Gene/Reaction Knockout Strategies

Learning Objectives

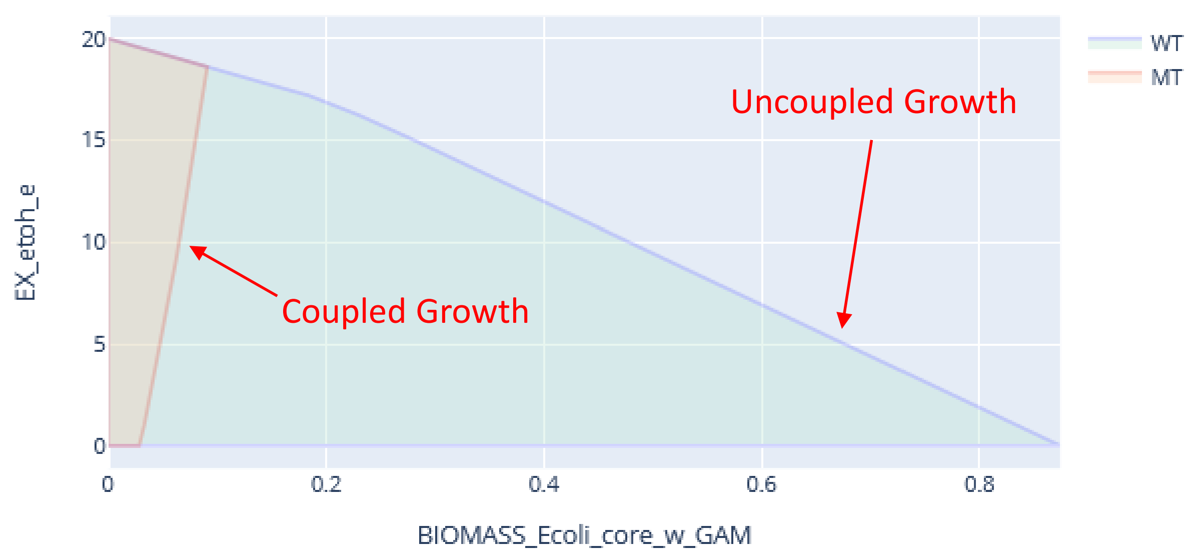
Each student should be able to:

- 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 OptGene.

Lesson Outline

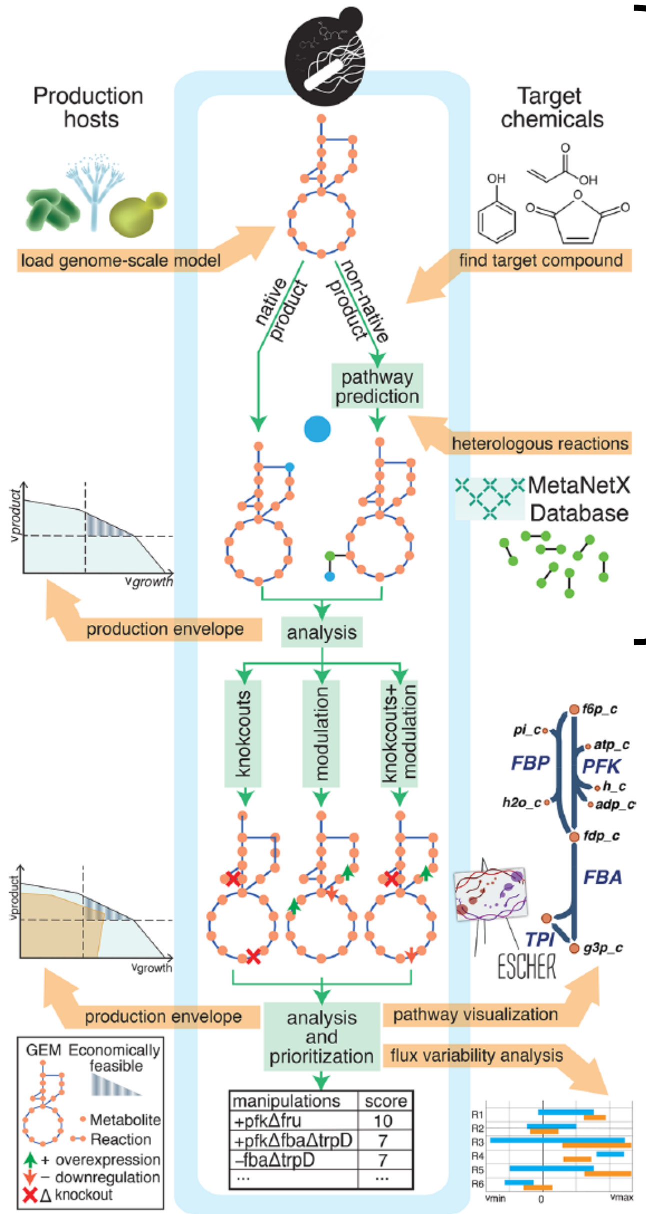
- Overview
- Yields
- OptKnock
- OptGene

