Gene/Reaction Knockouts

Learning Objectives

- Explain the purpose of a gene/reaction knockout
- Explain growth-coupled bioproduction.
- Explain the purpose of a production envelope plot.
- Explain the capabilities and limitations of OptKnock.
- Explain the capabilities and limitations of the Genetic Design Local Search (GDLS) tool.
- Explain the capabilities and limitations of OptGene.

Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

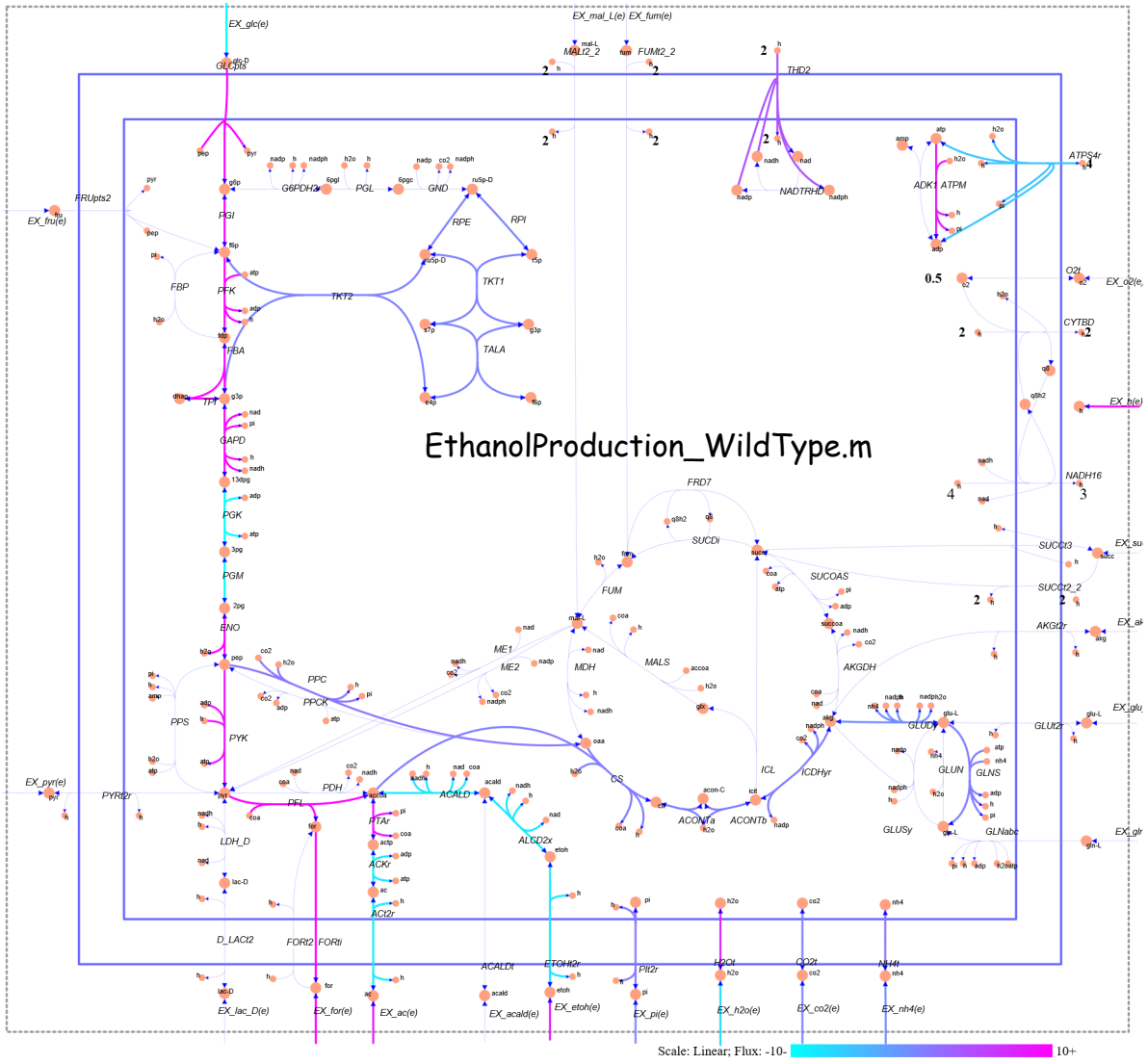
Gene/Reaction Knockouts

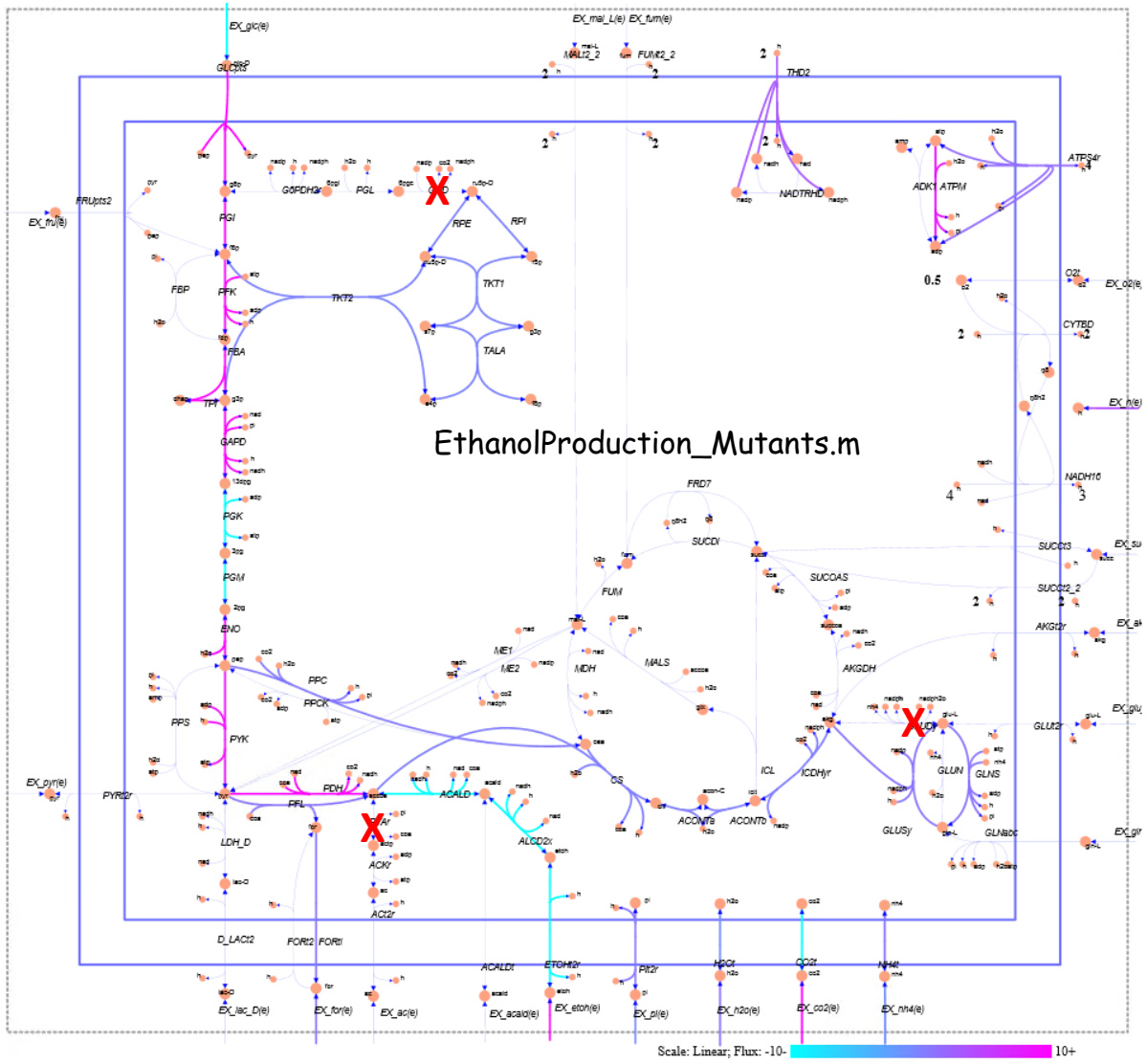
- Metabolic engineering has been successful is using the recombinant DNA technology to selectively alter cell metabolism (new strain design) and improve a targeted cellular function (bioproduct production).
- The use of metabolic genome scale metabolic reconstructions represents a major opportunity for the field of metabolic engineering to use whole-cell networks and systems-level analysis to determine optimal metabolic engineering strategies.
- Constraint-based techniques can be used for metabolic engineering where FBA-based algorithms, such as OptKnock, GDLS, or OptGene, predict the gene/reaction knockouts that can generate a desired phenotype to produce specific metabolites by an organism
- Using this approach, the desired phenotype will show an increase in biomass yield coupled to an increase in the production rate of a desired by-product (metabolite). In other words, the cell will be able to grow faster only by producing more of the desired metabolite. The resulting knockout strain (mutant) could have significant metabolite production at a desired growth rate.
- These knockout strains would theoretically be stable strains that can produce specific metabolites.

Simulating Gene/Reaction Knockouts

- Just as growth in different environments can be simulated with FBA, gene/reaction knockouts can also be simulated by changing reaction bounds.
- To simulate the knockout of any gene, its associated reaction or reactions can simply be constrained to not carry flux. By setting **both the upper and lower bounds** ('b') of a reaction to 0 mmol gDW^{-1} hr^{-1}, a reaction is essentially knocked out, and is restricted from carrying flux
- The COBRA Toolbox contains a function called **deleteModelGenes** that uses the GPRs to constrain the reactions associated with a given gene. Then FBA may be used to predict the model phenotype with gene knockouts.

Creating a Mutant Strain: Anaerobic Ethanol Production





Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

OptKnock
A Reaction Deletion Strategy

- The OptKnock framework suggests a reaction **deletion** strategy that leads to the overproduction of specific chemical compounds.
- This is accomplished by ensuring that the production of the desired chemical becomes an required byproduct of growth by "shaping" the connectivity of the metabolic network.
- OptKnock identifies and subsequently removes metabolic reactions that are capable of **uncoupling** cellular growth from chemical production.
- To reduce the computation time of OptKnock the number of candidate reactions for knockout should be minimized.
- Requires Gurobi or CPLEX solvers! Built-in Matlab solvers will not work.

Maximize: Bioengineering Objective
(through reaction knockouts)

Subject to:
- **Maximize:** cellular objective
(over fluxes)
- **Subject to:** Fixed substrate uptake
Network Stoichiometry
Blocked reactions identified
by the outer problem

Number of knockouts ≤ limit

Bilevel optimization structure of OptKnock

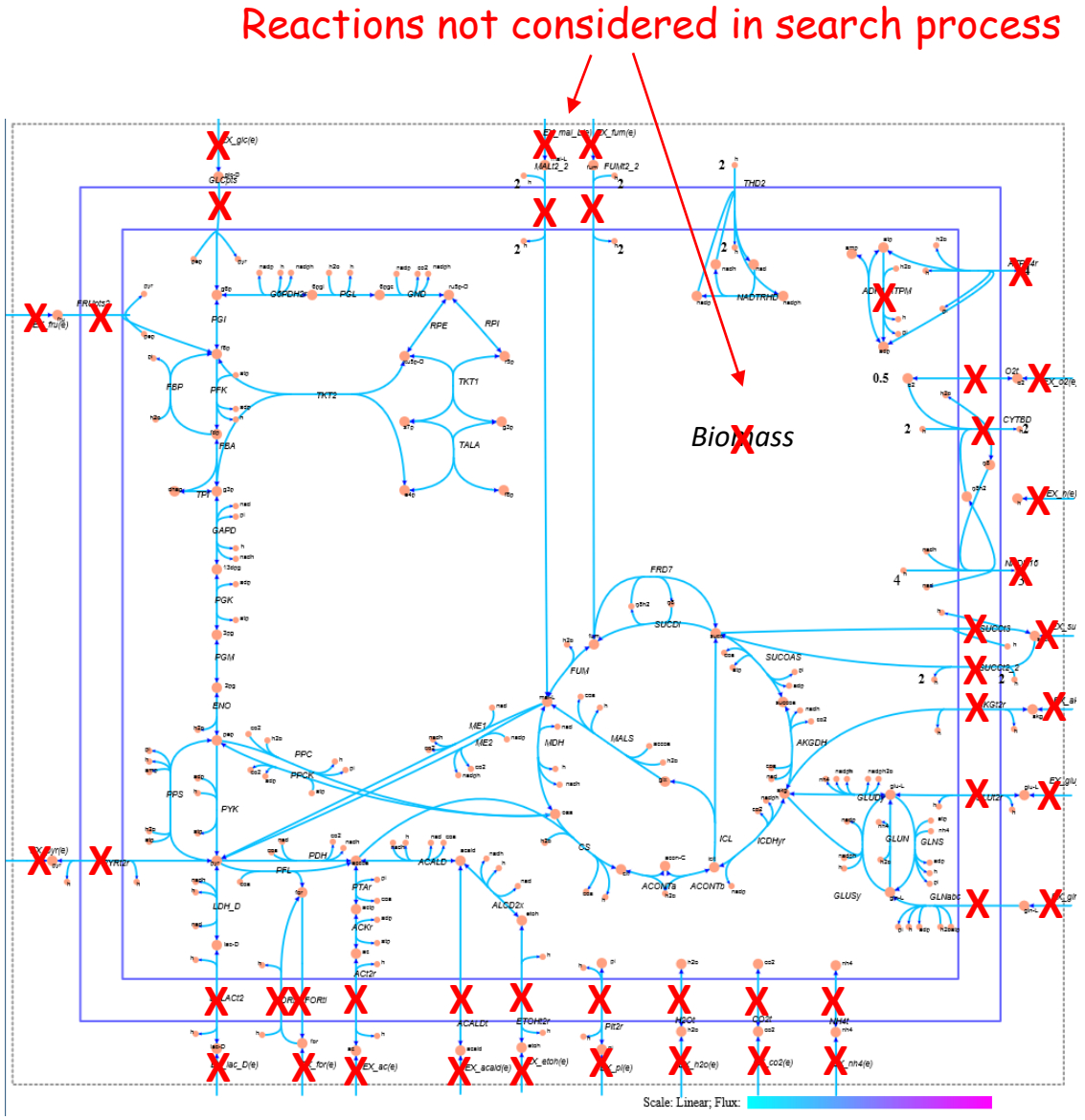
Burgard, A. P., P. Pharkya, et al. (2003). "Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization." Biotechnology and bioengineering 84(6): 647-657.

Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

Reducing OptKnock Computational Time

- OptKnock can take a large amount of time to identify the reactions to knockout. The computational time can be significantly reduced by limiting the number of potential knockout reactions.
- When building the list of reactions, for deletion, "selectRxns," exclude exchange and transport reactions, and biomass and ATP maintenance requirements.
  - ✓ ATP maintenance requirements: ATPM
  - ✓ Exchange Reactions: EX_?
  - ✓ Biomass Objective Function:
    'Biomass_Ecoli_core_N(w/GAM)_Nmet2'
  - ✓ Transport Reactions:
    ?tipp , ?tex, ?tpp, ?t2pp, ?t3pp, ?t2rpp, ?abcpp, etc.



Identifying Unwanted Knockout Reactions

- When building the list of reactions, for deletion, "selectRxns," exclude exchange and transport reactions, and biomass and ATP maintenance requirements.
  - ✓ ATP maintenance requirements: ATPM
  - ✓ Exchange Reactions: EX_?
  - ✓ Biomass Objective Function: 'Biomass_Ecoli_core_N(w/GAM)_Nmet2'
  - ✓ Transport Reactions:?tipp , ?tex, ?tpp, ?t2pp, ?t3pp, ?t2rpp, ?abcpp, etc.
- Removing unwanted reactions

```
[transRxns,nonTransRxns] = findTransRxns(model,true); % Identify non-transport/exchange reactions
[tmp,ATPMnumber] = ismember('ATPM',nonTransRxns); % Identify ATPM reaction number
[tmp,BioMassnumber] = ismember('BiomassEcoli',nonTransRxns); % Identify biomass reaction number
nonTransRxnsLength = length(nonTransRxns); % Find number of non-transport reactions

selectedRxns = {nonTransRxns{[1:ATPMnumber-1, ATPMnumber+1:BioMassnumber-1, …
BioMassnumber+1:nonTransRxnsLength]}}; % Reactions to be used by OptKnock
```

Transport & Non-transport Reactions

[transRxns,nonTransRxns] = findTransRxns(model,true)

Transport (transRxns) Reactions

'ACALDt'
'ACt2r'
'AKGt2r'
'ATPS4r'
'CO2t'
'CYTBD'
'D_LACt2'
'ETOHt2r'
'EX_ac(e)'
'EX_acald(e)'
'EX_akg(e)'
'EX_co2(e)'
'EX_etoh(e)'
'EX_for(e)'
'EX_fru(e)'
'EX_fum(e)'
'EX_glc(e)'
'EX_gln_L(e)'
'EX_glu_L(e)'
'EX_h2o(e)'
'EX_h(e)'
'EX_lac_D(e)'
'EX_mal_L(e)'
'EX_nh4(e)'
'EX_o2(e)'
'EX_pi(e)'
'EX_pyr(e)'
'EX_succ(e)'
'FORt2'
'FORti'
'FRUpts2'
'FUMt2_2'
'GLCpts'
'GLNabc'
'GLUt2r'
'H2Ot'
'MALt2_2'
'NADH16'
'NH4t'
'O2t'
'PIt2r'
'PYRt2r'
'SUCCt2_2'
'SUCCt3'
'THD2'

Non-transport (nonTransRxns) Reactions

'ACALD'
'ACKr'
'ACONTa'
'ACONTb'
'ADK1'
'AKGDH'
'ALCD2x'
'ATPM'
'Biomass'
'CS'
'ENO'
'FBA'
'FBP'
'FRD7'
'FUM'
'G6PDH2r'
'GAPD'
'GLNS'
'GLUDy'
'GLUN'
'GLUSy'
'GND'
'ICDHyr'
'ICL'
'LDH_D'
'MALS'
'MDH'
'ME1'
'ME2'
'NADTRHD'
'PDH'
'PFK'
'PFL'
'PGI'
'PGK'
'PGL'
'PGM'
'PPC'
'PPCK'
'PPS'
'PTAr'
'PYK'
'RPE'
'RPI'
'SUCDi'
'SUCOAS'
'TALA'
'TKT1'
'TKT2'
'TPI'

IdentifyingUnwantedKnockoutReactions.m

Identifying the Potential Knockout Reactions

```
% IdentifyingUnwantedKnockoutReactions.m
clear; clc;

% Input the E.coli core model and set environmental conditions
model = readCbModel('ecoli_core_model.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Perform FBA with Biomass_Ecoli_core_N(w/GAM)-Nmet2 as the objective,
FBAsolution = optimizeCbModel(model,'max')

% Identify non-tranport reactions
[transRxns,nonTransRxns] = findTransRxns(model,true);

% Removing ATPM and biomass function
[tmp,ATPMnumber] = ismember('ATPM',nonTransRxns); % Identify ATPM reaction number
[tmp,BioMassnumber] = ismember('Biomass_Ecoli_core_N(w/GAM)-Nmet2',nonTransRxns); % Identify biomass reaction number
nonTransRxnsLength = length(nonTransRxns); % Find number of non-transport reactions
selectedRxns = {nonTransRxns{[1:ATPMnumber-1, ATPMnumber+1:BioMassnumber-1, BioMassnumber+1:nonTransRxnsLength]}};

disp('Transport Reactions');
disp(transRxns);

disp('Nontransport Reactions');
disp(nonTransRxns);

disp('selectedRxns');
disp(selectedRxns');
```

Transport, Non-transport & Selected Reactions

Transport (transRxns) Reactions

'ACALDt'	'EX_h(e)'	'PIt2r'
'ACt2r'	'EX_lac_D(e)'	'PYRt2r'
'AKGt2r'	'EX_mal_L(e)'	'SUCCt2_2'
'ATPS4r'	'EX_nh4(e)'	'SUCCt3'
'CO2t'	'EX_o2(e)'	'THD2'
'CYTBD'	'EX_pi(e)'	
'D_LACt2'	'EX_pyr(e)'	
'ETOHt2r'	'EX_succ(e)'	
'EX_ac(e)'	'FORt2'	
'EX_acald(e)'	'FORti'	
'EX_akg(e)'	'FRUpts2'	
'EX_co2(e)'	'FUMt2_2'	
'EX_etoh(e)'	'GLCpts'	
'EX_for(e)'	'GLNabc'	
'EX_fru(e)'	'GLUt2r'	
'EX_fum(e)'	'H2Ot'	
'EX_glc(e)'	'MALt2_2'	
'EX_gln_L(e)'	'NADH16'	
'EX_glu_L(e)'	'NH4t'	
'EX_h2o(e)'	'O2t'	

Non-transport (nonTransRxns) Reactions

'ACALD'	'GLUSy'	'PTAr'
'ACKr'	'GND'	'PYK'
'ACONTa'	'ICDHyr'	'RPE'
'ACONTb'	'ICL'	'RPI'
'ADK1'	'LDH_D'	'SUCDi'
'AKGDH'	'MALS'	'SUCOAS'
'ALCD2x'	'MDH'	'TALA'
'ATPM'	'ME1'	'TKT1'
'Biomass'	'ME2'	'TKT2'
'CS'	'NADTRHD'	'TPI'
'ENO'	'PDH'	
'FBA'	'PFK'	
'FBP'	'PFL'	
'FRD7'	'PGI'	
'FUM'	'PGK'	
'G6PDH2r'	'PGL'	
'GAPD'	'PGM'	
'GLNS'	'PPC'	
'GLUDy'	'PPCK'	
'GLUN'	'PPS'	

Selected Reactions

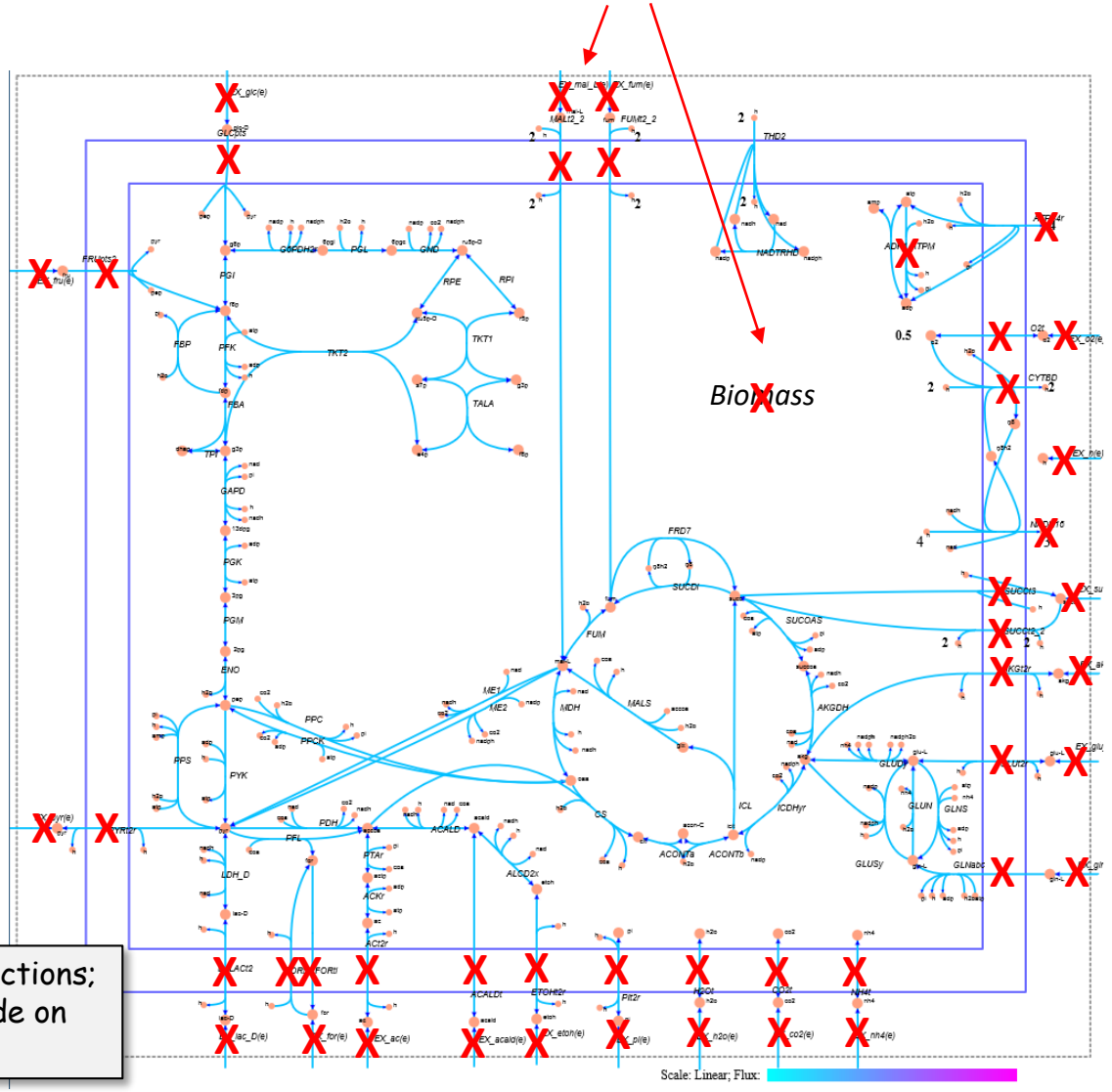
'ACALD'	'GLUSy'	'PTAr'
'ACKr'	'GND'	'PYK'
'ACONTa'	'ICDHyr'	'RPE'
'ACONTb'	'ICL'	'RPI'
'ADK1'	'LDH_D'	'SUCDi'
'AKGDH'	'MALS'	'SUCOAS'
'ALCD2x'	'MDH'	'TALA'
'CS'	'ME1'	'TKT1'
'ENO'	'ME2'	'TKT2'
'FBA'	'NADTRHD'	'TPI'
'FBP'	'PDH'	
'FRD7'	'PFK'	
'FUM'	'PFL'	
'G6PDH2r'	'PGI'	
'GAPD'	'PGK'	
'GLNS'	'PGL'	
'GLUDy'	'PGM'	
'GLUN'	'PPC'	
	'PPCK'	
	'PPS'	

IdentifyingUnwantedKnockoutReactions.m

Potential Knockout Reactions

1	'ACALD'	26	'EX_fru(e)'	51	'GLNS'	76	'PGL'
2	'ACALDt'	27	'EX_fum(e)'	52	'GLNabc'	77	'PGM'
3	'ACKr'	28	'EX_glc(e)'	53	'GLUDy'	78	'PIt2r'
4	'ACONTa'	29	'EX_gln_L(e)'	54	'GLUN'	79	'PPC'
5	'ACONTb'	30	'EX_glu_L(e)'	55	'GLUSy'	80	'PPCK'
6	'ACt2r'	31	'EX_h2o(e)'	56	'GLUt2r'	81	'PPS'
7	'ADK1'	32	'EX_h(e)'	57	'GND'	82	'PTAr'
8	'AKGDH'	33	'EX_lac_D(e)'	58	'H2Ot'	83	'PYK'
9	'AKGt2r'	34	'EX_mal_L(e)'	59	'ICDHyr'	84	'PYRt2r'
10	'ALCD2x'	35	'EX_nh4(e)'	60	'ICL'	85	'RPE'
11	'ATPM'	36	'EX_o2(e)'	61	'LDH_D'	86	'RPI'
12	'ATPS4r'	37	'EX_pi(e)'	62	'MALS'	87	'SUCCt2_2'
13	'Biomass'	38	'EX_pyr(e)'	63	'MALt2_2'	88	'SUCCt3'
14	'CO2t'	39	'EX_succ(e)'	64	'MDH'	89	'SUCDi'
15	'CS'	40	'FBA'	65	'ME1'	90	'SUCOAS'
16	'CYTBD'	41	'FBP'	66	'ME2'	91	'TALA'
17	'D_LACt2'	42	'FORt2'	67	'NADH16'	92	'THD2'
18	'ENO'	43	'FORti'	68	'NADTRHD'	93	'TKT1'
19	'ETOHt2r'	44	'FRD7'	69	'NH4t'	94	'TKT2'
20	'EX_ac(e)'	45	'FRUpts2'	70	'O2t'	95	'TPI'
21	'EX_acald(e)'	46	'FUM'	71	'PDH'		
22	'EX_akg(e)'	47	'FUMt2_2'	72	'PFK'		
23	'EX_co2(e)'	48	'G6PDH2r'	73	'PFL'		
24	'EX_etoh(e)'	49	'GAPD'	74	'PGI'		
25	'EX_for(e)'	50	'GLCpts'	75	'PGK'		

Reactions not considered in search process



Black – Potential knockout reactions;
Red – Reactions to NOT include on optKnock process;

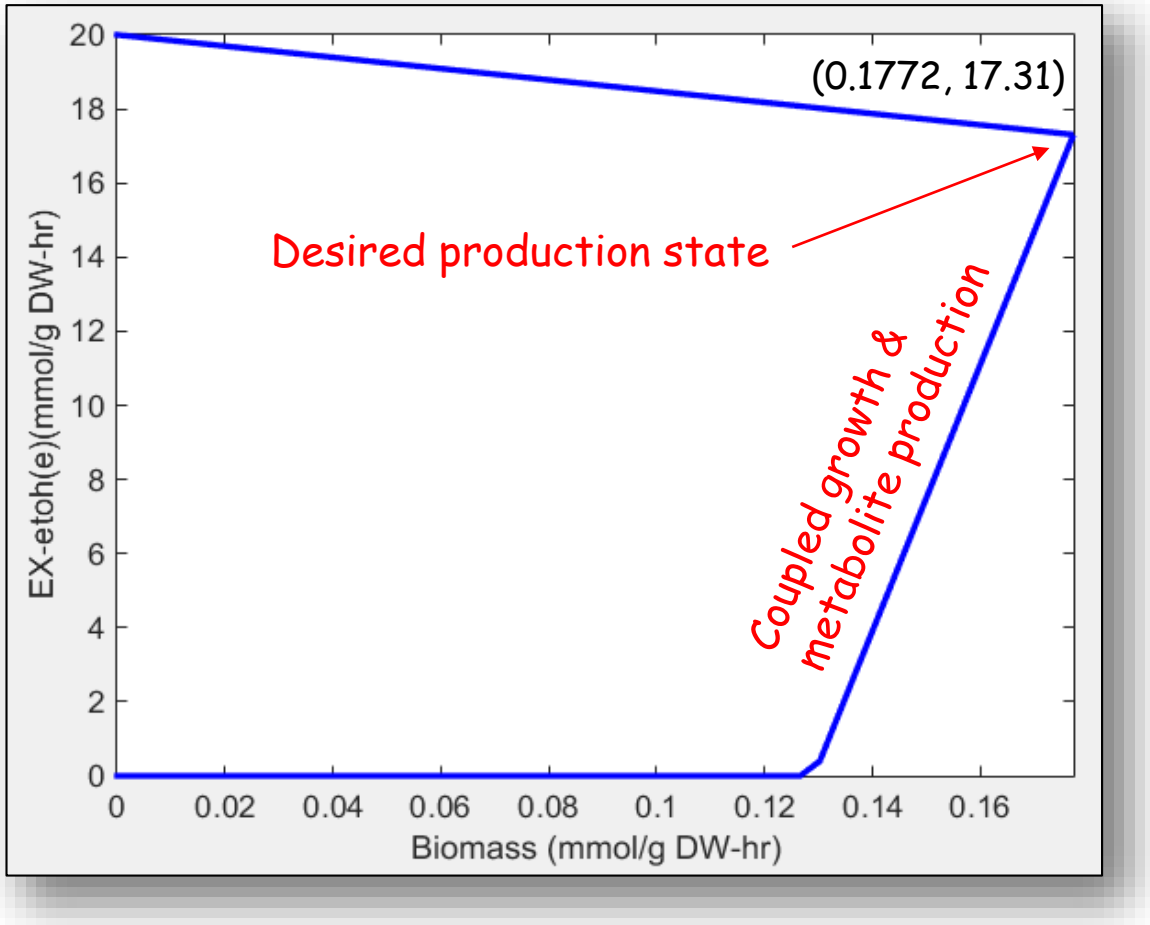
Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

