What is flux balance analysis?

Supplementary Tutorial

In this tutorial, several examples of how FBA can be used to analyze constraint-based models are presented. These examples utilize the COBRA Toolbox[1] and *E. coli* core model[2], which were introduced in the main text. Many COBRA methods, including some that are covered in this tutorial, are also presented in the original COBRA Toolbox paper. The forthcoming COBRA Toolbox 2.0 will also include a test suite of functions that will demonstrate many of these methods. A map of the core model is shown in **Supplementary Figure 1**. Formal reaction name abbreviations are listed in blue text, formal metabolite name abbreviations are listed in purple text, and COBRA Toolbox code is listed in `Courier text`.

Beginner COBRA methods

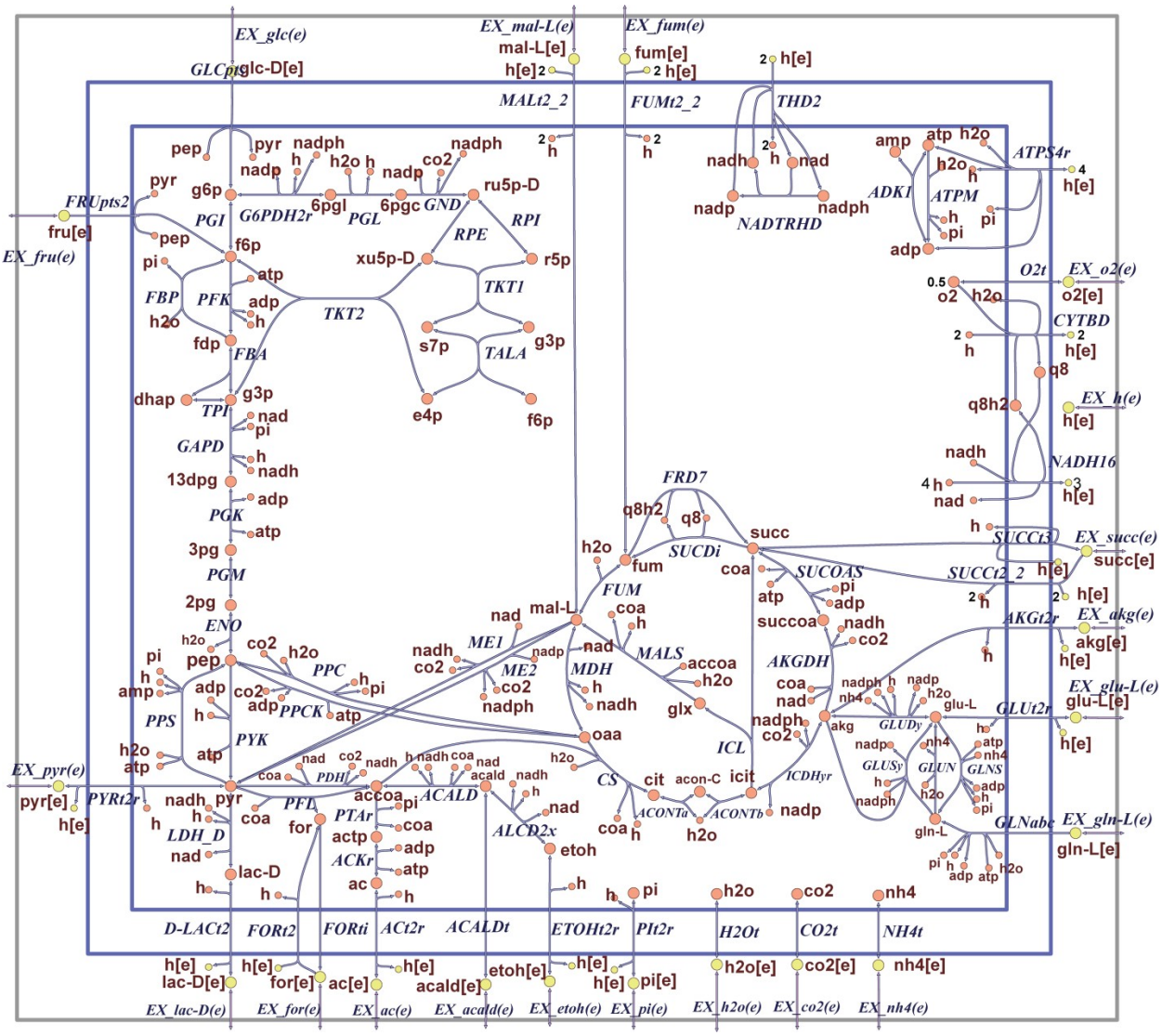
Example from the main text: Calculating growth rates
Supplementary Example 1. Growth on alternate substrates
Supplementary Example 2. Production of cofactors and biomass precursors
Supplementary Example 3. Alternate optimal solutions
Supplementary Example 4. Robustness analysis
Supplementary Example 5. Phenotypic phase planes
Supplementary Example 6. Simulating gene knockouts
Supplementary Example 7. Which genes are essential for which biomass precursor?
Supplementary Example 8. Which non-essential gene knockouts have the greatest effect on the network flexibility?

Non-FBA based COBRA methods

Supplementary Example 9. Analysis of the topological features of the **S** matrix
Supplementary Example 10. Characterization of functional states by random sampling

Advanced COBRA methods

FBA with regulation (Covert *et al.*[3])
MOMA (minimization of metabolic adjustment) (Segre *et al.*[4])
ROOM (regulatory on/off minimization) (Shlomi *et al.*[5])
SMILEY (Reed *et al.*[6])
OMNI (Herrgard et al.[7])
GapFind and GapFill (Satish Kumar et al.[8])
GrowMatch (Kumar and Maranas[9])
OptKnock (Burgard *et al.*[10])
OptGene (Patil *et al.*[11])
OptStrain (Pharkya et al.[12])



Supplementary Figure 1 Map of the core *E. coli* metabolic network. Orange circles represent cytosolic metabolites, yellow circles represent extracellular metabolites, and the blue arrows represent reactions. Reaction name abbreviations are uppercase and metabolite name abbreviations are lowercase.

Example from the main text: calculating growth rates

This section will demonstrate how to perform the FBA calculations referenced in the main text. Growth of *E. coli* on glucose can be simulated under aerobic conditions. To set the maximum glucose uptake rate to 18.5 mmol gDW^{-1} hr^{-1} (millimoles per gram dry cell weight per hour, the default flux units used in the COBRA Toolbox), enter into Matlab:

```
model = changeRxnBounds(model,'EX_glc(e)',-18.5,'l');
```

This changes the lower bound (`'l'`) of the glucose exchange reaction to -18.5, a biologically realistic uptake rate. By convention, exchange reactions are written as

export reactions (e.g. '`glc[e] <==>`'), so import of a metabolite is a negative flux. To allow unlimited oxygen uptake, enter:

```
model = changeRxnBounds(model,'EX_o2(e)',-1000,'l');
```

By setting the lower bound of the oxygen uptake reaction to such a large number, it is practically unbounded. Next, to ensure that the biomass reaction is set as the objective function, enter:

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

To perform FBA with maximization of the biomass reaction as the objective, enter:

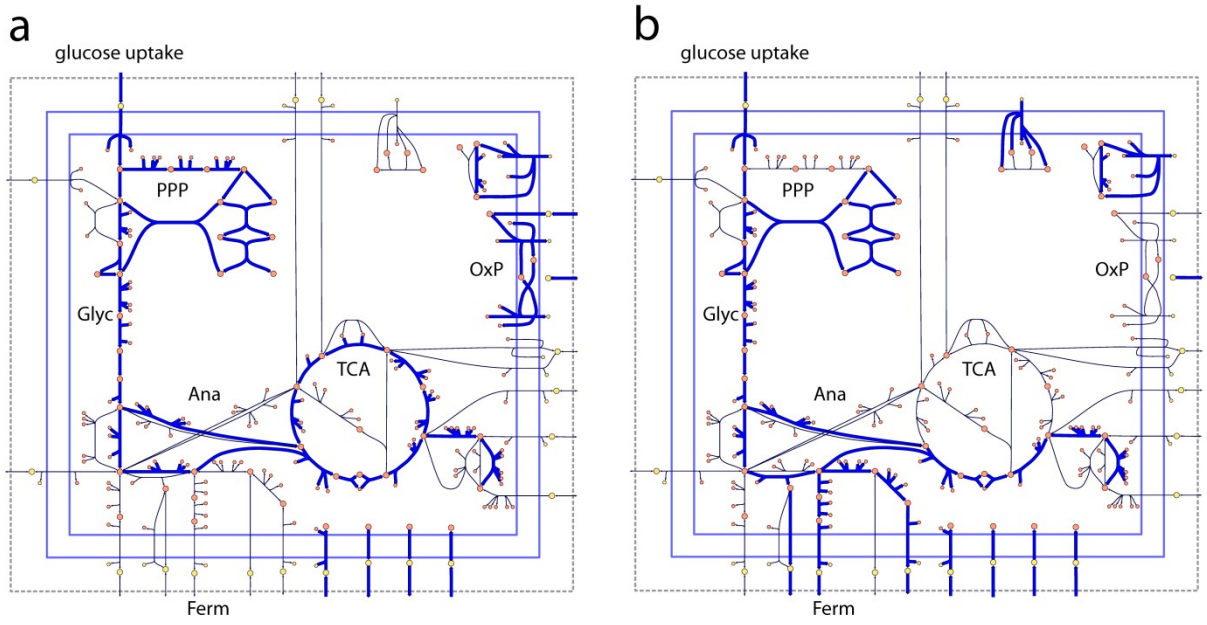
```
FBAsolution = optimizeCbModel(model,'max');
```

`FBAsolution.f` then gives the value of the objective function (Z) as 1.6531. This means that the model predicts a growth rate of 1.6531 hr^{-1}. Inspection of the flux distribution vector `FBAsolution.x` (**v**) shows that there is high flux in the glycolysis, pentose phosphate, TCA cycle, and oxidative phosphorylation pathways, and that no organic by-products are secreted (**Supplementary Figure 2a**).

Next, the same simulation is performed under anaerobic conditions. With the same model, enter:

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

The lower bound of the oxygen exchange reaction is now 0, so oxygen may not enter the system. When `optimizeCbModel` is used as before, the resulting growth rate is now much lower, 0.4706 hr^{-1}. The flux distribution shows that oxidative phosphorylation is not used in these conditions, and that acetate, formate, and ethanol are produced by fermentation pathways (**Supplementary Figure 2b**).



Supplementary Figure 2 Flux distributions computed by FBA can be visualized on network maps. In these two examples, the thick blue arrows represent reactions carrying flux, and the thin black arrows represent unused reactions. These maps show the state of the *E. coli* core model with maximum growth rate as the objective (*Z*) under aerobic (**a**) and anaerobic (**b**) conditions. Reactions that are in use have thick blue arrows, while reactions that carry 0 flux have thin black arrows. The metabolic pathways shown in these maps are glycolysis (Glyc), pentose phosphate pathway (PPP), TCA cycle (TCA), oxidative phosphorylation (OxP), anaplerotic reactions (Ana), and fermentation pathways (Ferm). These flux maps were drawn using SimPheny and edited for clarity with Adobe Illustrator.