Production Envelope

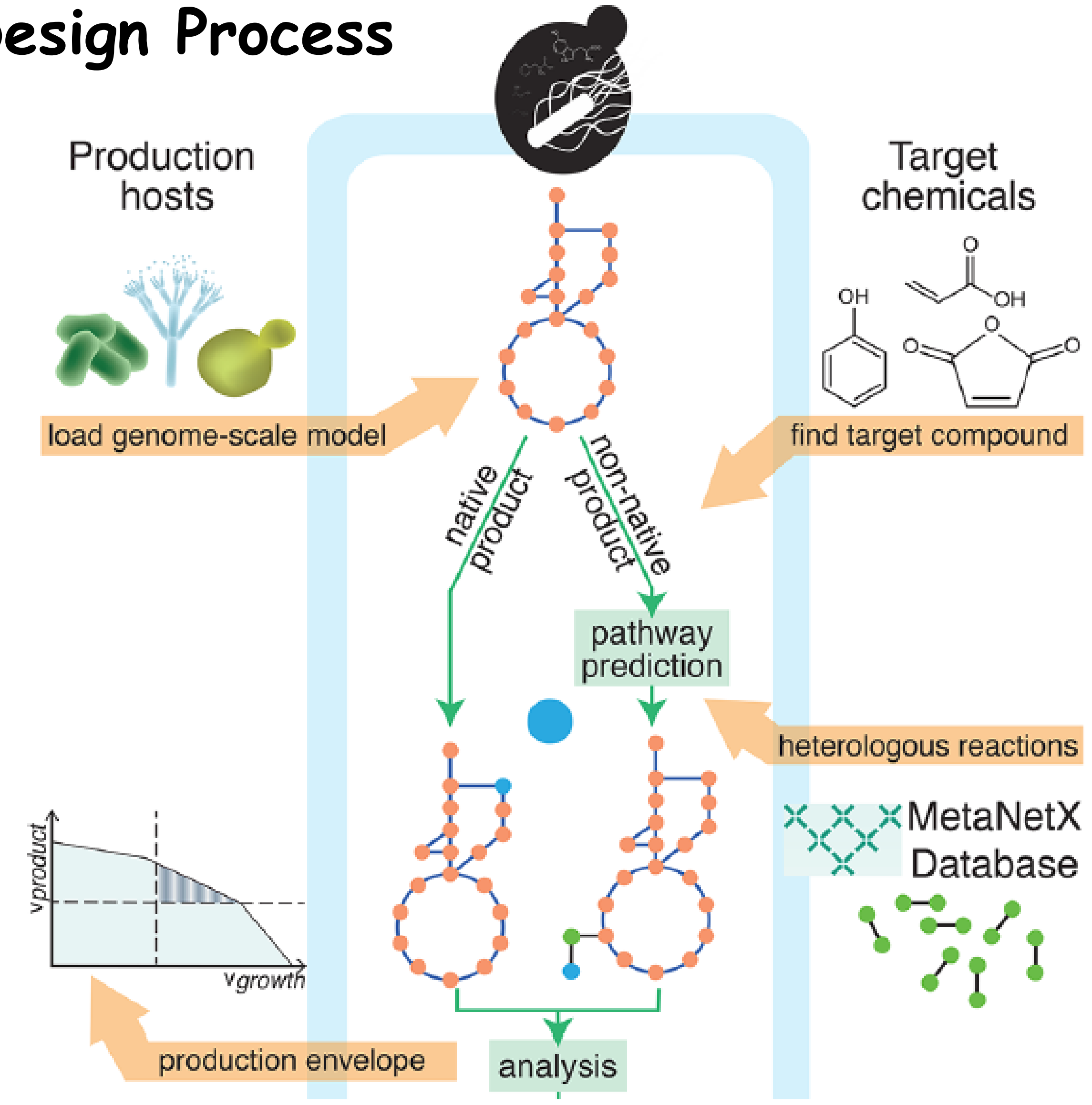


Constraint-based Metabolic Reconstructions & Analysis

Strain Design Process

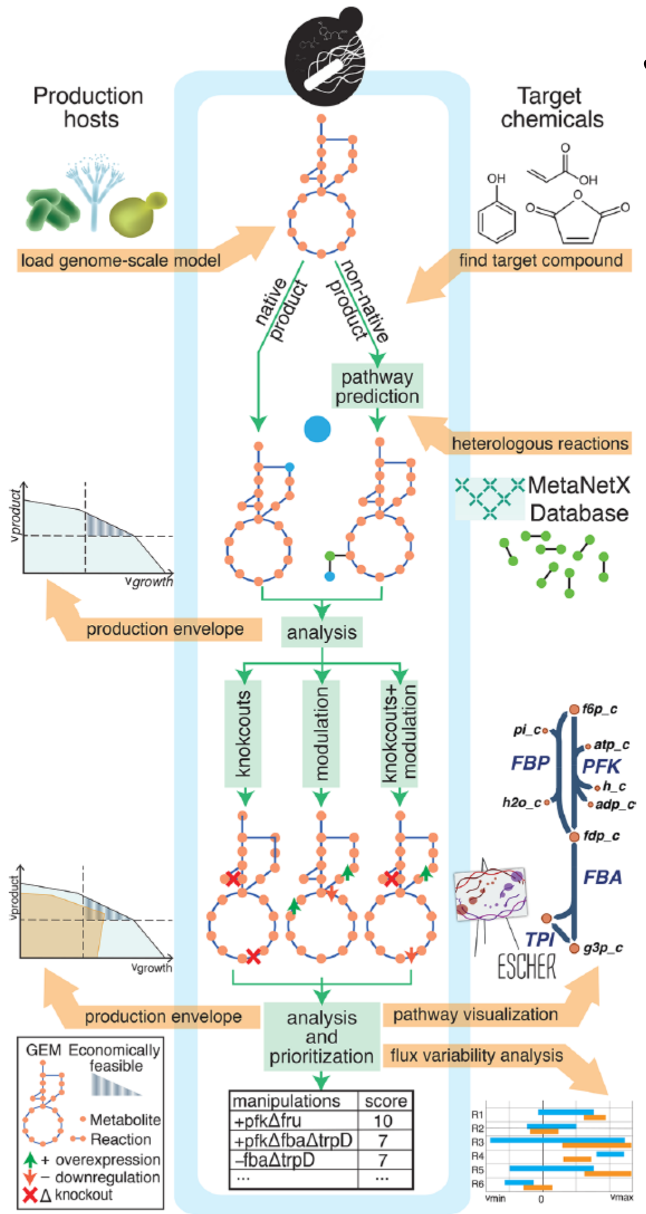


manipulations	score
+pfkΔfru	10
+pfkΔfbaΔtrpD	7
−fbaΔtrpD	7
...	...

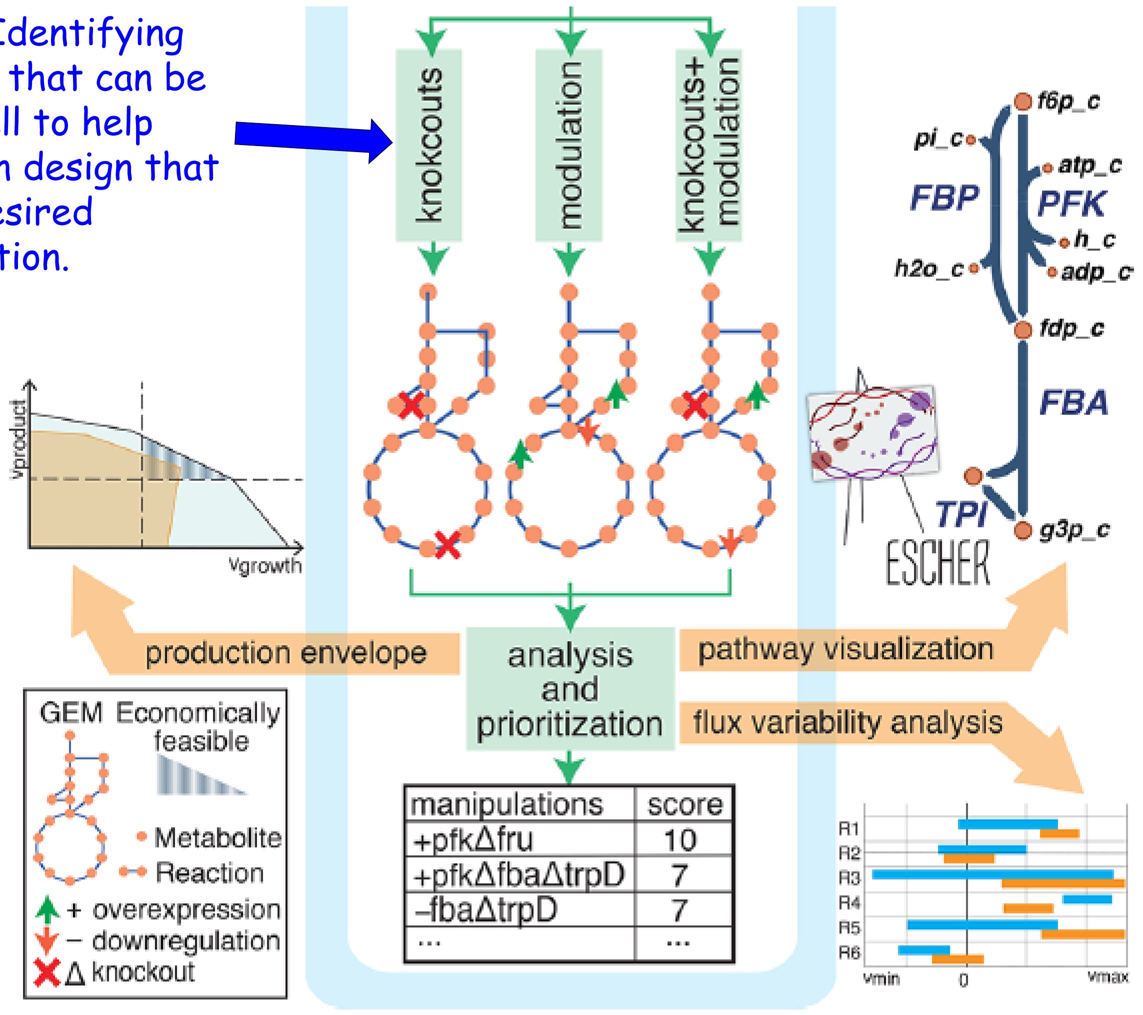


Cardoso, João GR, et al. "Cameo: a Python library for computer aided metabolic engineering and optimization of cell factories." *ACS synthetic biology* 7.4 (2018): 1163-1166.

