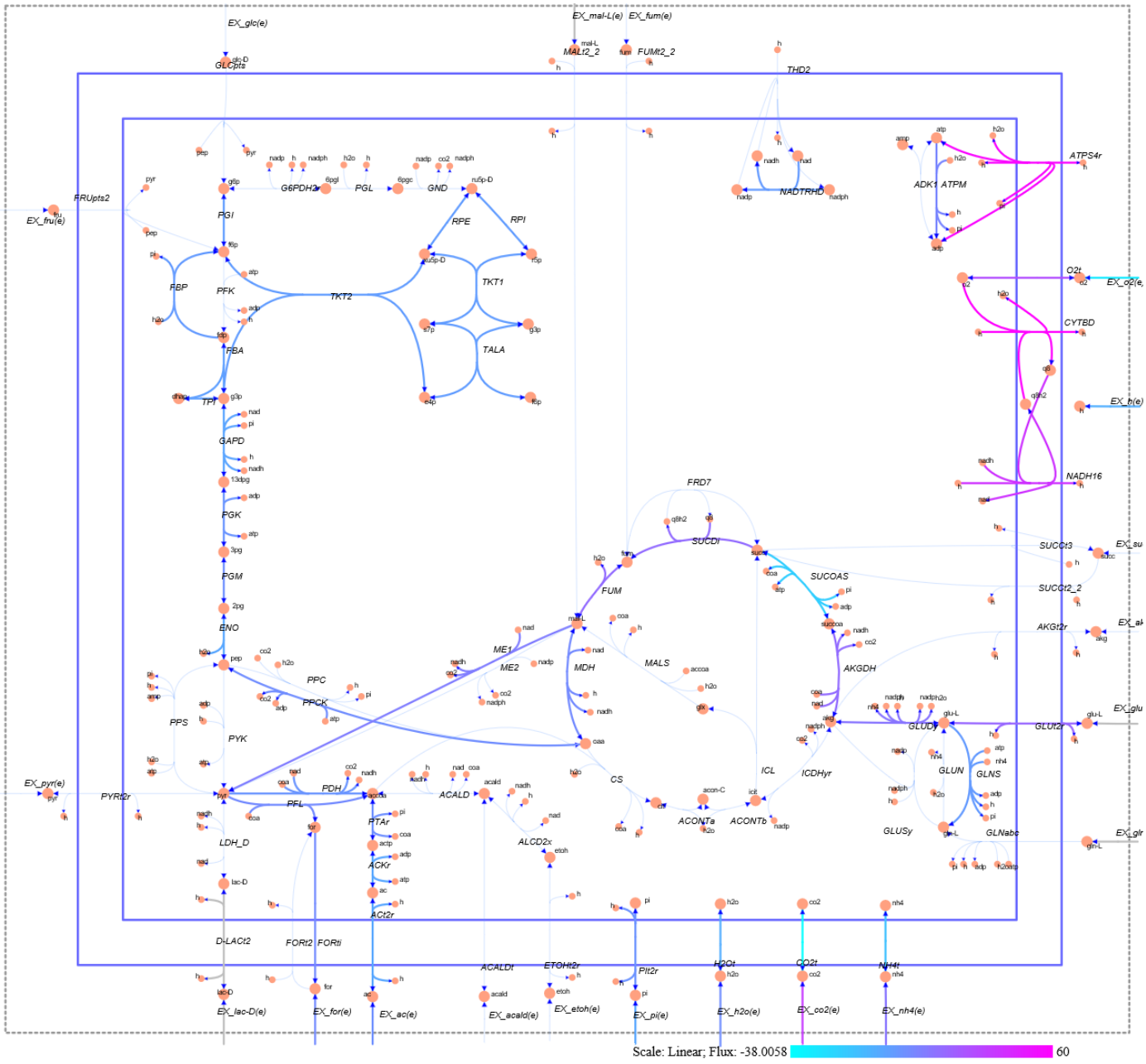


Figure 31. A screenshot of glutamate serving as both carbon and nitrogen source.

In this figure, it can be seen that glutamate enters the cell in the lower right. It passes through the nitrogen metabolism producing 2-oxogluarate (akg[c]) which then feeds the upper part of the TCA cycle. The anapleurotic reactions and gluconeogenesis support the flux necessary to create the the 4-, 5- and 7-carbon precursors. Part of the flux from the TCA cycle is also directed to the fermentation pathway precursors in addition to secreting both formate and acetate.