Supplementary Example 1. Growth on alternate substrates

Just as FBA was used to calculate growth rates of *E. coli* on glucose in the main text, it can also be used to simulate growth on other substrates. The core *E. coli* model contains exchange reactions for 13 different organic compounds, each of which can be used as the sole carbon source under aerobic conditions. For example, to simulate growth on succinate instead of glucose, first use the `changeRxnBounds` function to set the lower bound of the glucose exchange reaction (EX_glc(e)) to 0. Then use `changeRxnBounds` to set the lower bound of the succinate exchange reaction (EX_succ(e)) to -20 mmol gDW^{-1} hr^{-1} (an arbitrary uptake rate). As in the glucose examples, make sure that Biomass_Ecoli_core_w_GAM is set as the objective (the function `checkObjective` can be used to identify the objective reaction(s)), and use `optimizeCbModel` to perform FBA. The growth rate, given by `FBAsolution.f`, will be 0.8401 hr^{-1}. The full code to calculate growth on succinate (with the model starting with its default bounds and objective) is:

```
model = changeRxnBounds(model,'EX_glc(e)',0,'l');
model = changeRxnBounds(model,'EX_succ(e)',-20,'l');
FBAsolution = optimizeCbModel(model,'max');
```

Growth can also be simulated under anaerobic conditions with any substrate by using `changeRxnBounds` to set the lower bound of the oxygen exchange reaction (EX_o2(e)) to 0 mmol gDW^{-1} hr^{-1}, so no oxygen can enter the system. When this constraint is applied and succinate is the only organic substrate, `optimizeCbModel` returns a growth rate of 0 hr^{-1}, and does not calculate a flux vector **v** (depending on which linear programming solver is used with the COBRA Toolbox, a growth rate may not be calculated at all). In this case, FBA predicts that growth is not possible on succinate under anaerobic conditions. Because the maximum amount of ATP that can be produced from this amount of succinate is less than the minimum bound of 8.39 mmol gDW^{-1} hr^{-1} of the ATP maintenance reaction, ATPM, there is no feasible solution. FBA predicted growth rates for all 13 organic substrates in the *E. coli* core model under both aerobic and anaerobic conditions are shown in **Supplementary Table 1**. The growth rates are all much lower anaerobically (0 hr^{-1} in most cases) because the electron transport chain cannot be used to fully oxidize the substrates and generate as much ATP.

Supplementary Table 1 The maximum growth rate of the core *E. coli* model on its 13 different organic substrates, computed by FBA. Growth rate was calculated for both aerobic and anaerobic conditions for each substrate, and the maximum substrate uptake rate was set to 20 mmol gDW^{-1} hr^{-1} for every substrate.

	Growth Rate (hr^{-1})	
Substrate	**Aerobic**	**Anaerobic**
acetate	0.3893	0
acetaldehyde	0.6073	0
2-oxoglutarate	1.0982	0
ethanol	0.6996	0
D-fructose	1.7906	0.5163
fumarate	0.7865	0
D-glucose	1.7906	0.5163
L-glutamine	1.1636	0
L-glutamate	1.2425	0
D-lactate	0.7403	0
L-malate	0.7865	0
pyruvate	0.6221	0.0655
succinate	0.8401	0

Supplementary Example 2. Production of cofactors and biomass precursors

FBA can also be used to determine the maximum yields of important cofactors and biosynthetic precursors from glucose and other substrates[13]. In this example, the maximum yields of the cofactors ATP, NADH, and NADPH from glucose under

aerobic conditions are calculated. To calculate optimal ATP production, first use `changeRxnBounds` to constrain the glucose exchange reaction (EX_glc(e)) to exactly -1 mmol gDW^{-1} hr^{-1} by setting both the lower and upper bounds to -1 (`'b'`). Next, set the ATP maintenance reaction (ATPM) as the objective to be maximized using `changeObjective`. ATPM is a stoichiometrically balanced reaction that hydrolyzes ATP (atp[c]) and produces ADP (adp[c]), inorganic phosphate (pi[c]), and a proton (h[c]). It works as an objective for maximizing ATP production because in order to consume the maximum amount of ATP, the network must first produce ATP by the most efficient pathways available by recharging the produced ADP. The constraint on this reaction should be removed by using `changeRxnBounds` to set the lower bounds to 0. By default, this reaction has a lower bound of 8.39 mmol gDW^{-1} hr^{-1} to simulate non-growth associated maintenance costs. Use `optimizeCbModel` to calculate the maximum yield of ATP, which is 17.5 mol ATP/mol glucose. The full COBRA Toolbox code to perform this calculation (with the model starting with its default bounds and objective) is:

```
model = changeRxnBounds(model,'EX_glc(e)',-1,'b');
model = changeObjective(model,'ATPM');
model = changeRxnBounds(model,'ATPM',0,'l');
FBAsolution = optimizeCbModel(model,'max');
```

Calculation of the yields of NADH and NADPH one at a time can be performed in a similar manner. First, constrain ATPM to 0 mmol gDW^{-1} hr^{-1} flux (`'b'`) so the cell is not required to produce ATP, and also cannot consume any ATP using this reaction. Add stoichiometrically balanced NADH and NADPH consuming reactions using the function `addReaction`, and set these as the objectives using `changeObjective`. The maximum yields of ATP, NADH, and NADPH are shown in **Supplementary Table 2**. The full code to calculate the maximum NADH yield is:

```
model = changeRxnBounds(model,'EX_glc(e)',-1,'b');
model = changeRxnBounds(model,'ATPM',0,'b');
model = addReaction(model,'NADH_drain','nadh[c] -> nad[c] + h[c]');
model = changeObjective(model,'NADH_drain');
FBAsolution = optimizeCbModel(model,'max');
```

The sensitivity of an FBA solution is indicated by either shadow prices or reduced costs. Shadow prices are the derivative of the objective function with respect to the exchange flux of a metabolite. They indicate how much the addition of that metabolite will increase or decrease the objective. Reduced costs are the derivatives of the objective function with respect to an internal reaction with 0 flux, indicating how much each particular reaction affects the objective. In the COBRA Toolbox, shadow prices and reduced costs can be calculated by `optimizeCbModel`. The vector of *m* shadow prices is `FBAsolution.y` and the vector of *n* reduced costs is `FBAsoltuion.w`.

ATP production is limited by cellular proton balancing. The shadow price of cytosolic protons (h[c]) is -0.25, indicating that the addition of 4 mol protons/mol glucose to the system reduces ATP yield by 1 mol ATP/mol glucose. Protons are produced by various metabolic reactions and are also pumped into the cell by the ATP synthase reaction (ATPS4r). In order for the system to be at steady-state (**Sv** = 0), an equal number of protons must be pumped out by the electron transport chain

reactions or by excreting metabolites with symporters. If more ATP were to be produced by ATP synthase, it would import additional protons that have no way to escape the cell. The flux distribution for optimal ATP production is shown in **Supplementary Figure 3**.

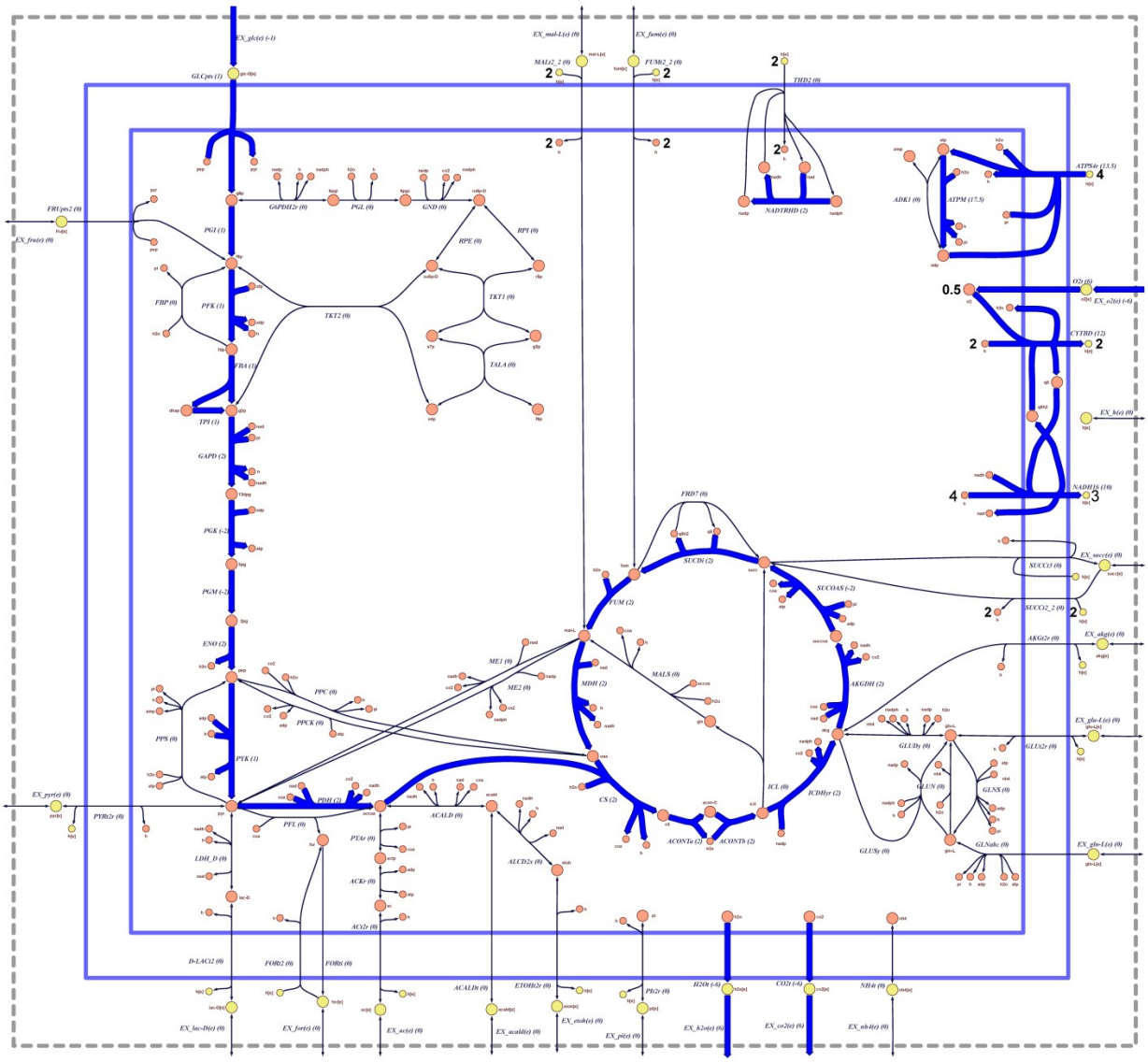
NADH and NADPH production are also ultimately limited by proton balancing. For maximum NADH yield, the proton shadow price is -0.1429. For maximum NADPH yield, the proton shadow price is -0.1111. The protons produced in metabolism are removed by ATPS4r in reverse (with a negative flux), which consumes ATP. The stoichiometry of the network also limits the production of NADH and NADPH. Glucose has 12 electron pairs that can be donated to redox carriers such as NAD^+ or $NADP^+$, but when the TCA cycle is used, two of these electron pairs are used to reduce the quinone q8[c] in the succinate dehydrogenase reaction (SUCDi), and thus cannot be used to produce NADH or NADPH. The only pathway that can reduce 12 redox carriers with one molecule glucose is the pentose phosphate pathway, but this is infeasible because this pathway generates no ATP, which is needed to pump out the protons that are produced.

The production of these cofactors can also be computed under anaerobic conditions by setting the lower bound of the oxygen exchange reaction (EX_o2(e)) to 0 mmol gDW^{-1} hr^{-1}. The results of these calculations are shown in **Supplementary Table 3**.

The core *E. coli* model contains 12 basic biosynthetic precursor compounds that are used to build macromolecules such as nucleic acids and proteins. The maximum yield of each of these precursor metabolites from glucose can be calculated by adding a demand reaction for each one (a reaction that consumes the compound without producing anything) and setting these as the objectives for FBA. Maximum yields of each of the 12 precursors are listed in **Supplementary Table 4**. Note that the drain reactions for acetyl-CoA (accoa[c]) and succinyl-CoA (succoa[c]) produce coenzyme-A (coa[c]), since this carrier molecule is not produced from glucose in the core model.