Strain Design Process



Gene knockouts: Identifying genes or reactions that can be removed from a cell to help create a new strain design that can improve the desired bioproduct production.



manipulations	score
+pfkΔfru	10
+pfkΔfbaΔtrpD	7
–fbaΔtrpD	7
...	...

Cardoso, João GR, et al. "Cameo: a Python library for computer aided metabolic engineering and optimization of cell factories." *ACS synthetic biology* 7.4 (2018): 1163-1166.

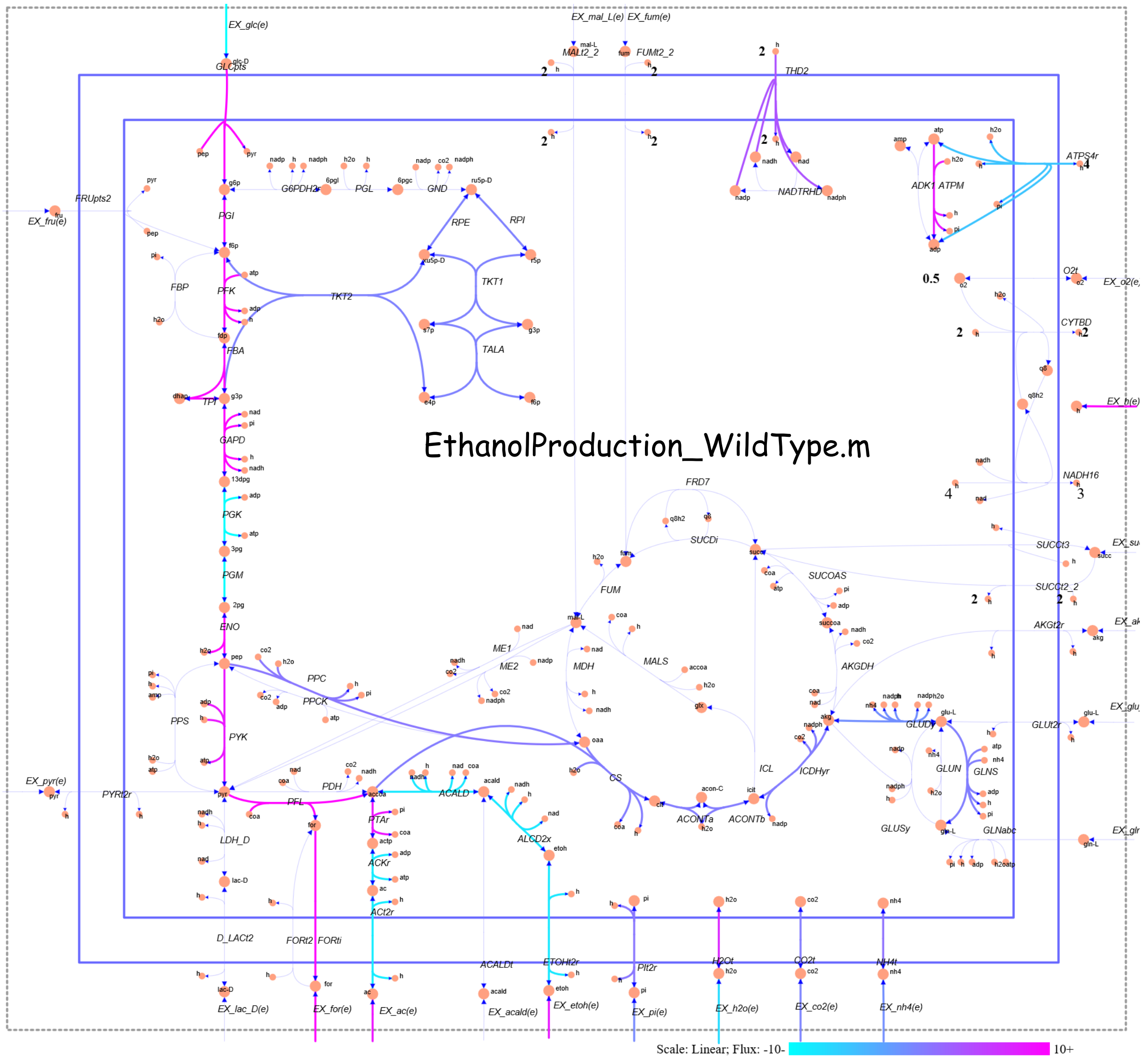
Gene/Reaction Knockouts

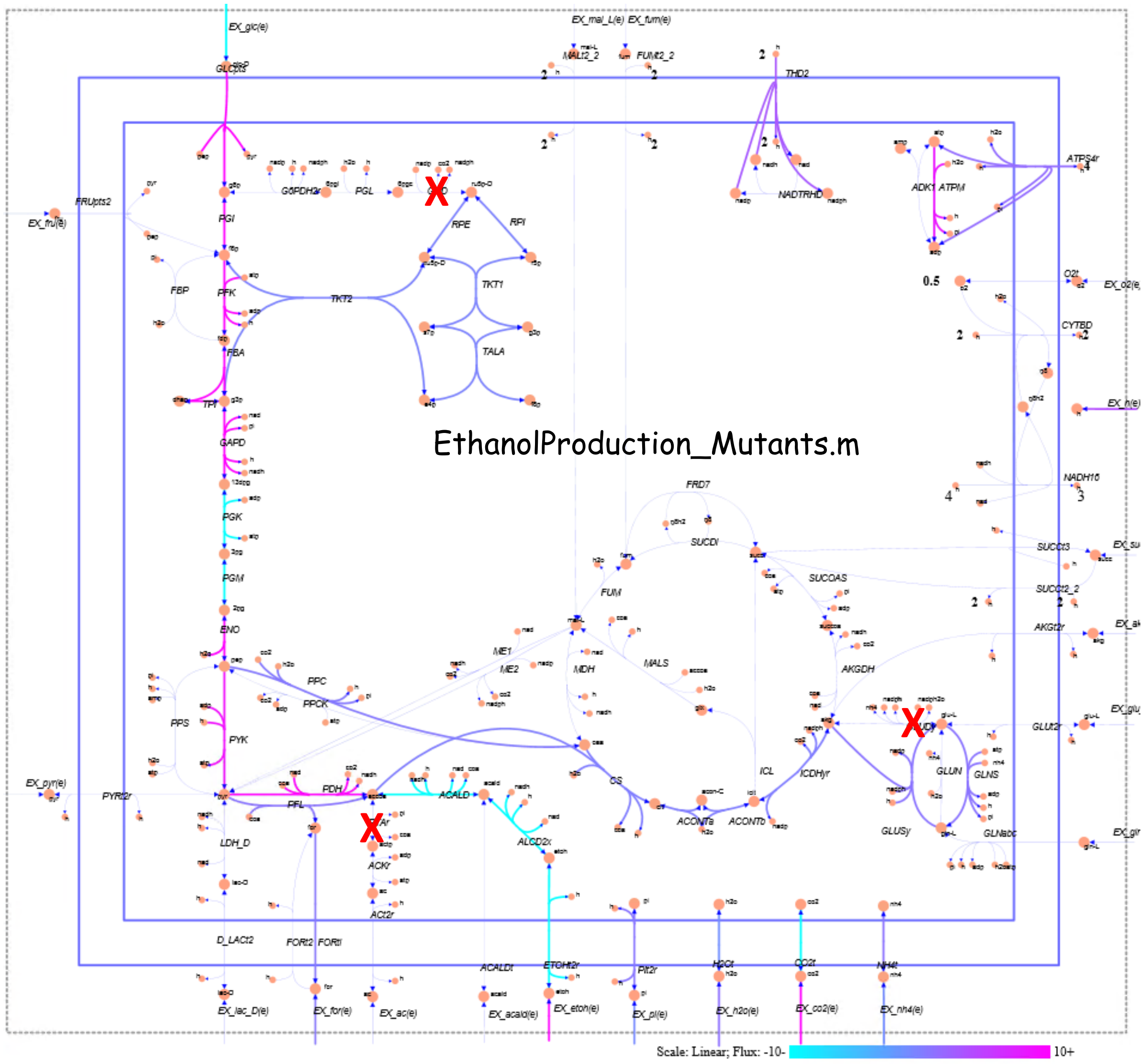
- Metabolic engineering has been successful in using the recombinant DNA technology to selectively alter cell metabolism (new strain design) and improve a targeted cellular function (bioproduct production).