The fluxes for this example are shown below: *[Timing: Seconds]*

```
printFluxVector(model,FBAsolution.x,true) % only prints nonzero reactions
```

```
ACKr -4.71094
ACt2r -4.71094
AKGDH 18.8317
ATPM 8.39
ATPS4r 57.8269
Biomass_Ecoli_core_w_GAM 1.08283
```

```
CO2t -38.0058
CYTBD 60
ENO -4.49838
EX_ac(e) 4.71094
EX_co2(e) 38.0058
EX_for(e) 6.49219
EX_glu_L(e) -20
EX_h(e) -7.0754
EX_h2o(e) 5.89349
EX_nh4(e) 14.0956
EX_o2(e) -30
EX_pi(e) -3.98339
FBA -1.07709
FBP 1.07709
FORti 6.49219
FUM 18.8317
GAPD -2.87847
GLNS 0.276878
GLUDy 14.3724
GLUt2r 20
H2Ot -5.89349
MDH 18.8317
NADH16 41.1683
NADTRHD 0.265508
NH4t -14.0956
O2t 30
PDH 2.27697
PFL 6.49219
PGI -0.221979
PGK 2.87847
PGM 4.49838
PIt2r 3.98339
PPCK 16.8971
PTAr 4.71094
PYK 11.8366
RPE -0.778335
RPI -0.778335
SUCDi 18.8317
SUCOAS -18.8317
TALA -0.193717
TKT1 -0.193717
TKT2 -0.584617
TPI -1.07709
```

Since the normal source of nadh[c] from the glycolysis pathway is not availabe during gluconeogenesis, let's explore where the nadh[c] is produced and consumed. *[Timing: Seconds]*

```
surfNet(model,'nadh[c]',0,FBAsolution.x,1,1)
```

```
Met #51  nadh[c], Nicotinamide-adenine-dinucleotide-reduced, C21H27N7O14P2
Consuming reactions with non-zero fluxes :
  #49  GAPD (-2.87847), Bd: -1000 / 1000, glyceraldehyde-3-phosphate dehydrogenase
  g3p[c] + nad[c] + pi[c] <=> 13dpg[c] + h[c] + nadh[c]
  #67  NADH16 (41.1683), Bd: 0 / 1000, NADH dehydrogenase (ubiquinone-8 & 3 protons)
  4 h[c] + nadh[c] + q8[c] -> 3 h[e] + nad[c] + q8h2[c]
Producing reactions with non-zero fluxes :
  #8  AKGDH (18.8317), Bd: 0 / 1000, 2-Oxoglutarate dehydrogenase
  akg[c] + coa[c] + nad[c] -> co2[c] + nadh[c] + succoa[c]
  #13  Biomass_Ecoli_core_w_GAM (1.08283), Bd: 0 / 1000, Biomass Objective Function with GAM
  1.496 3pg[c] + 3.7478 accoa[c] + 59.81 atp[c] + 0.361 e4p[c] + 0.0709 f6p[c] + 0.129 g3p[c] + 0.205 g6
  2.8328 pyr[c] + 0.8977 r5p[c] -> 59.81 adp[c] + 4.1182 akg[c] + 3.7478 coa[c] + 59.81 h[c] + 3.547 nad
  #64  MDH (18.8317), Bd: -1000 / 1000, malate dehydrogenase
  mal-L[c] + nad[c] <=> h[c] + nadh[c] + oaa[c]
  #68  NADTRHD (0.26551), Bd: 0 / 1000, NAD transhydrogenase
  nad[c] + nadph[c] -> nadh[c] + nadp[c]
```

```
#71  PDH (2.27697), Bd: 0 / 1000, pyruvate dehydrogenase
coa[c] + nad[c] + pyr[c] -> accoa[c] + co2[c] + nadh[c]

Show previous steps...
```

We can see here that there are many sources of nadh[c] production including: AKGDH and MDH from the reductive pathways of the TCA cycle, the anapleurotic reaction ME1, PDH from the fermentation metabolism, and even with the energy management reactions where excess nadph[c] is converted to nadh[c]. The consumers are primarily NADH16 where it provides the reducing power necessary for the electron transport chain and GAPD which is required for the operaton of gluconeogenesis.

5. Conclusion

This wraps up the tutorial on the *E.coli* core model. It has attempted to show how the COBRA toolbox can be used to explore a genome-scale metabolic network reconstruction using the core model as an example. Now with this beginning skill set you can start exploring the larger and more accurate network reconstructions!