Supplementary Table 2 The maximum yields of the cofactors ATP, NADH, and NADPH from glucose under aerobic conditions. ATP Shadow Price is the shadow price of the metabolite atp[c], and indicates how much the addition of ATP to the system will increase the yield of the cofactor. Constraint indicates what is limiting constraints on the yields are. Energy constraints are due to the limited availability of ATP, while stoichiometry constraints indicate that the structure of the network prevents maximum yield.

Cofactor	Yield (mol/mol glc)	ATP Shadow Price	Constraint
ATP	17.5	0	H^+ balancing
NADH	10	0.5714	Energy, Stoichiometry
NADPH	8.778	0.4444	Energy, Stoichiometry



Supplementary Figure 3 Flux map for maximum ATP yield from glucose under aerobic conditions. Thick blue lines indicate reactions carrying flux in this particular solution vector. This is a unique solution (see **Supplementary Example 3**).

Supplementary Table 3 The maximum yields of the cofactors ATP, NADH, and NADPH from glucose under anaerobic conditions. ATP Shadow Price is the shadow price of the metabolite atp[c], and indicates how much the addition of ATP to the system will increase the yield of the cofactor. Constraint indicates what is limiting constraints on the yields are.

Cofactor	Yield (mol/mol glc)	ATP Shadow Price	Constraint
ATP	2.75	0	H^+ balancing
NADH	6	1	Energy
NADPH	4	1.333	Energy

Supplementary Table 4 The maximum yields of different biosynthetic precursors from glucose under aerobic conditions. The precursors are 3pg (3-phospho-D-glycerate), pep

(phosphoenolpyruvate), pyr (pyruvate), oaa (oxaloacetate), g6p (D-glucose-6-phosphate), f6p (D-fructose-6-phosphate), r5p (α-D-ribose-5-phosphate), e4p (D-erythrose-4-phosphate), g3p (glyceraldehyde-3-phosphate), accoa (acetyl-CoA), akg (2-oxoglutarate), and succoa (succinyl-CoA). Carbon Conversion indicates what percentage of the carbon atoms in glucose are converted to the precursor compound. ATP Shadow Price is the shadow price of the metabolite atp[c]. Constraint indicates what the limiting constraints on the yields are, preventing a yield of at least 100%. Some precursors have a yield of over 100% because carbon from CO_2 can be fixed in some reactions.

Precursor	Yield (mol/mol glc)	Carbon Conversion	ATP Shadow Price	Constraint
3pg	2	100%	0	-
pep	2	100%	0	-
pyr	2	100%	0	-
oaa	2	133.33%	0	-
g6p	0.8916	89.16%	0.0482	Energy
f6p	0.8916	89.16%	0.0482	Energy
r5p	1.0571	88.10%	0.0571	Energy
e4p	1.2982	86.55%	0.0702	Energy
g3p	1.6818	84.09%	0.0909	Energy
accoa	2	66.67%	0	Stoichiometry
akg	1	83.33%	0	Stoichiometry
succoa	1.64	109.33%	0	-

Supplementary Example 3. Alternate optimal solutions

As discussed in the main text, the flux distribution calculated by FBA is often not unique. In many cases, it is possible for a biological system to achieve the same objective value by using alternate pathways, so phenotypically different alternate optimal solutions are possible. A method that uses FBA to identify alternate optimal solutions is Flux Variability Analysis (FVA)[14]. This is a method that identifies the maximum and minimum possible fluxes through a particular reaction with the objective value constrained to be close to or equal to its optimal value. Performing FVA on a single reaction using the basic COBRA Toolbox functions is simple. First, use functions `changeRxnBounds`, `changeObjective`, and `optimizeCbModel` to perform FBA as described in the previous examples. Get the optimal objective value *Z* (`FBAsolution.f`), and then set both the lower and upper bounds of the objective reaction to exactly this value. Next, set the reaction of interest as the objective, and use FBA to minimize and maximize this new objective in two separate steps. This will give the minimum and maximum possible fluxes through this reaction while contributing to the optimal objective value.

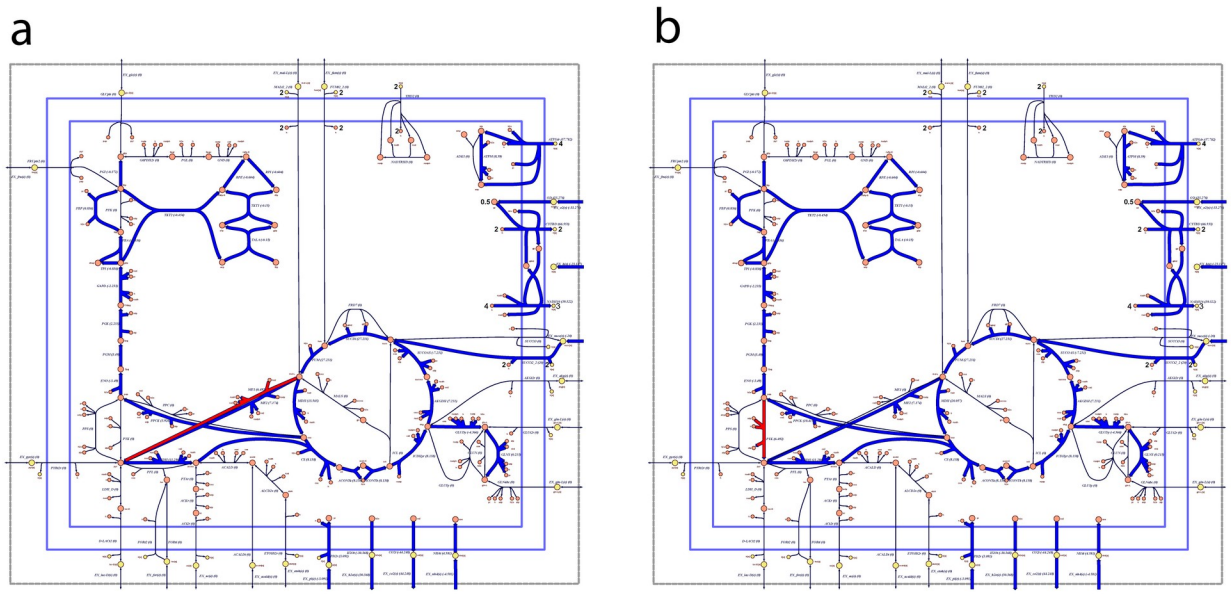
For example, consider the variability of the malic enzyme reaction (ME1) in *E. coli* growing on succinate. The minimum possible flux through this reaction is 0 mmol gDW^{-1} hr^{-1} and the maximum is 6.49 mmol gDW^{-1} hr^{-1}. In one alternate optimal solution, the ME1 reaction is used, but in another, it is not used at all. The full code to set the model to these conditions and perform FVA on this reaction is:

```
model = changeRxnBounds(model,'EX_glc(e)',0,'l');
model = changeRxnBounds(model,'EX_succ(e)',-20,'l');
FBAsolution = optimizeCbModel(model,'max');
model = changeRxnBounds(model,'Biomass_Ecoli_core_w_GAM',FBAsolution.f,'b');
model = changeObjective(model,'ME1');
FBAsolutionMin = optimizeCbModel(model,'min');
FBAsolutionMax = optimizeCbModel(model,'max');
```

The COBRA Toolbox includes a built in function for performing FVA called `fluxVariability`. This function is useful because it performs FVA on every reaction in a model. When FVA is performed on every reaction in the *E. coli* core model for growth on succinate, eight reactions are found to be variable (**Supplementary Table 5**). Inspection of the variable reactions shows that conversion of L-matate to pyruvate may occur through several different pathways, each leading to the same maximum growth rate. Flux distributions using combinations of these pathways are also valid solutions. Two of these alternate solutions are shown in **Supplementary Figure 4**.

Supplementary Table 5 Variable reactions for growth on succinate (uptake rate = 20 mmol gDW^{-1} hr^{-1}) under aerobic conditions. The minimum and maximum possible flux for every reaction was calculated at the maximum growth rate and only reactions with variable fluxes are shown here. FRD7 (fumarate reductase) and SUCDi (succinate dehydrogenase) always have highly variable fluxes in this model because they form a cycle that can carry any flux. Physiologically, these fluxes are not relevant. The other variable reactions are MDH (malate dehydrogenase), ME1 (malic enzyme (NAD)), ME2 (malic enzyme (NADP)), NADTRHD (NAD transhydrogenase), PPCK (phosphoenolpyruvate carboxykinase), and PYK (pyruvate kinase).

Reaction	Minimum Flux (mmol gDW^{-1} hr^{-1})	Maximum Flux (mmol gDW^{-1} hr^{-1})
FRD7	0	972.77
MDH	13.56	20.06
ME1	0	6.49
ME2	7.17	13.67
NADTRHD	0	6.49
PPCK	3.93	10.42
PYK	0	6.49
SUCDi	27.23	1000



Supplementary Figure 4 Flux maps for two alternate solutions for maximum aerobic growth on succinate. In (**a**), the reaction ME1 is used to convert L-malate to pyruvate, but in (**b**), this reaction is not used at all, and the reaction PYK is used. The two alternative reactions are highlighted in red.

Supplementary Example 4. Robustness analysis

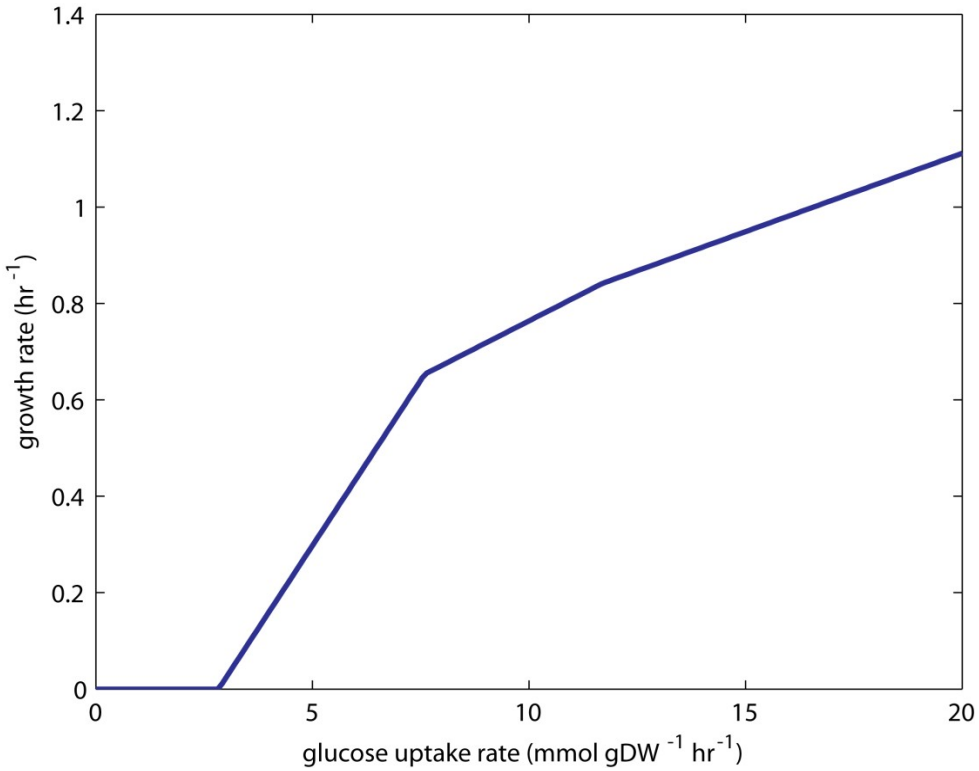
Another method that uses FBA to analyze network properties is robustness analysis[15]. In this method, the flux through one reaction is varied and the optimal objective value is calculated as a function of this flux. This reveals how sensitive the objective is to a particular reaction. There are many insightful combinations of reaction and objective that can be investigated with robustness analysis. One example is examination of the effects of nutrient uptake on growth rate. To determine the effect of varying glucose uptake on growth, first use `changeRxnBounds` to set the oxygen uptake rate (EX_o2(e)) to 17 mmol gDW^{-1} hr^{-1}, a realistic uptake rate. Then, use a `for` loop to set both the upper and lower bounds of the glucose exchange reaction to values between 0 and -20 mmol gDW^{-1} hr^{-1}, and use `optimizeCbModel` to perform FBA with each uptake rate. Be sure to store each growth rate in a vector or other type of Matlab list. The COBRA Toolbox also contains a function for performing robustness analysis (`robustnessAnalysis`) that can perform these functions. The full code to perform this robustness analysis is:

```
model = changeRxnBounds(model,'EX_o2(e)',-17,'b');
growthRates = zeros(21,1);
for i = 0:20
      model = changeRxnBounds(model,'EX_glc(e)',-i,'b');
      FBAsolution = optimizeCbModel(model,'max');
      growthRates(i+1) = FBAsolution.f;
end
```