Production Envelopes

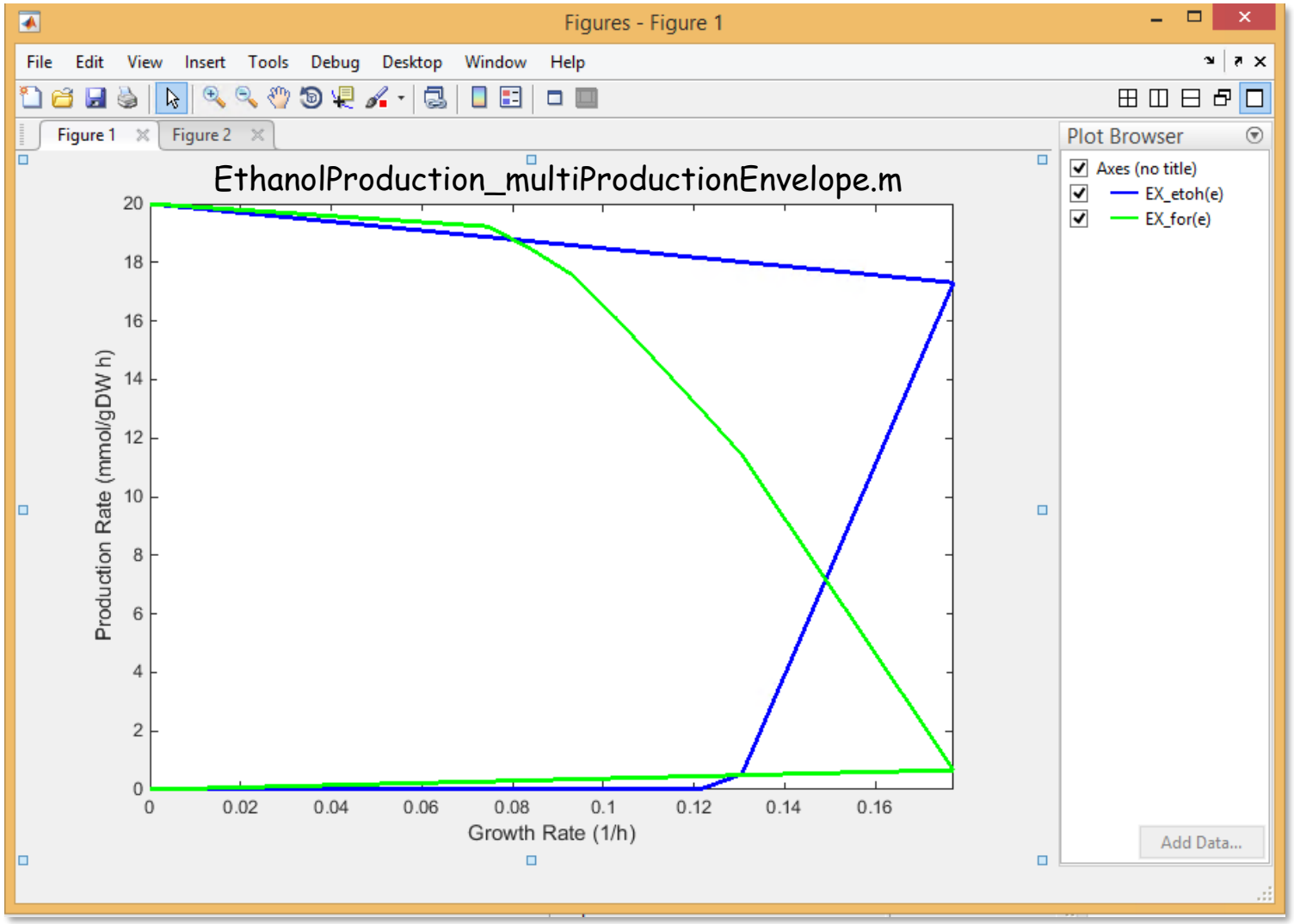
(EthanolProduction_OptKnock.m)

Production envelope - A graph that shows both the maximum and minimum bioproduct production with respect to the growth rate




```
lineColor = 'b';
targetRxn = 'EX_etoh(e)';
biomassRxn = 'Biomass_Ecoli_core_N(w/GAM)-Nmet2';
geneDelFlag = false; % Genes(true) or reactions(false)
nPts = 50;
deletions = {'FUM','G6PDH2r','GLUDy','PTAr','SUCDi'};
[biomassValues,targetValues] = productionEnvelope(model,deletions,lineColor,targetRxn,biomassRxn,geneDelFlag,nPts);
xlabel('Biomass (mmol/g DW-hr)')
ylabel('EX-etoh(e)(mmol/g DW-hr)')
```

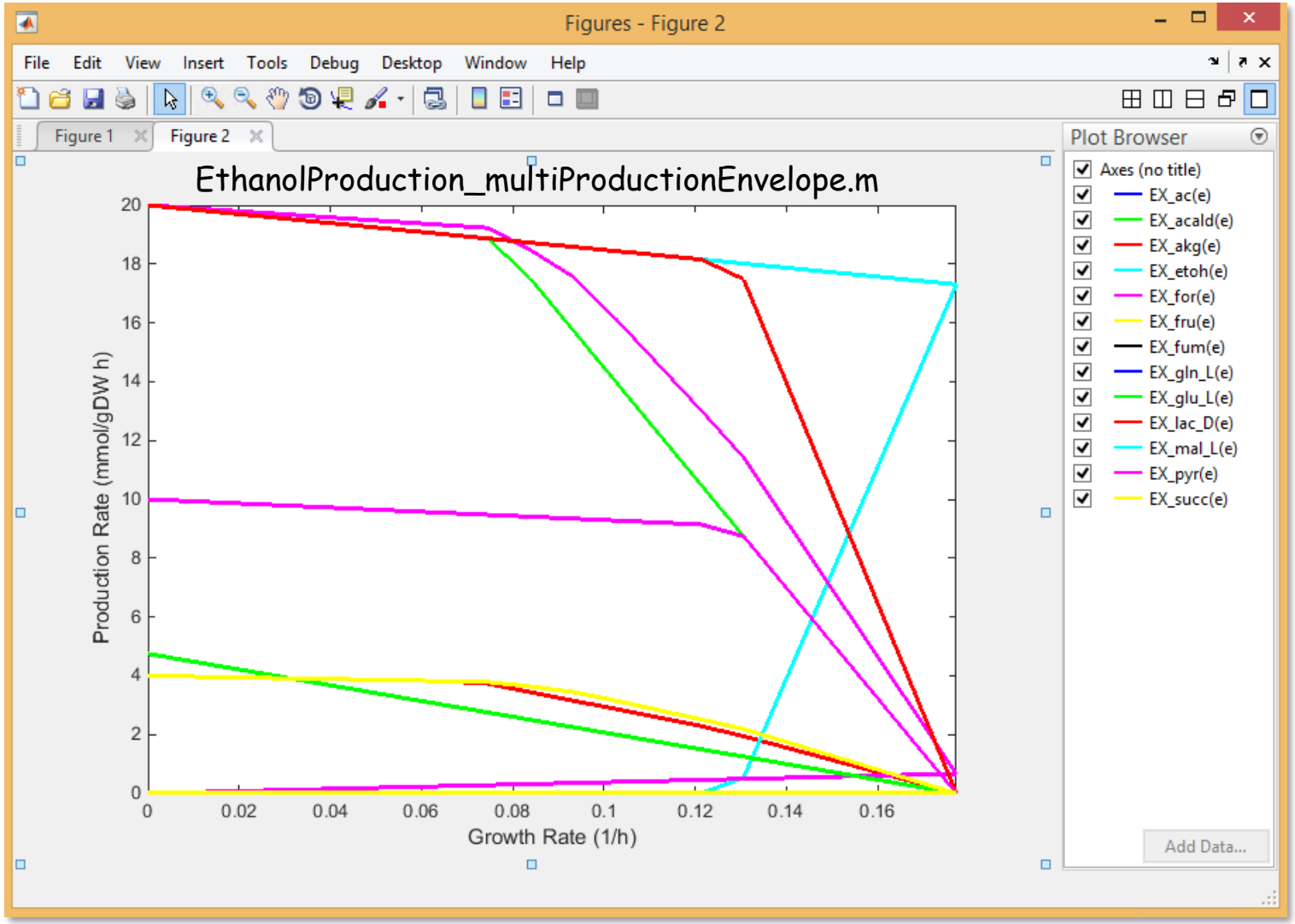
Multiproduction Envelopes of All Growth-coupled Secreted Metabolites




```
% Reactions to be deleted
deletions = {'ACKr','GLUDy','ME2','PGL','PYK'};

% Biomass function
biomassRxn = {'Biomass_Ecoli_core_N(w/GAM)-Nmet2'};

% Show only growth coupled metabolites
[biomassValues,targetValues] = multiProductionEnvelope(model,deletions,biomassRxn,false,20,false);
```

Don't plot all secreted metabolites only those that are growth coupled

Multiproduction Envelopes of All the Secreted Metabolites




```
% Reactions to be deleted
deletions = {'ACKr','GLUDy','ME2','PGL','PYK'};

% Biomass function
biomassRxn = {'Biomass_Ecoli_core_N(w/GAM)_Nmet2'};

%Show all secreted metabolites
[biomassValues,targetValues] = multiProductionEnvelope(model,deletions,biomassRxn,false,20,true);
```

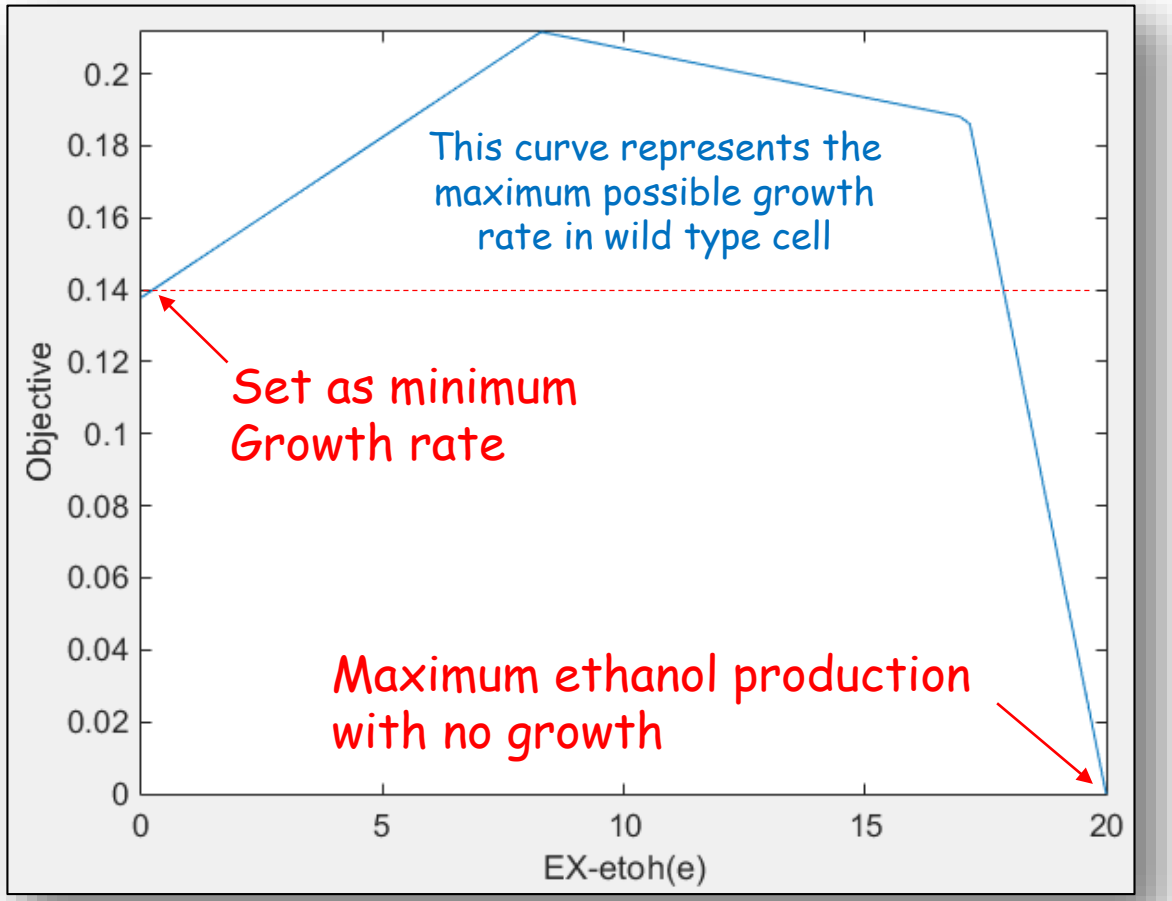
Plot all secreted metabolites

Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

What Do We Know About Ethanol Production?

```
% AnaerobicEthanolRA.m

clear; clc;

% Input the E.coli core model

model = readCbModel('ecoli_core_model.mat');

% Set uptake rates

model = changeRxnBounds(model,'EX_glc(e)',-10,'b');

model = changeRxnBounds(model,'EX_o2(e)',-0,'b');

% Set optimization objective to Biomass_Ecoli_core_N(w/GAM)-Nmet2

model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Using robustnessAnalysis, plot the objective function as a function

% of the ethanol secretion rate

[controlFlux, objFlux] = robustnessAnalysis(model,'EX_etoh(e)',100);
```

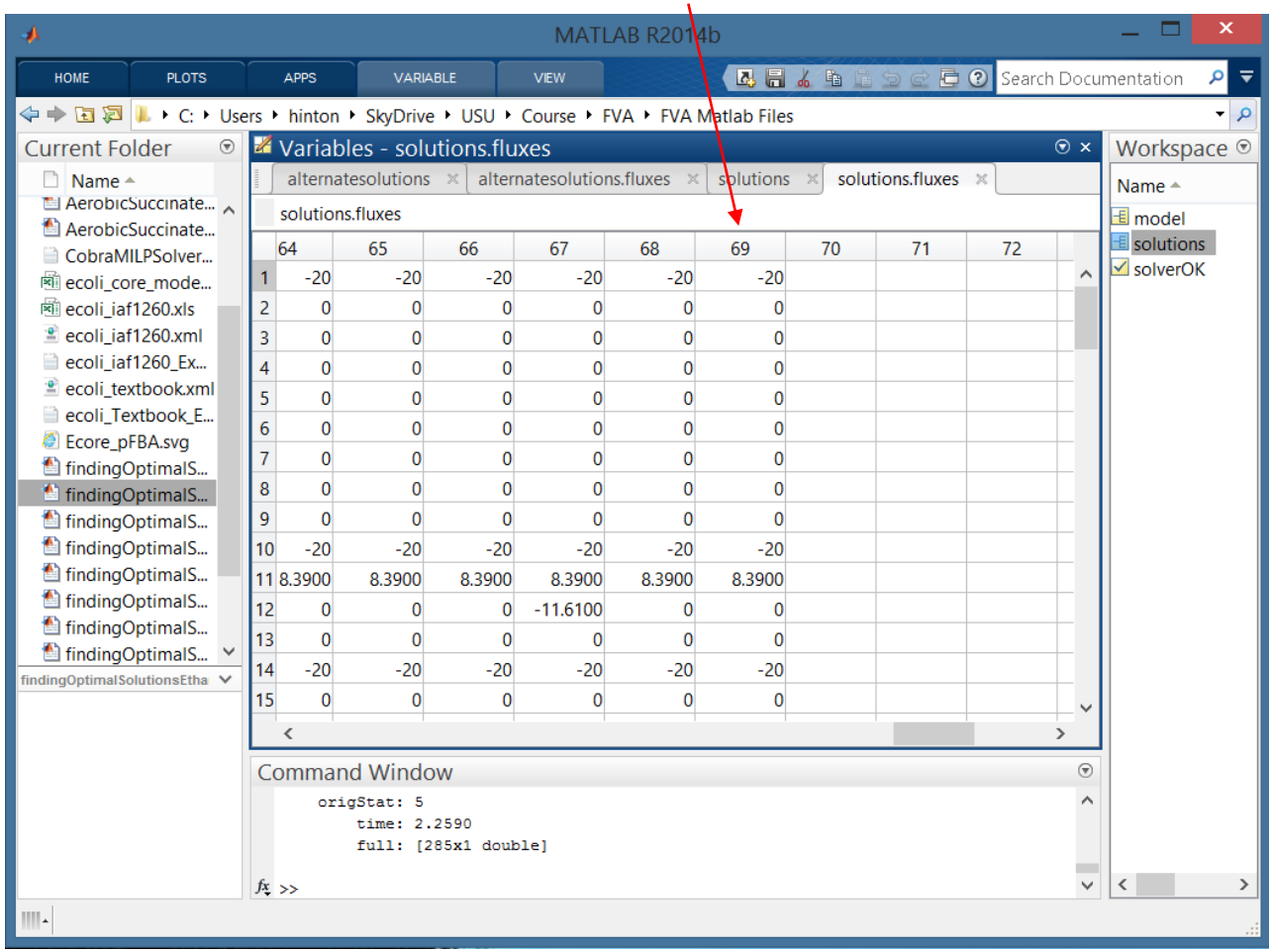


Robustness Analysis

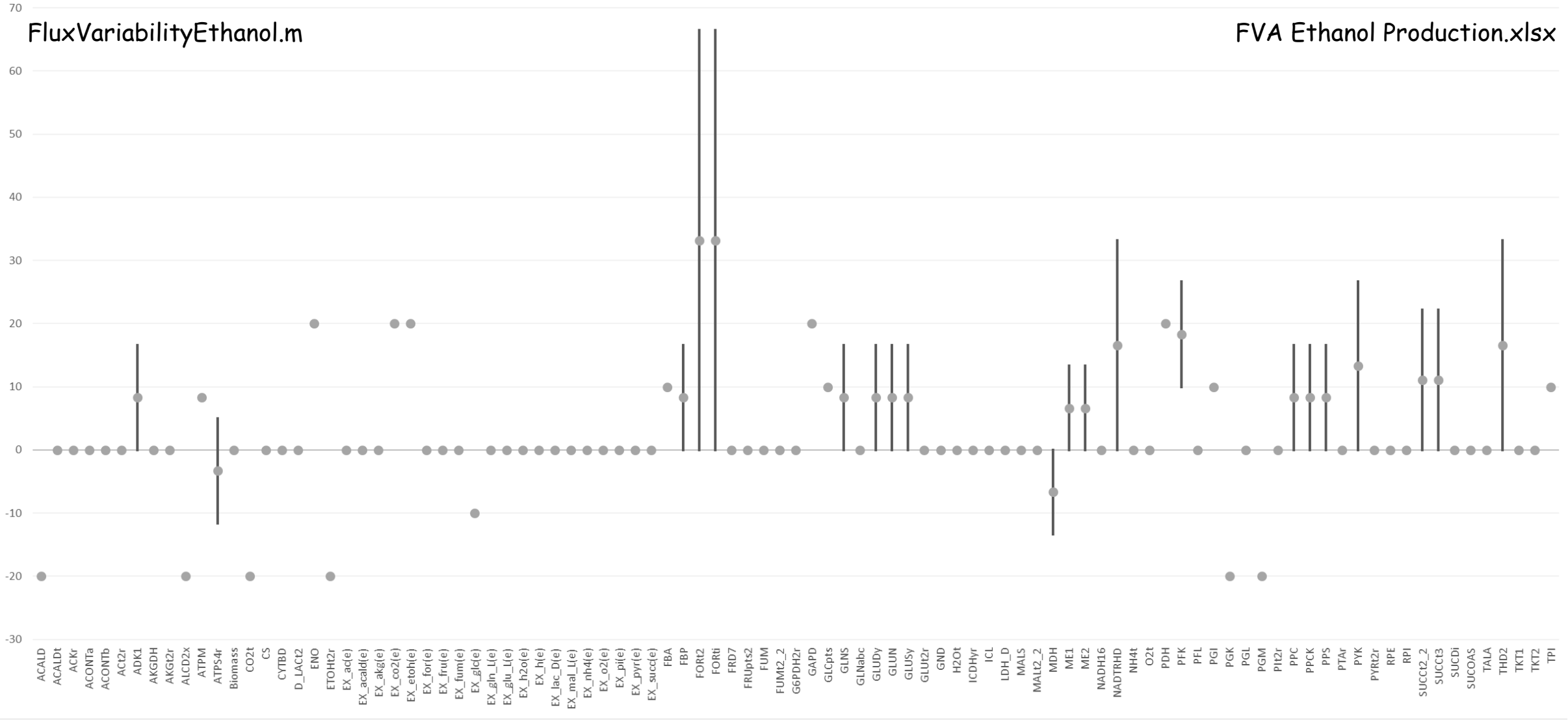
Alternate Optimal Flux Vectors for Ethanol Production

```
% findingOptimalSolutionsEthanol.m
clear;

model = readCbModel('ecoli_core_model.mat');

model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');

model = changeObjective(model,'EX_etoh(e)');

solverOK = changeCobraSolver('glpk');

% List optimal solutions
[solutions] = enumerateOptimalSolutions(model);
```

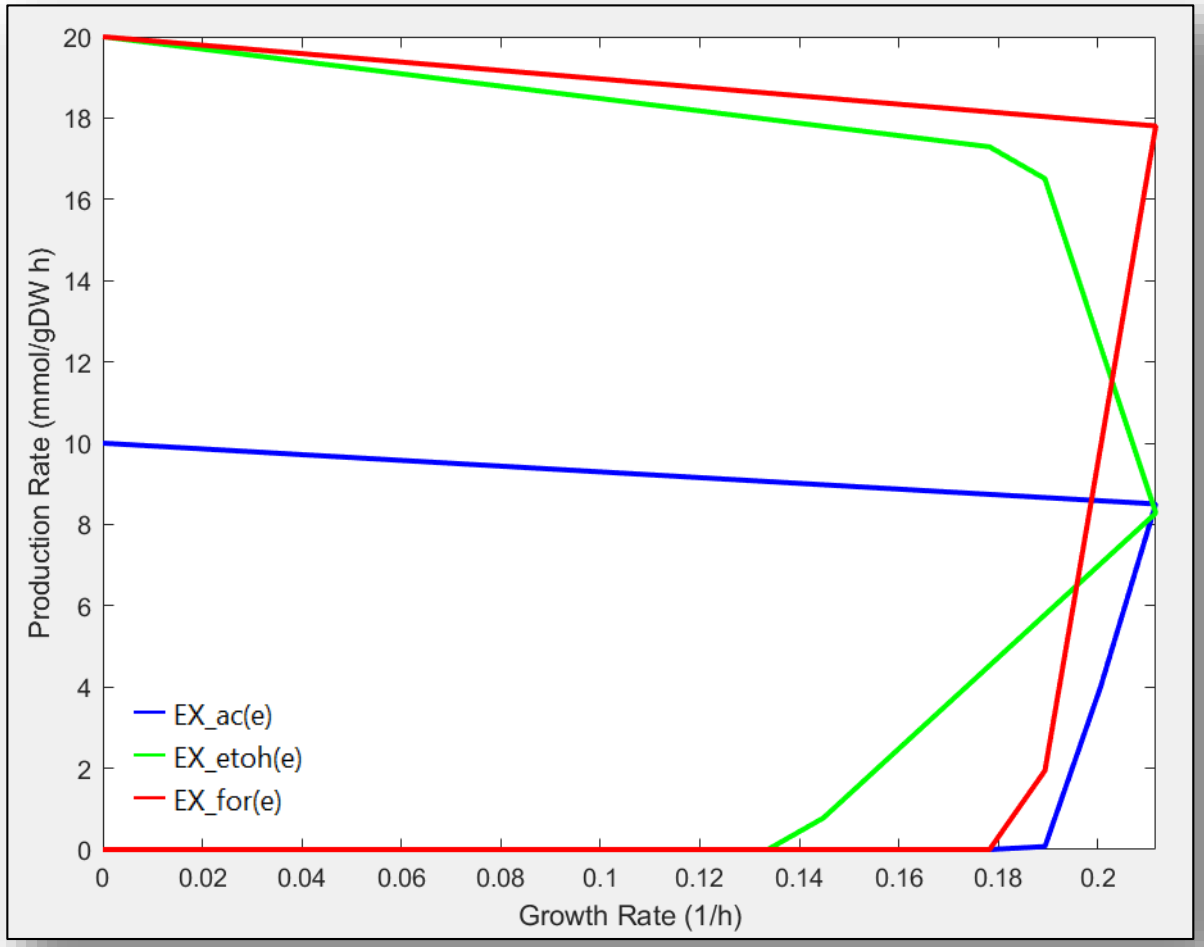
69 Alternate Optimal Flux Vectors



FVA Chart for Glucose-based Anaerobic Ethanol Production



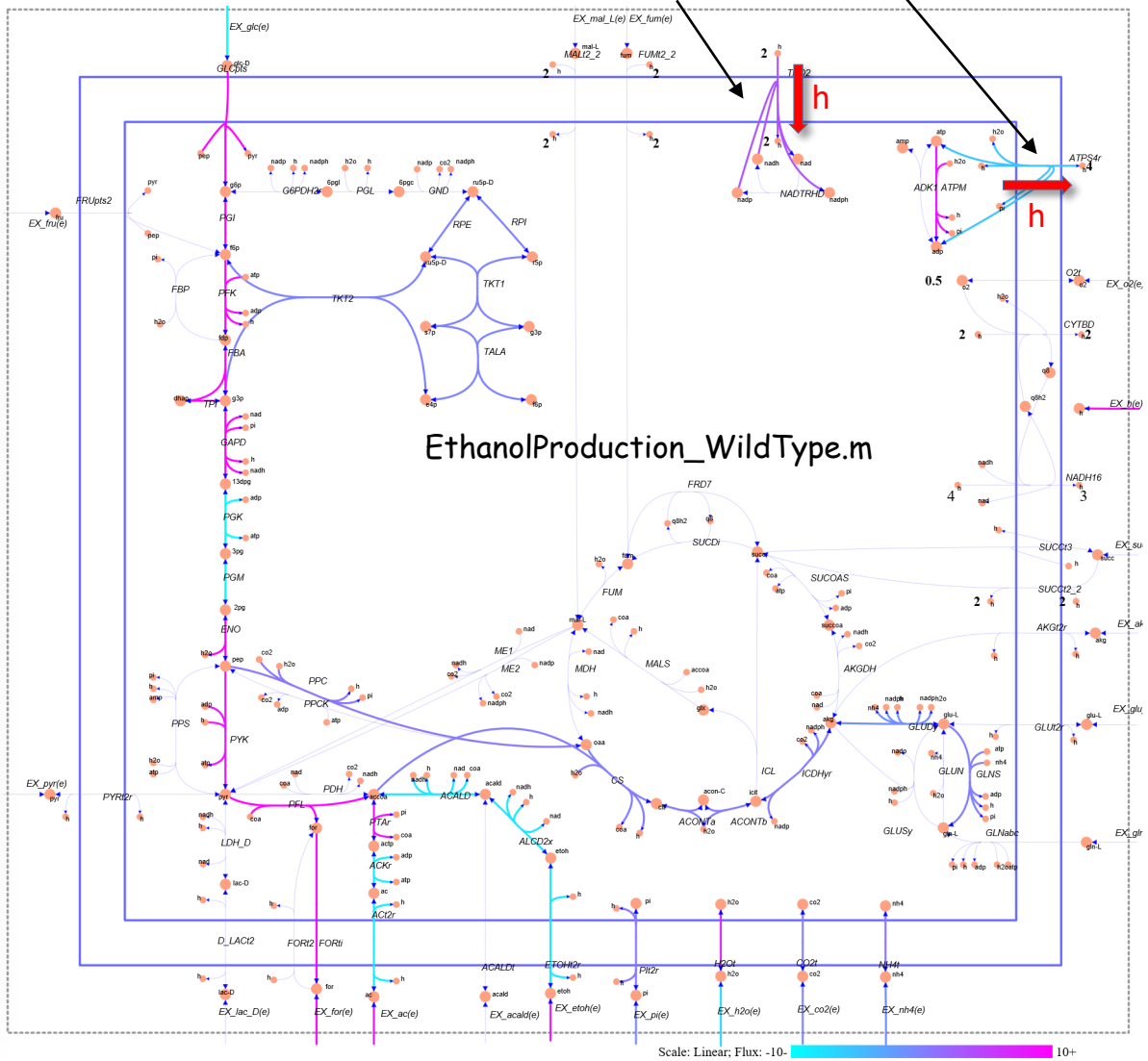
Ethanol Production: Wild Type



EthanolProduction_multiProductionEnvelope.m

Pumps protons to create protomotive force

Creates nad[c] and nadph[c]



Enhanced OptKnock Example: Ethanol Production

(EthanolProduction_OptKnock.m)

```
% EthanolProduction_OptKnock_Gurobi_Revised.m
clear; % Input the E.coli core model
load('ecoli_textbook.mat');
% Set carbon source and oxygen uptake rates
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)_Nmet2');

% Select reactions to be explored by the optKnock algorithm
[transRxns,nonTransRxns] = findTransRxns(model,true);
[tmp,ATPMnumber] = ismember('ATPM',nonTransRxns); % Identify ATPM reaction number
[tmp,BioMassnumber] = ismember('Biomass_Ecoli_core_N(w/GAM)_Nmet2',nonTransRxns); % Identify biomass reaction number
nonTransRxnsLength = length(nonTransRxns); % Find number of non-transport reactions
selectedRxns = {nonTransRxns{[1:ATPMnumber-1, ATPMnumber+1:BioMassnumber-1, BioMassnumber+1:nonTransRxnsLength]}};

% Optknock analysis for Ethanol secretion
options.targetRxn = 'EX_etoh(e)';
options.vMax = 1000;
options.numDel = 3;
options.numDelSense = 'L';
constrOpt.rxnList = {'Biomass_Ecoli_core_N(w/GAM)_Nmet2','ATPM'};
constrOpt.values = [0.14, 8.39];
constrOpt.sense = 'GE';

optKnockSol = OptKnock(model, selectedRxns, options, constrOpt);
deletions = optKnockSol.rxnList'

% Print out growth rate and minimum & maximum secretion rate
[growthRate,minProd,maxProd] = testOptKnockSol(model,'EX_etoh(e)',optKnockSol.rxnList)
```

deletions = 'ACKr','G6PDH2r','GLUDy'

growthRate = 0.1772

minProd = 17.31; maxProd = 17.31

Candidate Reactions to be knocked out (want to keep as small as possible)