- The use of metabolic genome scale metabolic reconstructions represents a major opportunity for 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 and 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 the production rate of a desired by-product (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

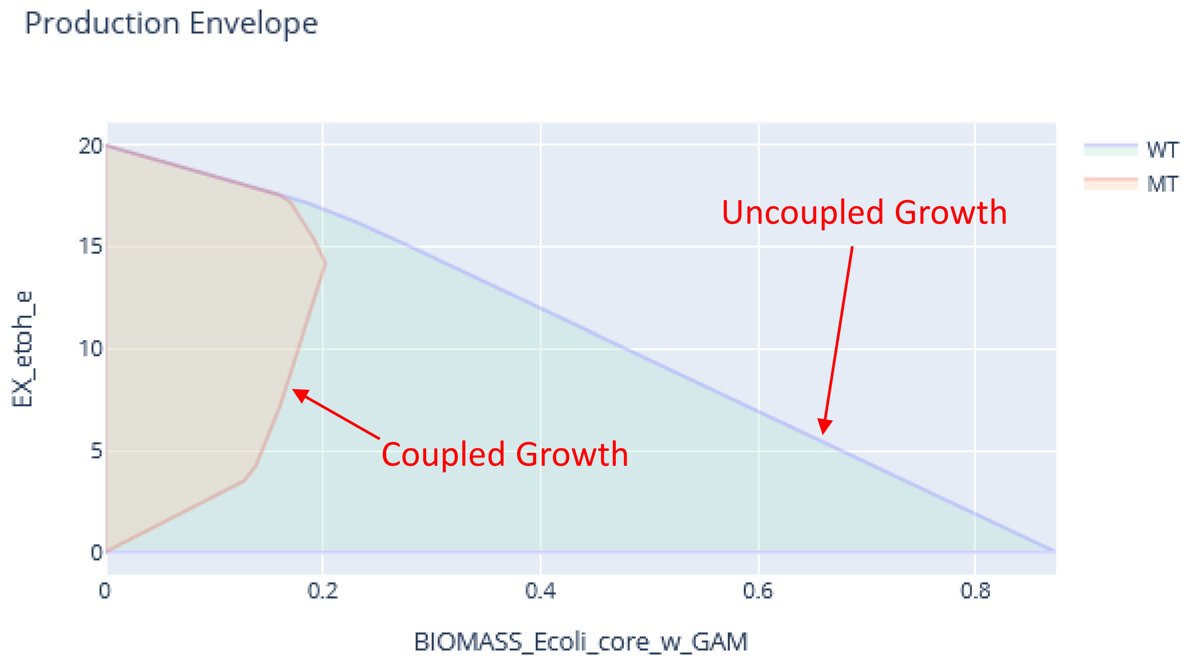
- Just as growth in different environments can be simulated with FBA, gene/reaction knockouts can also be simulated by changing reaction bounds or using the knock_out() method.
- To simulate the knockout of a gene use "model.genes.gene_locus.knock_out()"
- To simulate the knockout of a reaction use "model.rections.reaction_id.knock_out()"
- 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 each reaction to 0 mmol gDW^{-1} hr^{-1}. In this case, each reaction is knocked out by restricting it from carrying flux.

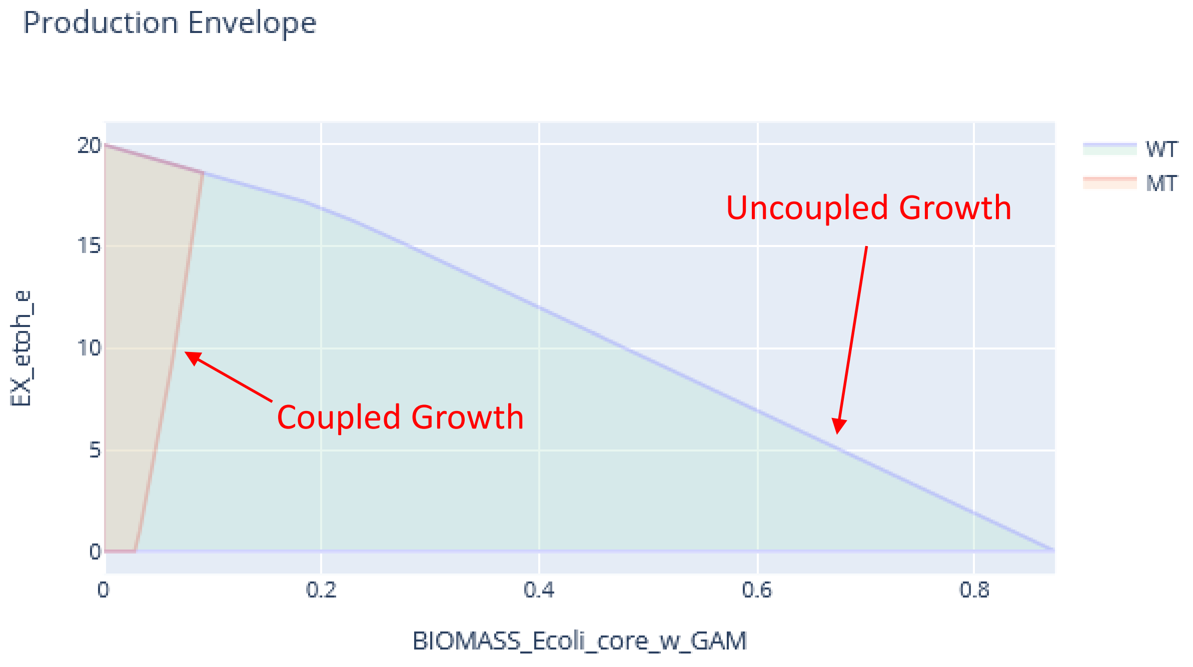
Creating a Mutant Strain: Anaerobic Ethanol Production