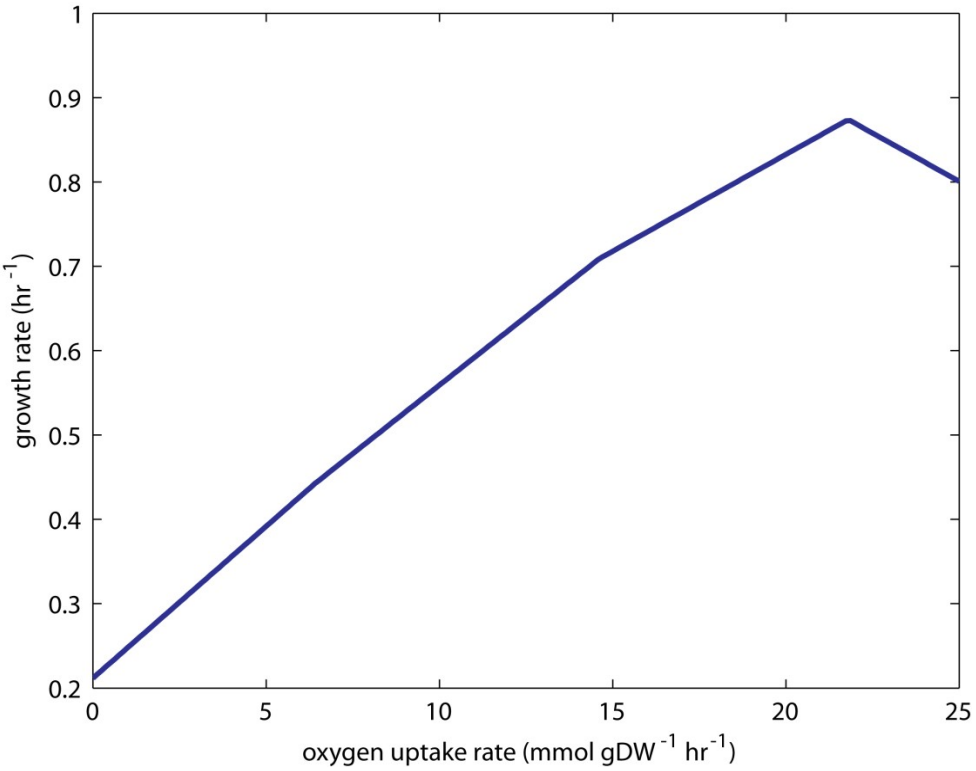
The results can then be plotted, as in **Supplementary Figure 5**. As expected, with a glucose uptake of 0 mmol gDW^{-1} hr^{-1}, the maximum possible growth rate is 0 hr^{-1}.

Growth remains at 0 hr^{-1} until a glucose uptake rate of about 2.83 mmol gDW^{-1} hr^{-1}, because with such a small amount of glucose, the system cannot make 8.39 mmol gDW^{-1} hr^{-1} ATP to meet the default lower bound of the ATP maintenance reaction (ATPM). This reaction simulates the consumption of ATP by non-growth associated processes such as maintenance of electrochemical gradients across the cell membrane. Once enough glucose is available to meet this ATP requirement, growth increases rapidly as glucose uptake increases. At an uptake rate of about 7.59 mmol gDW^{-1} hr^{-1}, growth does not increase as rapidly. It is at this point that oxygen and not glucose limits growth. Excess glucose cannot be fully oxidized, so the fermentation pathways are used.

The oxygen uptake rate can also be varied with the glucose uptake rate held constant. With glucose uptake fixed at 10 mmol gDW^{-1} hr^{-1}, growth rate increases steadily as oxygen uptake increases (**Supplementary Figure 6**). At an oxygen uptake of about 21.80 mmol gDW^{-1} hr^{-1}, growth actually begins to decrease as oxygen uptake increases. This is because glucose becomes limiting at this point, and glucose that would have been used to produce biomass must instead be used to reduce excess oxygen. In the previous example, growth rate continues to increase once oxygen become limiting because *E. coli* can grow on glucose without oxygen. In this example, *E. coli* cannot grow with oxygen but not glucose (or another carbon source), so growth decreases when excess oxygen is added.



Supplementary Figure 5 Robustness analysis for maximum growth rate while varying glucose uptake rate with oxygen uptake fixed at 17 mmol gDW^{-1} hr^{-1}.



Supplementary Figure 6 Robustness analysis for maximum growth rate while varying oxygen uptake rate with glucose uptake fixed at 10 mmol gDW^{-1} hr^{-1}.

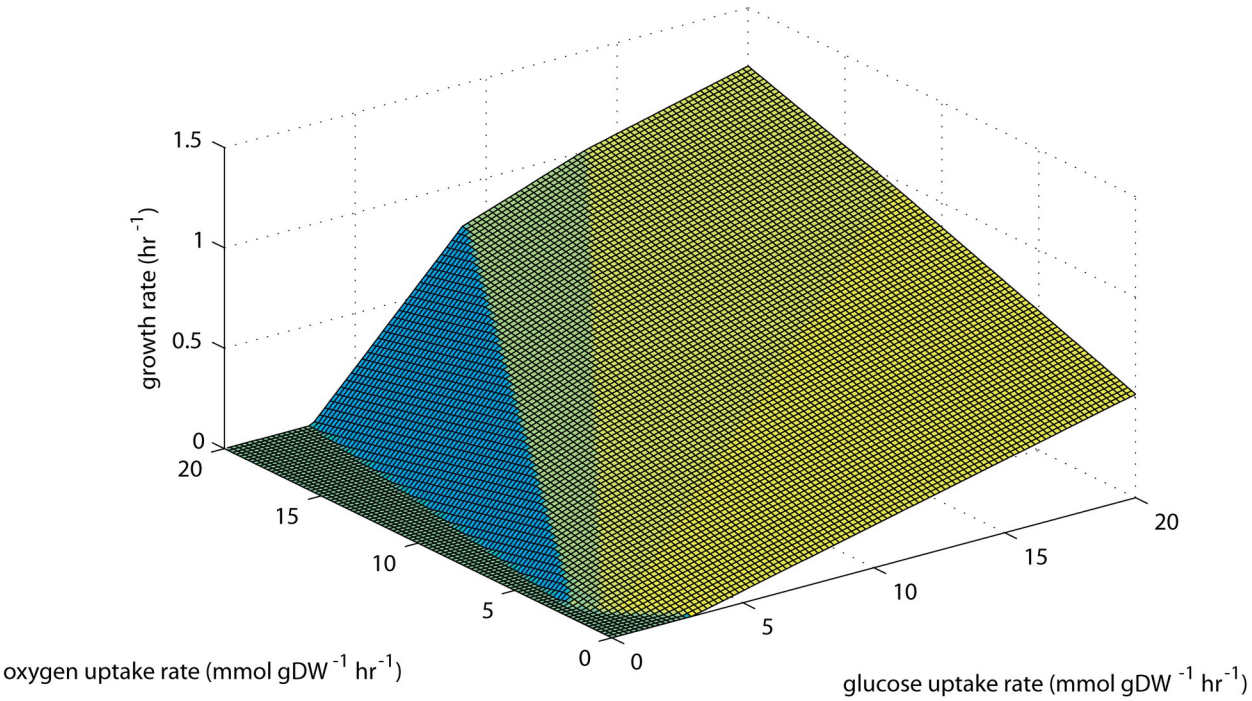
Supplementary Example 5. Phenotypic phase plane analysis

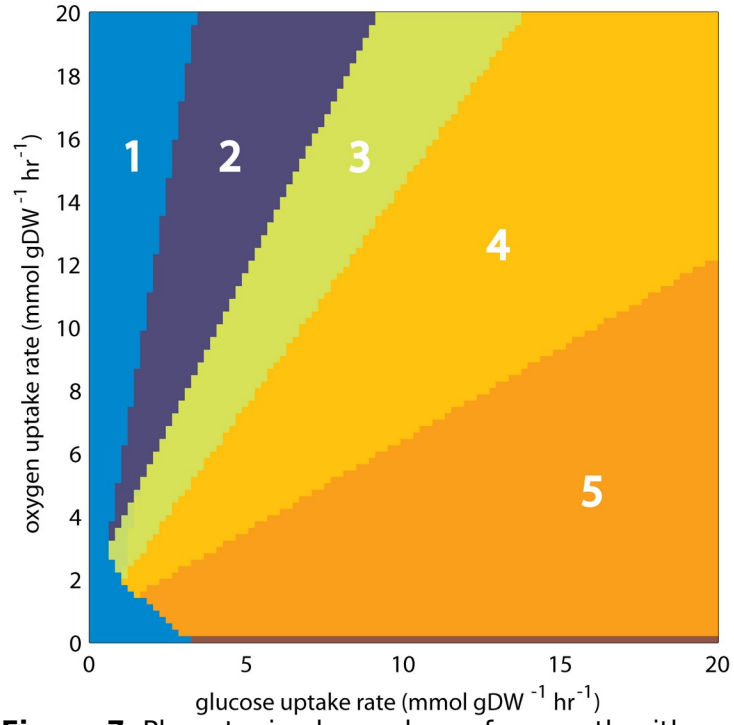
When performing robustness analysis, one parameter is varied and the network state is calculated. It is also possible to vary two parameters simultaneously and plot the results as a phenotypic phase plane[16]. These plots can reveal the interactions between two reactions in interesting ways. As an example, the phenotypic phase plane for maximum growth while varying glucose and oxygen uptake rates will be calculated. Although more sophisticated methods for computing phenotypic phase planes exist[17], they can be easily computed in a manner similar to the calculations for robustness analysis. Instead of using one `for` loop to vary one reaction, two nested `for` loops are used to vary two reactions. In this case, use `for` loops to vary the bounds of the glucose exchange reaction (EX_glc(e)) and oxygen exchange reaction (EX_o2(e)) between 0 and -20 mmol gDW^{-1} hr^{-1}. Use `optimizeCbModel` to perform FBA at each combination of glucose and oxygen uptake rates. The full code to perform the calculations is:

```
growthRates = zeros(21);
for i = 0:20
      for j = 0:20
            model = changeRxnBounds(model,'EX_glc(e)',-i,'b');
            model = changeRxnBounds(model,'EX_o2(e)',-j,'b');
            FBAsolution = optimizeCbModel(model,'max');
            growthRates(i+1,j+1) = FBAsolution.f;
```

```
        end
end
```

The resulting growth rates can be plotted as a 2-D graph or as a 3-D surface (**Supplementary Figure 7**). It is clear from these plots that this surface has 5 distinct regions, and each one is a flat plane. This is a general feature of phenotypic phase planes. They do not form curved surfaces or other shapes. Each of these phases has a qualitatively distinct phenotype, and all of the shadow prices (`FBAsolution.y`) are constant within each phase. Phase 1 (on the far left of the plots) is characterized by 0 growth. There is not enough glucose to meet the ATP maintenance requirement imposed by the ATPM reaction. In phase 2, growth is limited by oxygen. o2[e] has a shadow price of -0.0229 because there is not enough glucose to reduce all of the oxygen and produce biomass optimally. The line between phase 2 and phase 3 is where glucose and oxygen are perfectly balanced and growth yield is highest. In phases 3, 4, and 5, oxygen and glucose are both limiting growth. There is not enough oxygen to fully oxidize glucose, so various compounds are produced by fermentation.