Exchange reaction to be maximized

Maximum flux

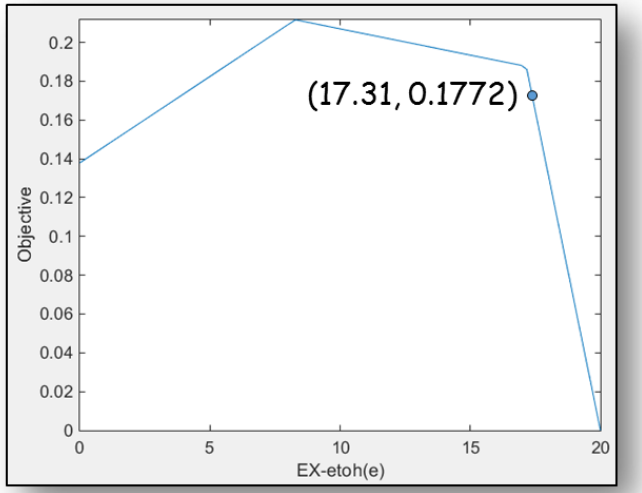
Maximum number of knockouts

Reactions to be constrained

Values of constraints

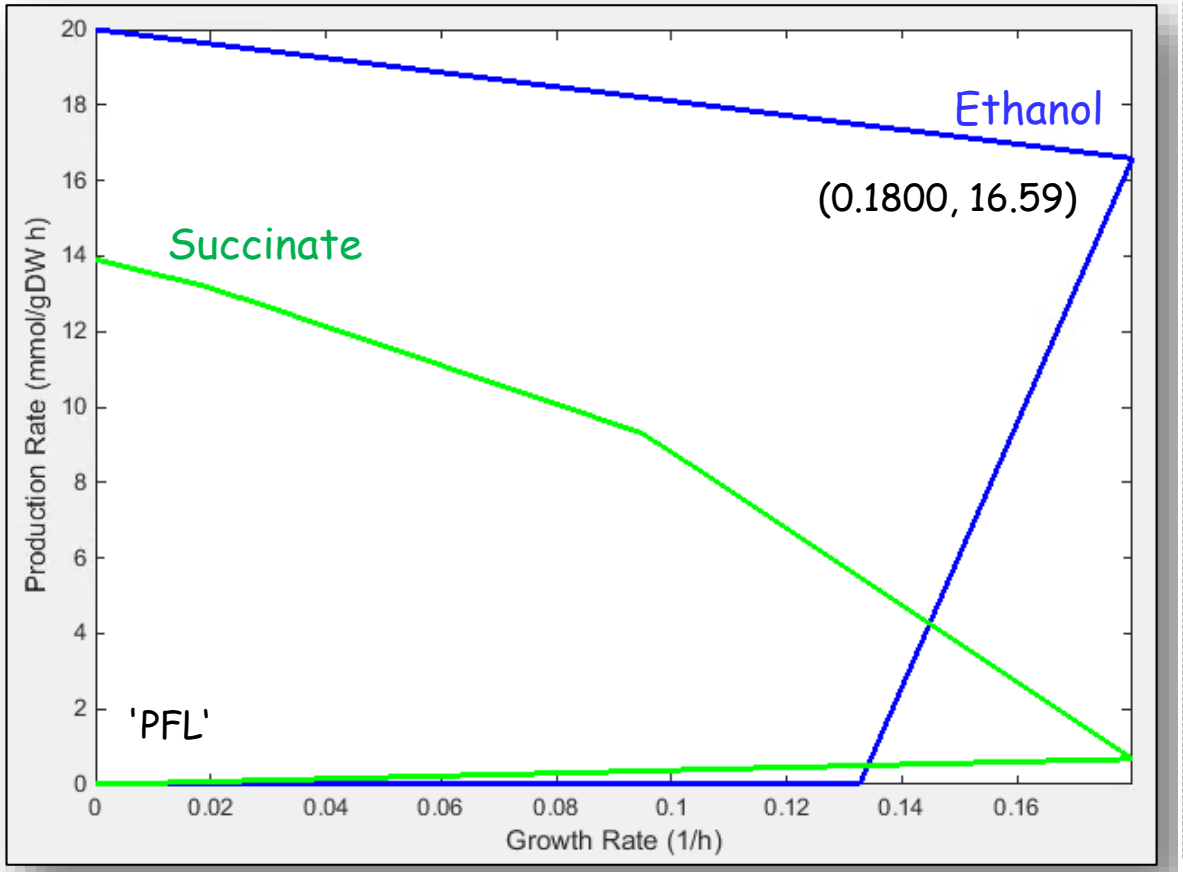
Condition for constraints

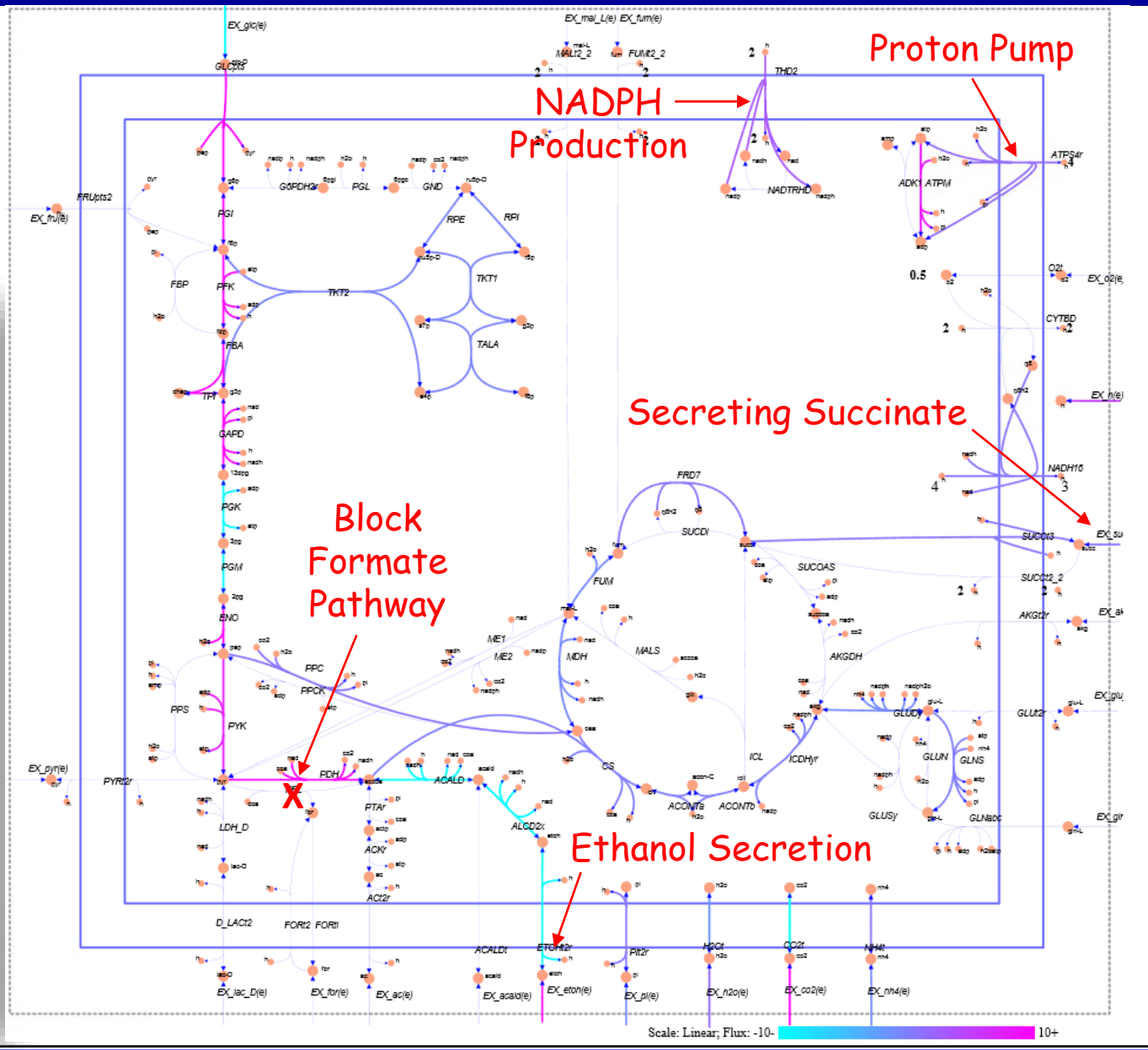
OptKnock function



1 Knockout

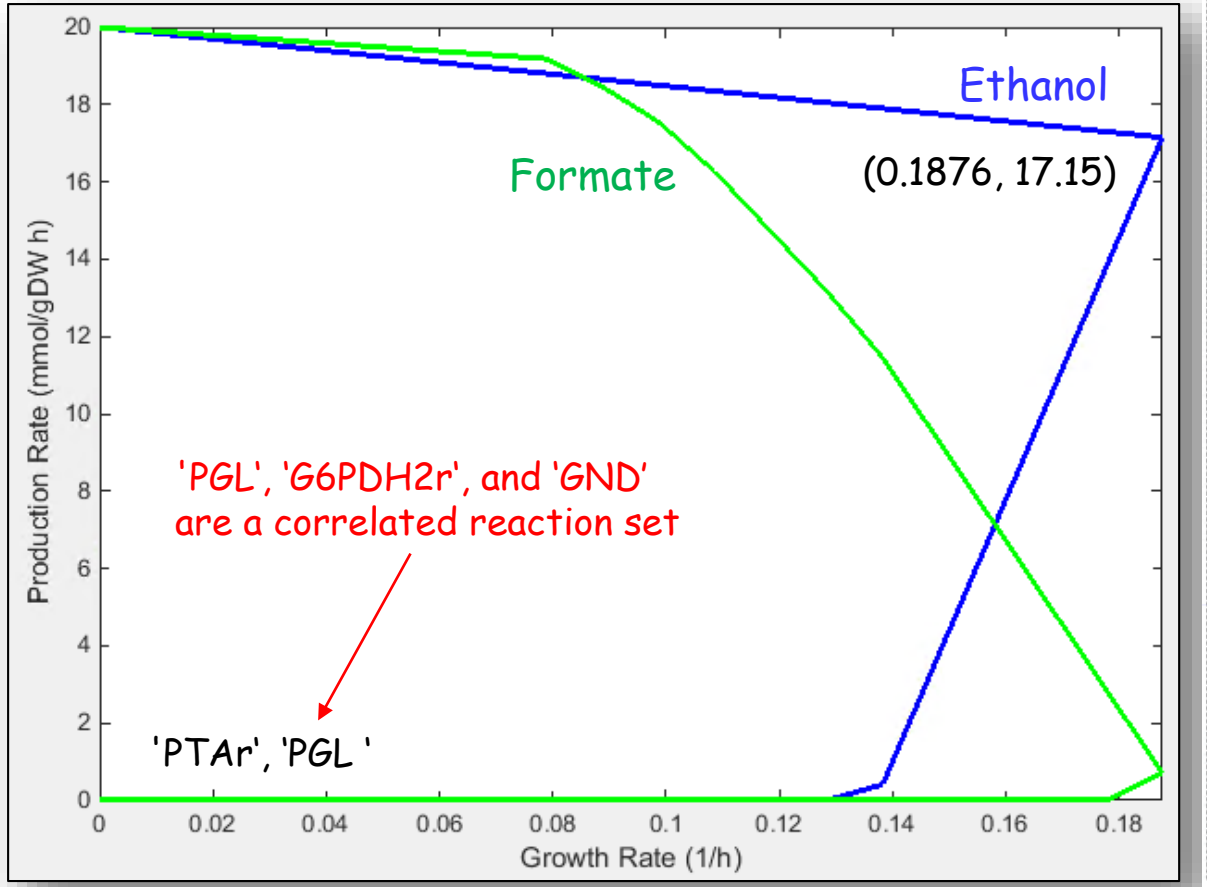
EthanolProduction_Mutant.m
EthanolProduction_multiProductionEnvelope.m

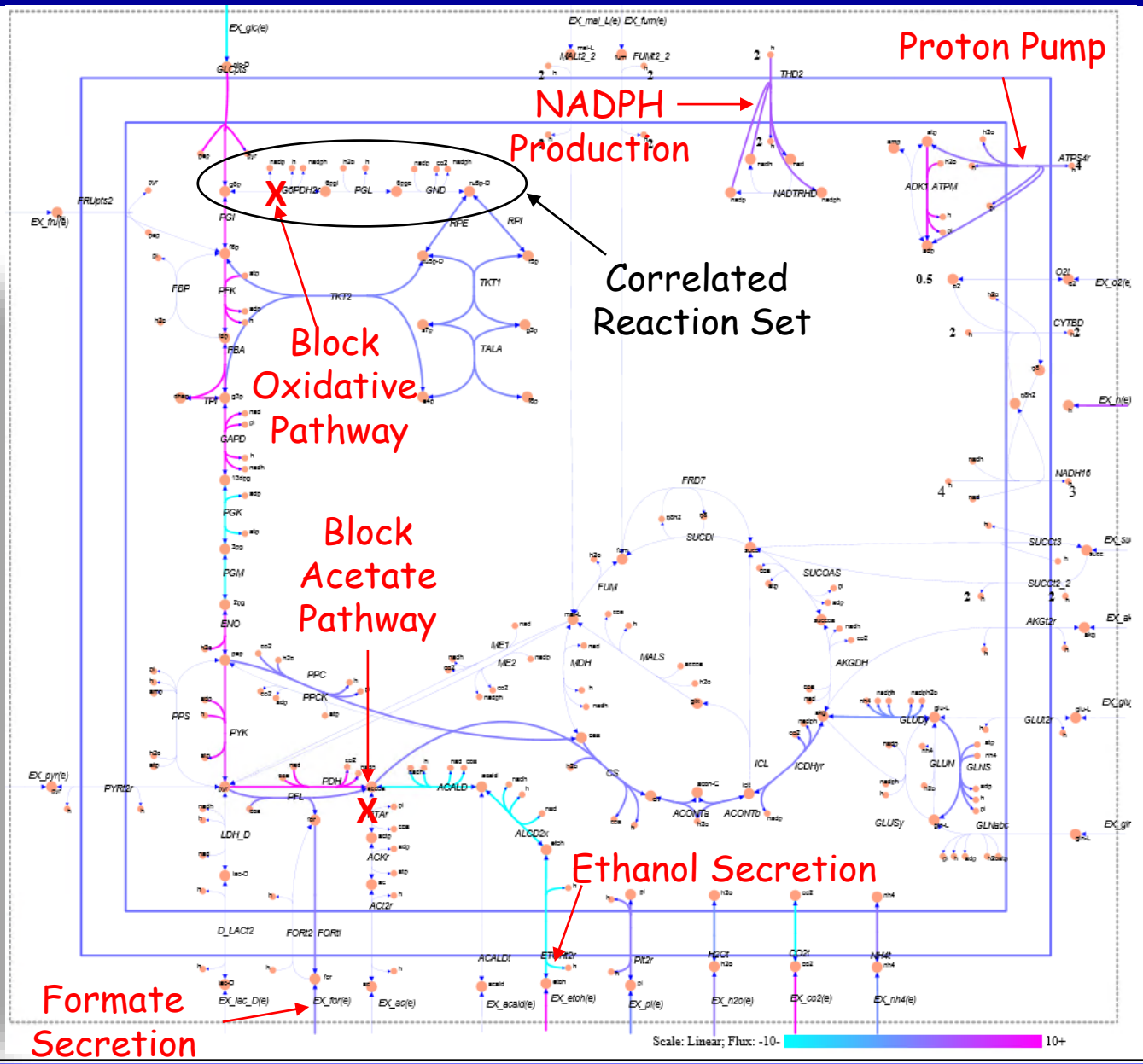




2 Knockouts

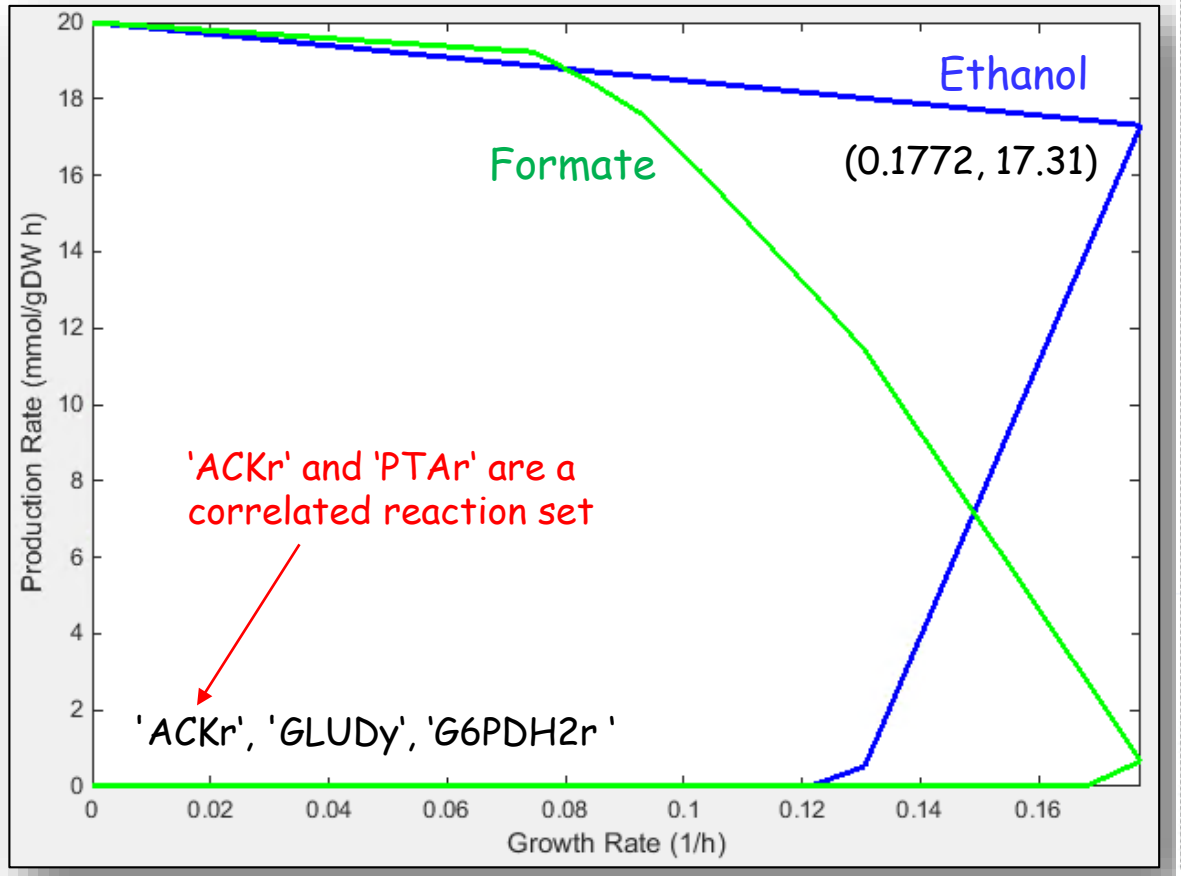
EthanolProduction_Mutant.m
EthanolProduction_multiProductionEnvelope.m

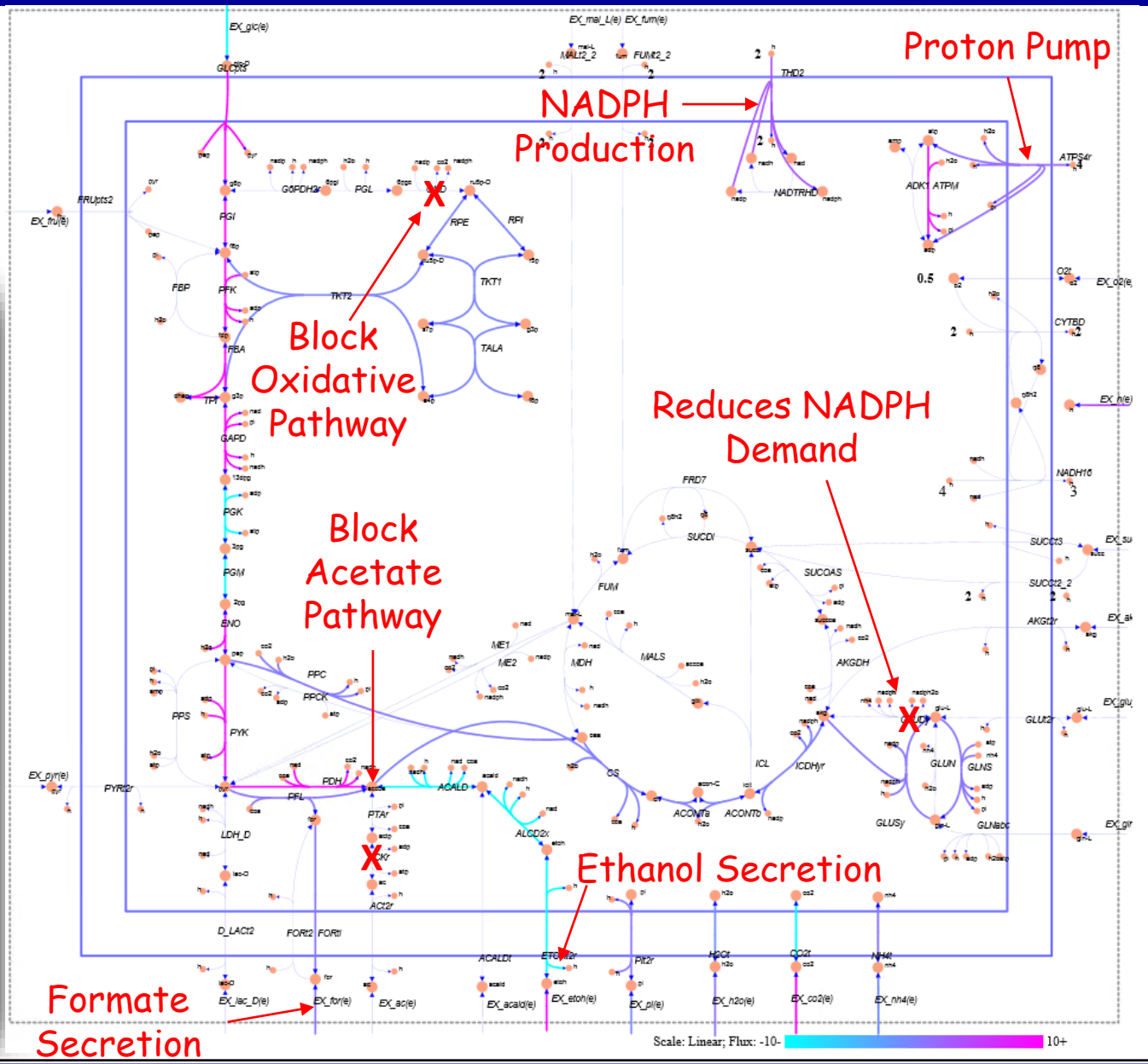




3 Knockouts

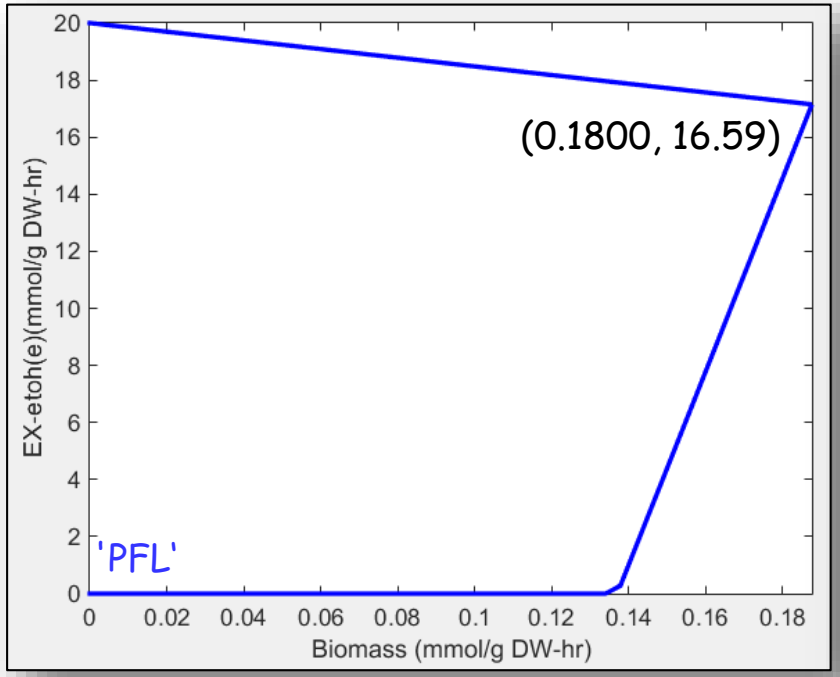
EthanolProduction_Mutant.m

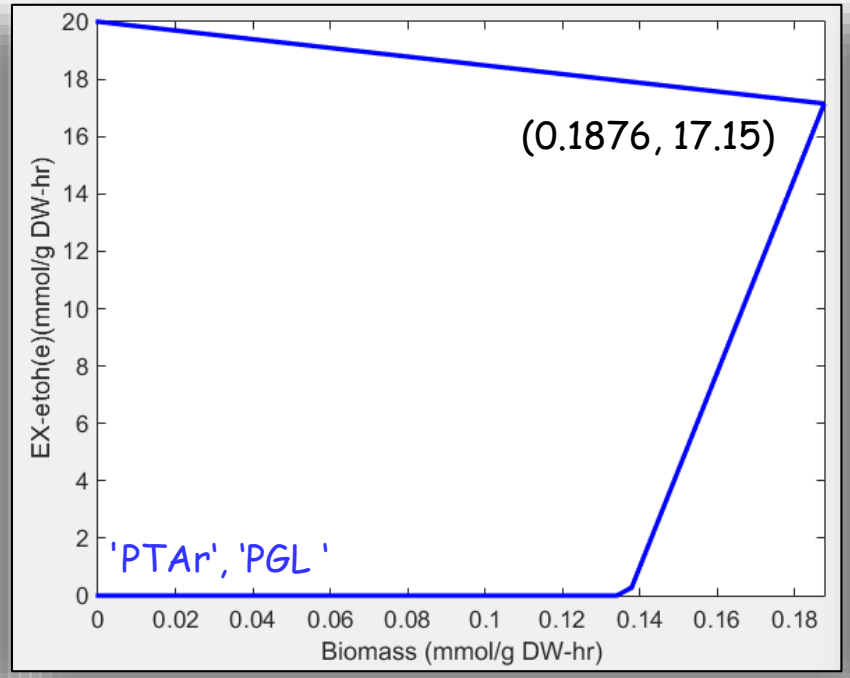


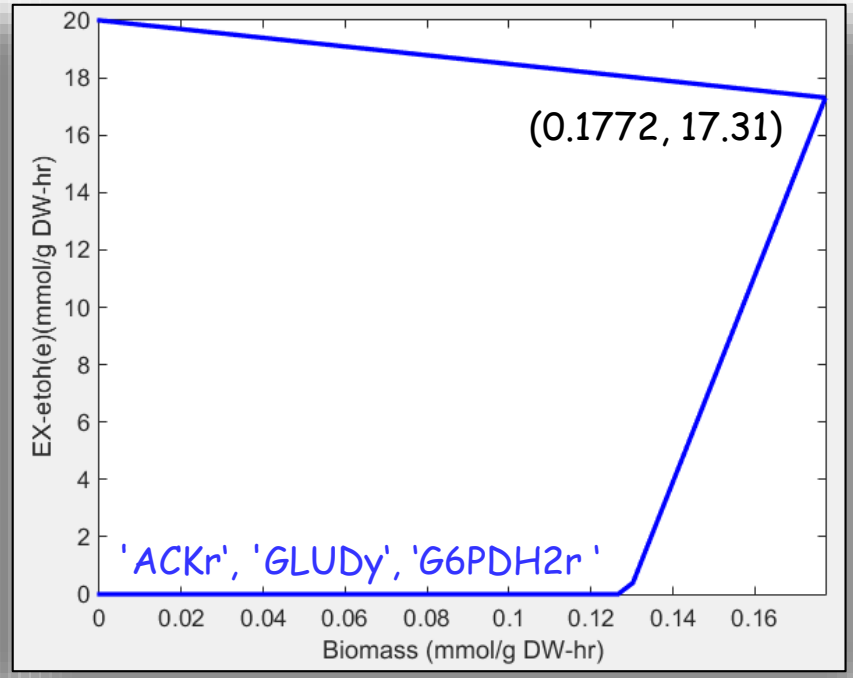


Production Envelopes for Different Ethanol Producing Knockouts

Maximum Growth-rate ≥ 0.14



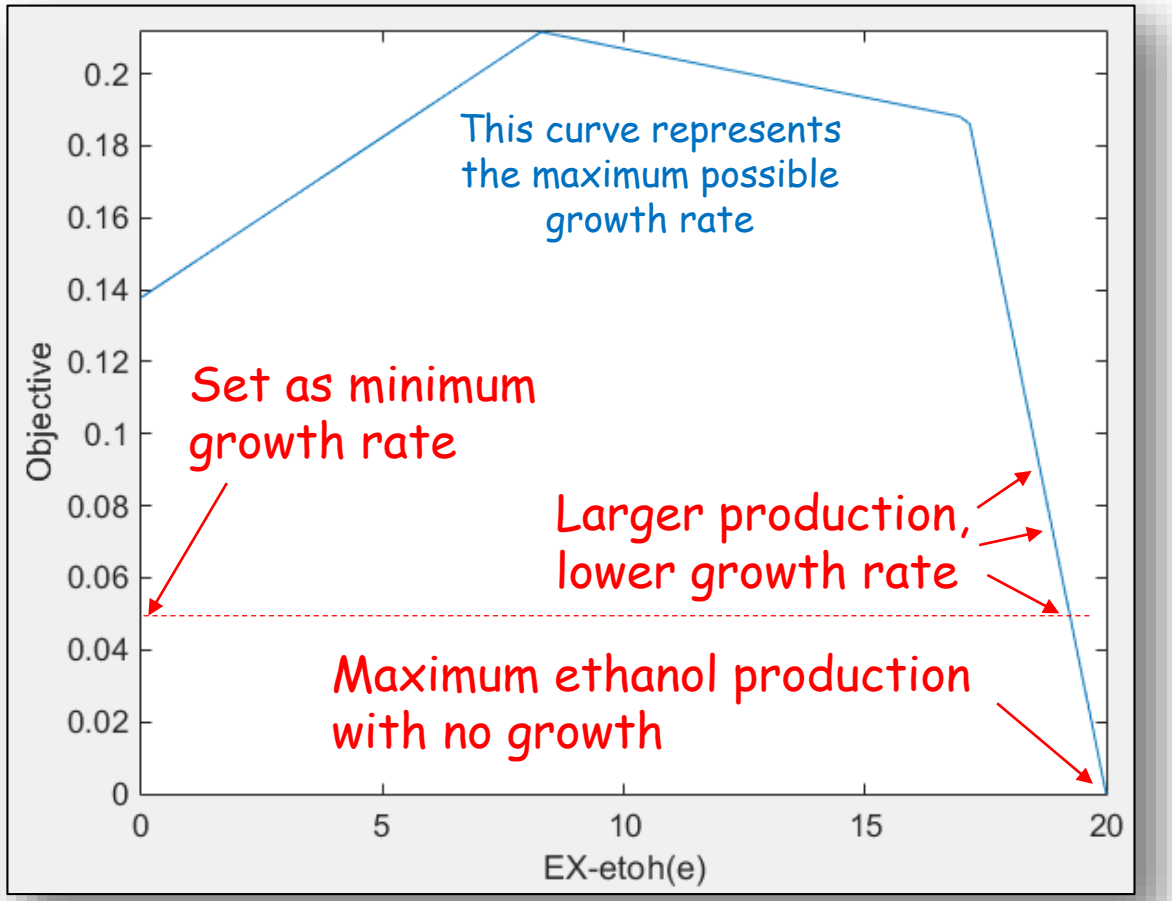




EthanolProduction_OptKnock.m

Can we Increase the Ethanol Production?

```
% AnaerobicEthanolRA.m

clear; clc;

% Input the E.coli core model

model = readCbModel('ecoli_core_model.mat');

% Set uptake rates

model = changeRxnBounds(model,'EX_glc(e)',-10,'b');

model = changeRxnBounds(model,'EX_o2(e)',-0,'b');

% Set optimization objective

model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Using robustnessAnalysis, plot the objective function as a function

% of the ethanol secretion rate

[controlFlux, objFlux] = robustnessAnalysis(model,'EX_etoh(e)',100);
```