Coupled Growth and Bioproduct Production

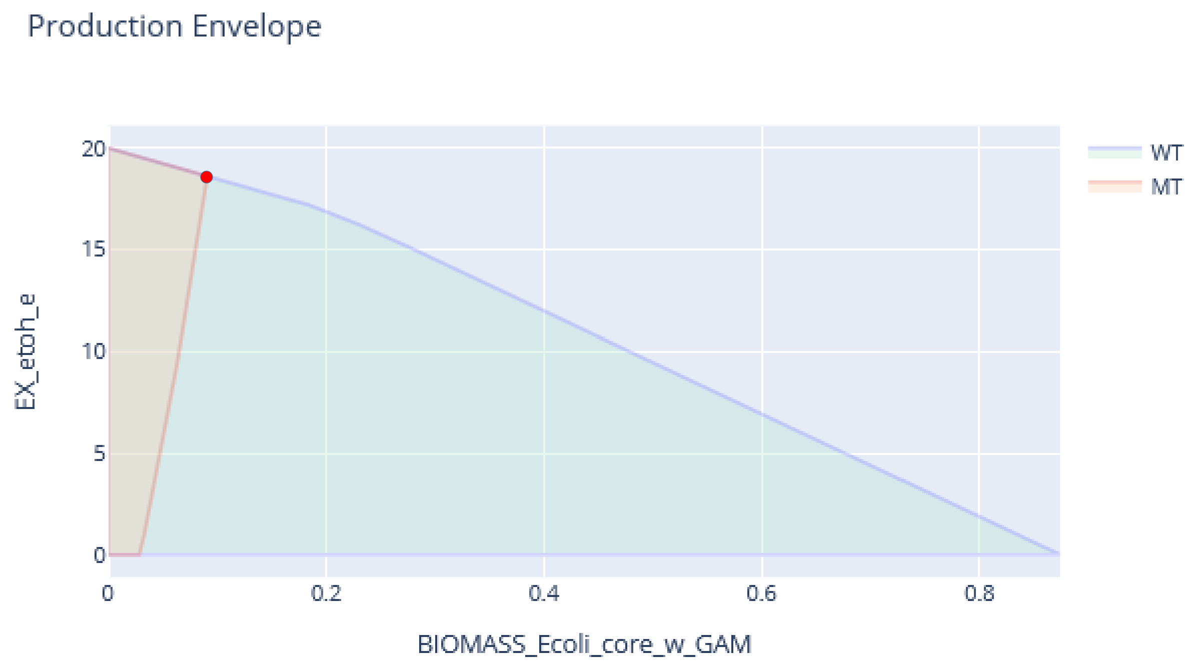




Coupled growth – the production of a bioproduct is coupled to the growth of the cell

Lesson Outline

- Overview
- ➡ Yields
- OptKnock
- OptGene



Molar and Mass Yields

Molar Yield

$$M_y = \frac{Bioproduct\ (moles)}{Carbon\ Source\ (moles)} = \frac{Bioproduct\ (\frac{mmol}{gDW*hr})}{Carbon\ Source\ (\frac{mmol}{gDW*hr})} = \frac{Bioproduct\ (flux)}{-\ Carbon\ Source\ (flux)}$$

myield = solution.fluxes[‘Product'] / (-1. * solution.fluxes[‘Carbon Source'])

Mass Yield

$$G_y = \frac{Bioproduct\ (grams)}{Carbon\ Source\ (grams)} = \frac{Bioproduct\ (\frac{mmol*MW}{gDW*hr})}{Carbon\ Source\ (\frac{mmol*MW}{gDW*hr})} = \frac{Bioproduct\ (flux*MW)}{-\ Carbon\ Source\ (flux*MW)}$$

MW_product = model.metabolites. *Product*.formula_weight # Molecular weight of product
MW_cs = model.metabolites.*Carbon_source*.formula_weight # Molecular weight of carbon source
gyield = MW_ac*solution.fluxes[' *Product* '] / (-1. * solution.fluxes['*Carbon_source*']*MW_glc)

Carbon and Biomass Yields

Carbon Yield

$$C_y = \frac{Bioproduct\ (moles * N_p)}{Carbon\ Source\ (moles * N_c)} = \frac{Bioproduct\ (\frac{mmol * N_p}{gDW * hr})}{Carbon\ Source\ (\frac{mmol * N_c}{gDW * hr})} = \frac{Bioproduct\ (flux * N_p)}{-\ Carbon\ Source\ (flux * N_c)}$$

N_c = model.metabolites.*Carbon_source*.elements['C'] # Number of carbon atoms in carbon source

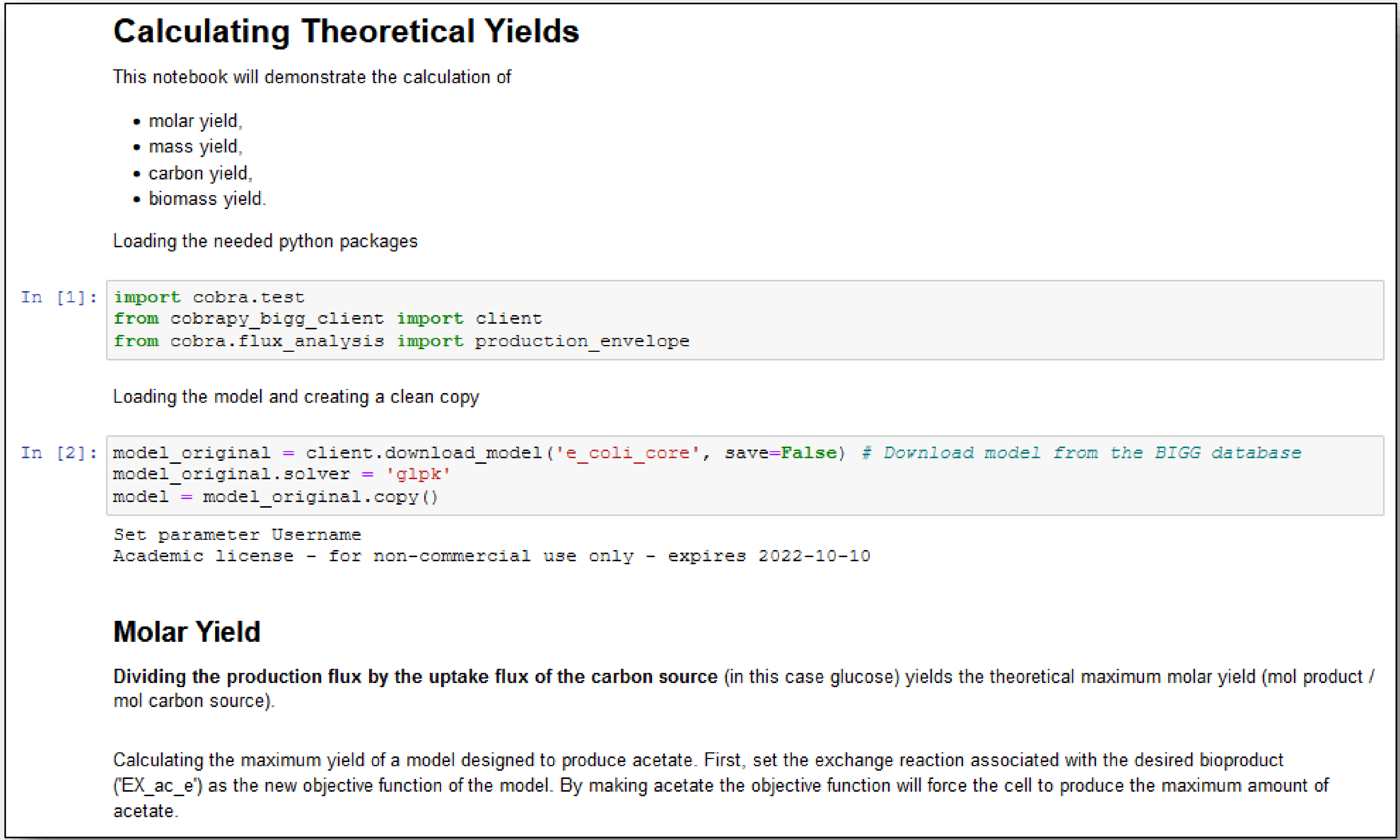
N_p = model.metabolites.*Product*.elements['C'] # Number of carbon atoms in product