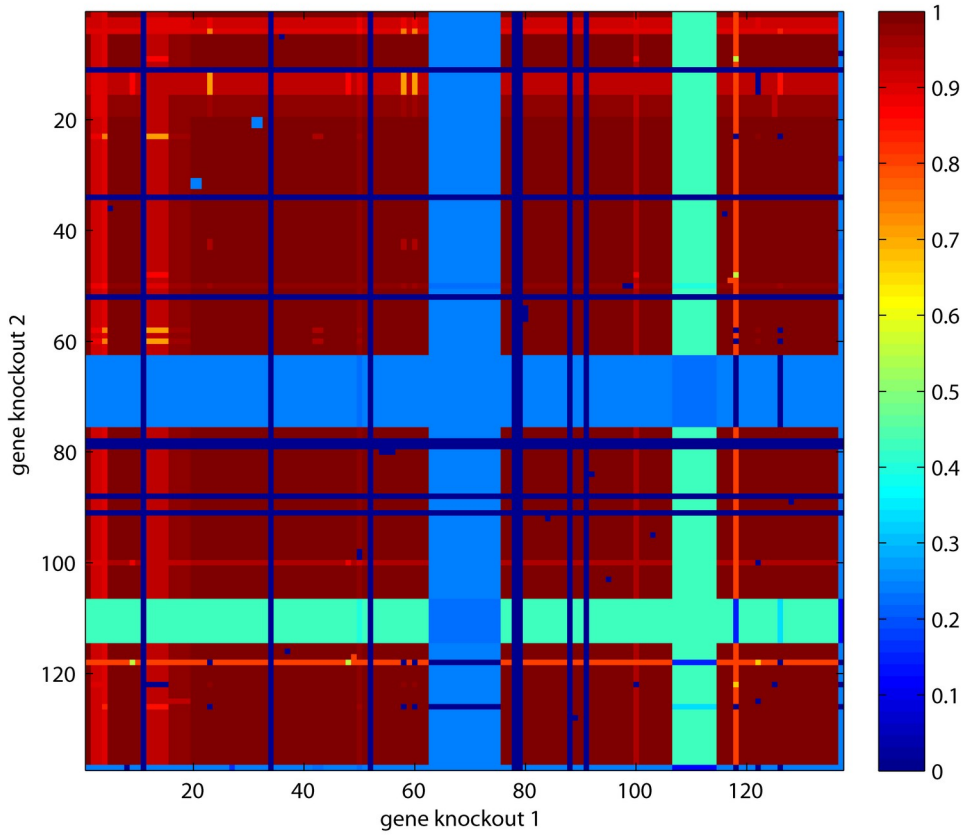
Supplementary Figure 7 Phenotypic phase planes for growth with varying glucose and oxygen uptake rates. In phase 1, no growth is possible. In phase 2, growth is limited by excess oxygen. In phase 3, acetate is secreted; in phase 4, acetate and formate are secreted; and in phase 5, acetate, formate, and ethanol are secreted. The 3-D plot was created using the Matlab function `surfl` and the 2-D plot was created with `pcolor`.

Supplementary Example 6. Simulating gene knockouts

Just as growth in different environments can be simulated with FBA, gene 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 of a reaction to 0 mmol gDW^{-1} hr^{-1}, a reaction is essentially knocked out, and is restricted from carrying flux. The *E. coli* core model, like many other constraint-based models, contains a list of gene-protein-reaction interactions (GPRs), a list of Boolean rules that dictate which genes are connected with each reaction in the model. When a reaction is catalyzed by isozymes (two different enzymes that catalyze the same reaction), the associated GPR contains an "or" rule, where either of two or more genes may be knocked out but the reaction will not be constrained. For example, the GPR for phosphofructokinase (PFK) is "b1723 (*pfkB*) or b3916 (*pfkA*)," so according to this Boolean rule, both *pfkB* and *pfkA* must be knocked out to restrict this reaction. When a reaction is catalyzed by a protein with multiple essential subunits, the GPR contains an "and" rule, and if any of the genes are knocked out the reaction will be constrained to 0 flux. Succinyl-CoA synthetase (SUCOAS), for example, has the GPR "b0728 (*sucC*) and b0729 (*sucD*)," so knocking out either of these genes will restrict this reaction. Some reactions are catalyzed by a single gene product, while others may be associated with ten or more genes in complex associations.

The COBRA Toolbox contains a function called `deleteModelGenes` that uses the GPRs to constrain the correct reactions. Then FBA may be used to predict the model phenotype with gene knockouts. For example, growth can be predicted for *E. coli* growing aerobically on glucose with the gene b1852 (*zwf*), corresponding to the reaction glucose-6-phospahte dehydrogenase (G6PDH2r), knocked out. The FBA predicted growth rate of this strain is 1.6329 hr^{-1}, which is slightly lower than the growth rate of 1.6531 hr^{-1} for wild-type *E. coli* because the cell can no longer use the oxidative branch of the pentose phosphate pathway to generate NADPH. Using FBA to predict the phenotypes of gene knockouts is especially useful in predicting essential genes. When the gene b2779 (*eno*), corresponding to the enolase reaction (ENO), is knocked out, the resulting growth rate on glucose is 0 hr^{-1}. Growth is no longer possible because there is no way to convert glucose into TCA cycle intermediates without this glycolysis reaction, so this gene is predicted to be essential. Because FBA can compute phenotypes very quickly, it is feasible to use it for large scale computational screens for gene essentiality, including screens for two or more simultaneous knockouts. **Supplementary Figure 8** shows the results of a double knockout screen, in which every pairwise combination of the 136 genes in the *E. coli* core model were knocked out. The code to produce this figure is:

```
[grRatio,grRateKO,grRateWT] = doubleGeneDeletion(model);
imagesc(grRatio)
xlabel('gene knockout 1')
ylabel('gene knockout 2')
```



Supplementary Figure 8 Gene knockout screen on glucose under aerobic conditions. Each of the 136 genes in the core *E. coli* model were knocked out in pairs, and the resulting relative growth rates were plotted. In this figure, genes are ordered by their b number. Some genes are always essential, and result in a growth rate of 0 when knocked out no matter which other gene is also knocked out. Other genes form synthetic lethal pairs, where knocking out only one of the genes has no effect on growth rate, but knocking both out is lethal. Growth rates in this figure are relative to wild-type.

Supplementary Example 7. Which genes are essential for which biomass precursor?

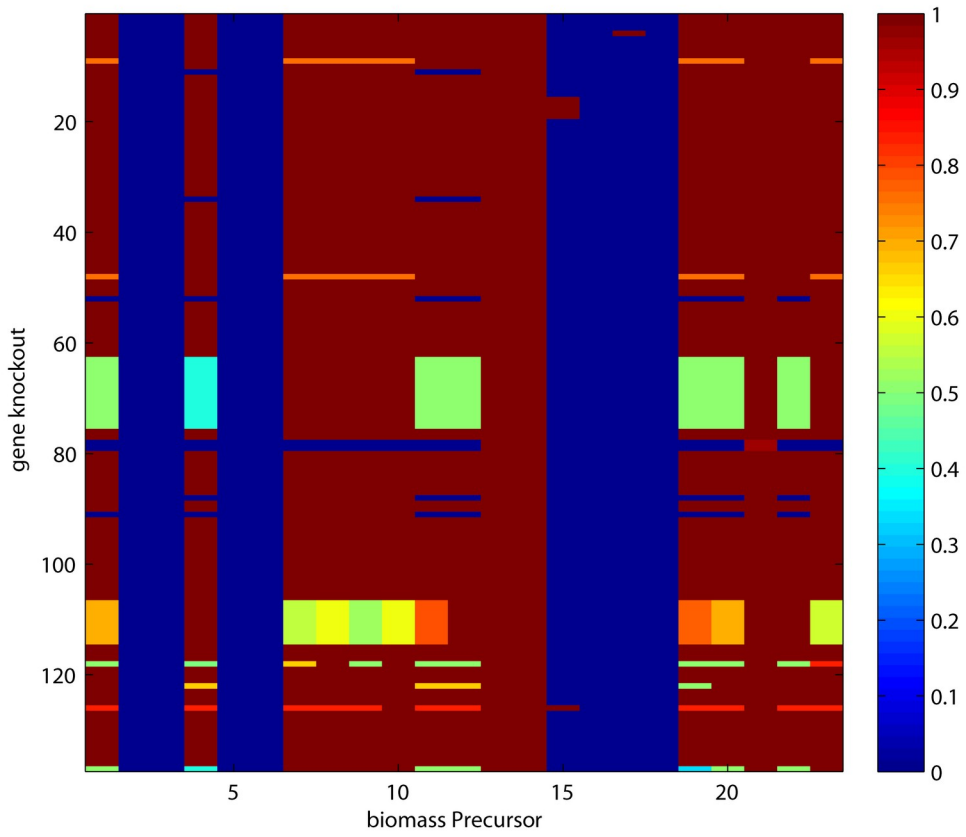
Earlier, we saw how to determine essential genes for growth. Here we will repeat the same analysis but instead of optimizing for the biomass production, we will optimize for the synthesis of all biomass precursors individually. Therefore, we have to add a demand reaction for each biomass precursor to the model and perform a gene deletion study for each demand reaction. First, the components of the biomass reaction can be identified and demand reactions can be added by using the `addDemandReaction` function:

```
[biomassComponents,biomassFraction] = printBiomass(model,13);
[modelBiomass,rxnNames] = addDemandReaction(model,biomassComponents);
```

Next, gene knockout screens can be performed with each of these demand reactions set as the objective, one at a time:

```
for i = 1:length(rxnNames)
    modelBiomass = changeObjective(modelBiomass,rxnNames{i});
    [grRatio,grRateKO,grRateWT,hasEffect,delRxns,fluxSolution] =
singleGeneDeletion(modelBiomass);
    biomassPrecursorGeneEss(:,i) = grRateKO;
    biomassPrecursorGeneEssRatio(:,i) = grRatio;
end
```

The resulting matrix `biomassPrecursorGeneEssRatio` is plotted with the `imagesc` function in **Supplementary Figure 9**, and indicate which biomass precursors become blocked by certain gene knockouts. Some precursors (like atp[c]) cannot be produced by any of the gene knockout strains because of the demand reactions that consume them. The `addDemandReacton` function produces a demand reaction that does not regenerate ADP when ATP is consumed or NAD^+ when NADH is consumed, so these reactions cannot carry flux at steady-state.



Supplementary Figure 9 Gene essentiality for biomass precursor synthesis. Heat map shows the relative biomass precursor synthesis rate of mutant compared to wild-type. The 23 biomass precursors are 3pg, accoa, adp, akg, atp, coa, e4p, f6p, g3p, g6p, gln-L, glu-L, h2o, h, nad, nadh, nadp, nadph, oaa, pep, pi, pyr, r5p.

Supplementary Example 8. Which non-essential gene knockouts have the greatest effect on the network flexibility?

Another question one might be interested in asking is what are the consequences for the network of deleting a non-essential gene? To address this question, first delete the network genes individually (a single gene deletion study) and then perform FVA (**Supplementary Example 3**) on non-essential gene deletion strains. In many cases one would expect that the deletion of a gene has only minor consequences to the network, especially if the deletion does not alter the maximal growth rate. In some cases however, the deletion may reduce the overall network flexibility or maybe even increase the flux range through specific reactions. In fact, this latter property is used to design optimal production strains (e.g., using OptKnock[10]), by redirecting fluxes through the network.