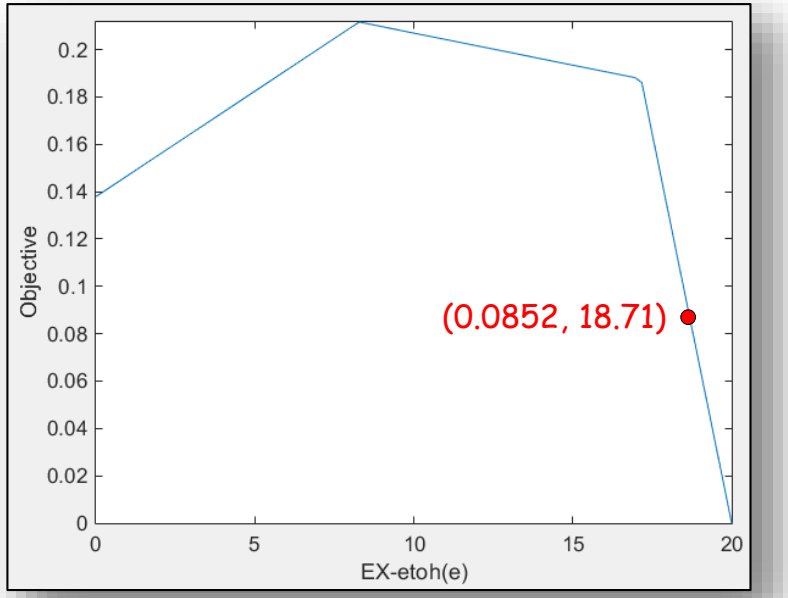
Robustness Analysis

OptKnock Example: Maximizing Ethanol Production

(EthanolProduction_OptKnock.m)

```
% EthanolProduction_OptKnock.m
clear; clc;
% Input the E.coli core model
load('ecoli_textbook.mat');

% Set carbon source and oxygen uptake rates
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)_Nmet2');

% Select reactions to be explored by the optKnock algorithm
[transRxns,nonTransRxns] = findTransRxns(model,true);
[tmp,ATPMnumber] = ismember('ATPM',nonTransRxns); % Identify ATPM reaction number
[tmp,BioMassnumber] = ismember('Biomass_Ecoli_core_N(w/GAM)_Nmet2',nonTransRxns); % Identify biomass reaction number
nonTransRxnsLength = length(nonTransRxns); % Find number of non-transport reactions
selectedRxns = {nonTransRxns{[1:ATPMnumber-1, ATPMnumber+1:BioMassnumber-1, BioMassnumber+1:nonTransRxnsLength]}};

% Optknock analysis for Ethanol secretion
options.targetRxn = 'EX_etoh(e)';
options.vMax = 1000;
options.numDel = 5;
options.numDelSense = 'L';
constrOpt.rxnList = {'Biomass_Ecoli_core_N(w/GAM)_Nmet2','ATPM'};
constrOpt.values = [0.05, 8.39];
constrOpt.sense = 'GE';

optKnockSol = OptKnock(model, selectedRxns, options, constrOpt);
deletions = optKnockSol.rxnList'

% Print out growth rate and minimum & maximum secretion rate
[growthRate,minProd,maxProd] = testOptKnockSol(model,'EX_etoh(e)',optKnockSol.rxnList)
```

Reduce the growth rate to increase bioproduction (0.14 -> 0.05)



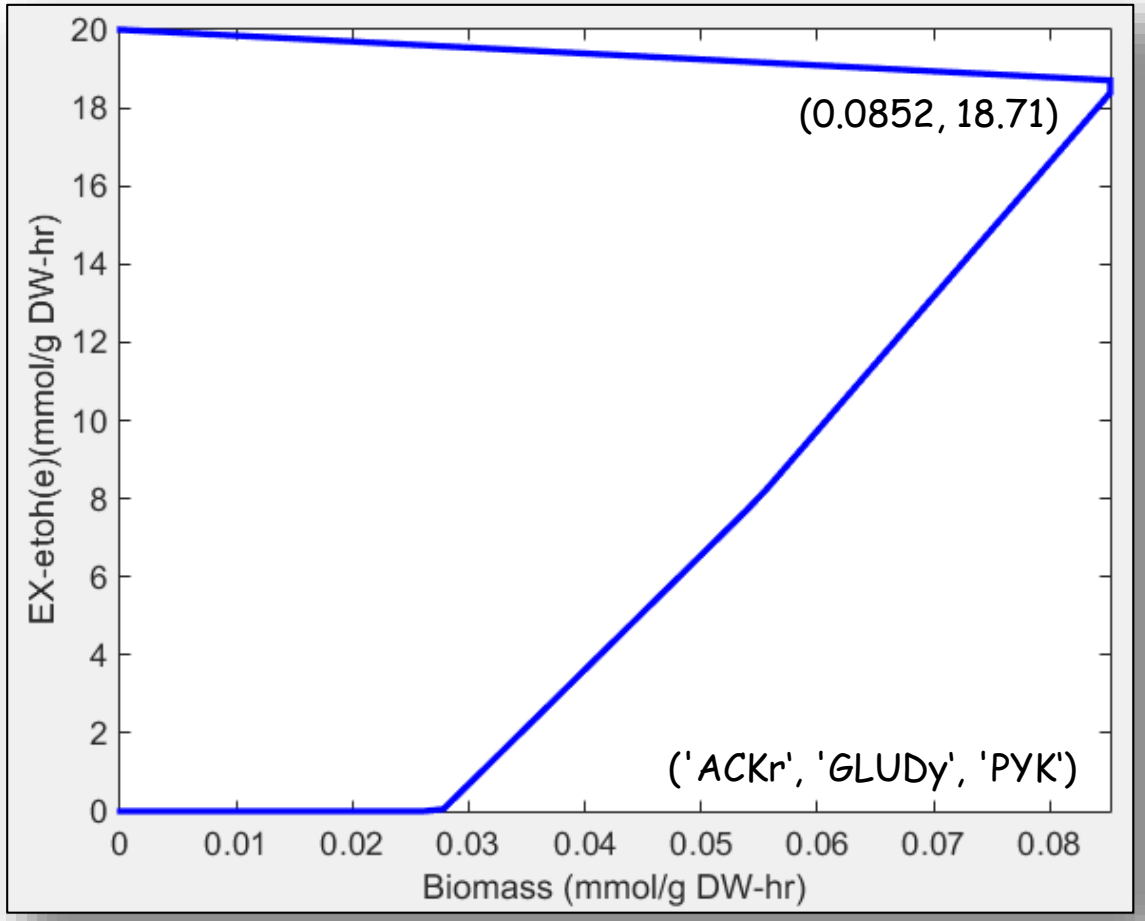
deletions = 'PTAr' , 'GLUDy', 'PYK'

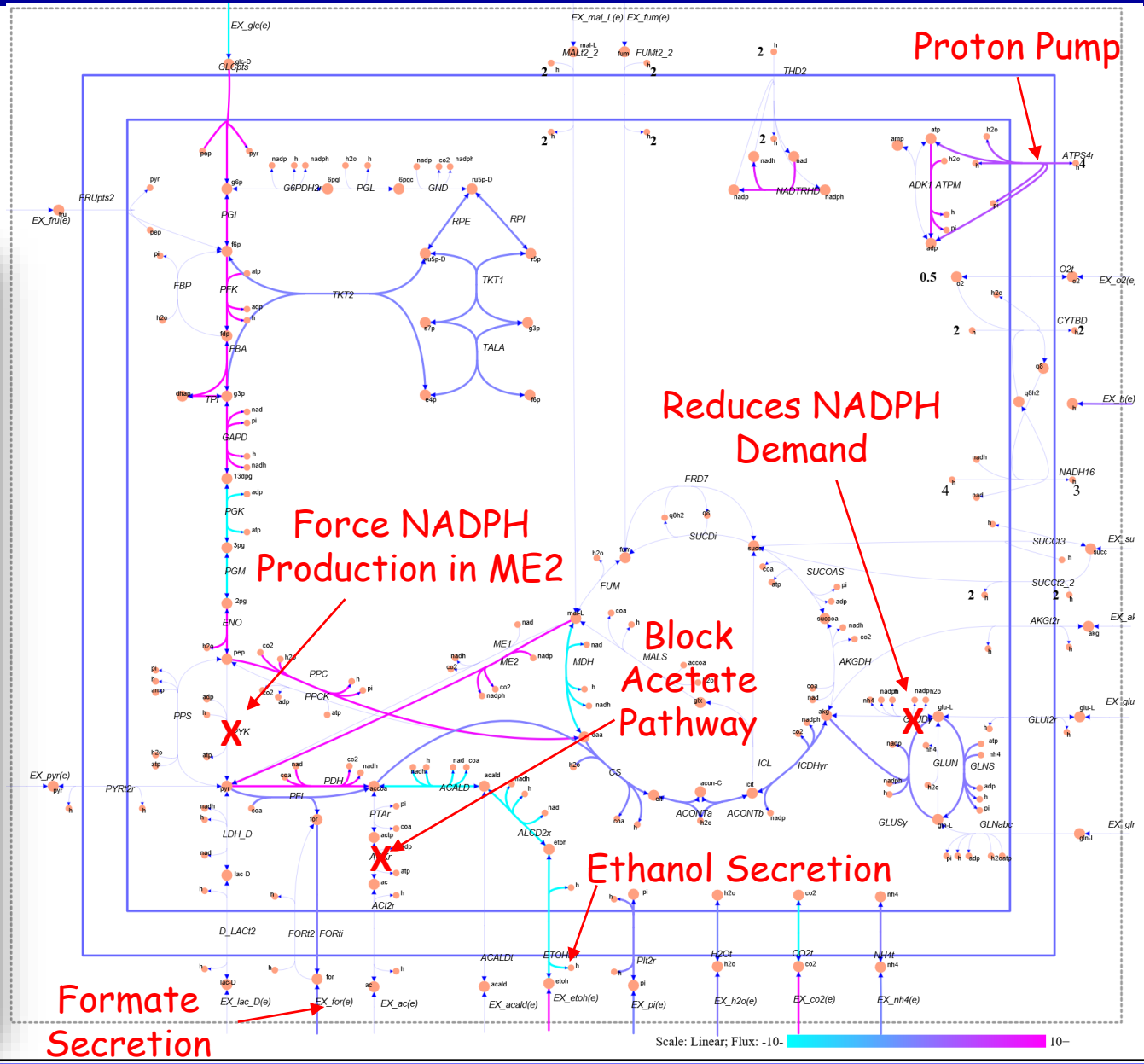
growthRate = 0.0852

minProd = 18.3845; maxProd = 18.7054

3 Knockouts

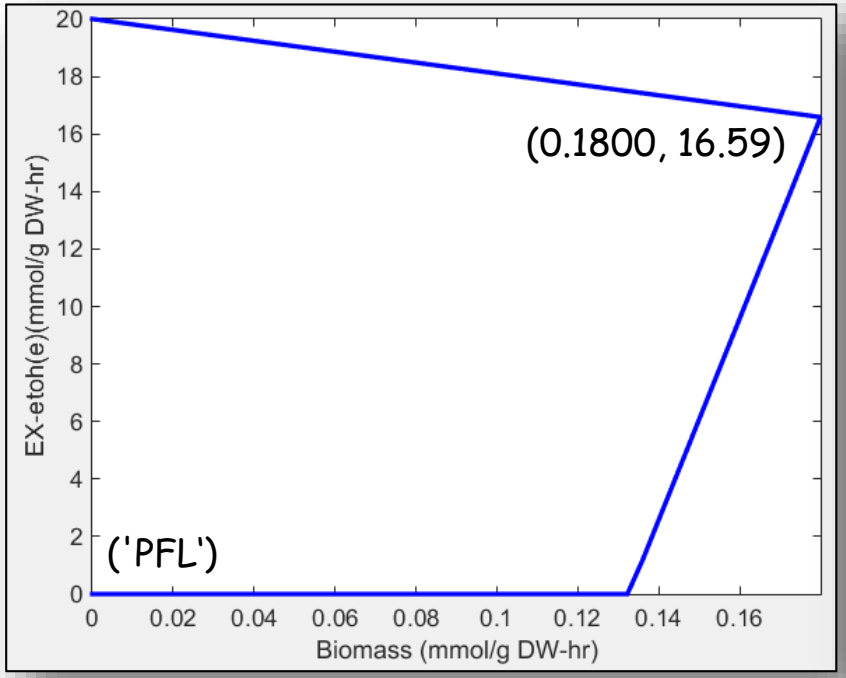
EthanolProduction_Mutant.m

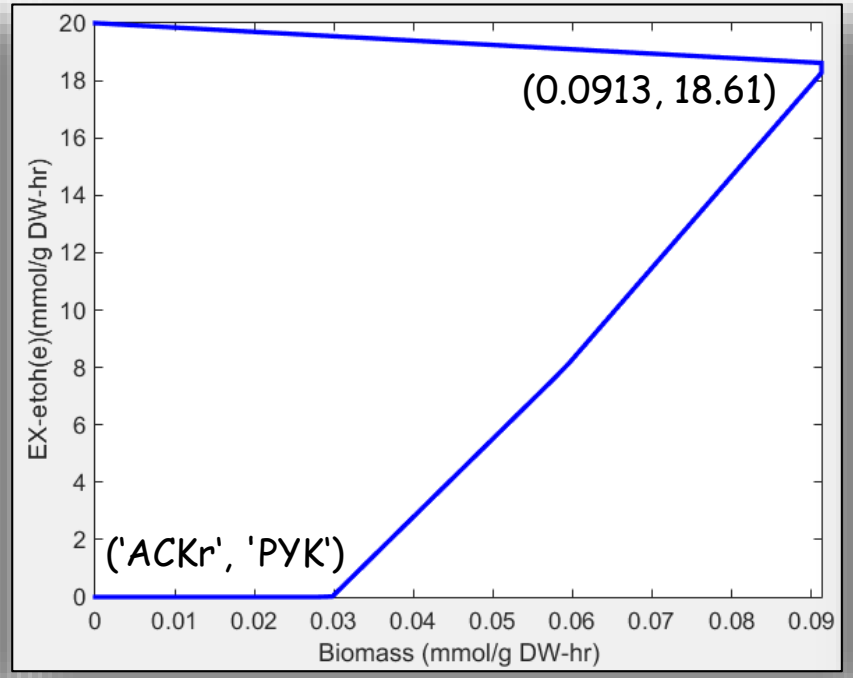


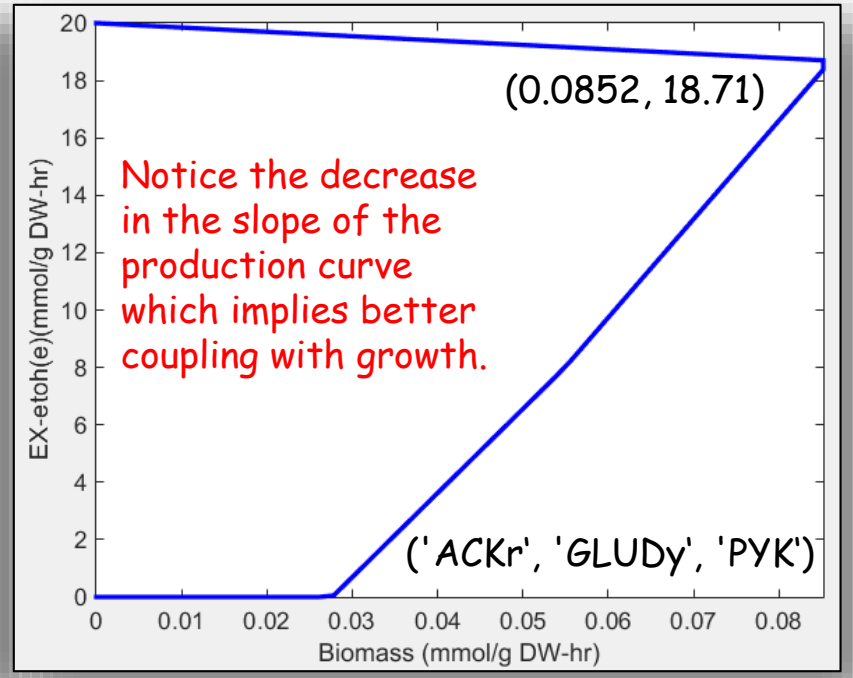


Production Envelopes for Different Ethanol Producing Knockouts

Maximum Growth-rate ≥ 0.05



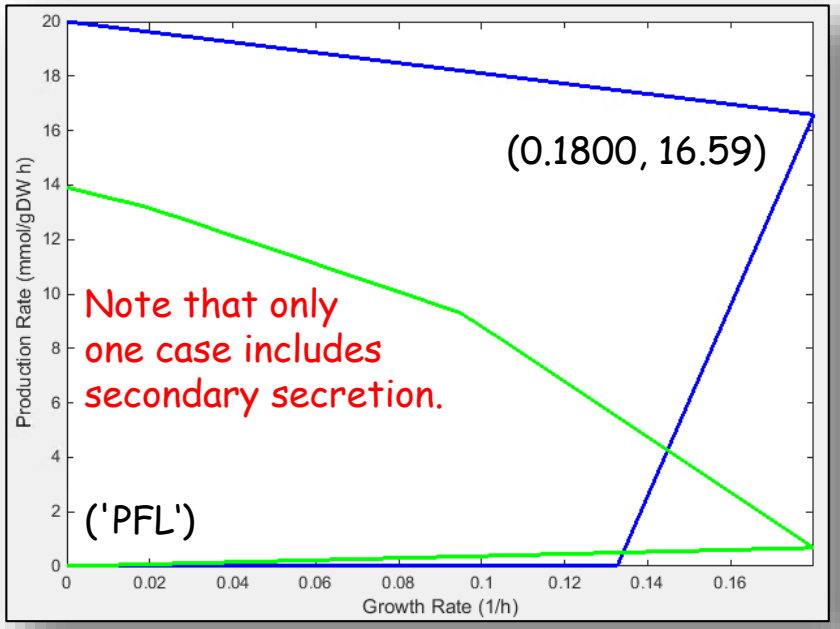


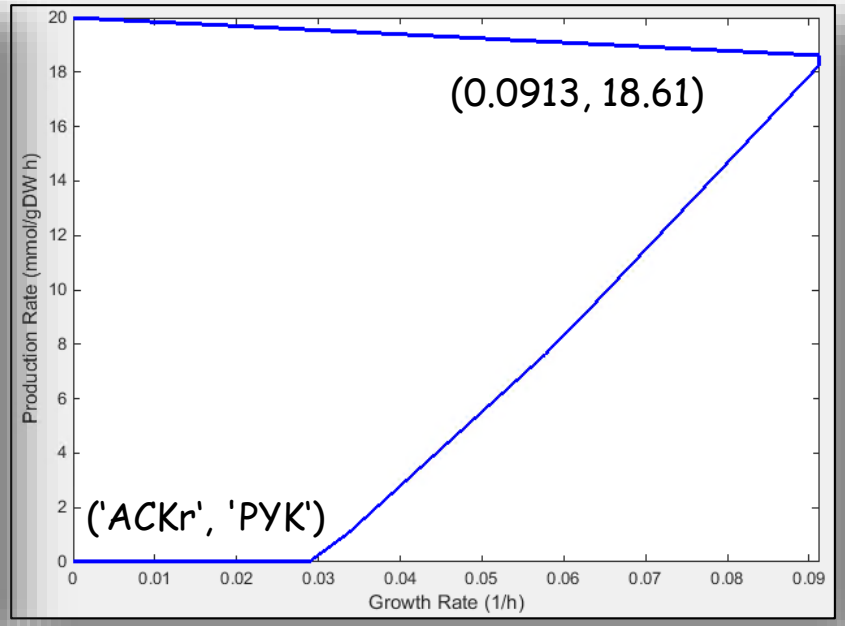


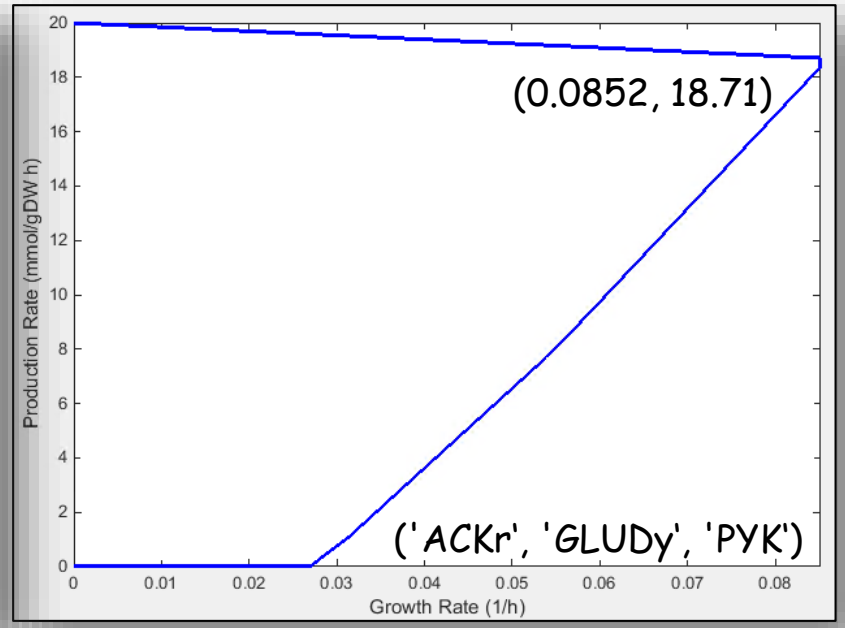
EthanolProduction_OptKnock.m

Multiproduction Envelopes for Different Knockouts

Maximum Growth-rate ≥ 0.05



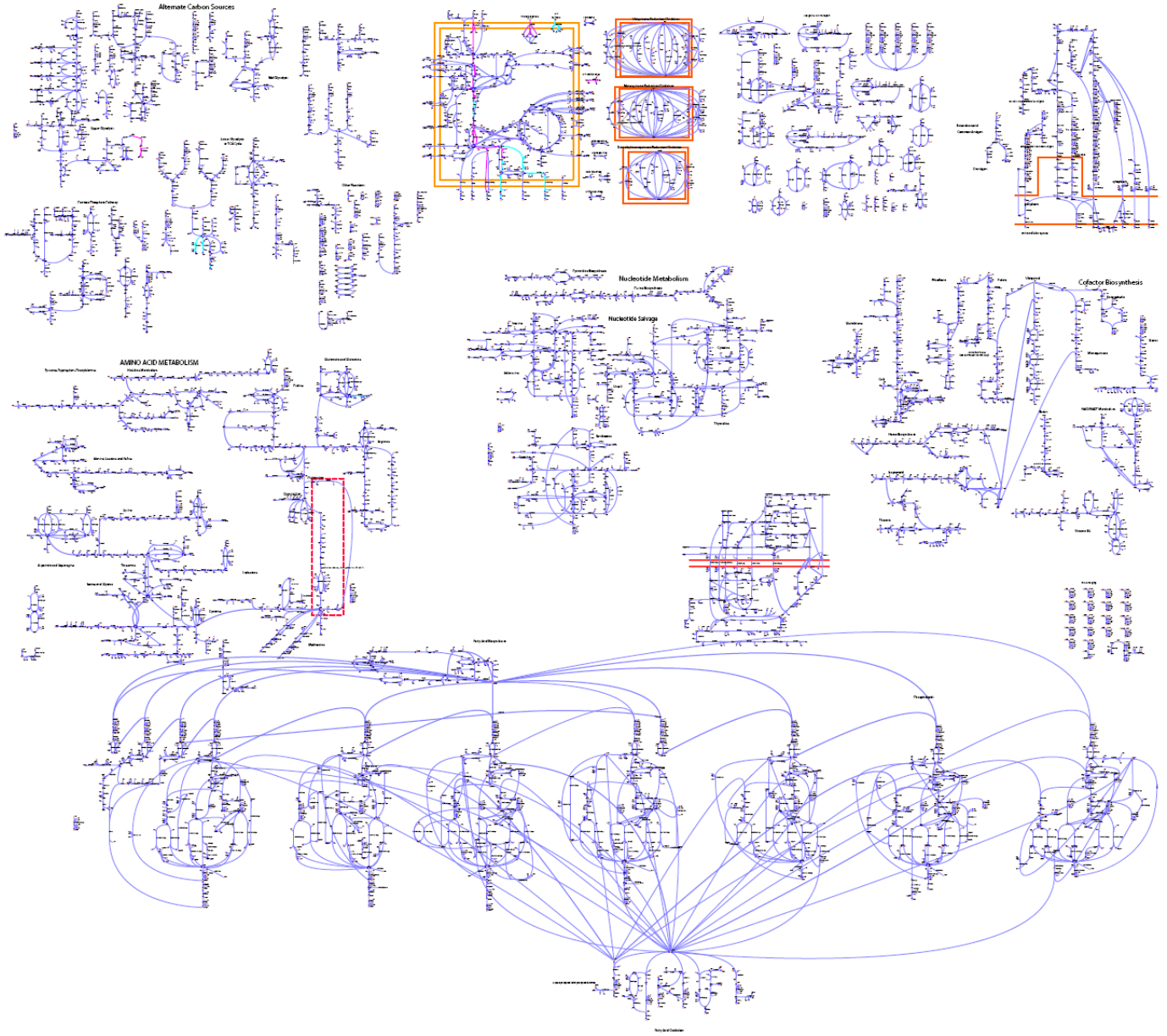




EthanolProduction_OptKnock.m (CPLEX solver)

iAF1260 Model

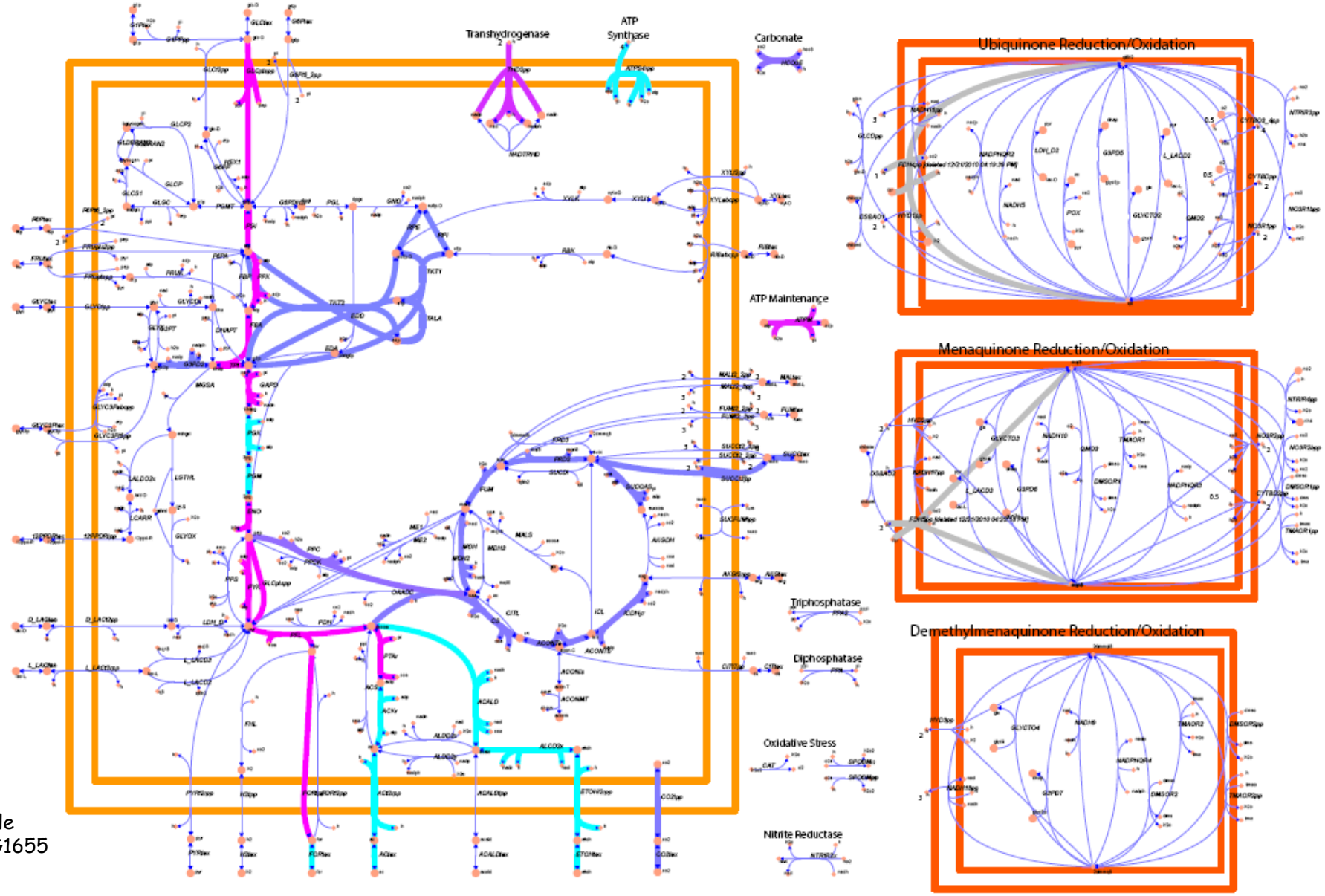
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