6. Reflective Questions

- What is the difference between glycolysis and gluconeogenesis?
- What reactions make-up the glycolysis pathway?
- What metabolites are created in the glycolysis pathway?
- What is the final metabolite created by the glycolysis pathway?
- What are the biosynthetic precursors created by the glycolysis pathway?
- What are the biosynthetic precursors created by the pentose phosphate pathway?
- What is the difference between the oxidative and non-oxidative pathways of the pentose phosphate pathway?
- What reactions make-up the pentose phosphate pathway?
- What metabolites are created in the pentose phosphate pathway?
- What are the different names for the TCA cycle?
- What are the biosynthetic precursors created by the TCA cycle?
- What is the oxidative pathway in the TCA cycle?
- What reactions make-up the TCA cycle?
- What metabolites are created in the TCA cycle?
- What is the anapleurotic pathway?
- What is the glycoxylate cycle?
- What reactions make-up the anapleurotic pathway and the glycoxylate cycle?
- What metabolites are created in the anapleurotic pathway and the glycoxylate cycle?
- What reactions make-up the core models oxidative phosphorylation and electron transfer chain?
- What metabolites are created in the core models oxidative phosphorylation and electron transfer chain?
- What reactions make-up the fermentation pathways?
- What metabolites are created in the fermentation pathways?
- What are the biosynthetic precursors created by the nitrogen metabolism?
- What reactions make-up the nitrogen metabolism?
- What metabolites are created in the nitrogen metabolism?
- What is the purpose of the "changeCobraSolver" function?
- What is the purpose of the "readCbMap" function?
- What are geneIDs?
- What is the purpose of the "printLabeledData" function?
- What is the purpose of the "findRxnsFromGenes" function?
- Describe the capabilities of the "surfNet" function?
- What are the default model constraints for the *E.coli* core model?

- What is the purpose of the "findRxnIDs" function?
- What is the purpose of the objective function?
- What is the purpose of the biomass reaction?
- Describe the capabilities of the "printFluxVector" function?
- What are the units of flux in the COBRA models?
- What is the purpose of the "computeFluxSplits" function?
- Describe the capabilities of the "optimizeCbModel" function?
- What is the purpose of the "changeRxnBounds" function?
- What are the outputs produced by the "optimizeCbModel" function?

7. Tutorial Understanding Enhancement Problems

1. Find the maximum atp[c], nadh[c], and nadph[c] that can be produced by the *E.coli* core model in an aerobic environment assuming a fixed glucose uptake rate of -1 $\mathrm{mmol} \cdot \mathrm{gDW}^{-1} \cdot \mathrm{hr}^{-1}$. Hint: For atp[c] you can set ATPM as the objective function but for nadh[c] and nadph[c] you will need to create separate demand functions. See Chapter 19 of Palsson's book [1].
2. Compare the difference in the aerobic vs anaerobic flux rate through the glycolysis pathway by setting biomass function to a fixed rate of 0.8739 h^{-1}. Why is the anaerobic flux so much higher than the aerobic flux? Hint: Set the objective function to the glucose exchange reaction.

References

1. Palsson, B. (2015). Systems biology : constraint-based reconstruction and analysis. Cambridge, United Kingdom, Cambridge University Press.
2. Palsson, B. (2006). Systems biology : properties of reconstructed networks. Cambridge ; New York, Cambridge University Press.
3. Orth, Fleming, and Palsson (2010), *EcoSal Chapter 10.2.1 - Reconstruction and Use of Microbial Metabolic Networks: the Core* Escherichia coli *Metabolic Model as an Educational Guide* - http://www.asmscience.org/content/journal/ecosalplus/10.1128/ecosalplus.10.2.1#backarticlefulltext
4. Becker, S. et al., "Quantitative prediction of cellular metabolism with constraint-based models: The COBRA Toolbox", *Nat. Protoc* **2**, 727-738 (2007).
5. Schellenberger J, Que R, Fleming RMT, Thiele I, Orth JD, Feist AM, Zielinski DC, Bordbar A, Lewis NE, Rahmanian S et al., Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0 *Nat. Protoc* **6**(9):1290-307 (2011).
6. Feist, A. M., Herrgard, M. J., Thiele, I., Reed , J . L., and Palsson, B. 0., (2009). Reconstruction of biochemical networks in microorganisms. Nat. Rev Microbiol 7 : 129-143.
7. Price, N. D., Papin, J . A., Schilling, C. H. , and Palsson, B. 0., (2003) Genome-scale microbial *in silico* models: the constraints-based approach. Trends Biotechnol 21:162-169.
8. Orth, J. D., I. Thiele, et al. (2010). "What is flux balance analysis?" Nature biotechnology 28(3): 245-248.
9. Schellenberger, J., Park, J. O., Conrad, T. M., and Palsson, B. O., (2010) "BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions", *BMC Bioinformatics*, **11**:213.
10. King, Z.A., Lu, J., Draeger, A., Miller, P., Federowicz, S., Lerman, J.A., Palsson, B.O., Lewis, N.E., (2015) "BiGG models: A platform for integrating, standardizing and sharing genome-scale models", *Nucleic Acids Research.*
11. Schaechter, M., Ingraham, J. L., Neidhardt, F. C. (2006), "Microbe", ASM Press, Washington, D. C.
12. Edwards, J. S. and B. O. Palsson (2000). "Robustness analysis of the Escherichia coli metabolic network." Biotechnology progress 16(6): 927-939.