cyield = N_p*solution.fluxes['Product'] / (-1. * N_c*solution.fluxes['Carbon source'])

Biomass Yield

$$B_y = \frac{Biomass\ (moles)}{Carbon\ Source\ (moles)} = \frac{Biomass\ (\frac{1}{hr})}{Carbon\ Source\ (\frac{mmol}{gDW * hr})} = \frac{Bioproduct\ (gDW)}{-\ Carbon\ Source\ (mmol)}$$

byield = solution.fluxes[‘Biomass reaction'] / (-1. * solution.fluxes['Carbon source'])



Calculating Theoretical Yields

This notebook will demonstrate the calculation of

- molar yield,
- mass yield,
- carbon yield,
- biomass yield.

Loading the needed python packages

```
In [1]: import cobra.test
        from cobrapy_bigg_client import client
        from cobra.flux_analysis import production_envelope
```

Loading the model and creating a clean copy

```
In [2]: model_original = client.download_model('e_coli_core', save=False) # Download model from the BIGG database
        model_original.solver = 'glpk'
        model = model_original.copy()

        Set parameter Username
        Academic license - for non-commercial use only - expires 2022-10-10
```

Molar Yield

Dividing the production flux by the uptake flux of the carbon source (in this case glucose) yields the theoretical maximum molar yield (mol product / mol carbon source).

Calculating the maximum yield of a model designed to produce acetate. First, set the exchange reaction associated with the desired bioproduct ('EX_ac_e') as the new objective function of the model. By making acetate the objective function will force the cell to produce the maximum amount of acetate.

Yields.ipynb

Lesson Outline

- Overview
- Yields
- ➡ OptKnock
- OptGene

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 $\leq$ limit

Bilevel optimization structure of OptKnock

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 a 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.

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 $\leq$ 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.

OptKnock

OptKnock solves a bi-level optimization problem, finding the set of knockouts that allows maximal target production under optimal growth.

```
from cameo.strain_design.deterministic.linear_programming import OptKnock
```

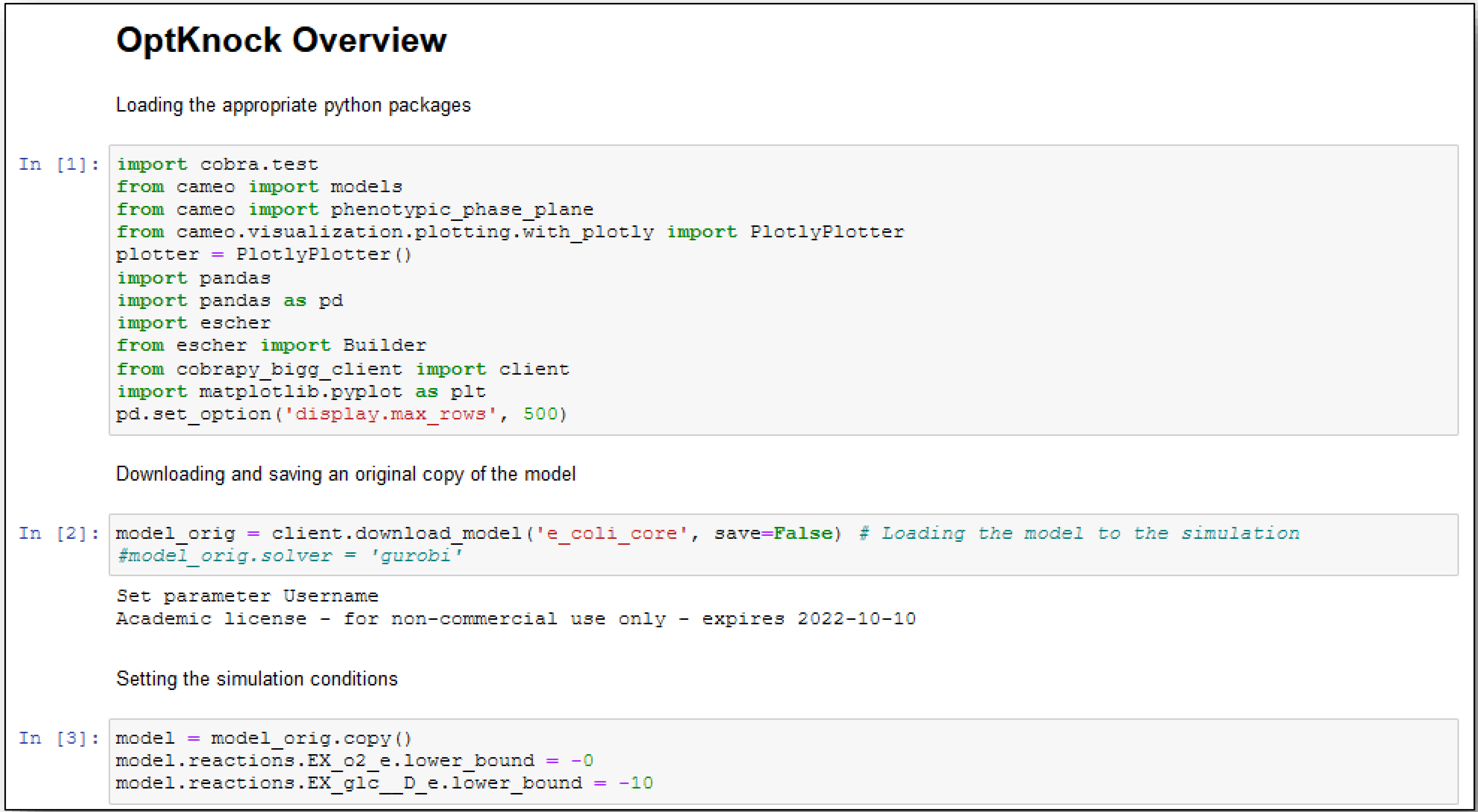
OptKnock(model, exclude_reactions=None, remove_blocked=True, fraction_of_optimum=0.1, exclude_non_gene_reactions=True, use_nullspace_simplification=True)