iAF1260 Metabolic Core

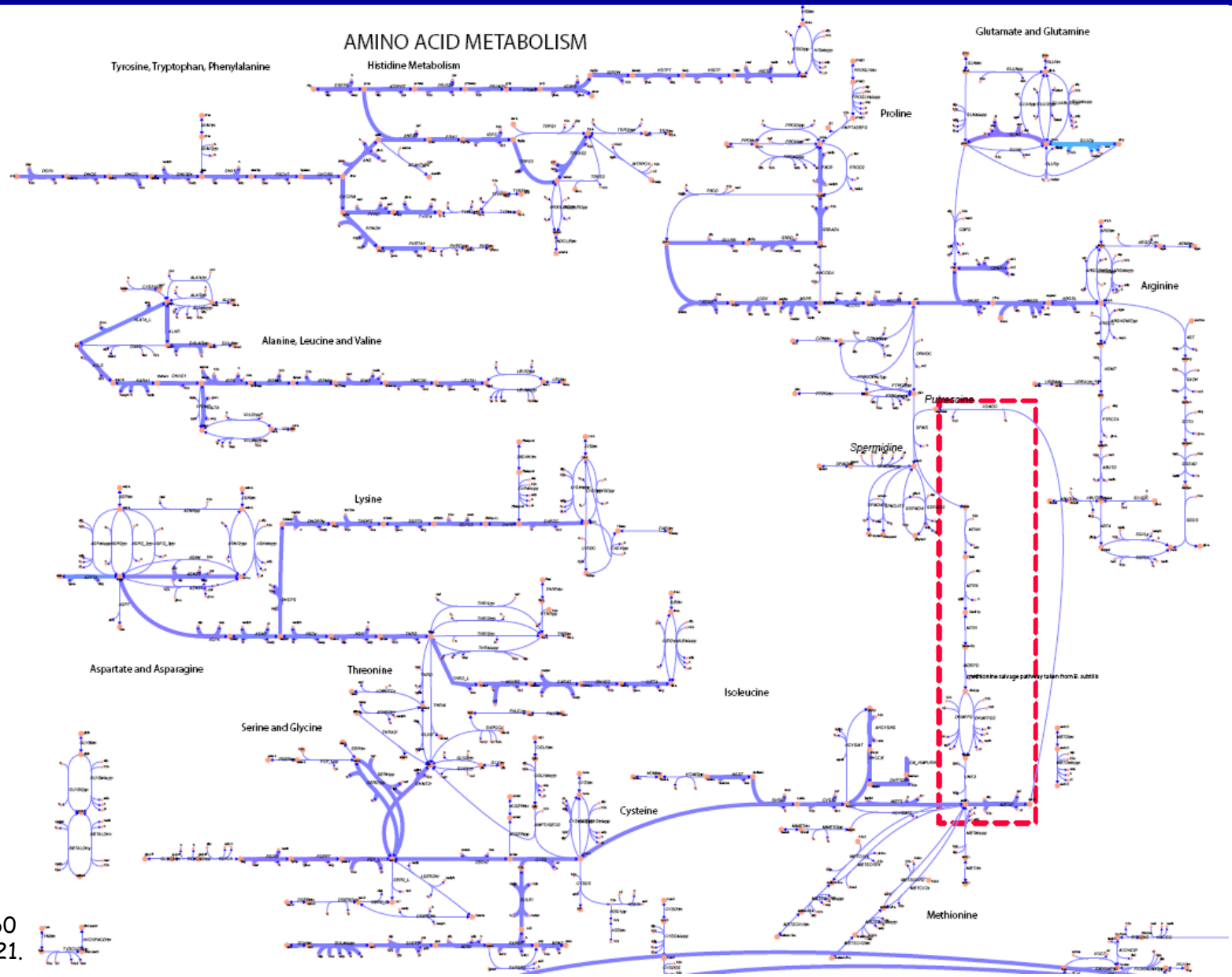
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

iAF1260 Amino Acid Metabolism

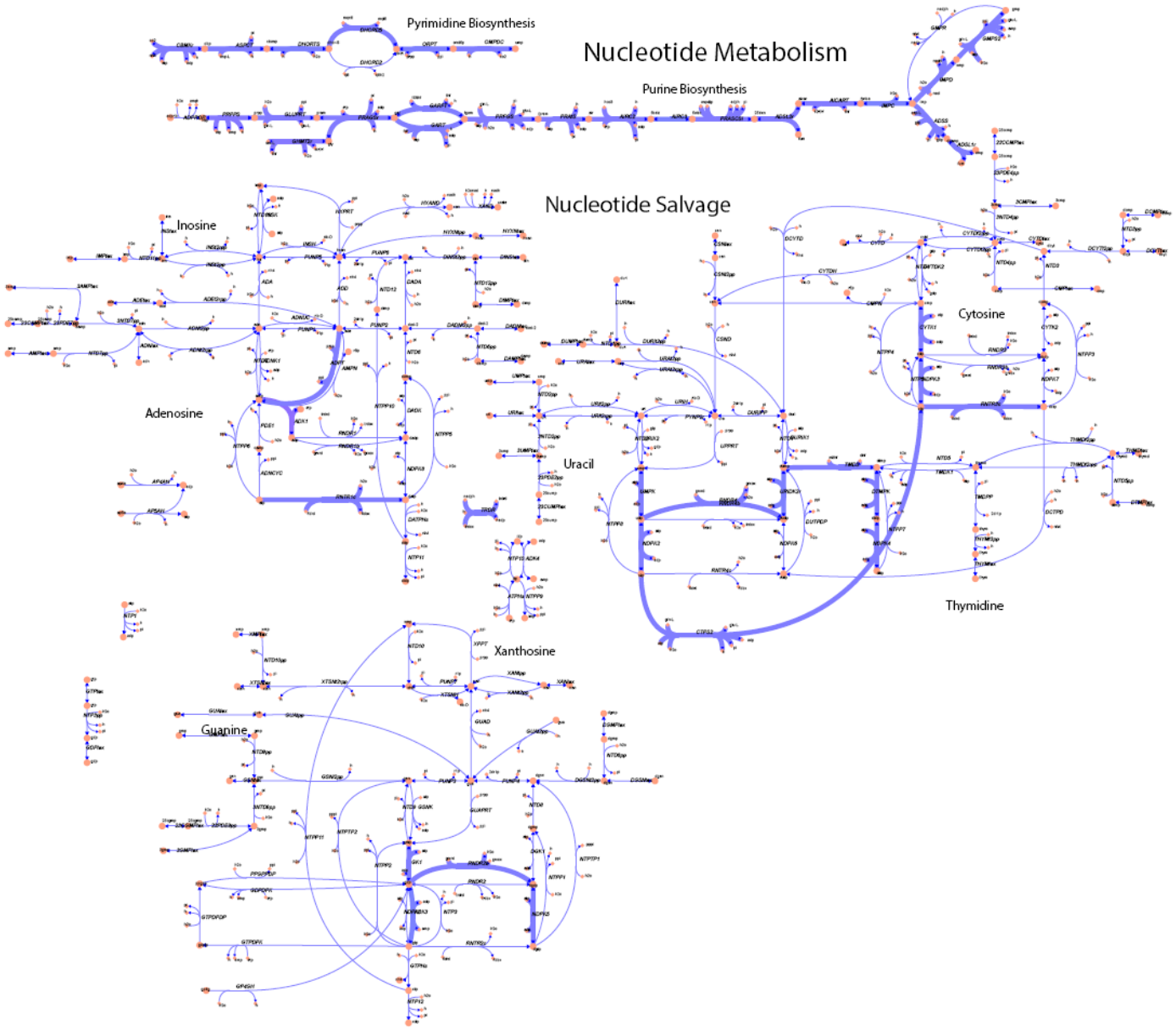
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

iAF1260 Necleotide Metabolism

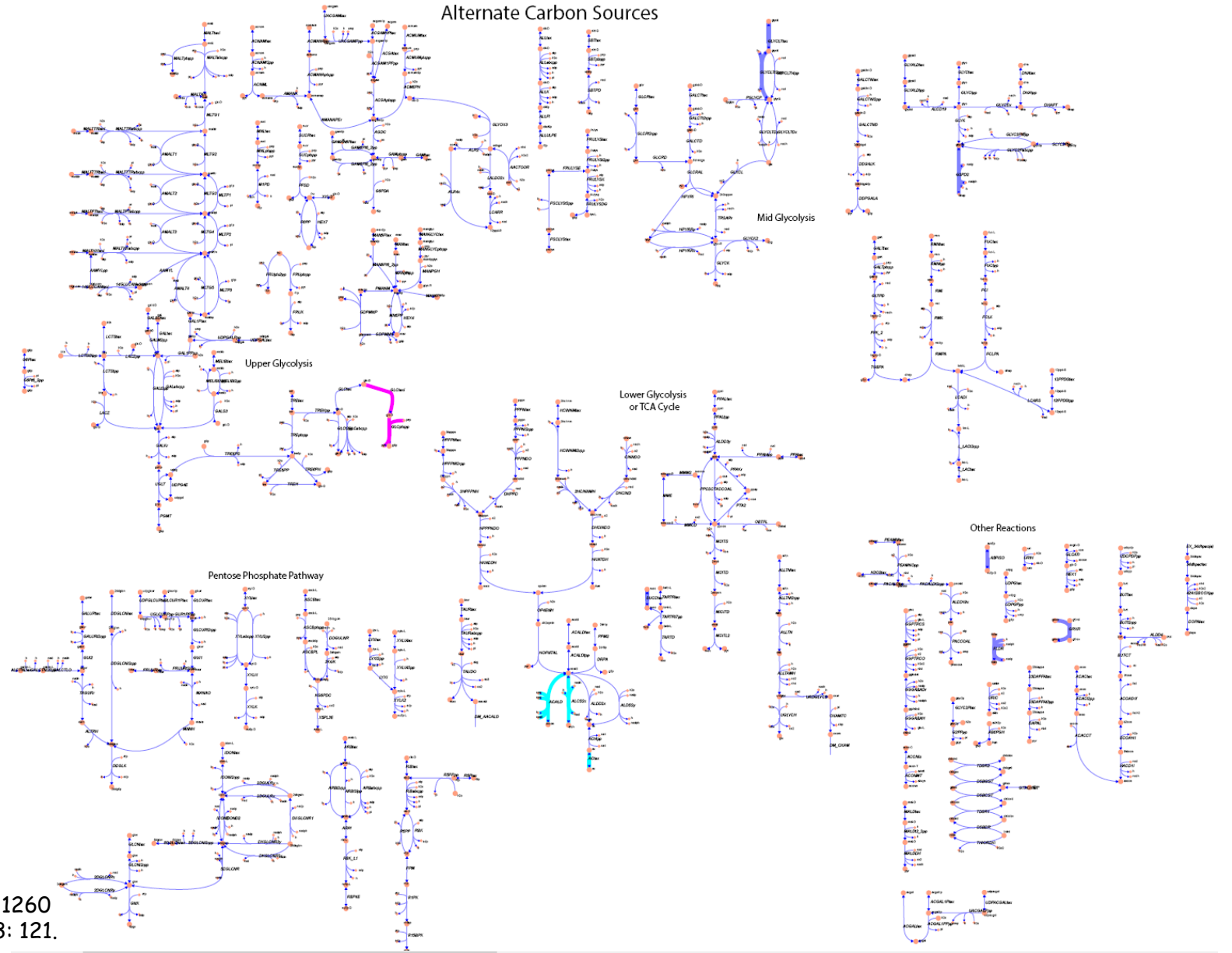
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

iAF1260 Alternate Carbon Sources

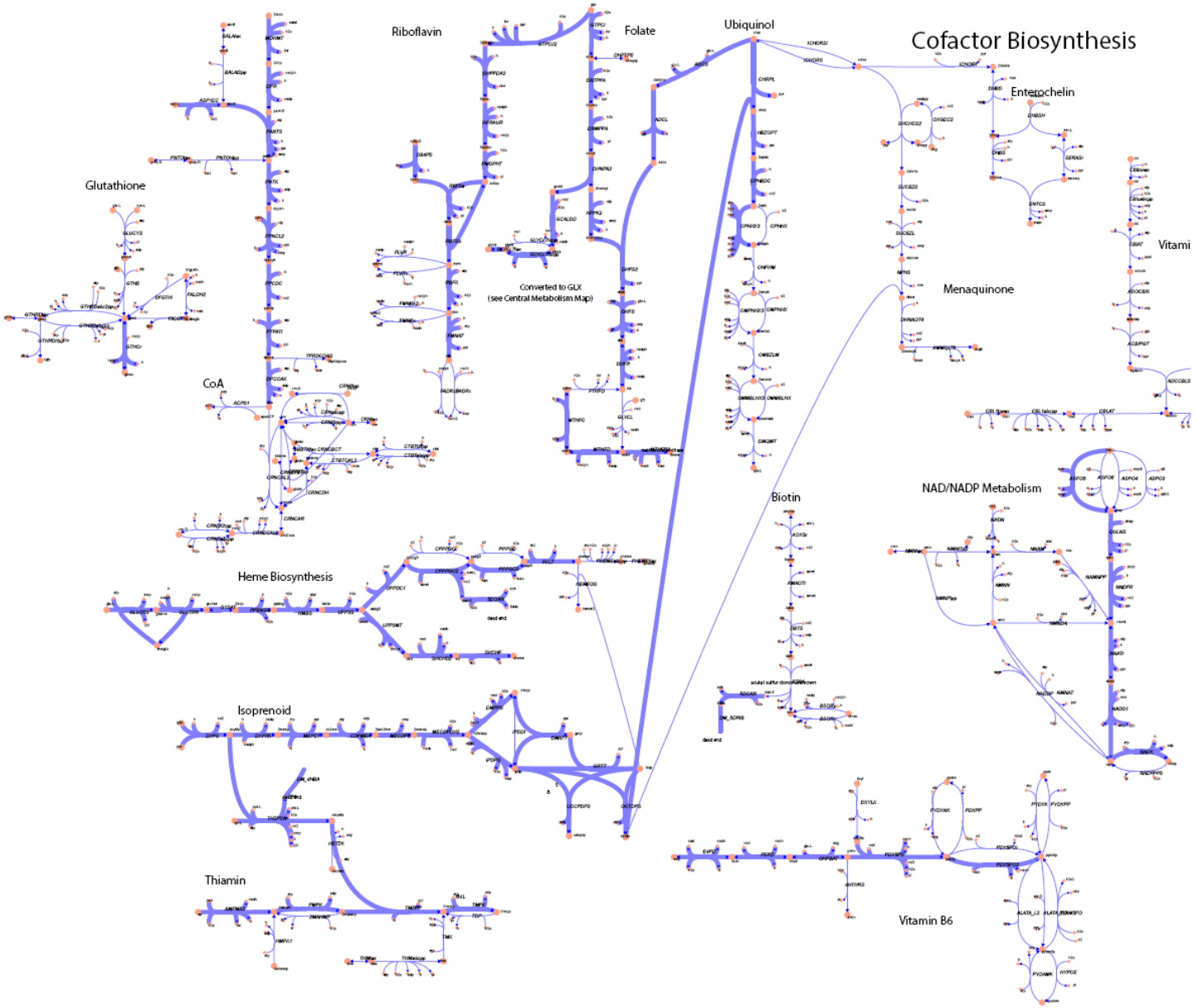
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

iAF1260 Cofactor Biosynthesis

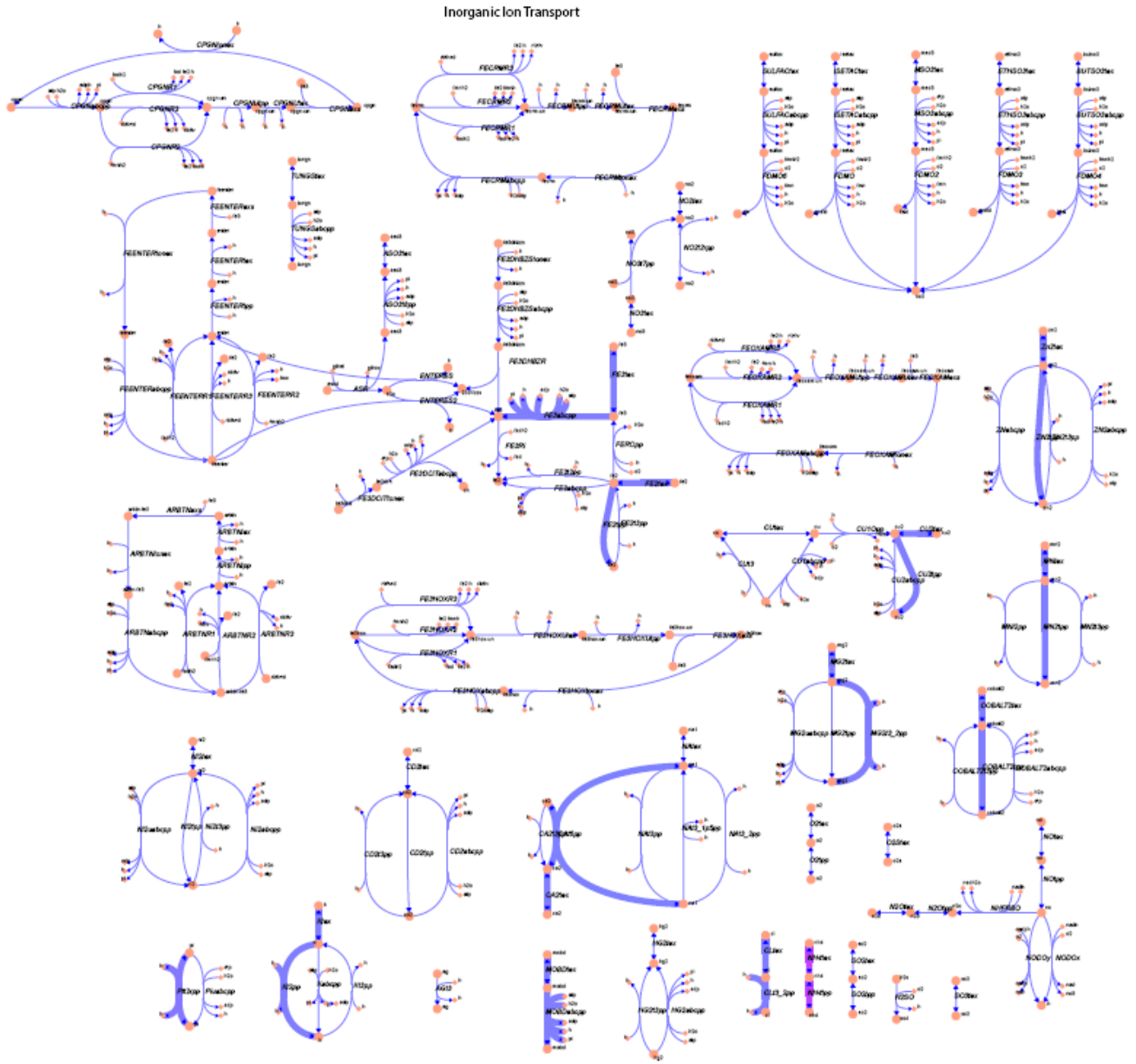
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

iAF1260 Inorganic Ion Transport

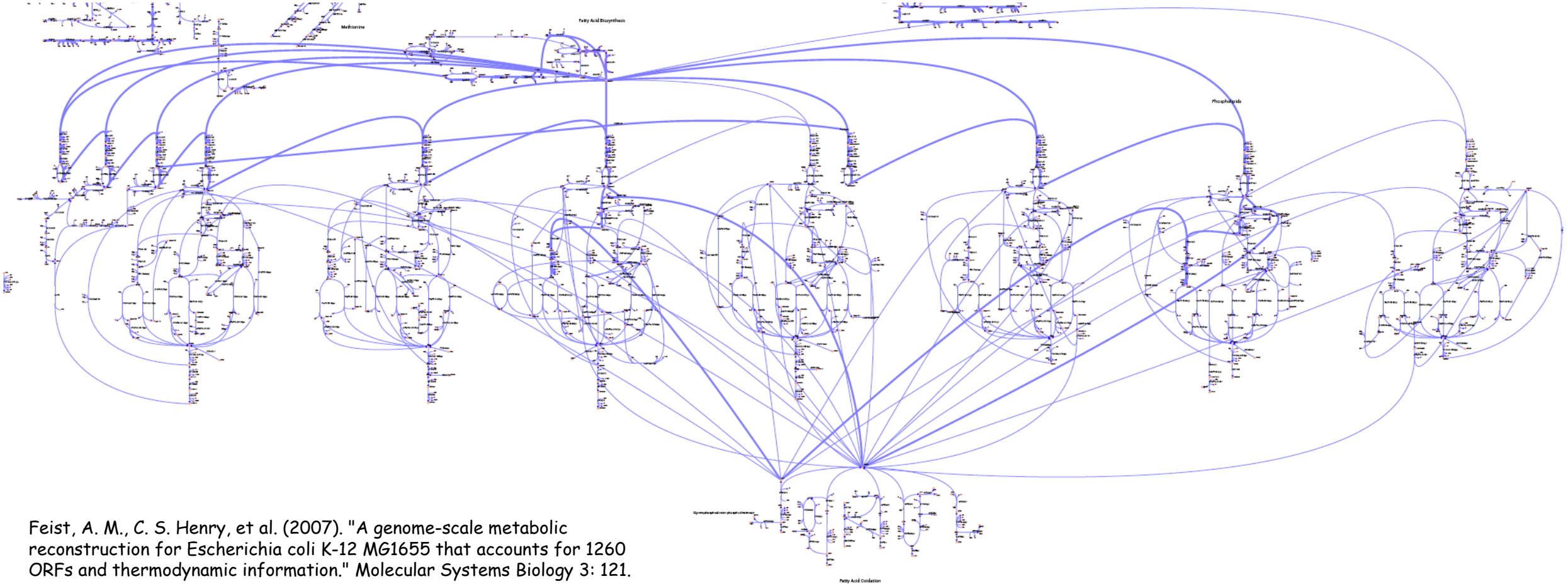
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

Fatty Acid Biosynthesis

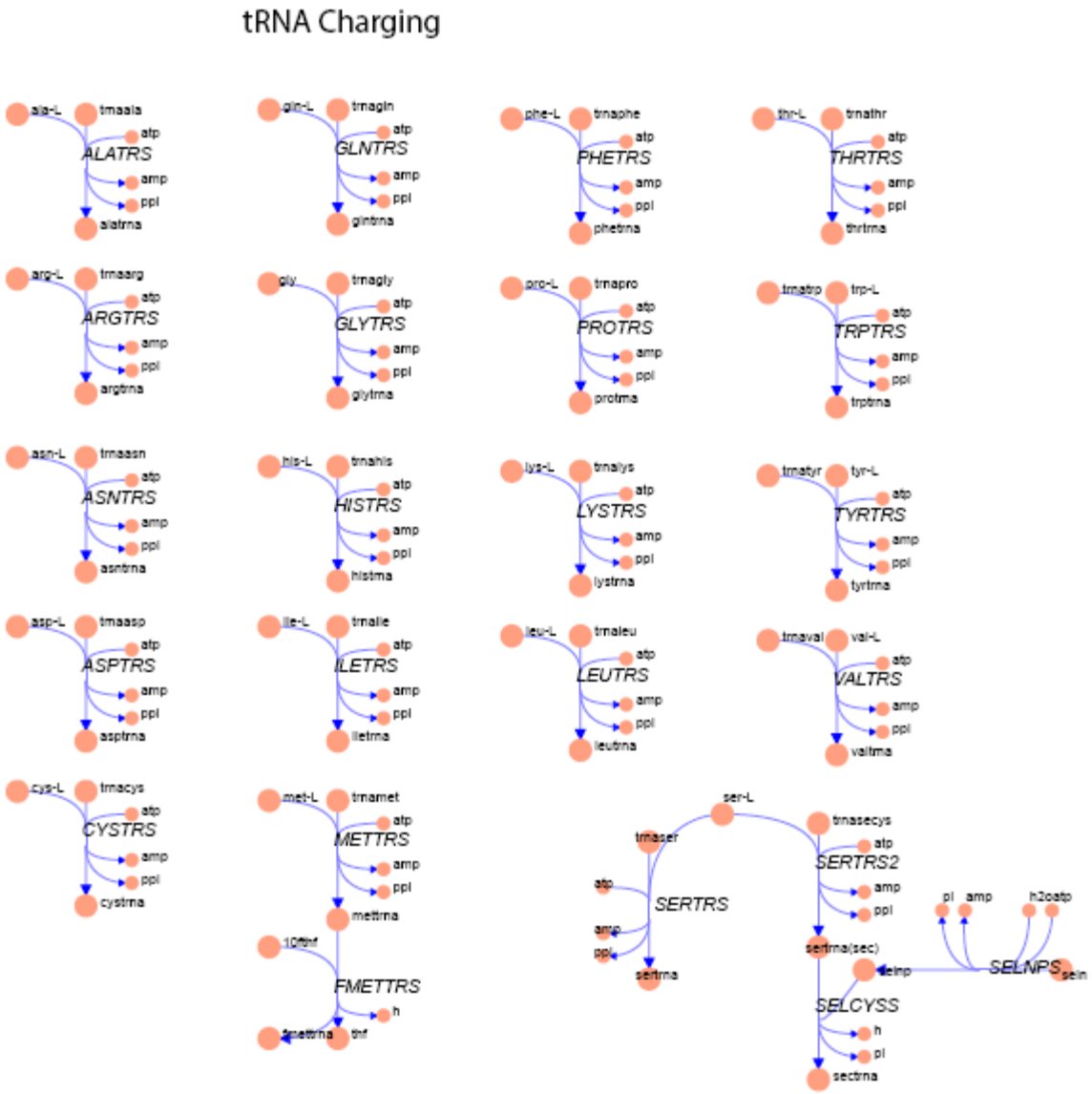
(GlucoseEthanol_iaf1260.pdf)



Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

tRNA Charging

(GlucoseEthanol_iaf1260.pdf)



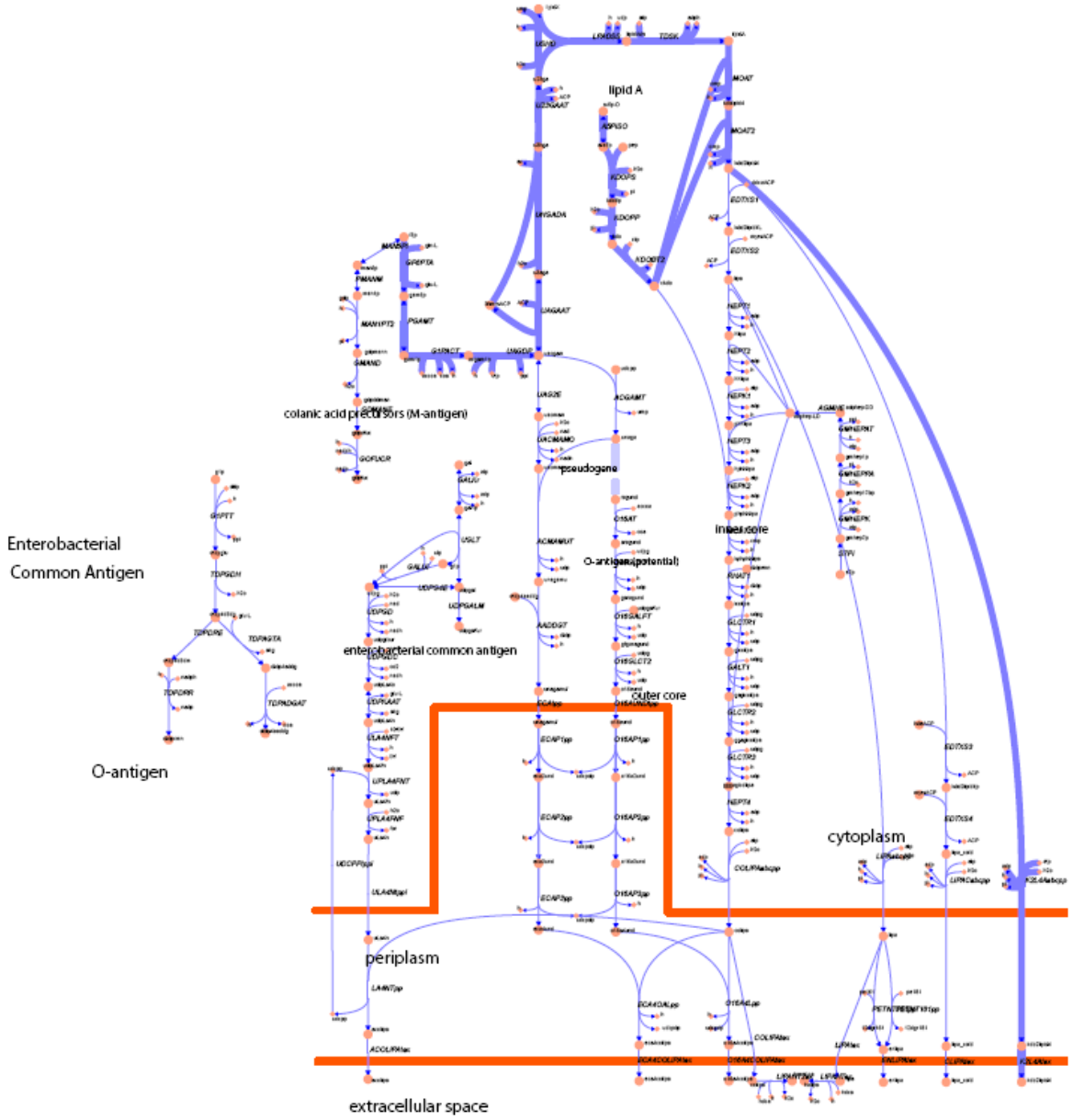
Feist, A. M., C. S. Henry, et al. (2007). "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information." Molecular Systems Biology 3: 121.

Enterobacterial Common Antigen

(GlucoseEthanol_iaf1260.pdf)

Enterobacterial common antigen (ECA), also referred to as an endotoxin, is a family-specific surface antigen shared by all members of the Enterobacteriaceae and is restricted to this family. It is found in freshly isolated wild-type strains as well as in laboratory strains like Escherichia coli K-12. ECA is located in the outer leaflet of the outer membrane. It is a glycophospholipid built up by an aminosugar heteropolymer linked to an L-glycerophosphatidyl residue.

Kuhn, H. M., U. Meier-Dieter, et al. (1988). "ECA, the enterobacterial common antigen." FEMS microbiology reviews 4(3): 195-222.