The absolute flux span is a measure of flux range for each reaction. It is calculated as $f_i = \text{abs}(v_{max,i} - v_{min,i})$, where $v_{min,i}$ and $v_{max,i}$ are the minimal and maximal flux rate as determined using FVA. First, calculate wild-type flux variability:

```
[minFluxAll(:,1),maxFluxAll(:,1)] = fluxVariability(model);
```

Next, calculate the knockout strain flux variablities and flux spans:

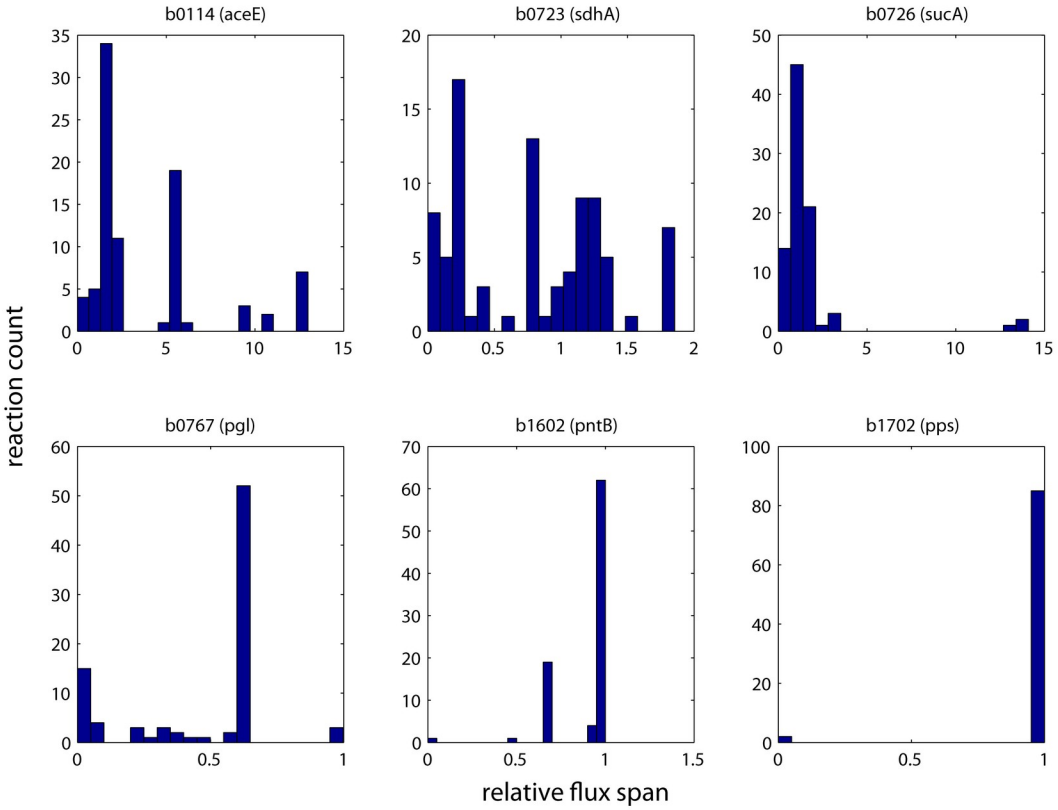
```
genes=model.genes([2,14,16,23,42,48]);
for i = 1 : length(genes)
    [modelDel] = deleteModelGenes(model,genes{i});
    [minFluxAll(:,i+1),maxFluxAll(:,i+1)] = fluxVariability(modelDel);
end

fluxSpan = abs(maxFluxAll - minFluxAll);
for i = 1 : size(fluxSpan,2)
    fluxSpanRelative(:,i) = fluxSpan(:,i)./fluxSpan(:,1);
end
```

Finally, histograms can be plotted to show how many reactions have increased or decreased flux spans for each knockout:

```
for i =2:7
    subplot(2,3,i-1)
    hist(fluxSpanRelative(:,i),20);
    title(genes{i-1});
end
```

This example with six mutant strains shows that the majority of the network reactions have reduced flux span compared to wild-type (**Supplementary Figure 10**). However, some of these knockout strains have a few reactions which have much higher flux span than wildtype. For example, in one of the knockout strains (Δ*sucA*), the flux span is increased 14 times through the phosphoenolpyruvate carboxykinase reaction (PPCK).



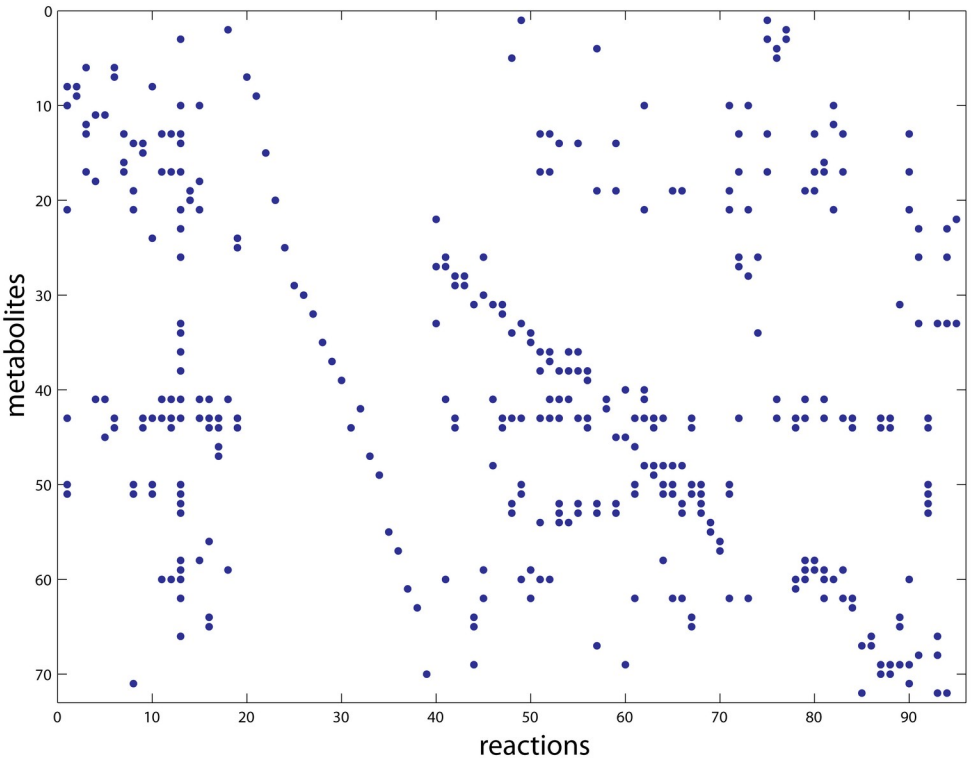
Supplementary Figure 10 Histograms of relative flux spans for 6 gene knockout mutants.

Supplementary Example 9. Topological features of the S matrix: metabolite connectivity and reaction participation

Visualization of the S matrix

The core *E. coli* model **S** matrix can be visualized by using the spy command in Matlab. This command will represent all non-zero entries in **S** with a dot, as shown in **Supplementary Figure 11**. The code to produce this figure is:

```
spy(model.S);
```



Supplementary Figure 11 The (72*95) **S** matrix of the *E. coli* core model. All non-zero entries are marked with a dot.

Determination of the number of reactions the metabolites occur in

The **S** matrix can be converted into a binary matrix ($\mathbf{S}_{bin}$), by replacing all non-zero elements in **S** with a '1' in $\mathbf{S}_{bin}$. Then, all ones in each row of the $\mathbf{S}_{bin}$ can be summed to determine the number of reactions a metabolite occurs in. The full code to binarize the **S** matrix and calculate and plot metabolite connectivity is:

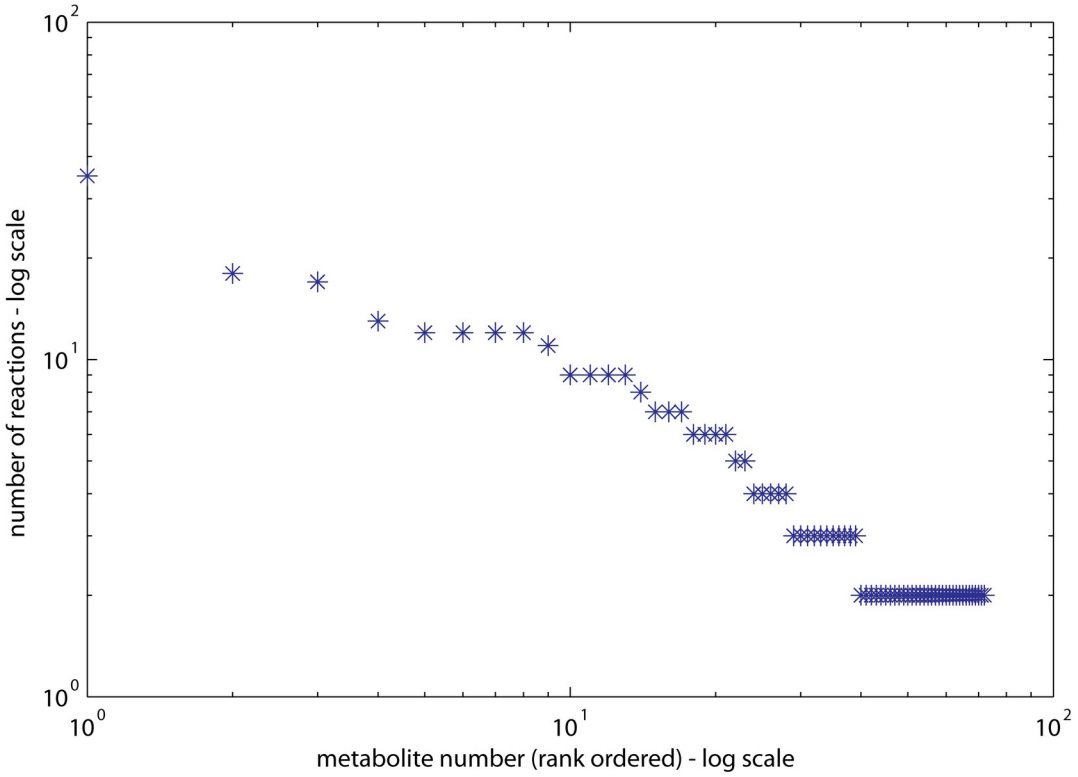
```
Sbin = zeros(size(model.S));
Sbin(find(model.S))=1;

for i = 1 : length(model.mets)
metConnectivity(i,1) = sum(Sbin(i,:));
end

loglog(sort(metConnectivity,'descend'),'*')
xlabel('metabolite number (rank ordered) - log scale');
ylabel('number of reactions - log scale')
```

As **Supplementary Figure 12** shows, there are very few metabolites that are highly connected, while most metabolites participate only in a few reactions. The approximate linear appearance of the curve of connectivities is surprising and corresponds to a power law distribution of metabolite connectivity. The few highly connected metabolites are "global" players, similar to hubs in protein-protein-

interaction networks, while the low connectivity metabolites are “local” players, many of which only occur in linear pathways. The power law distribution indicates that the networks are scale-free[18, 19].



Supplementary Figure 12 Connectivity of the core *E. coli* metabolites (loglog plot).

Reaction participation is the number of metabolites per reaction
To determine reaction participation for every reaction, the number of non-zero elements per column in $\mathbf{S}_{bin}$ is counted:

```
for i = 1 : length(model.rxns)
     rxnParticipation(i,1) = sum(Sbin(:,i));
end
```

In the *E. coli* core model, there are on average 3.8 metabolites per reaction (`mean(rxnParticipation)`). The most common type of reaction in the *E. coli* core metabolic network is the bi-linear reaction involving two substrates and two products.