Parameters

- ***model*** (cobra.Model) – A model to be used for finding optimal knockouts. Always set a non-zero lower bound on biomass reaction before using OptKnock.
- ***exclude_reactions*** (iterable of str or Reaction objects) – Reactions that will not be knocked out. Excluding reactions can give more realistic results and decrease running time. Essential reactions and exchanges are always excluded.
- ***remove_blocked*** (boolean (default True)) – If True, reactions that cannot carry flux (determined by FVA) will be removed from the model. This reduces running time significantly.
- ***fraction_of_optimum*** (If not None, this value will be used to constrain the inner objective (e.g. growth) to) – a fraction of the optimal inner objective value. If inner objective is not constrained manually this argument should be used. (Default: None)
- ***exclude_non_gene_reactions*** (If True (default), reactions that are not associated with genes will not be knocked out). This results in more practically relevant solutions as well as shorter running times.
- ***use_nullspace_simplification*** (Boolean (default True)) – Use a basis for the nullspace to find groups of reactions whose fluxes are multiples of each other. From each of these groups only 1 reaction will be included as a possible knockout

Methods

- **run**(*max_knockouts=5*, *biomass=None*, *target=None*, *max_results=1*)



OptKnock Overview

Loading the appropriate python packages

```
In [1]: import cobra.test
        from cameo import models
        from cameo import phenotypic_phase_plane
        from cameo.visualization.plotting.with_plotly import PlotlyPlotter
        plotter = PlotlyPlotter()
        import pandas
        import pandas as pd
        import escher
        from escher import Builder
        from cobrapy_bigg_client import client
        import matplotlib.pyplot as plt
        pd.set_option('display.max_rows', 500)
```

Downloading and saving an original copy of the model

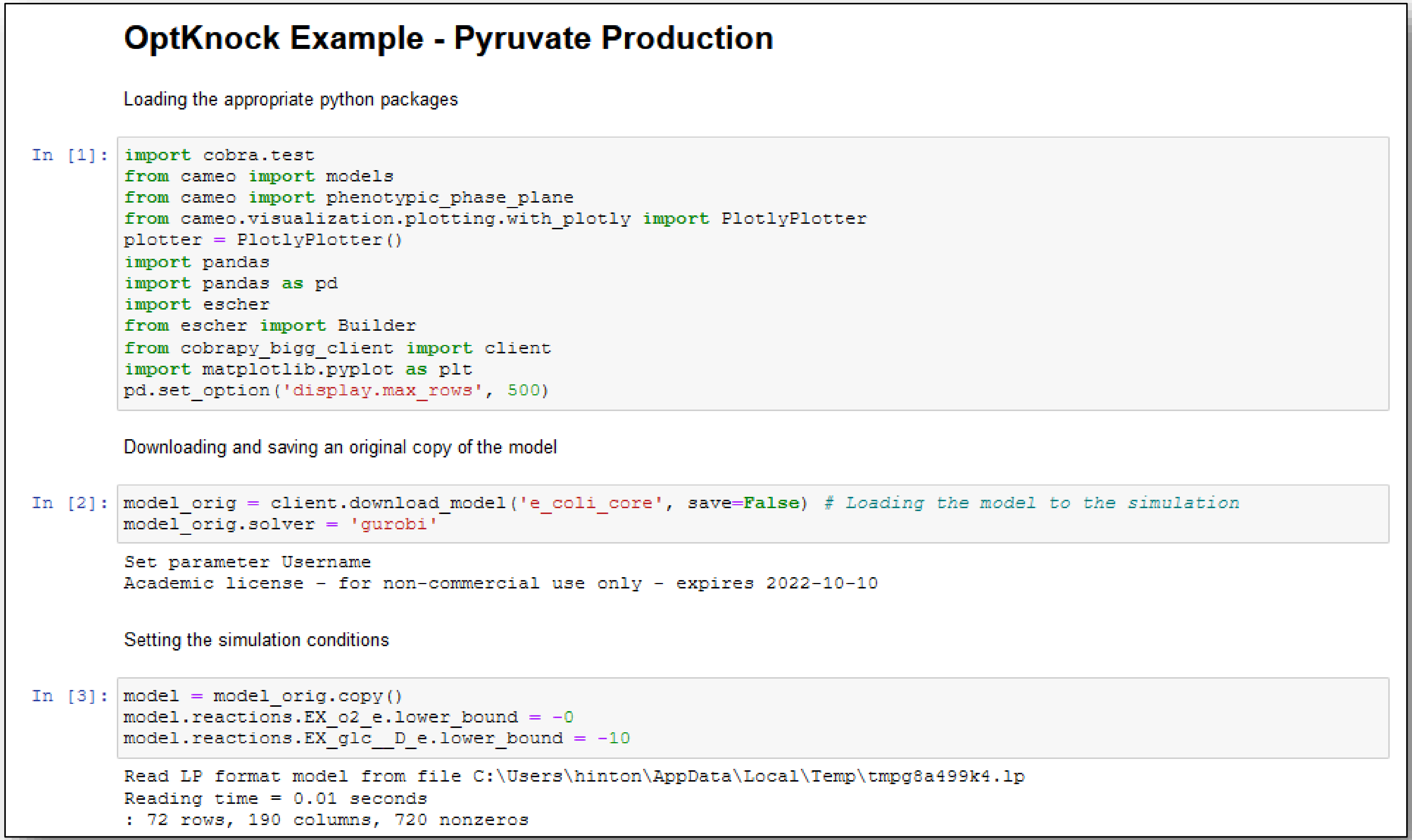
```
In [2]: model_orig = client.download_model('e_coli_core', save=False) # Loading the model to the simulation
        #model_orig.solver = 'gurobi'

        Set parameter Username
        Academic license - for non-commercial use only - expires 2022-10-10
```

Setting the simulation conditions

```
In [3]: model = model_orig.copy()
        model.reactions.EX_o2_e.lower_bound = -0
        model.reactions.EX_glc__D_e.lower_bound = -10
```

OptKnock_overview.ipynb



OptKnock Example - Pyruvate Production

Loading the appropriate python packages

```
In [1]: import cobra.test
        from cameo import models
        from cameo import phenotypic_phase_plane
        from cameo.visualization.plotting.with_plotly import PlotlyPlotter
        plotter = PlotlyPlotter()
        import pandas
        import pandas as pd
        import escher
        from escher import Builder
        from cobrapy_bigg_client import client
        import matplotlib.pyplot as plt
        pd.set_option('display.max_rows', 500)
```

Downloading and saving an original copy of the model

```
In [2]: model_orig = client.download_model('e_coli_core', save=False) # Loading the model to the simulation
        model_orig.solver = 'gurobi'

        Set parameter Username
        Academic license - for non-commercial use only - expires 2022-10-10
```