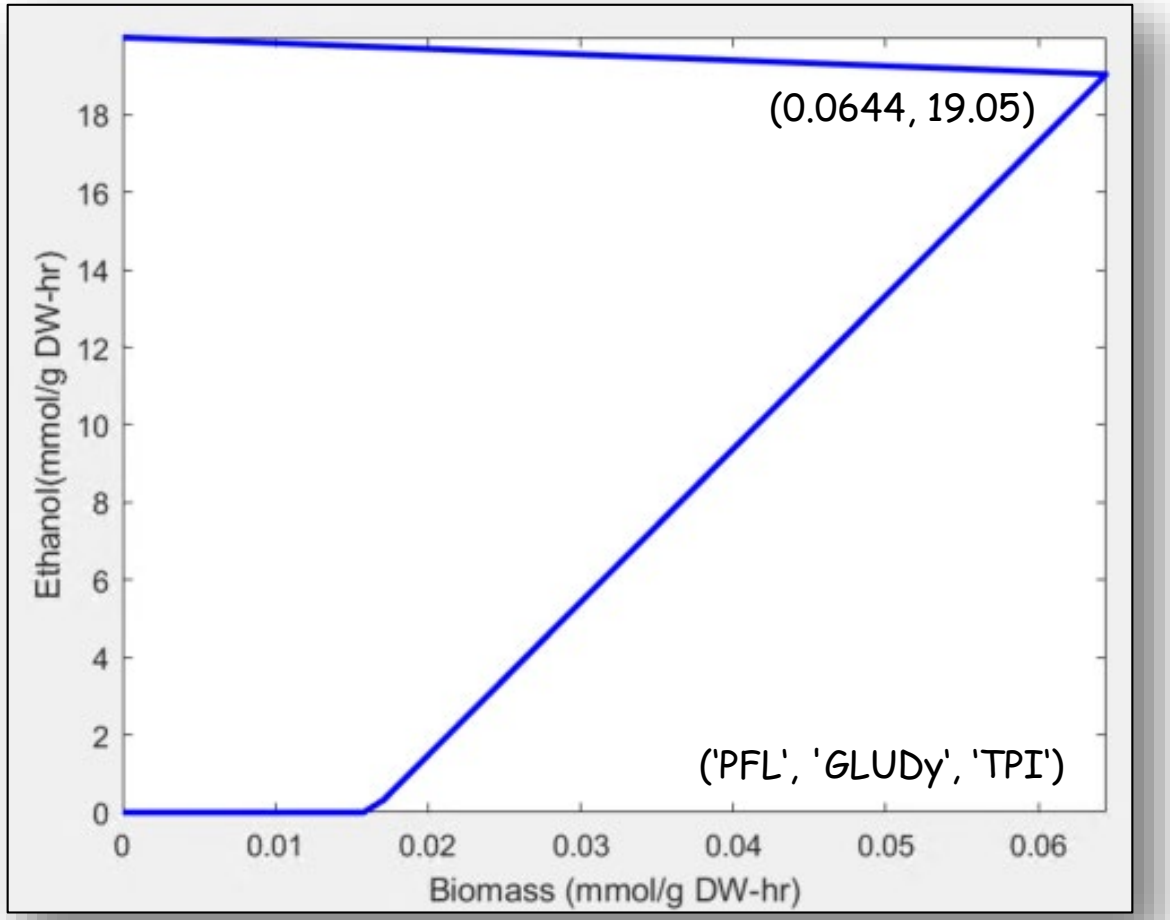


"optKnock" Knockouts for Ethanol Production using the iAF1260 Model

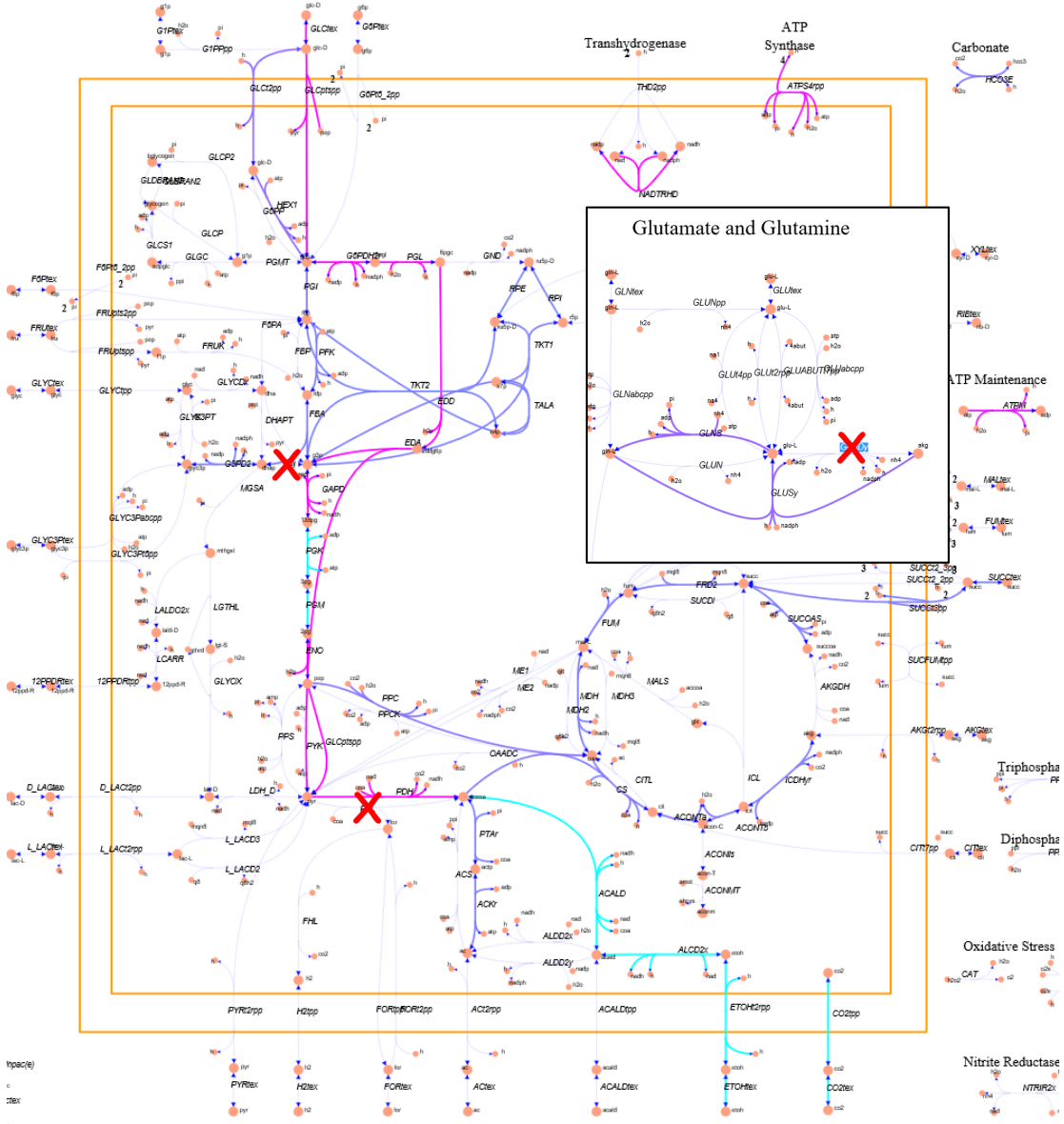
```
% GlucoseEthanolOptknock_iaf1260_Reduced_Reactions.m
clear;

% Set constraints
model = readCbModel('iAF1260.mat');
model = changeRxnBounds(model, {'EX_o2(e)', 'EX_glc(e)'}, [0 -10], 'l');
model = changeObjective(model,'Ec_biomass_iAF1260_core_59p81M');

[transRxns,nonTransRxns] = findTransRxns(model,true); % Remove transport reactions
includedSubSystems = {'Transport, Inner Membrane','Glycerophospholipid Metabolism','Transport, Outer Membrane Porin',...
    'Cell Envelope Biosynthesis','Nucleotide Salvage Pathway','Murein Recycling','Membrane Lipid Metabolism', ...
    'Glycerophospholipid Metabolism','Inorganic Ion Transport and Metabolism','Lipopolysaccharide Biosynthesis / Recycling',...
    'tRNA Charging','Unassigned','Membrane Lipid Metabolism','Murein Biosynthesis'};
[unwantedReactions,rxnPos] = findRxnsFromSubSystem(model,includedSubSystems)
[tf,rids] = ismember([unwantedReactions;{'ATPM'};{'Ec_biomass_iAF1260_core_59p81M'}], nonTransRxns);
idfinal=rids(tf);
nonTransRxns(idfinal) = [];
selectedRxns = nonTransRxns;

% Optknock analysis for Ethanol secretion
disp('Executing optKnock');
options.targetRxn = 'EX_etoh(e)';
options.vMax = 1000;
options.numDel = 3;
options.numDelSense = 'L';
constrOpt.rxnList = {'Ec_biomass_iAF1260_core_59p81M','ATPM'};
constrOpt.values = [0.05, 8.39];
constrOpt.sense = 'GE';

optKnockSol = OptKnock(model, selectedRxns, options, constrOpt);
deletions = optKnockSol.rxnList'

% Print out growth rate and minimum & maximum secretion rate
[growthRate,minProd,maxProd] = testOptKnockSol(model,'EX_etoh(e)',optKnockSol.rxnList)
```

3 Knockouts – iaf1260

GlycerolEthanolOptknock_iaf1260.m



GlucoseEthanolOptknock_iaf1260_Reduced_Reactions.m



GlucoseEthanol_Mutant_iaf1260.m

Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

analyzeOptKnock

```
% EthanolProduction_analyzeOptKnock.m
clear; clc;

model = readCbModel('ecoli_core_model.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Analyze OptKnok results for growth coupling
deletions = {'ACKr','GLUDy','ME2','PGL','PYK'};
biomassRxn = {'Biomass_Ecoli_core_N(w/GAM)-Nmet2'};
minGrowth = 0,05;
geneDelFlag = false;
target ='EX_etoh(e)';
[type,maxGrowth,maxProd] = analyzeOptKnock(model,deletions,target,biomassRxn,geneDelFlag)
```

Determines whether an optKnock solution is growth coupled or not and what the maximum growth and production rates are

```
minGrowth =

     0


type =

growth coupled non unique


maxGrowth =

    0.0767


maxProd =

   18.8342
```

testOptKnockSol

```
% EthanolProduction_OptKnock.m
clear; clc;
% Input the E.coli core model
model = readCbModel('ecoli_core_model.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Select reactions to be explored by the optKnock algorithm
[transRxns,nonTransRxns] = findTransRxns(model,true);
[tmp,ATPMnumber] = ismember('ATPM',nonTransRxns); % Identify ATPM reaction number
[tmp,BioMassnumber] = ismember('Biomass_Ecoli_core_N(w/GAM)-Nmet2',nonTransRxns); % Identify biomass reaction number
nonTransRxnsLength = length(nonTransRxns); % Find number of non-transport reactions
selectedRxns = {nonTransRxns{[1:ATPMnumber-1, ATPMnumber+1:BioMassnumber-1, BioMassnumber+1:nonTransRxnsLength]}};

% Optknock analysis for Ethanol secretion
options.targetRxn = 'EX_etoh(e)';
options.vMax = 1000;
options.numDel = 1;
options.numDelSense = 'L';
constrOpt.rxnList = {'Biomass_Ecoli_core_N(w/GAM)-Nmet2','ATPM'};
constrOpt.values = [0.14, 8.39];
constrOpt.sense = 'GE';

optKnockSol = OptKnock(model, selectedRxns, options, constrOpt);
deletions = optKnockSol.rxnList'

% Print out growth rate and minimum & maximum secretion rate
[growthRate,minProd,maxProd] = testOptKnockSol(model,'EX_etoh(e)',optKnockSol.rxnList)
```

Test an OptKnock knockout strain

```
deletions =

    'ACKr'    'GLUDy'    'GND'


growthRate =

    0.1772


minProd =

   17.3062


maxProd =

   17.3062
```

Command Window Printout

simpleOptKnock

```
% EthanolProduction_simpleOptKnock.m
clear; clc;

model=readCbModel('ecoli_textbook');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

deletions = {'ACKr','GLUDy','ME2','PGL','PYK'};

biomassRxn = {'Biomass_Ecoli_core_N(w/GAM)-Nmet2'};
minGrowth = 0,05;
geneDelFlag = false;
doubleDelFlag = false;

[wtRes,delRes] = simpleOptKnock(model,'EX_etoh(e)',deletions,false,minGrowth,doubleDelFlag)
```

Check all one gene or reaction deletions for growth-coupled metabolite production

```
minGrowth =

     0


wtRes =

     growth: 0.2117
    maxProd: 8.2795
    minProd: 8.2795


delRes =

    maxProd: [5x1 double]
    minProd: [5x1 double]
     growth: [5x1 double]
```

Variables - delRes.maxProd

delRes | delRes.maxProd

delRes.maxProd

	1	2	3
1	16.5843	←	ACKr
2	8.3731	←	GLUDy
3	8.2795	←	ME2
4	8.2795	←	PGL
5	9.0016	←	PYK

Maximum production for each individual knockout reaction

OptKnock Review Questions

- What is OptKnock?
- Why should the number of potential knockout reactions be limited?
- What type of reactions should not be included in an OptKnock search?
- How do you knockout a reaction using the Cobra Toolbox?
- What does it mean to couple the growth and metabolite production?
- What is a production envelope? What is a multiproduction envelope?
- How can a production envelope be created for all secreted metabolites?
- Why is there a trade-off between biomass growth and bioproduct production?
- How many knockouts can be identified by OptKnock?
- What are some of the key parameters needed for OptKnock?
- How can you simulate the engineered mutant cell using the knockouts identified by OptKnock?
- What are the limitations of OptKnock?
- What are some of the OptKnock supporting functions in the Cobra Toolbox?

Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

Genetic Design Local Search (GDLS)

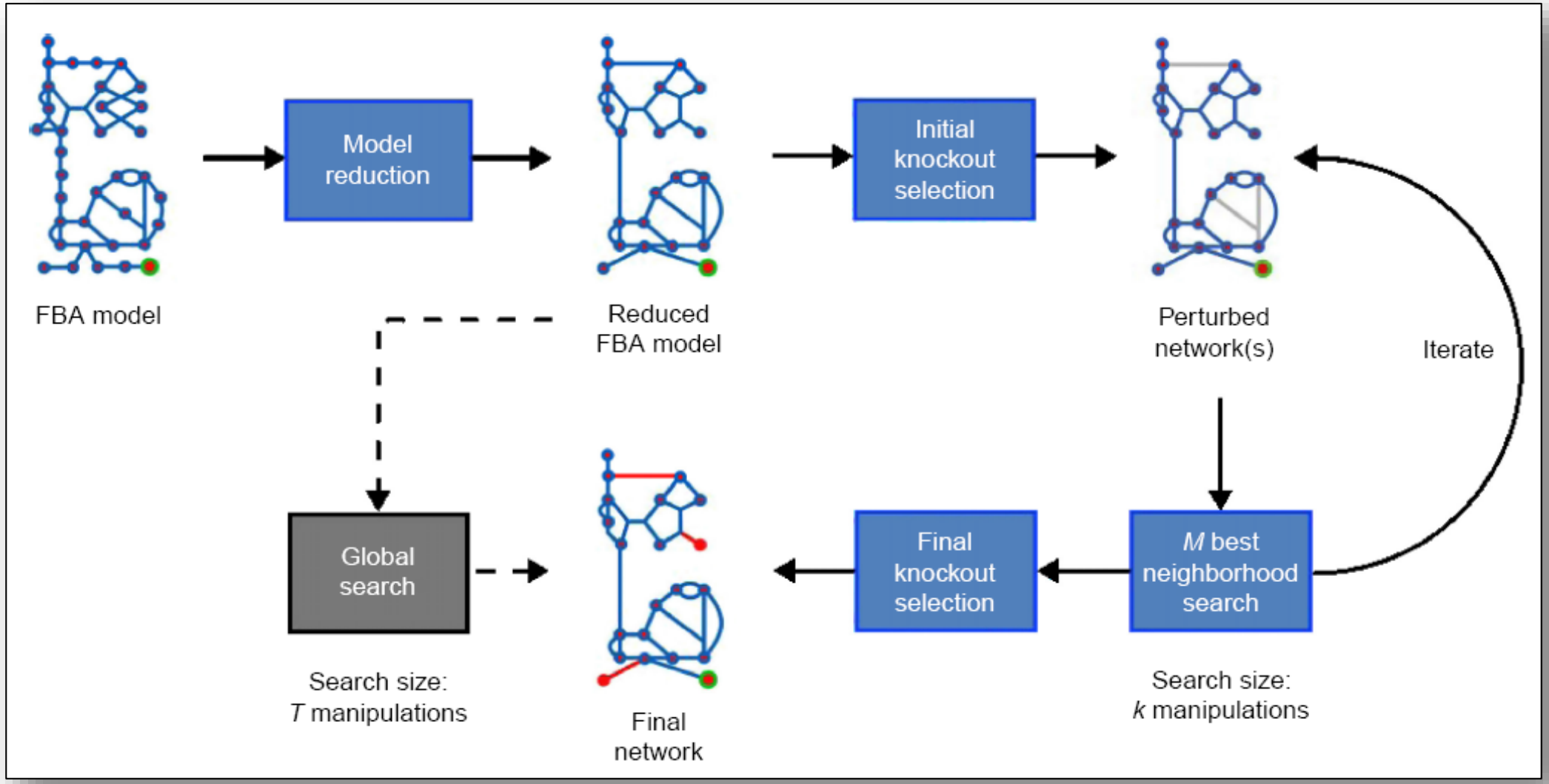
This algorithm typically runs faster than the global search performed by OptKnock, however, it is not guaranteed to identify the global optima.

[gdlsSolution, biLevelMILPproblem, gdlsSolutionStructs] = GDLS(model, varargin)

where: varargin are optional parameters; gdlsSolution is the knockout solution; biLevelMILPproblem is the bi-level MILP problem for the solution; and gdlsSolutionStructs contains the intermediate solutions.

Lun, D.S. et al. "Large-scale identification of genetic design strategies using local search," Mol Syst Biol 5 (2009).

Overview of the Genetic Design Local Search (GDLS)



Lun, D.S. et al. "Large-scale identification of genetic design strategies using local search," Mol Syst Biol 5 (2009).

Model Reduction

Change the model to reduce the number of reactions, metabolites, and boundary conditions to reduce the search space for optimization.

GDLS Model Reduction

- Remove dead-end reactions
- Replace linked reactions or equivalent variables (pairs of fluxes that are constrained to have the same value.)
- Finds the minimal bounds for the flux through each reaction.

reduceModel Cobra Function

- Removes from the model all of the reactions that are never used (max and min are < tol).
- Finds the minimal bounds for the flux through each reaction.
- Also returns the results for flux variability analysis (maxes, mins).

[gdlsSolution, bilevelMILPProblem, gdlsSolutionStructs] = GDLS(model, varargin)

INPUTS

model Cobra model structure

targetRxn Reaction(s) to be maximized (Cell array of strings)

OPTIONAL INPUTS

varargin parameters entered using either a structure or list of parameter, parameter value

List of optional parameters

'nbhdsz'	Neighborhood size (default: 1)
'M'	Number of search paths (default: 1)
'maxKO'	Maximum number of knockouts (default: 50)
'koCost'	Cost for knocking out a reaction, gene set, or gene A different cost can be set for each knockout. (default: 1 for each knockout)
'selectedRxns'	List of reactions/geneSets that can be knocked out
'koType'	What to knockout: reactions, gene sets, or genes {('rxns'), 'geneSets', 'genes'}
'iterationLimit'	Maximum number of iterations (default: 70)
'timeLimit'	Maximum run time in seconds (default: 252000)
'minGrowth'	Minimum growth rate

OUTPUTS

gdlsSolution GDLS solution structure (similar to OptKnock sol struct)

bilevelMILPProblem Problem structure used in computation

GDLS Example: Maximizing Ethanol Production

EthanolProduction_GDLS.m

```
% EthanolProduction_GDLS.m
clear; clc;

% Set operating conditions
model = readCbModel('ecoli_core_model.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Select reactions
[transRxns,nonTransRxns] = findTransRxns(model,true);
[tmp,ATPMnumber] = ismember('ATPM',nonTransRxns); % Identify ATPM reaction number
[tmp,BioMassnumber] = ismember('Biomass_Ecoli_core_N(w/GAM)-Nmet2',nonTransRxns); % Identify biomass reaction number
nonTransRxnsLength = length(nonTransRxns); % Find number of non-transport reactions
selectedRxns = {nonTransRxns{[1:ATPMnumber-1, ATPMnumber+1:BioMassnumber-1, BioMassnumber+1:nonTransRxnsLength]}};

% GDLS analysis for Ethanol secretion
[gdlsSolution, bilevelMILPproblem, gdlsSolutionStructs] = GDLS(model, 'EX_etoh(e)', 'minGrowth', 0.05, …
    'selectedRxns', selectedRxns, 'maxKO', 3, 'nbhdsz', 2);
```

deletions = 'ACKr', 'GLUDy', 'PYK'

growthRate = 0.0852

maxProd = 18.7054

GDLS Example: Iterations

EthanolProduction_GDLS.m

Iteration 1

Biomass flux: 0.091291
Synthetic flux: [18.268541, 18.612509]
Knockout cost: 2
Knockouts:

	ACKr PYK
Biomass	0.0912913
EX_co2(e)	18.5461
EX_etoh(e)	18.6125
EX_for(e)	0.343968
EX_glc(e)	-10
EX_h2o(e)	0.598561
EX_h(e)	2.17527
EX_nh4(e)	-0.497793
EX_pi(e)	-0.335833

Iteration 2

Biomass flux: 0.085178
Synthetic flux: [18.384498, 18.705430]
Knockout cost: 3
Knockouts:

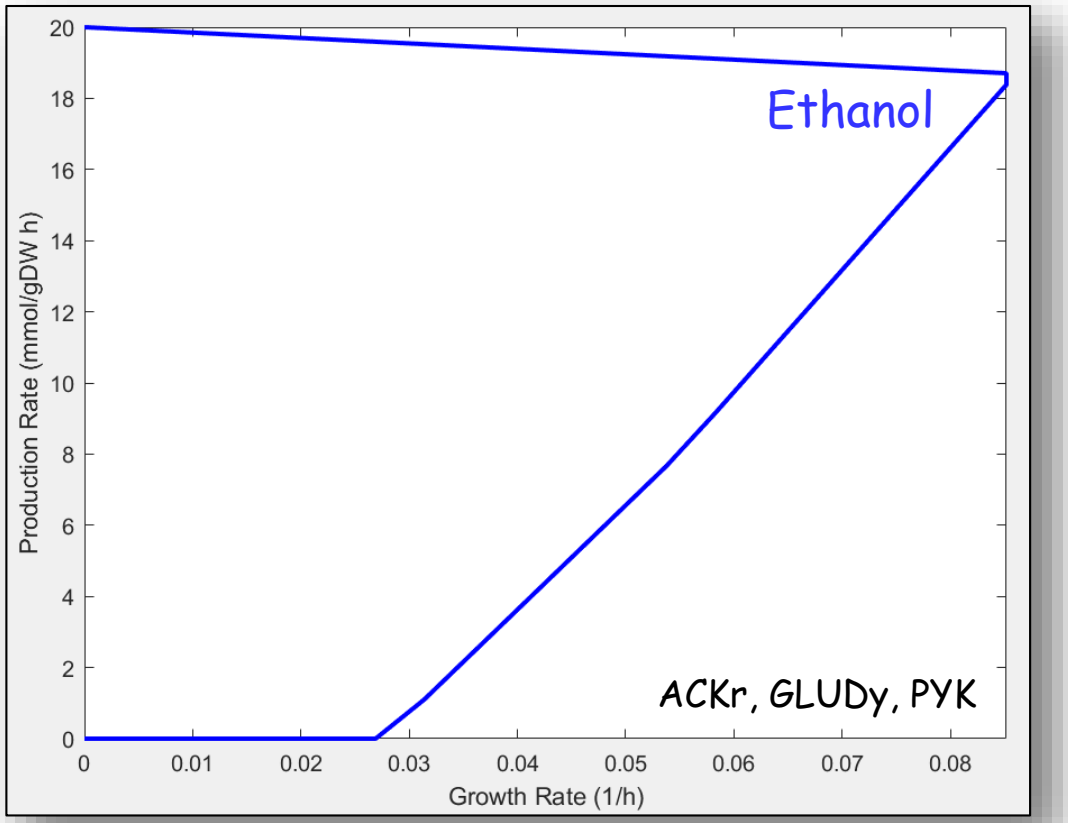
	ACKr GLUDy PYK
Biomass	0.0851775
EX_co2(e)	18.6434
EX_etoh(e)	18.7054
EX_for(e)	0.320932
EX_glc(e)	-10
EX_h2o(e)	0.558475
EX_h(e)	2.02959
EX_nh4(e)	-0.464456
EX_pi(e)	-0.313342

Iteration 3

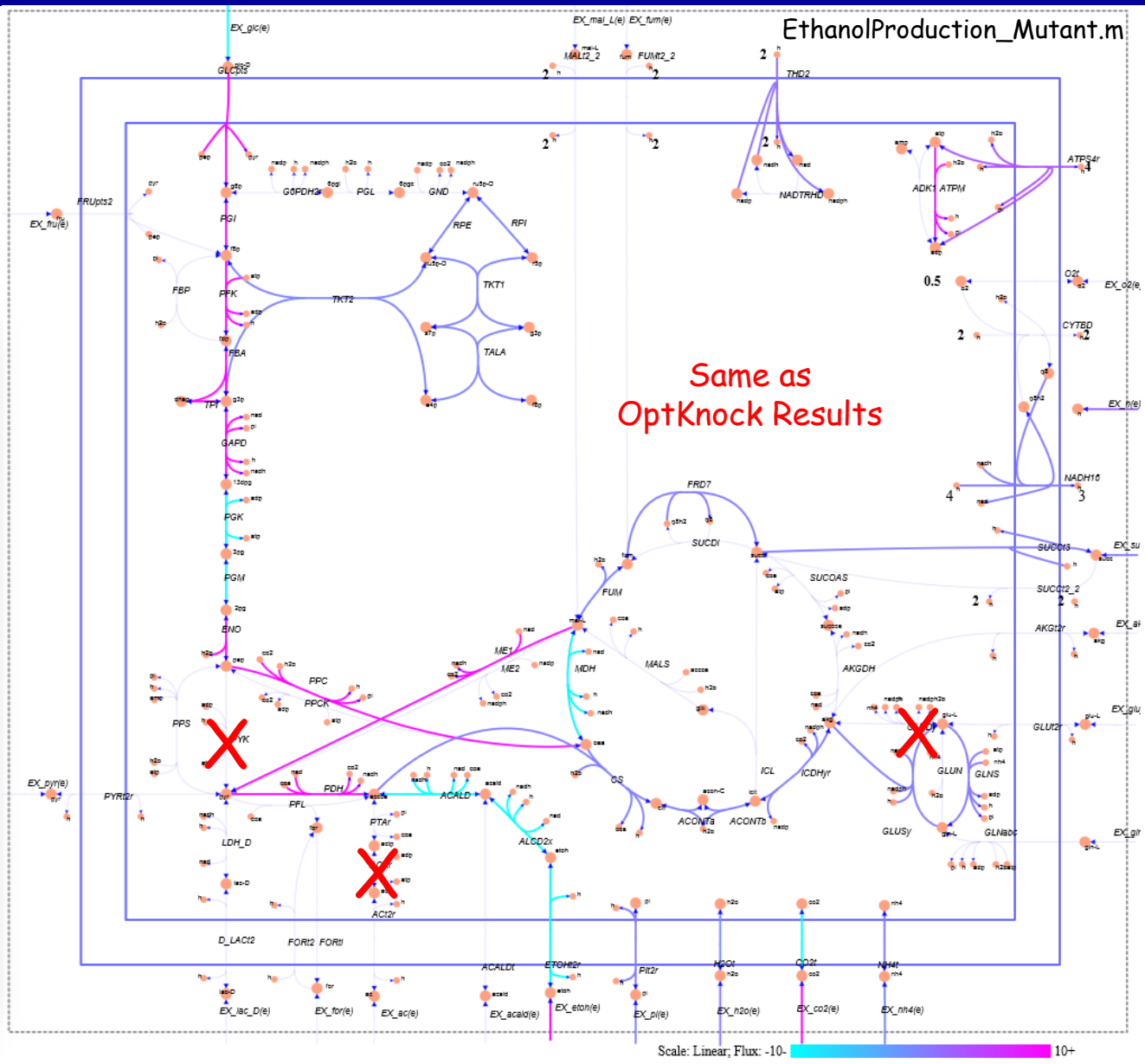
Biomass flux: 0.085178
Synthetic flux: [18.384498, 18.705430]
Knockout cost: 3
Knockouts:

	ACKr GLUDy PYK
Biomass	0.0851775
EX_co2(e)	18.6434
EX_etoh(e)	18.7054
EX_for(e)	0.320932
EX_glc(e)	-10
EX_h2o(e)	0.558475
EX_h(e)	2.02959
EX_nh4(e)	-0.464456
EX_pi(e)	-0.313342

GDLS Example: Location of Knockouts for the GDLS Mutant



EthanolProduction_multiProductionEnvelope.m



GDLS Example: Maximizing Ethanol Production using Genes

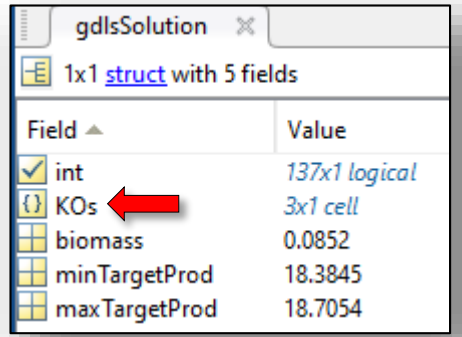
```
% EthanolProduction_GDLS_Gene.m
clear;

% Set operating conditions
load('ecoli_textbook.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)_Nmet2');

selectedRxns = model.genes; % Using all model genes

% GDLS analysis for Ethanol secretion
[gdlsSolution, bilevelMILPproblem, gdlsSolutionStructs] = GDLS(model, 'EX_etoh(e)', 'minGrowth', 0.05, ...
    'koType','genes','selectedRxns', selectedRxns, 'maxKO', 3, 'nbhdsz', 2);

gdlsSolution.KOs
[results ListResults] = findRxnsFromGenes(model, gdlsSolution.KOs); % Reaction names located in "results" structure
```