How many metabolites are co-occurring in two reactions?
To determine the number of metabolites common to two reactions, one needs to calculate the compound adjacency matrix ($\mathbf{A}_{comp}$): $\mathbf{A}_{comp} = \mathbf{S}_{bin} * \mathbf{S}_{bin}^{T}$. This multiplication leads to a square matrix of size $m*m$, where m is the number of metabolites in the network.

```
Acomp = Sbin*Sbin';
```

The diagonal elements correspond to the metabolite connectivity (computed above), and can be extracted as follows:

```
metConnectivity = diag(Acomp);
```

The off-diagonal elements correspond to the number of reactions two metabolites are co-occurring in. The cofactor pair nad[c] and nadh[c] occurs in 12 reactions together. Moreover, the connectivity of both metabolites is 12, thus, they only occur as pairs in the core *E. coli* model.

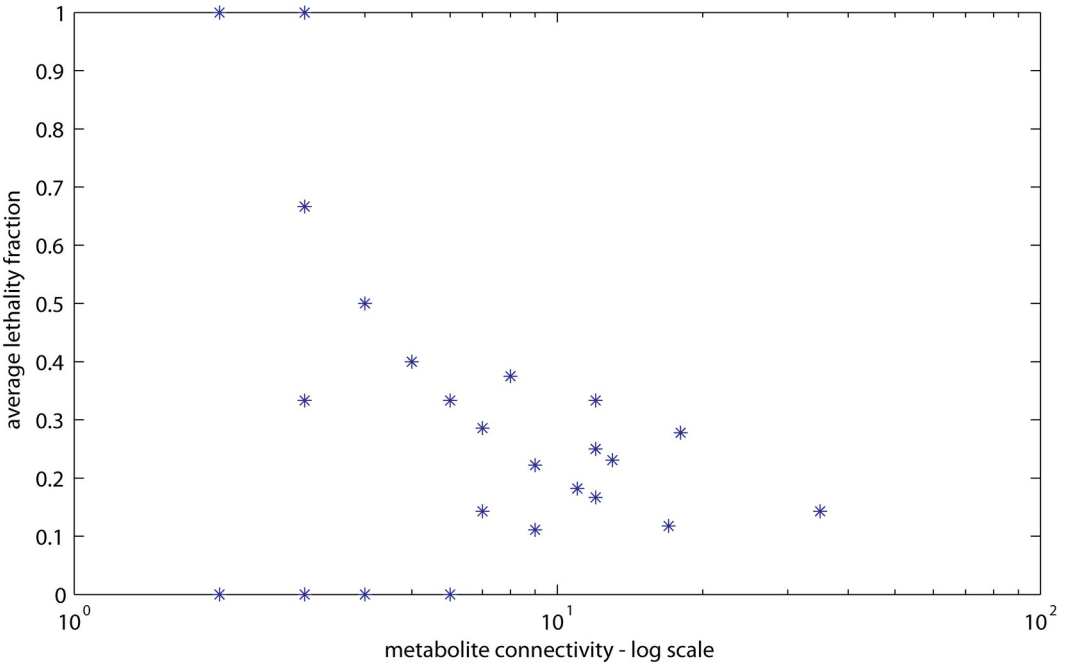
Is there any correlation between reaction essentiality and metabolite connectivity[20]?

One way to address this question is to delete all reactions in the network that are associated with a metabolite and ask if the network can still produce biomass. To find all reactions associated with a metabolite, one has to just scan through the corresponding rows in the **S** matrix. Then the associated genes can be knocked out, and FBA can be used to predict if growth is possible. The full code to perform these calculations and plot the results is:

```
growthDel = zeros(length(model.mets),1);
for i = 1 : length(model.mets)
    rxnID = find(model.S(i,:));
    for j = 1 : length(rxnID)
        modelDel = deleteModelRxn(model,model.rxns(rxnID(j)));
        FBAsolutionDel = optimizeCbModel(modelDel);
        if FBAsolutionDel.f <= 1e-6
            growthDel(i,1) = growthDel(i,1) + 1;
        end
    end
end

lethalityFraction = growthDel./metConnectivity;
semilogx(metConnectivity,lethalityFraction,'*')
xlabel('metabolite connectivity - log scale');
ylabel('average lethality fraction');
```

The results for the *E. coli* core model (**Supplementary Figure 13**) show that some less connected metabolites have a higher lethality fraction than highly connected metabolites. This has been found to be true for other, more complex metabolic networks[20]. In fact, for the *E. coli* core model, the average lethality fraction lies between 0.2 and 0.5 for the majority of the metabolites, regardless of their connectivity. There are a few compounds that have a connectivity of 2 and have a lethality fraction of 1 (e.g., cit[c], glc-D[e]). These metabolites often occur in a linear pathway, at the end of which an essential biomass precursor is produced.



Supplementary Figure 13 Correlation between metabolite connectivity and average lethality of reactions producing or consuming a particular metabolite. Less connected metabolites tend to occur in a higher fraction of essential reactions.

Supplementary Example 10. Characterization of functional states by random sampling

How probable is a flux through a reaction?

While flux variability analysis determines the minimum and maximum value of a reaction flux given some network constraints, Monte-Carlo-sampling can be used to determine a probability flux distribution for each network reaction[21, 22].

Sampling is an unbiased method, compared to many other biased methods in constraint-based modeling. Monte-Carlo-sampling involves a random walk from a starting point through the entire space. As the network size increases the number of steps required to reach all regions of the high-dimensional space will increase, making unbiased sampling more time-consuming. Consequently, a slightly biased approach (hit-and-run sampling) has been adapted for sampling of metabolic networks[22] that provides the sampling algorithm with a set of 'warm up points' (`warmupPts`)which are randomly generated within the feasible steady-state solution space. The hit-and-run sampler determines the geometric center (*cg*) of the high-dimensional solution space. Using this geometric center, the direction from a point *x* to the next step is biased by choosing a point x_1 along the line of *x* and *cg*, whereby the step size to x_1 is randomly determined. This approach has been shown to accelerate the random walk through the space. To ensure that the set of sampling points does indeed represent the solution space, i) there must be sufficient points (i.e., as the size of the solution space increases the number of stored points must increase, `nFiles*pointsPerFile` gives the total number of stored points to the `ACHRSampler`); and ii) all points must be random, i.e., the chosen path between two

subsequent sampling points has to be untraceable. Therefore, the number of points between two stored points (`stepsPerPoint`) must be chosen appropriately. For this latter property, it is also the case that the number of steps per points must increase as the dimensionality of the steady-state solution space increases. Note that the following calculations can be quite time consuming, even with a small scale model.

To sample the solution space of the *E. coli* core model with its default constraints (growth on glucose, aerobic):

```
warmupPts= createHRWarmupRand(model,500);
ACHRSampler(model,warmupPts,'samplingResults',10,5000,1000,warmupPts(:,1));
```

To the load the calculated data:

```
samples = loadSamples('samplingResults', 10, 5000);
```

Next, FVA (**Supplementary Example 3**) can be used to determine the minimum and maximum possible fluxes so the sampling results can be plotted for the first nine reactions in the model (**Supplementary Figure 14**):

```
[minFlux,maxFlux] = FluxVariability(model,0);

figure;
for i = 1 : 9
    subplot(3,3,i)
    hist(samples(i,:),10);
    hold on
    plot([minFlux(i) maxFlux(i)], [0 1],'*r');
    title(model.rxns{i});
end
```

The error of **Sv** = 0 for the sampled flux distributions can be calculated, and these errors are plotted in **Supplementary Figure 15**. The errors are all very small, indicating that these sampling results are valid:

```
errors = max(abs(model.S*samples));
plot(errors)
```

The correlation between the first ten network reactions can be calculated and plotted (**Supplementary Figure 16**) with:

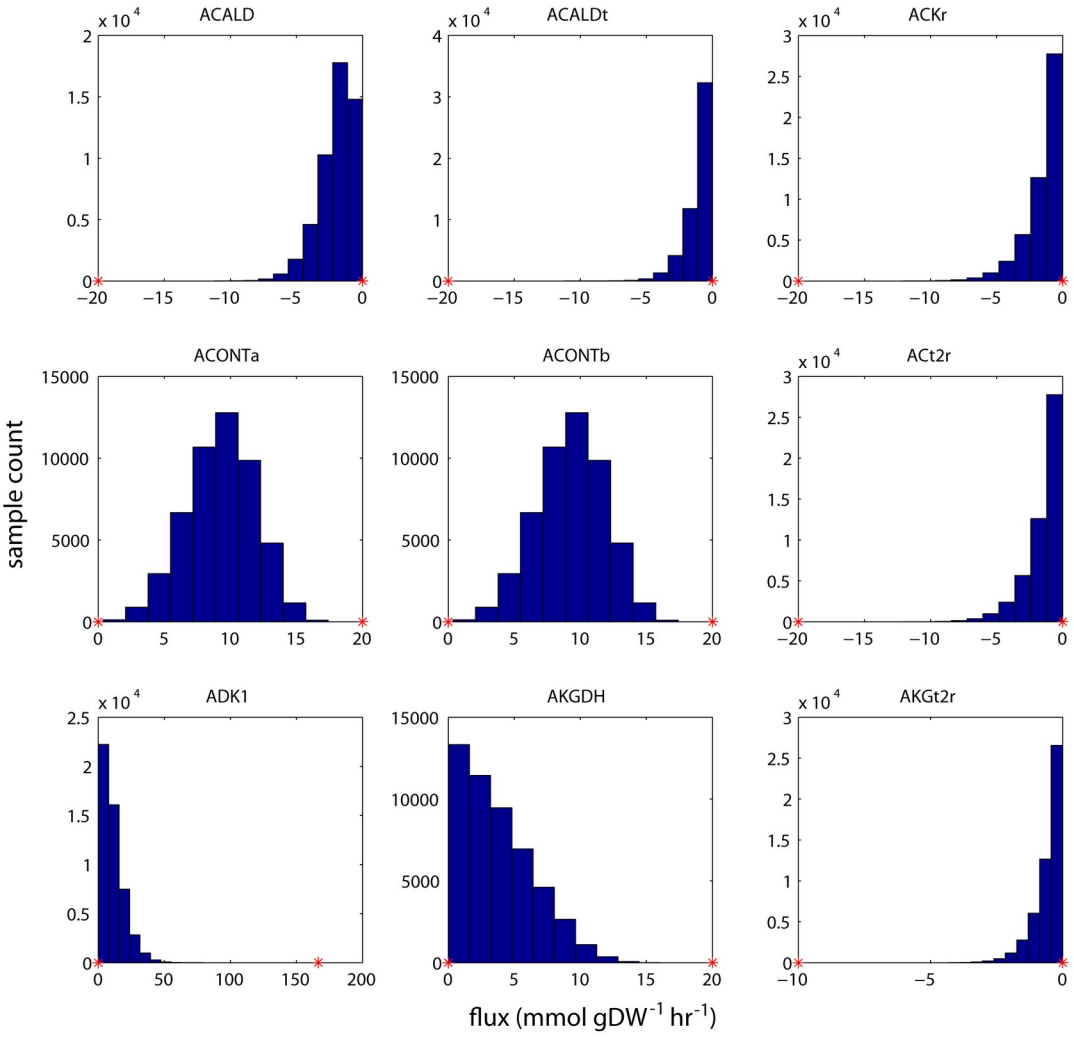
```
sampleScatterMatrix(model.rxns(1:10),model,samples);
```

This scatter plot allows us to visualize the interaction between two network reactions. For example, the aconitase reactions ACONTa and ACONTb are perfectly correlated in the tested condition (sample points between the two reactions align on a line). In contrast, the flux through the adenylate kinase reaction (ADK1) seems mostly independent from the flux through the acetaldehyde dehydrogenase reaction (ACALD). Considering the locations of these reactions on the metabolic map of the *E. coli* core model (**Supplementary Figure 1**), this observation is not astonishing.