Gene deletions = b1676, b1761, b2297

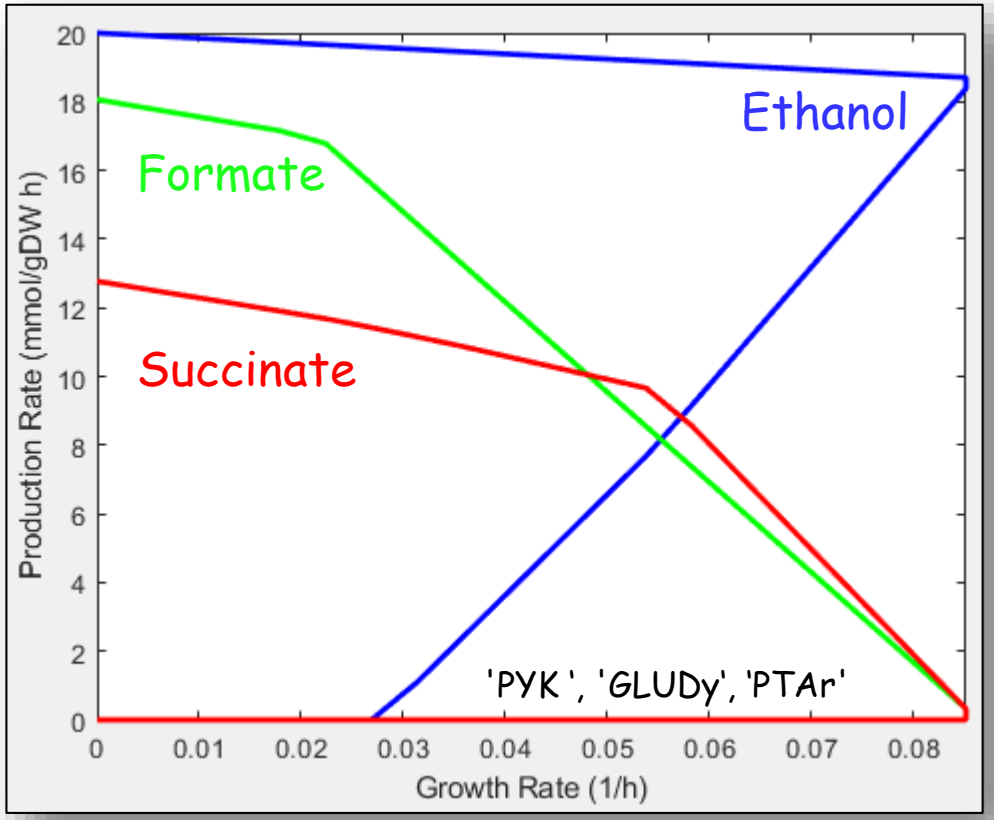
Reaction deletions = 'PYK', 'GLUDy', 'PTAr'

growthRate = 0.0852

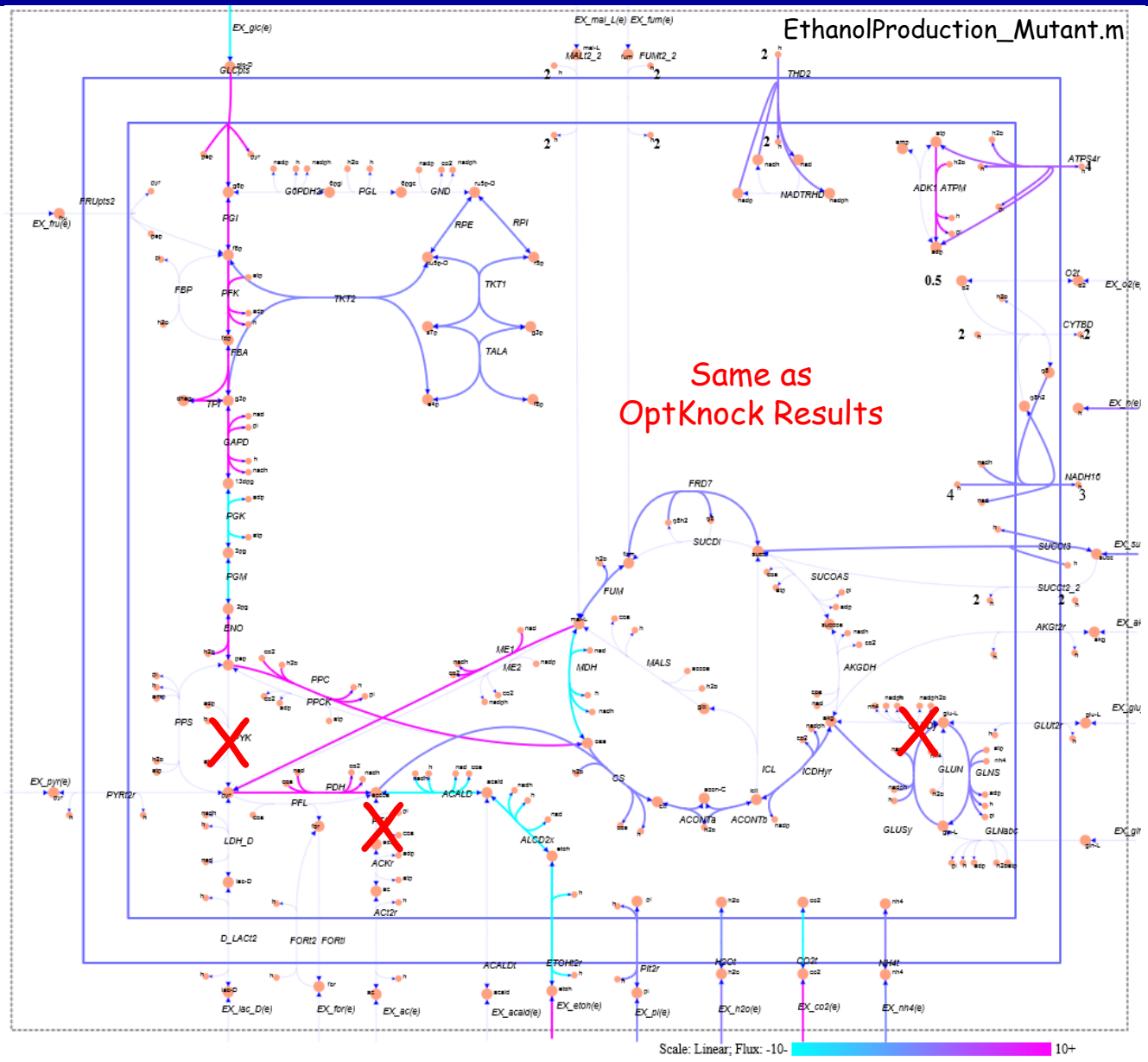
maxProd = 18.7054



GDLS Gene Example: Location of Knockouts for the GDLS Mutant

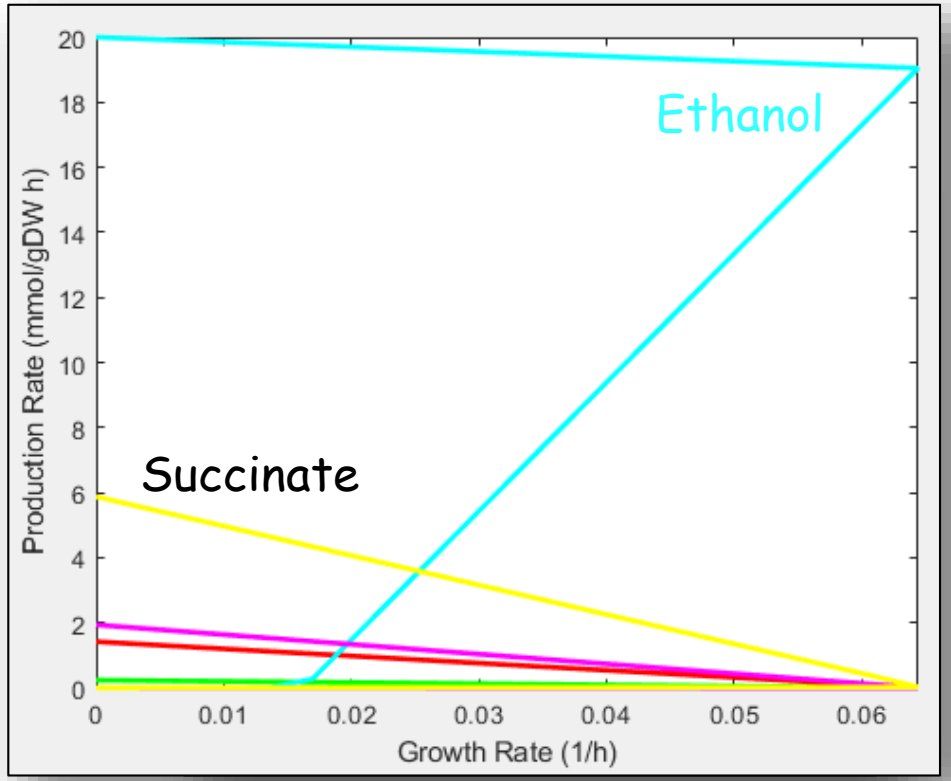


EthanolProduction_multiProductionEnvelope.m



"GDLS" Gene Knockouts for Ethanol Production Using the iAF1260 Model

```
% EthanolProduction_GDLS_Gene_iaf1260.m
clear;

% Set operating conditions
model = readCbModel('iAF1260.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Ec_biomass_iAF1260_core_59p81M');

selectedRxns = model.genes;

% GDLS analysis for Ethanol secretion
[gdlsSolution, bilevelMILPproblem, gdlsSolutionStructs] = GDLS(model, 'EX_etoh(e)', ...
'minGrowth', 0.05, 'koType','genes','selectedRxns', selectedRxns, 'maxKO', 3, 'nbhdsz', 2);

gdlsSolution.KOs
[results ListResults] = findRxnsFromGenes(model, gdlsSolution.KOs)
```

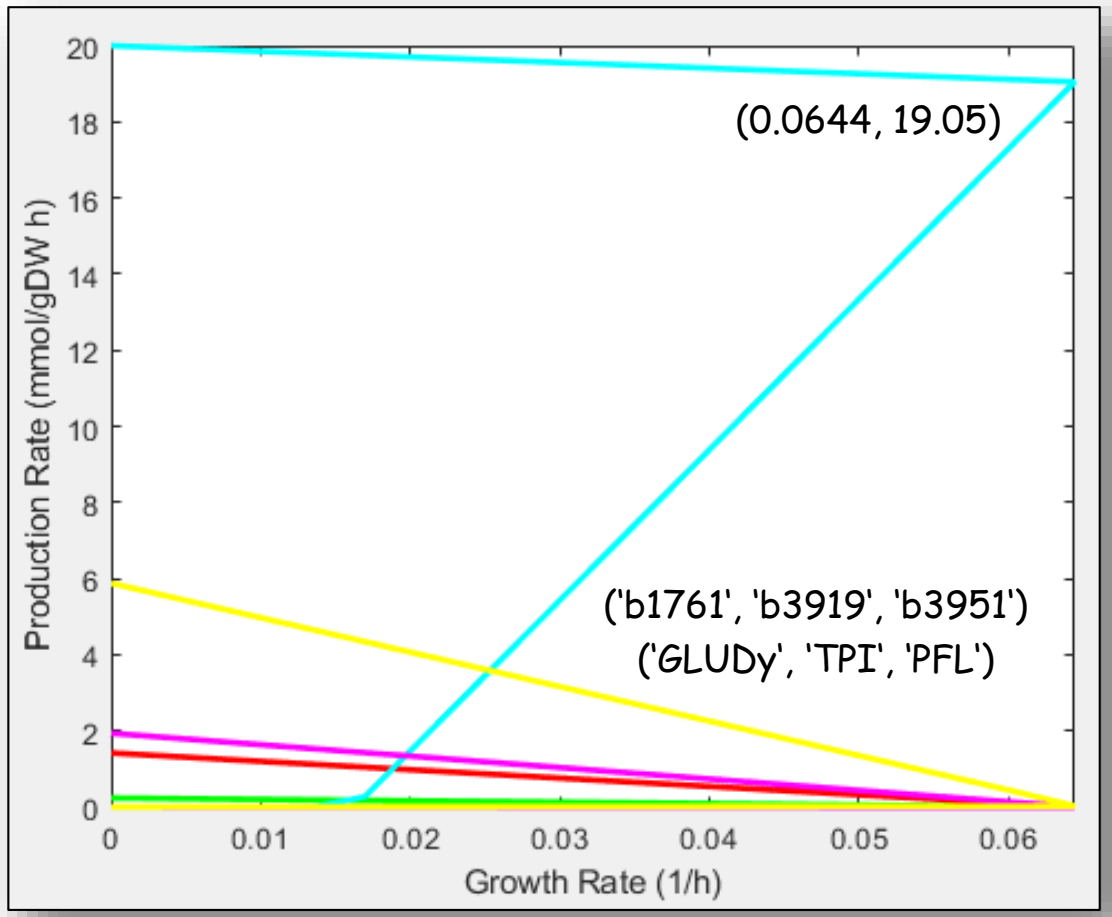


EthanolProduction_multiProductionEnvelope_iaf1260.m

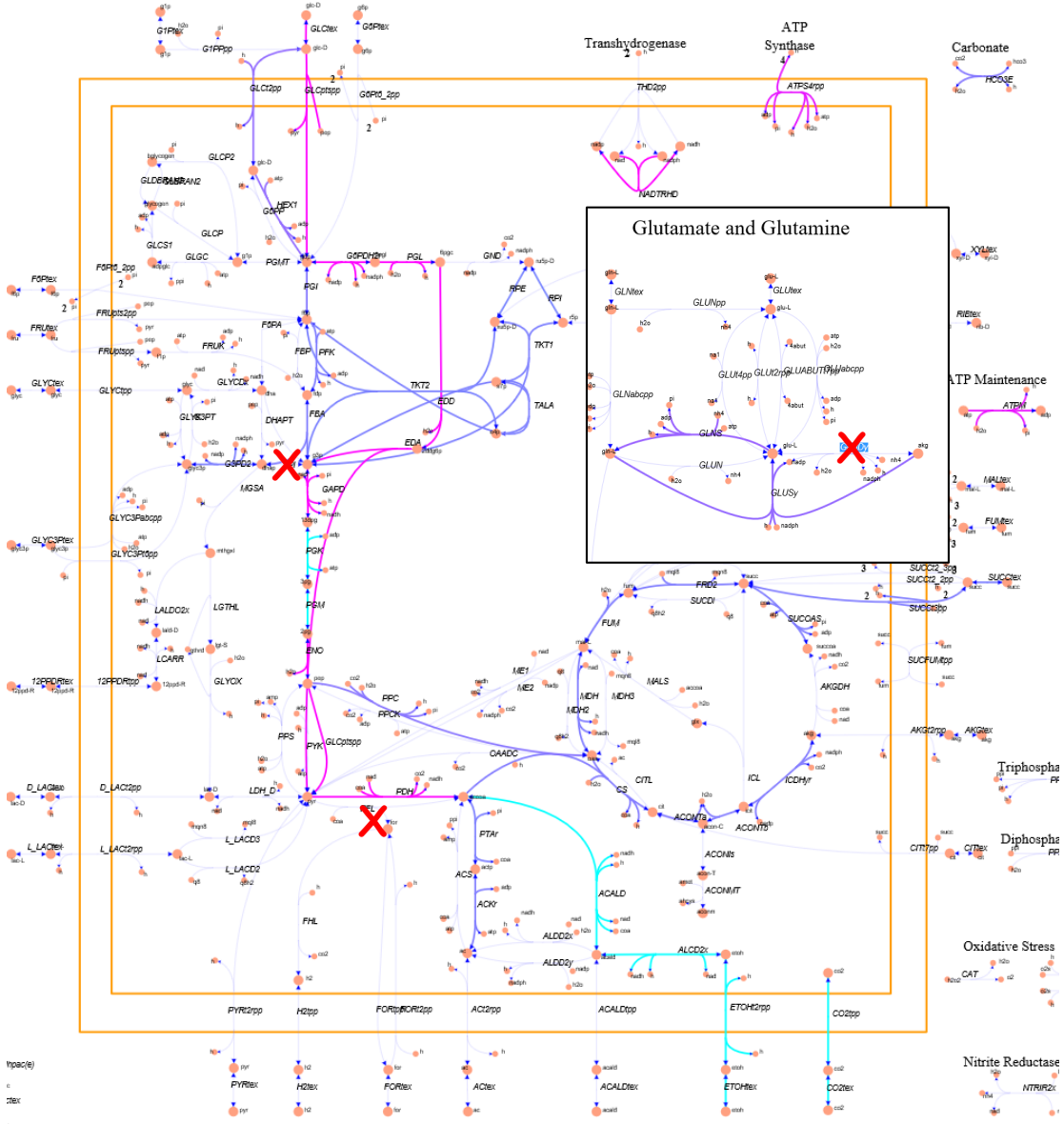
deletions = 'b1761' , 'b3919', 'b3951'
reactions = GLUDy', 'TPI', 'PFL'
growthRate = 0.0644
maxProd = 19.0497

3 Gene Knockouts - GDLS

GlucoseEthanolGDLS_iaf1260.m



EthanolProduction_multiProductionEnvelope_iaf1260.m



GlucoseEthanol_Mutant_iaf1260.m

GDLS Review Questions

- What is the purpose of GDLS?
- What is the difference between OptKnock, GDLS?
- What are some of the key parameters required by GDLS?
- What does it mean to reduce a model?
- How many iterations are required to complete the GDLS algorithm?
- How many knockouts can be identified by GDLS?
- What are the limitations of GDLS?
- Will OptKnock and GDLS give the same knockouts?

Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

OptGene

OptGene is an evolutionary programming-based method to determine gene knockout strategies for overproduction of a specific product. It can handle non-linear objective functions such as product flux multiplied by biomass.

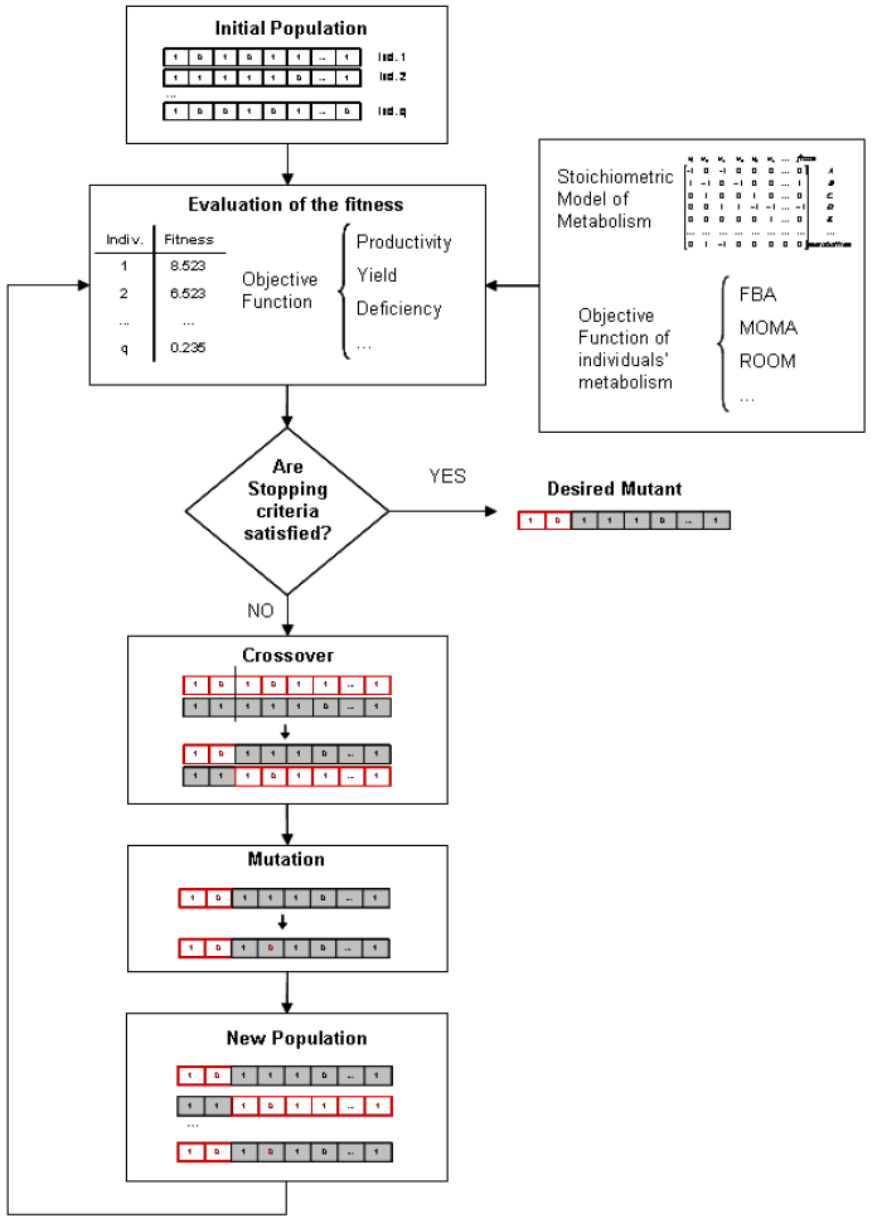
[x, population, scores, optGeneSol] = OptGene(model, targetRxn, substrateRxn, generxnList, maxKOs, [population])

where: targetRxn specifies the reaction to optimize; substrateRxn specifies the exchange reaction for the carbon source; generxnList is a cell array of strings that specifies which genes or reactions are allowed to be deleted; and maxKOs sets the maximum number of knockouts; x is the best scoring set as determined by the functions optGeneFitness or optGeneFitnessTilt; population is the binary matrix representing the knockout sets; and optGeneSol is the structure summarizing the results. If resuming a previous simulation, the binary matrix (population) can be specified.

Patil, K., Rocha, I., Forster, J. & Nielsen, J. Evolutionary programming as a platform for *in silico* metabolic engineering. BMC Bioinformatics 6, 308 (2005).

OptGene Algorithm

- OptGene begins with a predefined number of individuals, forming a population. Each column corresponds to a reaction.
- The fitness score of an individual is calculated using the desired objective function value. The best individuals are selected for crossover
- Selected individuals are then crossed to produce a new offspring.
- Individuals propagating to the new population are mutated (in our formulation, a gene is deleted) with a given probability.
- Mutation and crossover give rise to a new population, which can then again be subjected to a new round of evaluation, crossover and mutations.
- This cycle is repeated until an individual with a satisfactory phenotype is found.
- Uses Matlab's Genetic Algorithm (http://www.mathworks.com/discovery/genetic-algorithm.html)



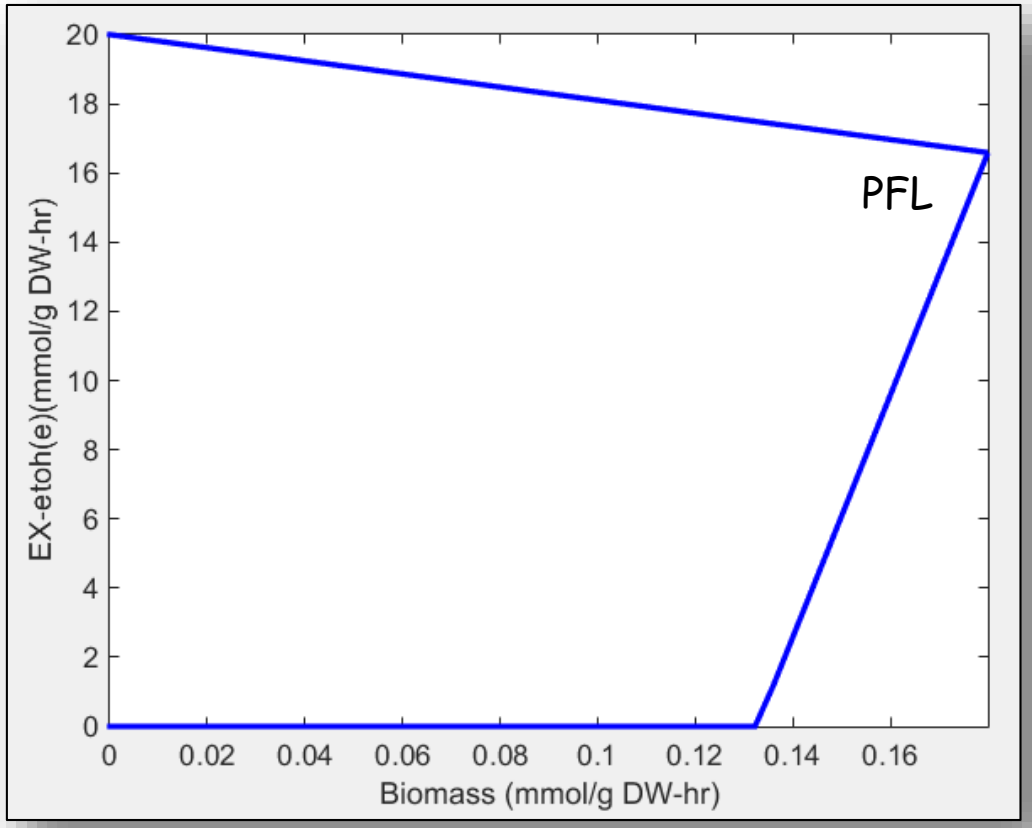
Patil, K., Rocha, I., Forster, J. & Nielsen, J. Evolutionary programming as a platform for *in silico* metabolic engineering. BMC Bioinformatics 6, 308 (2005).

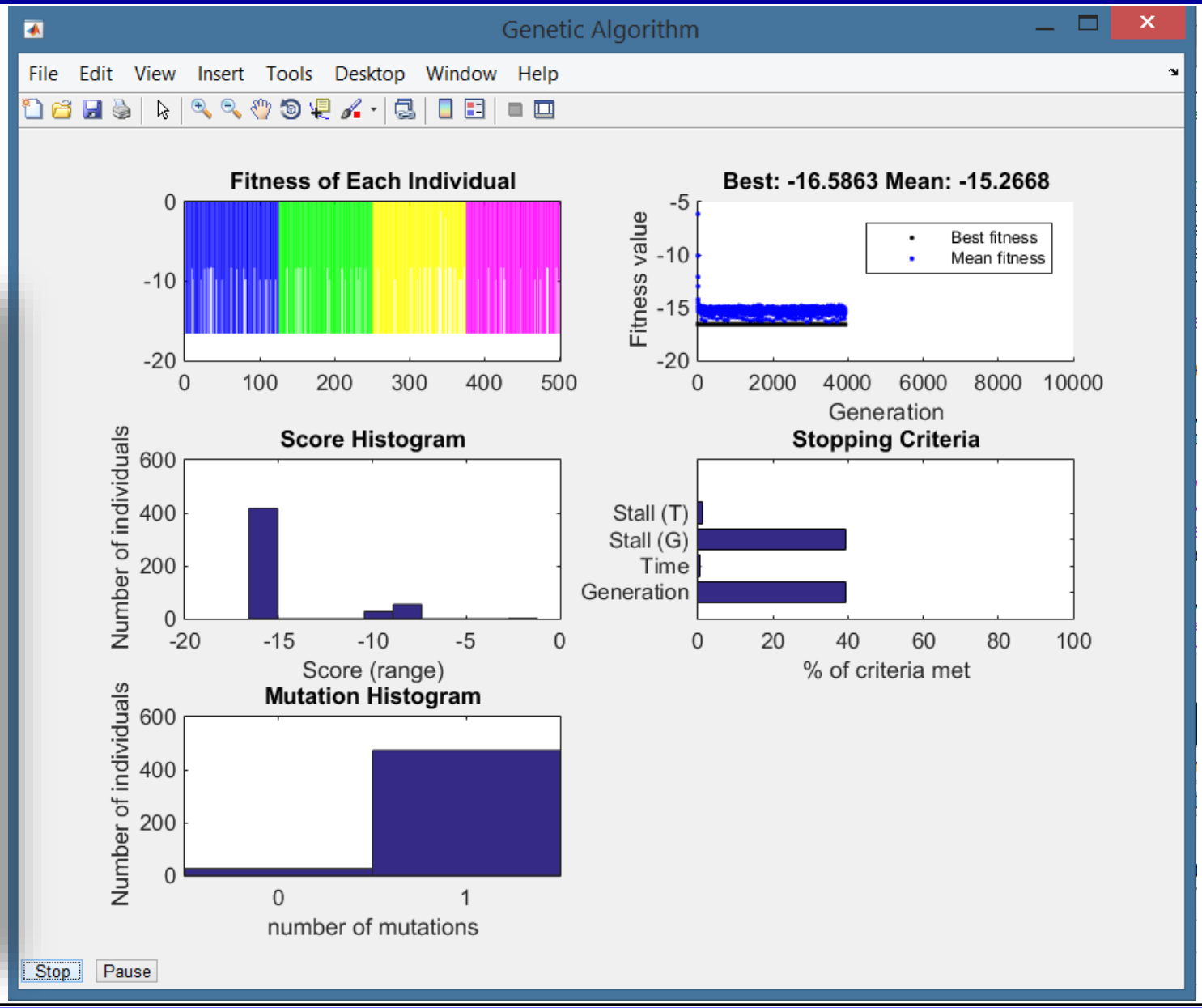
"optGene" Example

```
% Ethanol_OptGene_core.m
clear; clc;
% Set conditions
model = readCbModel('ecoli_core_model.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Select reactions
[transRxns,nonTransRxns] = findTransRxns(model,true);
[tmp,ATPMnumber] = ismember('ATPM',nonTransRxns); % Identify ATPM reaction number
[tmp,BioMassnumber] = ismember('Biomass_Ecoli_core_N(w/GAM)-Nmet2',nonTransRxns); % Identify biomass reaction
number
nonTransRxnsLength = length(nonTransRxns); % Find number of non-transport reactions
generxnList = {nonTransRxns{[1:ATPMnumber-1, ATPMnumber+1:BioMassnumber-1, BioMassnumber+1:nonTransRxnsLength]}};

% Run optGene
[~, ~, ~, optGeneSol] = optGene(model, 'EX_etoh(e)', 'EX_glc(e)', selectedRxnList, 'MaxKOs', 1, 'TimeLimit', 20);

% Graph production envelope
lineColor = 'b';
targetRxn = 'EX_etoh(e)';
biomassRxn = 'Biomass_Ecoli_core_N(w/GAM)_Nmet2';
geneDelFlag = false;
nPts = 50;
[biomassValues,targetValues] =
productionEnvelope(model,optGeneSol.rxnList,lineColor,targetRxn,biomassRxn,geneDelFlag,nPts);
xlabel('Biomass (mmol/g DW-hr)')
ylabel('EX-etoh(e)(mmol/g DW-hr)')
```

optGene Display Window

Ethanol_OptGene_core.m





OptGene Review Questions

- What is the purpose of OptGene?
- What is the difference between OptKnock, GDLS, and OptGene?
- Explain how to use OptGene?
- How many knockouts can be identified by OptGene?
- What are the limitations of OptGene?
- What solvers are required by OptGene?
- Are there changes that need to be made to the OptGene function before it can be used?
- How does OptGene's speed compare to OptKnock and GDLS?

Lesson Outline

- Overview
- OptKnock
  - ✓ OptKnock Philosophy
  - ✓ Identifying Potential Knockout Reactions
  - ✓ Production Envelopes
  - ✓ Knockout Calculation & Simulation
  - ✓ OptKnock Supporting Functions in the Cobra Toolbox
- GDLS
- OptGene

New Cobra Toolbox Functions

Identify non-transport/exchange reactions

```
[transRxns,nonTransRxns] = findTransRxns(model,true);
```

Plot a production envelope

```
[biomassValues,targetValues] = productionEnvelope(model,deletions,lineColor,targetRxn,biomassRxn,geneDelFlag,nPts);
```

Plot a multiproduction envelope

```
[biomassValues,targetValues] = productionEnvelope(model,deletions,lineColor,targetRxn,biomassRxn,geneDelFlag,nPts);
```

OptKnock Analysis

```
optKnockSol = OptKnock(model, selectedRxns, options, constrOpt);
```

OptKnock Supporting Functions

```
[type,maxGrowth,maxProd] = analyzeOptKnock(model,deletions,target,biomassRxn,geneDelFlag)
[growthRate,minProd,maxProd] = testOptKnockSol(model,target,optKnockSol.rxnList)
[wtRes,delRes] = simpleOptKnock(model,target,deletions,false,minGrowth,doubleDelFlag)
```

GDLS Analysis

[gdlsSolution, bilevelMILPProblem, gdlsSolutionStructs] = GDLS(model, varargin)

optGene Analysis

[x, population, scores, optGeneSol] = OptGene(model, targetRxn, substrateRxn, generxnList, maxKOs, [population])

References

Gene/Reaction Knockouts

1. Rocha, I., P. Maia, et al. (2010). "OptFlux: an open-source software platform for in silico metabolic engineering." BMC systems biology 4: 45.
2. Pharkya, P., A. P. Burgard, et al. (2004). "OptStrain: a computational framework for redesign of microbial production systems." Genome research 14(11): 2367-2376.

OptKnock

1. Burgard, A. P., P. Pharkya, et al. (2003). "Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization." Biotechnology and bioengineering 84(6): 647-657.
2. Schellenberger, J., R. Que, et al. (2011). "Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0." Nature protocols 6(9): pp. 1299, 1304, 1305.

GDLS

1. Lun, D.S. et al. "Large-scale identification of genetic design strategies using local search," Mol Syst Biol 5 (2009).
2. Schellenberger, J., R. Que, et al. (2011). "Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0." Nature protocols 6(9): pp. 1299, 1304, 1305.

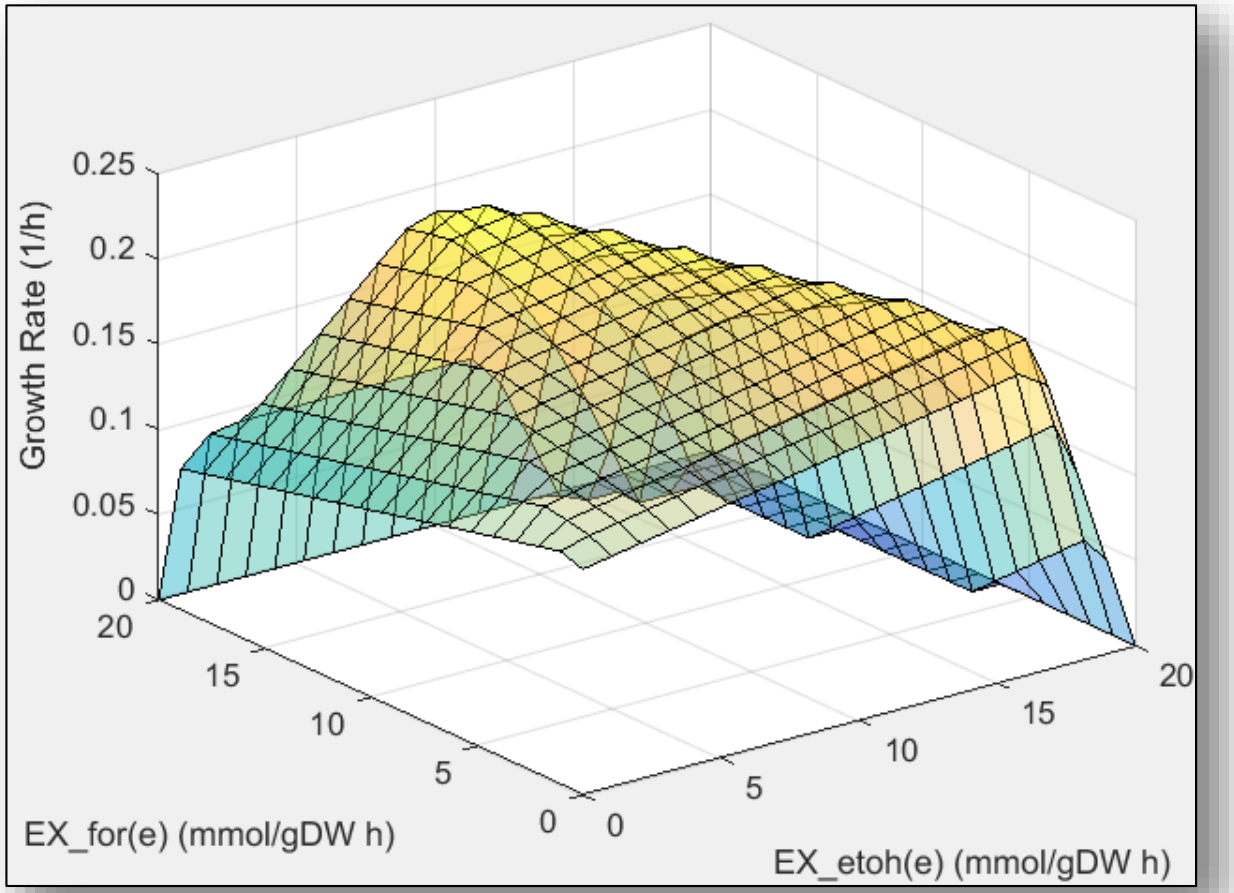
OptGene

1. Patil, K., Rocha, I., Forster, J. & Nielsen, J. Evolutionary programming as a platform for in silico metabolic engineering. BMC Bioinformatics 6, 308 (2005).
2. Schellenberger, J., R. Que, et al. (2011). "Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0." Nature protocols 6(9): pp. 1299.

Extra Slides

doubleProductionEnvelope

Plots maximum growth rate as a function of the output of two specified products.




```
% EthanolProduction_doubleProductionEnvelope.m
clear; clc;
model = readCbModel('ecoli_core_model.mat');
model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',-0,'l');

% Calculate doubleProductionEnvelope
deletions = {};
biomassRxn = {'Biomass_Ecoli_core_N(w/GAM)-Nmet2'};
[x1,x2,y] = doubleProductionEnvelope(model,deletions,'EX_etoh(e)','EX_for(e)',biomassRxn,false,20);
```

Ethanol Production: Exchange Fluxes

3 Knockouts (EthanolProduction_Mutant.m)

```
clear; clc;
model = readCbModel('ecoli_core_model.mat');

model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Knockout reactions
model = changeRxnBounds(model,'ACKr',0,'b');
model = changeRxnBounds(model,'G6PDH2r',0,'b');
model = changeRxnBounds(model,'GLUDy',0,'b');

FBAsolution = optimizeCbModel(model,'max')
printFluxVector(model,FBAsolution.x, true, true)

map=readCbMap('ecoli_Textbook_ExportMap');
options.lb = -10;
options.ub = 10;
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);

robustnessAnalysis(model,'EX_etoh(e)',100);
```

Knockouts

Biomass	0.17724
EX_co2(e)	17.1772
EX_etoh(e)	17.3062
EX_for(e)	0.667805
EX_glc(e)	-10
EX_h2o(e)	1.16209
EX_h(e)	4.22324
EX_nh4(e)	-0.966455
EX_pi(e)	-0.652013

Exchange Fluxes

Ethanol Production: Fluxes

3 Knockouts (EthanolProduction_Mutant.m)

```
clear; clc;
model = readCbModel('ecoli_core_model.mat');

model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Knockout reactions
model = changeRxnBounds(model,'ACKr',0,'b');
model = changeRxnBounds(model,'G6PDH2r',0,'b');
model = changeRxnBounds(model,'GLUDy',0,'b');

FBAsolution = optimizeCbModel(model,'max')
printFluxVector(model,FBAsolution.x, true)

map=readCbMap('ecoli_Textbook_ExportMap');
options.lb = -10;
options.ub = 10;
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);

robustnessAnalysis(model,'EX_etoh(e)',100);
```

Knockouts

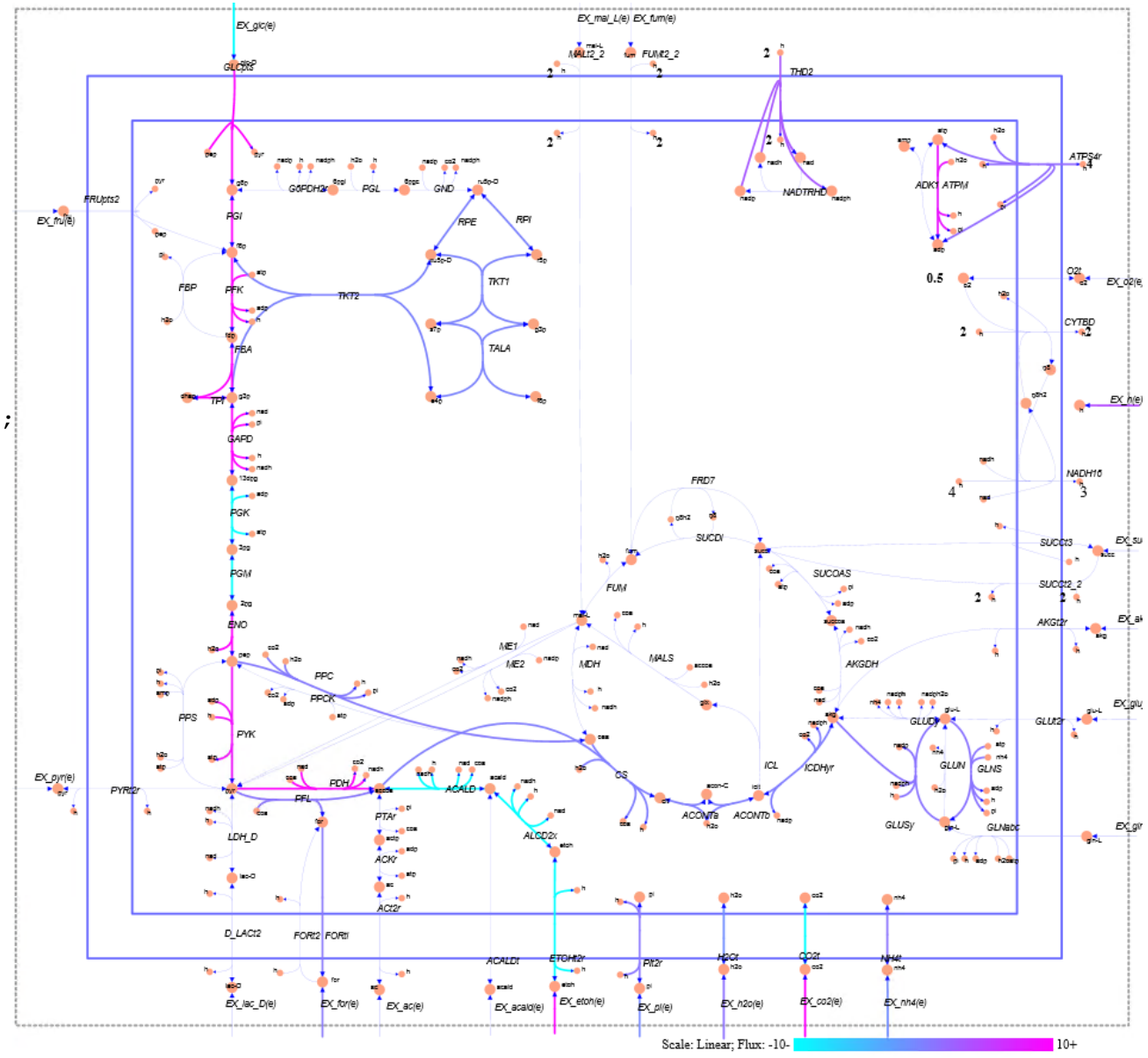
ACALD	-17.3062
ACONTa	0.191224
ACONTb	0.191224
ALCD2x	-17.3062
ATPM	8.39
ATPS4r	1.58825
Biomass	0.17724
CO2t	-17.1772
CS	0.191224
ENO	19.2637
ETOHt2r	-17.3062
EX_co2(e)	17.1772
EX_etoh(e)	17.3062
EX_for(e)	0.667805
EX_glc(e)	-10
EX_h2o(e)	1.16209
EX_h(e)	4.22324
EX_nh4(e)	-0.966455
EX_pi(e)	-0.652013
FBA	9.8237
FORti	0.667805
GAPD	19.5288
GLCpts	10
GLNS	0.966455
GLUSy	0.921134
H2Ot	-1.16209
ICDHyr	0.191224
NH4t	0.966455
PDH	17.4939
PFK	9.8237
PFL	0.667805
PGI	9.96367
PGK	-19.5288
PGM	-19.2637
PIt2r	0.652013
PPC	0.507899
PYK	8.66379
RPE	-0.1274
RPI	-0.1274
TALA	-0.0317082
THD2	3.03898
TKT1	-0.0317082
TKT2	-0.0956919
TPI	9.8237

Ethanol Production: Flux Map

3 Knockouts (EthanolProduction_Mutant.m)

```
clear; clc;
model = readCbModel('ecoli_core_model.mat');

model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Knockout reactions
model = changeRxnBounds(model,'ACKr',0,'b');
model = changeRxnBounds(model,'GLUDy',0,'b');
model = changeRxnBounds(model,'G6PDH2r',0,'b');

FBAsolution = optimizeCbModel(model,'max')
printFluxVector(model,FBAsolution.x, true, true)

map=readCbMap('ecoli_Textbook_ExportMap');
options.lb = -10;
options.ub = 10;
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);

robustnessAnalysis(model,'EX_etoh(e)',100);
```

Knockouts

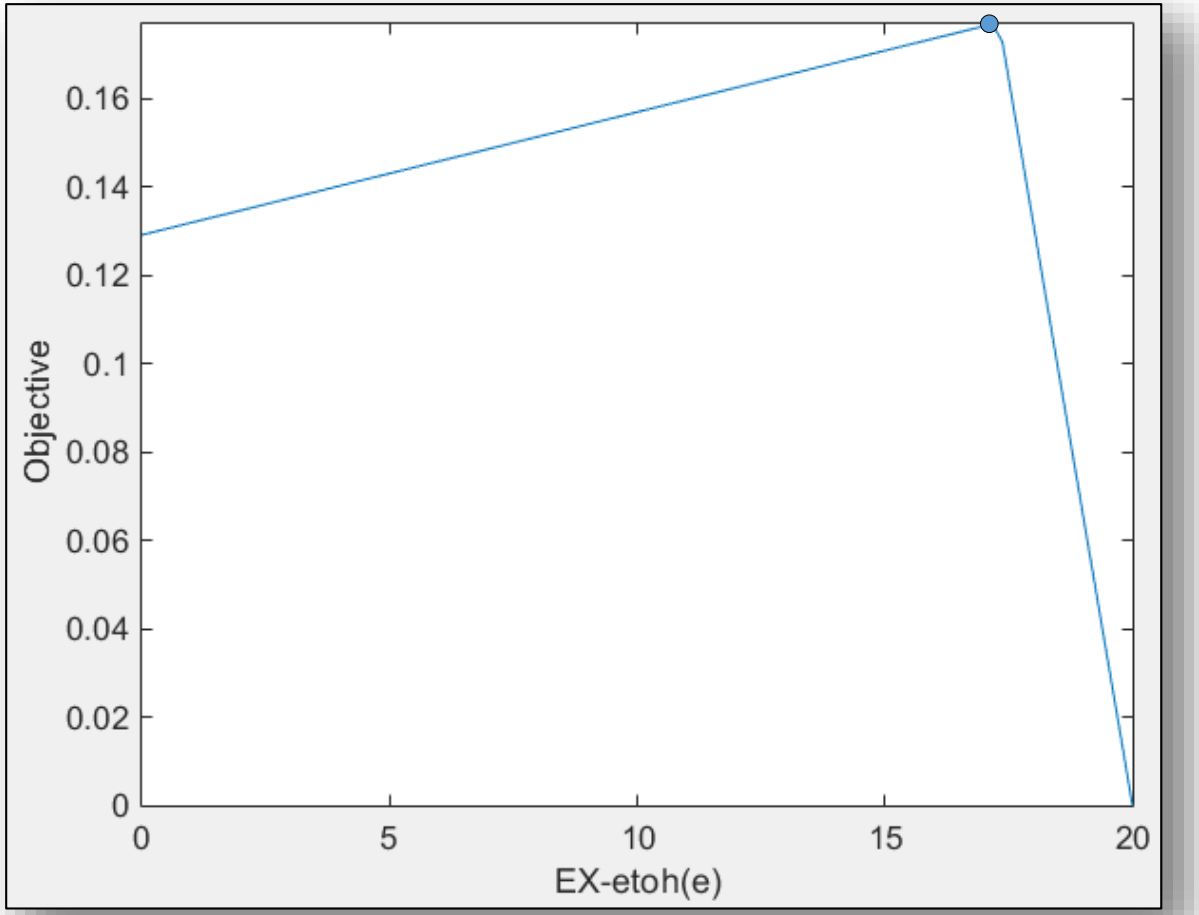


Ethanol Production: Robustness Analysis

3 Knockouts (EthanolProduction_Mutant.m)

```
clear; clc;
model = readCbModel('ecoli_core_model.mat');

model = changeRxnBounds(model,'EX_glc(e)',-10,'l');
model = changeRxnBounds(model,'EX_o2(e)',0,'l');
model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

% Knockout reactions
model = changeRxnBounds(model,'ACKr',0,'b');
model = changeRxnBounds(model,'G6PDH2r',0,'b');
model = changeRxnBounds(model,'GLUDy',0,'b');

FBAsolution = optimizeCbModel(model,'max')
printFluxVector(model,FBAsolution.x, true, true)

map=readCbMap('ecoli_Textbook_ExportMap');
options.lb = -10;
options.ub = 10;
options.zeroFluxWidth = 0.1;
options.rxnDirMultiplier = 10;
drawFlux(map, model, FBAsolution.x, options);

robustnessAnalysis(model,'EX_etoh(e)',100);
```

Knockouts

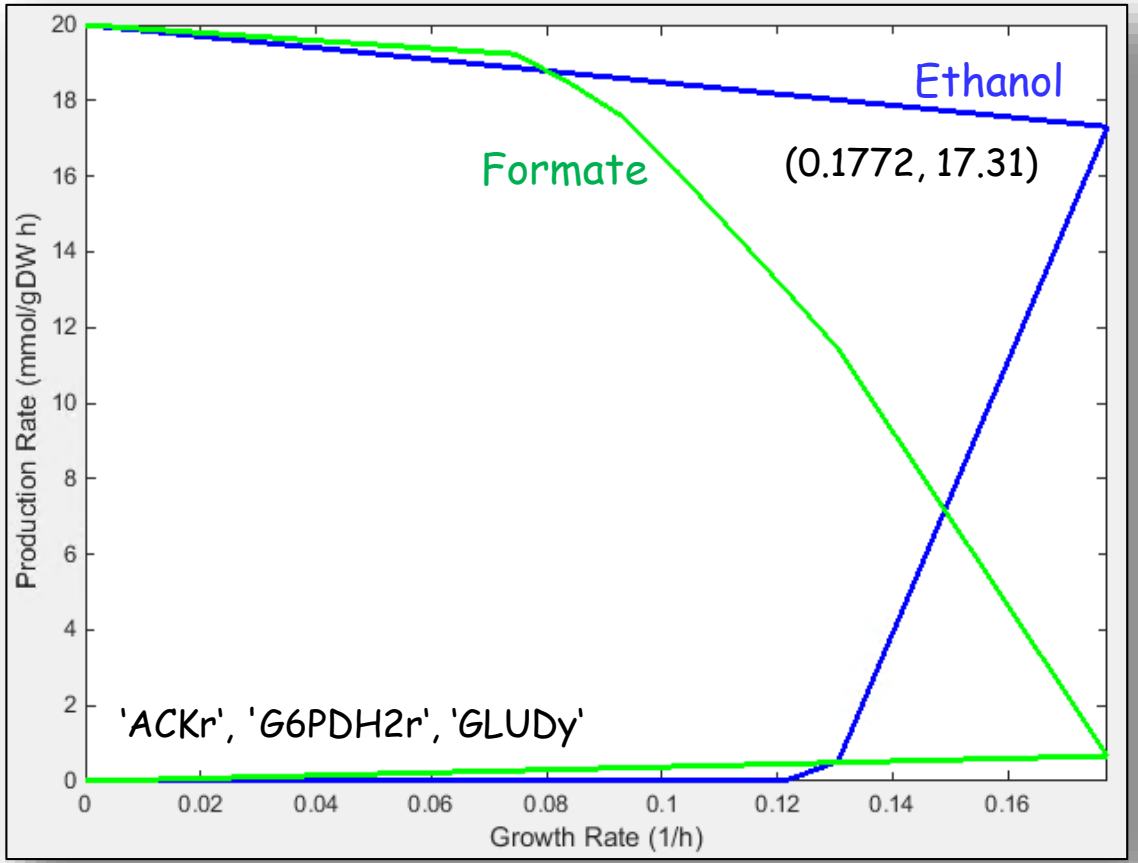


Ethanol Production: Production Envelope

3 Knockouts (EthanolProduction_multiProductionEnvelope.m)

```
% EthanolProduction_multiProductionEnvelope.m

clear;

model = readCbModel('ecoli_core_model.mat');

model = changeRxnBounds(model,'EX_glc(e)',-10,'l');

model = changeRxnBounds(model,'EX_o2(e)',-0,'l');

model = changeObjective(model,'Biomass_Ecoli_core_N(w/GAM)-Nmet2');

deletions = {'ACKr','GLUDy','G6PDH2r'};

biomassRxn = {'Biomass_Ecoli_core_N(w/GAM)_Nmet2'};

%Show only growth coupled metabolites

figure(1)

[biomassValues,targetValues] = multiProductionEnvelope(model,deletions,biomassRxn,false,20,false);
```

Knockouts



Mapping FVA Classifications

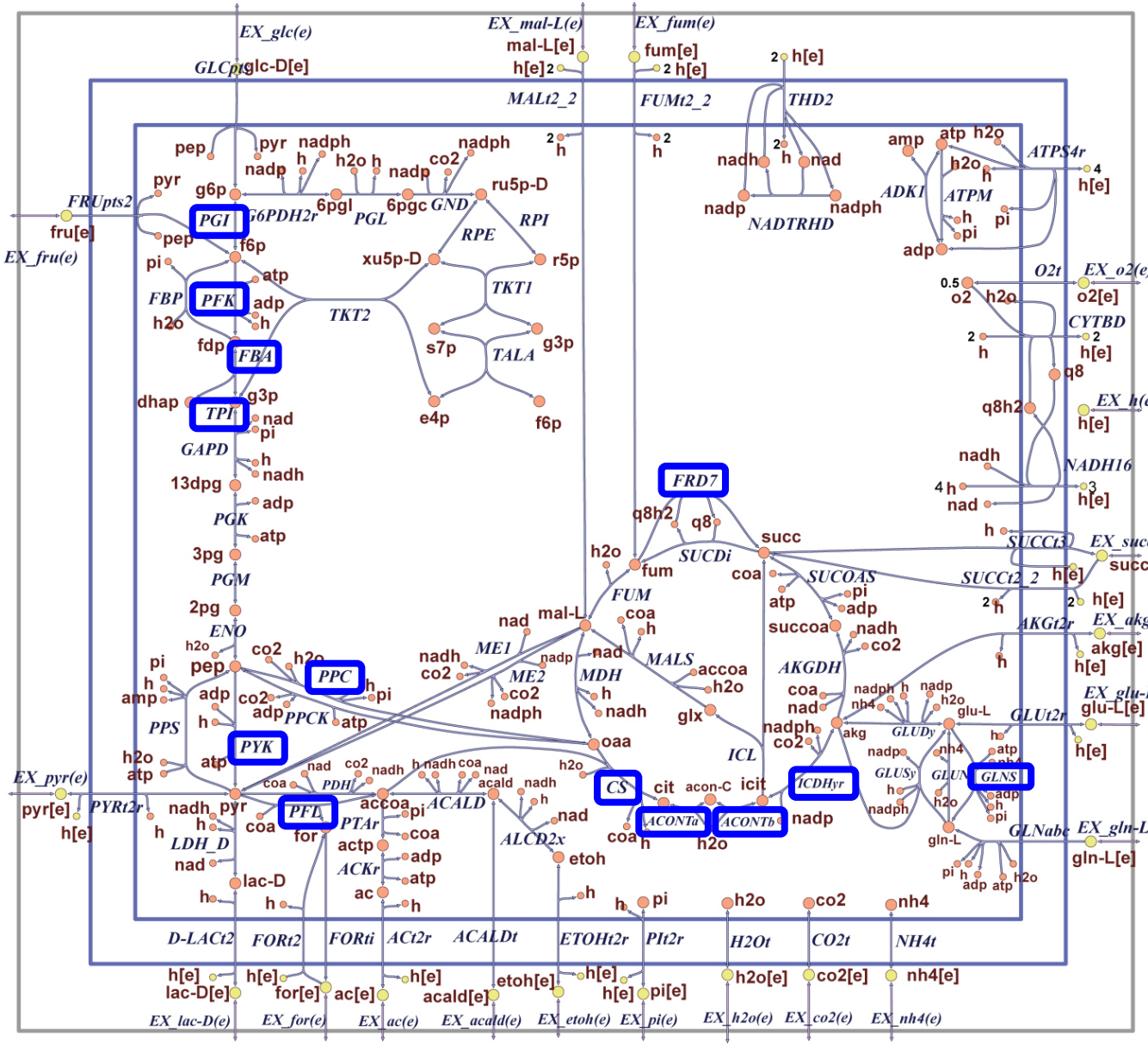
(Removing Transport and Exchange Reactions)

Partially-Coupled Reactions

'ACONTa'
'ACONTb'
'CS'
'FBA'
'FRD7'
'GLNS'
'ICDHyr'
'PFK'
'PFL'
'PGI'
'PPC'
'PTAr'
'PYK'
'TPI'

Not-Coupled Reactions

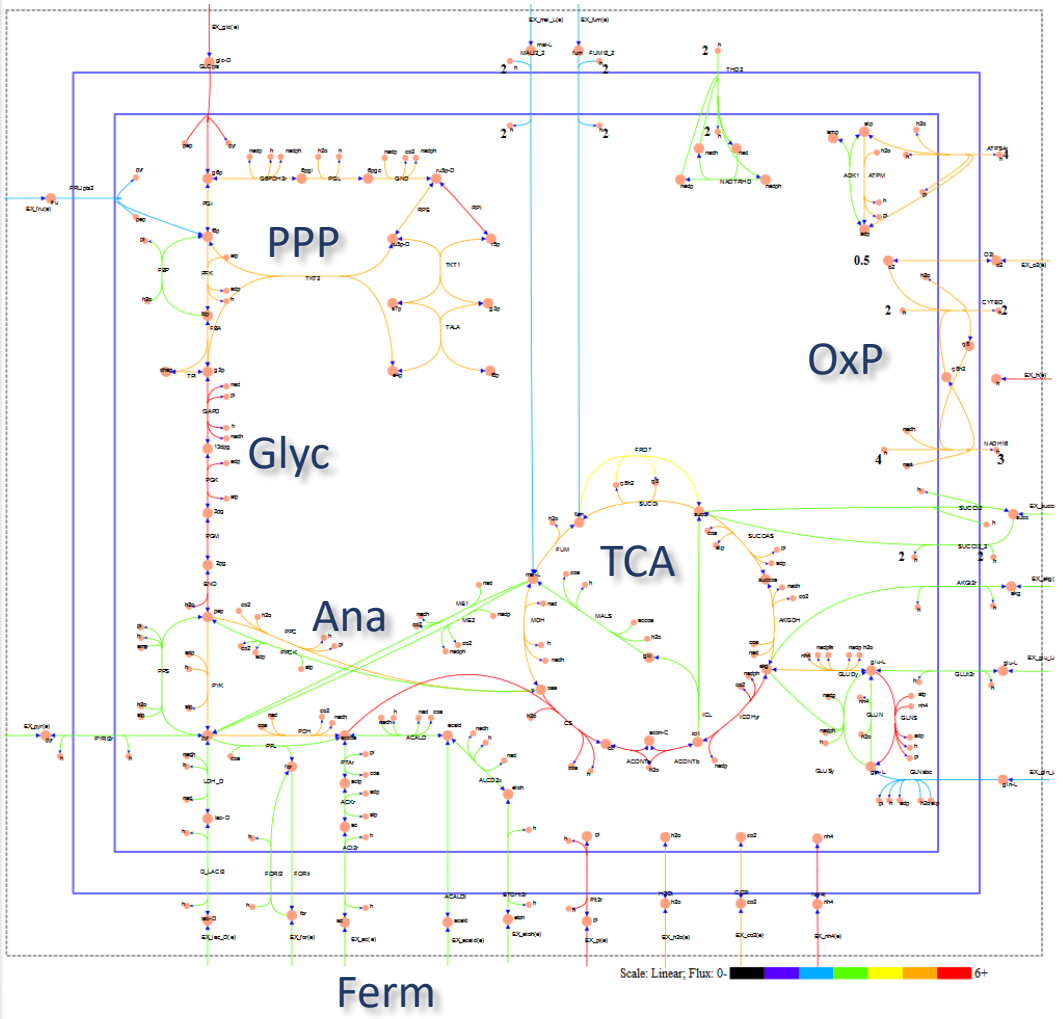
'ACALD'
'ACKr'
'ADK1'
'AKGDH'
'AKGt2r'
'ALCD2x'
'FBP'
'FUM'
'G6PDH2r'
'GLUDy'
'GLUN'
'GLUSy'
'GLUt2r'
'GND'
'ICL'
'LDH_D'
'MALS'
'MDH'
'ME1'
'ME2'
'NADH16'
'NADTRHD'
'PDH'
'PGK'
'PGL'
'PGM'
'PPCK'
'PPS'
'RPE'
'RPI'
'SUCDi'
'SUCOAS'
'TALA'
'THD2'
'TKT1'
'TKT2'



Parsimonious FBA For Glucose in an Anaerobic Environment

Essential	pFBA Optima	Enzymatically Less Efficient	Metabolically Less Efficient		pFBA No-flux	Blocked
'ACONTa'	'ACALD'	'FRD7'	'ACALDt'	'ME2'	'CYTBD'	
'ACONTb'	'ACKr'	'SUCDi'	'ADK1'	'NADH16'	'EX_fru(e)'	
Biomass'	'ACt2r'		'AKGDH'	'NADTRHD'	'EX_fum(e)'	
'CS'	'ALCD2x'		'AKGt2r'	'PDH'	'EX_gln_L(e)'	
'ENO'	'ATPM'		'D_LACt2'	'PGL'	'EX_mal_L(e)'	
'EX_glc(e)'	'ATPS4r'		'EX_acald(e)'	'PPCK'	'EX_o2(e)'	
'EX_h(e)'	'CO2t'		'EX_akg(e)'	'PPS'	'FRUpts2'	
'EX_nh4(e)'	'ETOHt2r'		'EX_glu_L(e)'	'PYRt2r'	'FUMt2_2'	
'EX_pi(e)'	'EX_ac(e)'		'EX_lac_D(e)'	'SUCCt2_2'	'GLNabc'	
'FBA'	'EX_co2(e)'		'EX_pyr(e)'	'SUCCt3'	'MALt2_2'	
'GAPD'	'EX_etoh(e)'		'EX_succ(e)'	'SUCOAS'	'O2t'	
'GLCpts'	'EX_for(e)'		'FBP'			
'GLNS'	'EX_h2o(e)'		'FORt2'			
'ICDHyr'	'FORti'		'FUM'			
'NH4t'	'GLUDy'		'G6PDH2r'			
'PFK'	'H2Ot'		'GLUN'			
'PGI'	'PFL'		'GLUSy'			
'PGK'	'PTAr'		'GLUt2r'			
'PGM'	'PYK'		'GND'			
'PIt2r'	'RPE'		'ICL'			
'PPC'	'TALA'		'LDH_D'			
'RPI'	'THD2'		'MALS'			
'TPI'	'TKT1'		'MDH'			
	'TKT2'		'ME1'			

pFBA_Ecoli_Core.m (Anaerobic)



FVA Chart for Glucose-based Anaerobic Ethanol Production