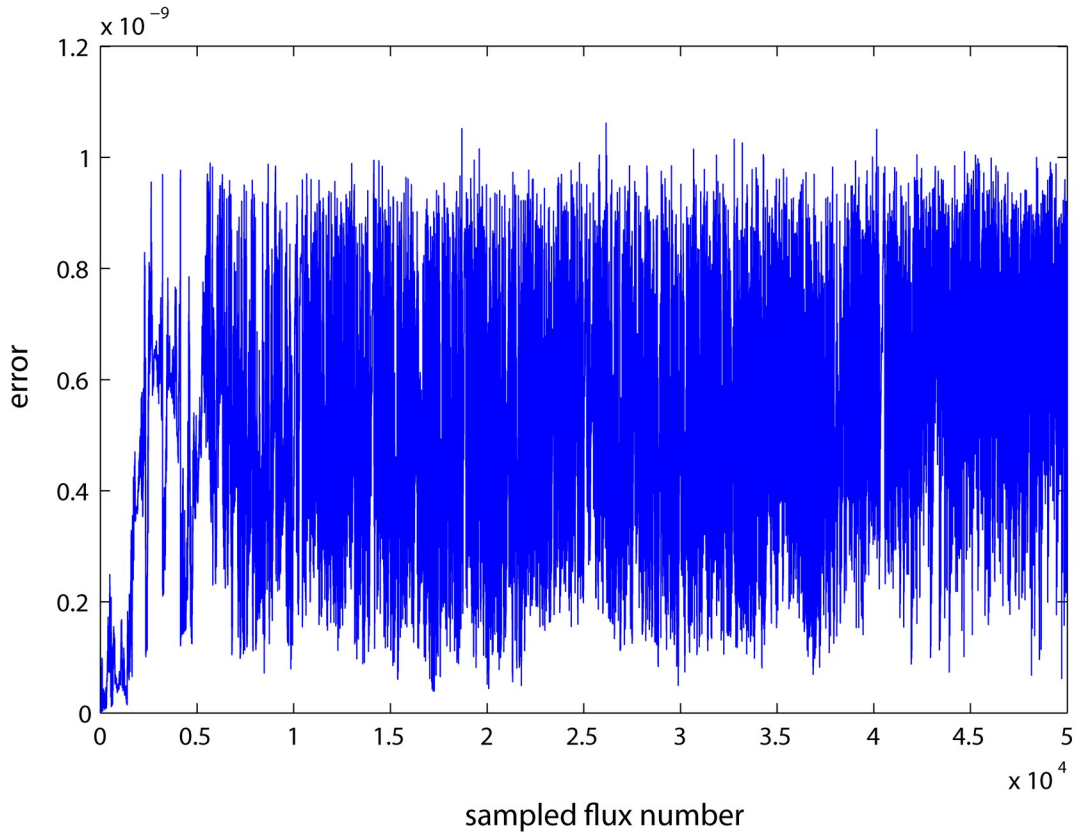
Which network reactions are perfectly correlated? Once the samples points have been calculated they can readily used to determine the correlation between any two reactions in the network. To obtain the list of perfectly correlated reactions (e.g., as in the example above, ACONTa and ACONTb), the following commands can be used:

```
[setsSorted,setNoSorted,setSize] = identifyCorrelSets(model,samples);
setNames = [];
setNumbers = [];
for i = 1 : length(setsSorted)
    setNames = [setNames; setsSorted{i}.names];
    setNumbers = [setNumbers; i*ones(length(setsSorted{i}.names),1)];
end
```

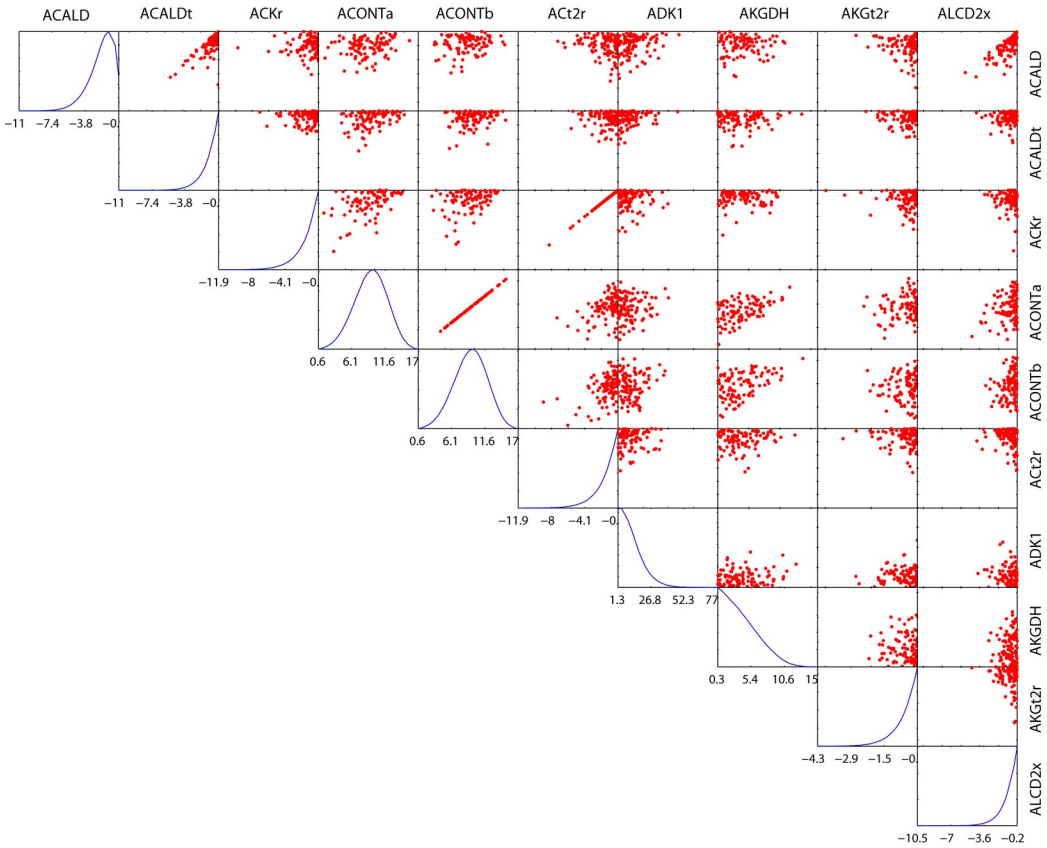
Two vectors will be returned (`setNames`, `setNumbers`) which can be copied into a spreadsheet to obtain a table view (see **Supplementary Table 6**). Interestingly, only 37 of the 95 network reactions are grouped into 17 perfectly correlated reaction sets. A main reason for this is that we did not require the growth rate to have a minimum value. Hence, many functional states are possible, some of which support growth while others don´t.



Supplementary Figure 14 Histograms of sampled fluxes for nine reactions.



Supplementary Figure 15 Error of **Sv** = 0 for the sampled fluxes.



Supplementary Figure 16 Scatter matrix of sample points in the *E. coli* core model, simulated in minimal medium glucose under aerobic conditions.

Supplementary Table 6: Correlated reaction sets in the *E. coli* core model, when grown on glucose minimal medium, aerobic conditions. Reactions that are not part of a correlated reaction set with more than one member are not listed.

Correlated Set #	Reaction Name	Correlated Set #	Reaction Name	Correlated Set #	Reaction Name
1	CYTBD	6	GND	12	EX_for(e)
1	EX_o2(e)	6	PGL	12	PFL
1	O2t	7	GAPD	13	ENO
2	ACONTa	7	PGK	13	PGM
2	ACONTb	8	FBA	14	D_LACt2
2	CS	8	TPI	14	LDH_D
3	ACKr	9	EX_pi(e)	15	CO2t
3	ACt2r	9	PIt2r	15	EX_co2(e)
3	PTAr	10	EX_nh4(e)	16	ALCD2x
4	TALA	10	NH4t	16	ETOHt2r
4	TKT1	11	EX_h2o(e)	17	ADK1
5	ICL	11	H2Ot	17	PPS
5	MALS				

Supplementary References

1. Becker, S.A. et al. Quantitative prediction of cellular metabolism with constraint-based models: The COBRA Toolbox. *Nat. Protocols* **2**, 727-738 (2007).
2. Orth, J.D., Fleming, R.M. & Palsson, B.O. in EcoSal - Escherichia coli and Salmonella Cellular and Molecular Biology. (ed. P.D. Karp) (ASM Press, Washington D.C.; 2009).
3. Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J. & Palsson, B.O. Integrating high-throughput and computational data elucidates bacterial networks. *Nature* **429**, 92-96 (2004).
4. Segre, D., Vitkup, D. & Church, G.M. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences of the United States of America* **99**, 15112-15117 (2002).
5. Shlomi, T., Berkman, O. & Ruppin, E. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *Proceedings of the National Academy of Sciences of the United States of America* **102**, 7695-7700 (2005).
6. Reed, J.L., Famili, I., Thiele, I. & Palsson, B.O. Towards multidimensional genome annotation. *Nat Rev Genet* **7**, 130-141 (2006).
7. Herrgard, M.J., Fong, S.S. & Palsson, B.O. Identification of genome-scale metabolic network models using experimentally measured flux profiles. *PLoS computational biology* **2**, e72 (2006).
8. Satish Kumar, V., Dasika, M.S. & Maranas, C.D. Optimization based automated curation of metabolic reconstructions. *BMC bioinformatics* **8**, 212 (2007).
9. Kumar, V.S. & Maranas, C.D. GrowMatch: an automated method for reconciling in silico/in vivo growth predictions. *PLoS Comput Biol* **5**, e1000308 (2009).
10. Burgard, A.P., Pharkya, P. & Maranas, C.D. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and bioengineering* **84**, 647-657 (2003).
11. Patil, K.R., Rocha, I., Forster, J. & Nielsen, J. Evolutionary programming as a platform for in silico metabolic engineering. *BMC bioinformatics* **6**, 308 (2005).
12. Pharkya, P., Burgard, A.P. & Maranas, C.D. OptStrain: a computational framework for redesign of microbial production systems. *Genome research* **14**, 2367-2376 (2004).
13. Varma, A. & Palsson, B.O. Metabolic capabilities of *Escherichia coli*: I. Synthesis of biosynthetic precursors and cofactors. *Journal of Theoretical Biology* **165**, 477-502 (1993).
14. Mahadevan, R. & Schilling, C.H. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic engineering* **5**, 264-276 (2003).
15. Edwards, J.S. & Palsson, B.O. Robustness analysis of the Escherichia coli metabolic network. *Biotechnology Progress* **16**, 927-939 (2000).
16. Edwards, J.S., Ramakrishna, R. & Palsson, B.O. Characterizing the metabolic phenotype: a phenotype phase plane analysis. *Biotechnology and bioengineering* **77**, 27-36. (2002).

17. Bell, S.L. & Palsson, B.O. Phenotype phase plane analysis using interior point methods. *Computers & Chemical Engineering* **29**, 481-486 (2005).
18. Palsson, B.O. Systems biology: properties of reconstructed networks. (Cambridge University Press, New York; 2006).
19. Barabasi, A.L. & Oltvai, Z.N. Network biology: understanding the cell's functional organization. *Nat Rev Genet* **5**, 101-113 (2004).
20. Mahadevan, R. & Palsson, B.O. Properties of metabolic networks: structure versus function. *Biophysical journal* **88**, L07-09 (2005).
21. Price, N.D., Schellenberger, J. & Palsson, B.O. Uniform Sampling of Steady State Flux Spaces: Means to Design Experiments and to Interpret Enzymopathies. *Biophysical journal* **87**, 2172-2186 (2004).
22. Thiele, I., Price, N.D., Vo, T.D. & Palsson, B.O. Candidate metabolic network states in human mitochondria: Impact of diabetes, ischemia, and diet. *The Journal of biological chemistry* **280**, 11683-11695 (2005).