Setting the simulation conditions

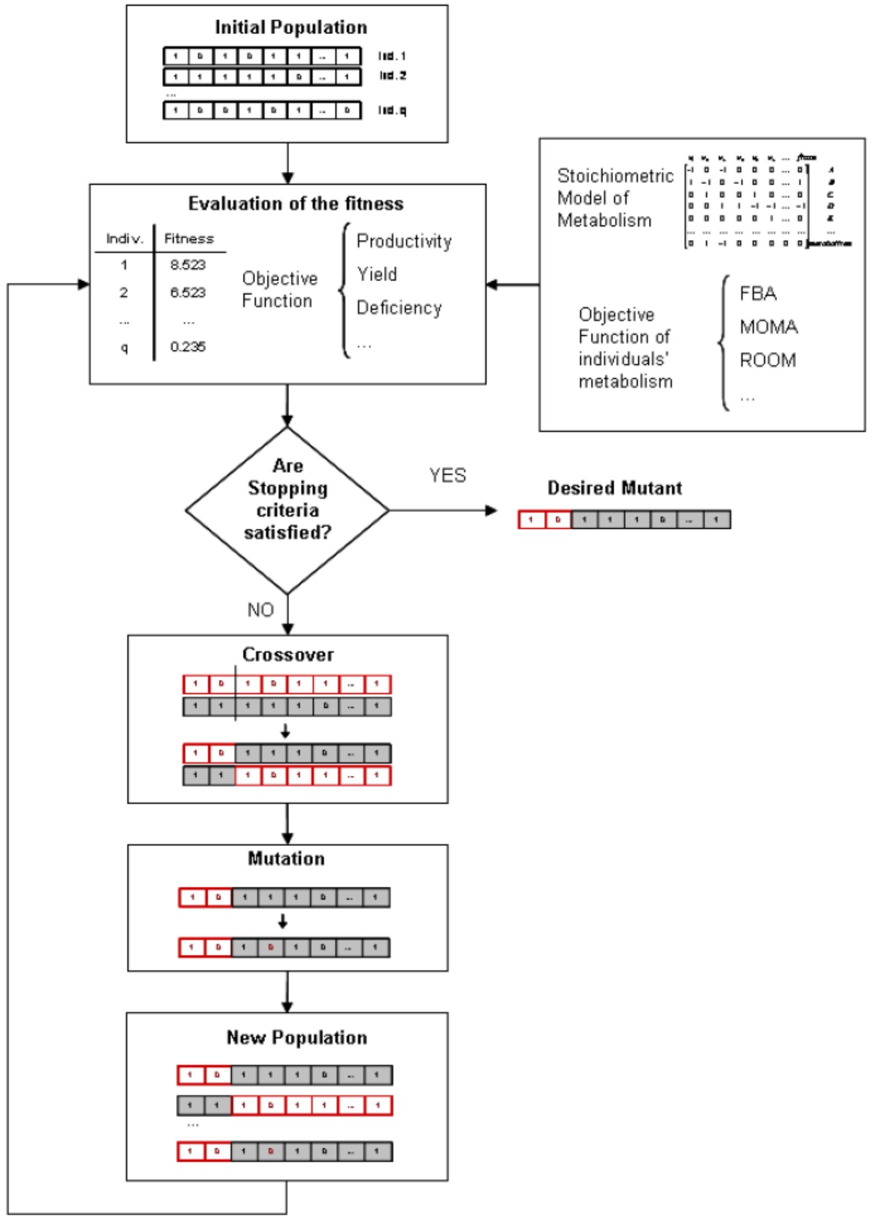
```
In [3]: model = model_orig.copy()
        model.reactions.EX_o2_e.lower_bound = -0
        model.reactions.EX_glc__D_e.lower_bound = -10

        Read LP format model from file C:\Users\hinton\AppData\Local\Temp\tmpg8a499k4.lp
        Reading time = 0.01 seconds
        : 72 rows, 190 columns, 720 nonzeros
```

OptKnock_example_pyruvate.ipynb

Lesson Outline

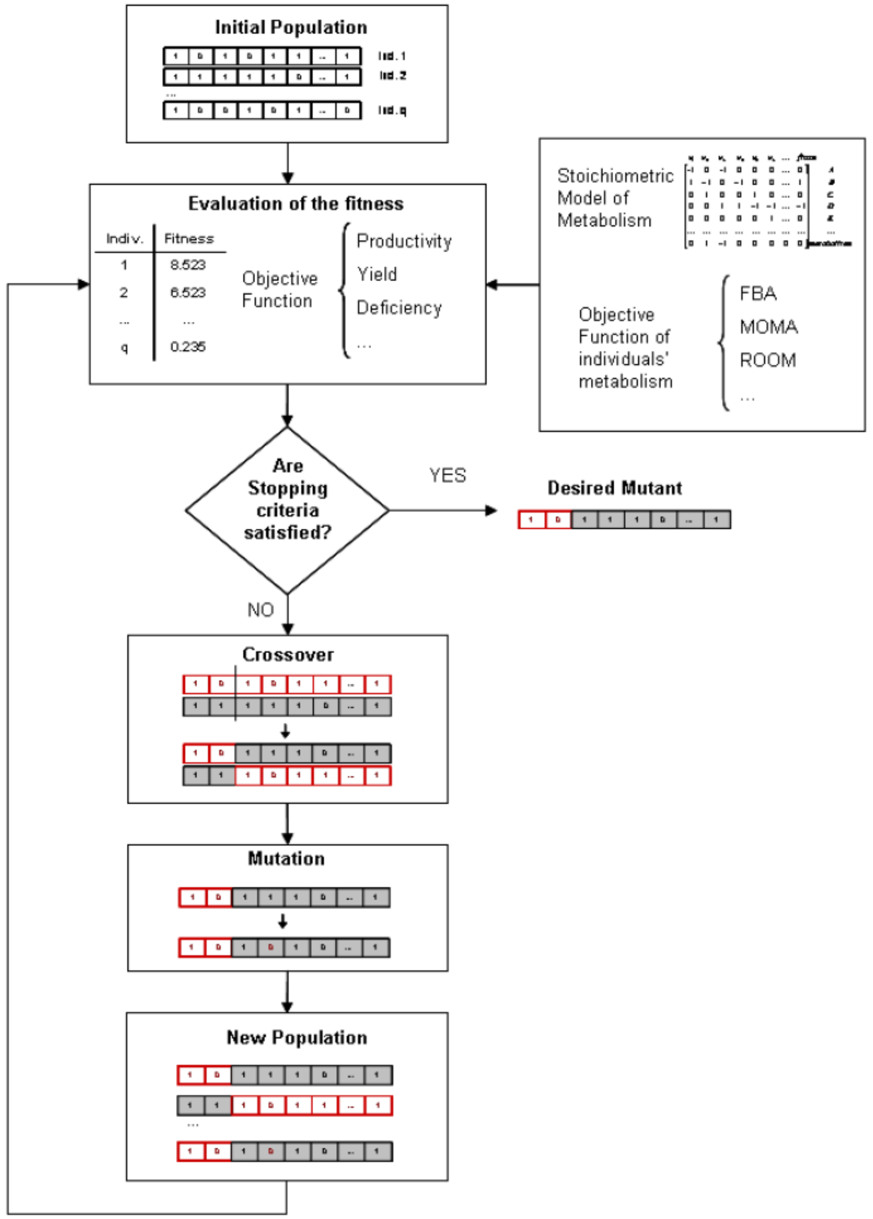
- Overview
- Yields
- OptKnock
- ➡ OptGene



OptGene Algorithm

- OptGene is an heuristic 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.
- 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.

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



OptGene

```
from cameo.strain_design.heuristic.evolutionary_based import OptGene
```

OptGene(*model, evolutionary_algorithm=<class 'inspyred.ec.ec.GA'>, manipulation_type='genes', essential_genes=None, essential_reactions=None, plot=True, exclude_non_gene_reactions=True, *args, **kwargs*)

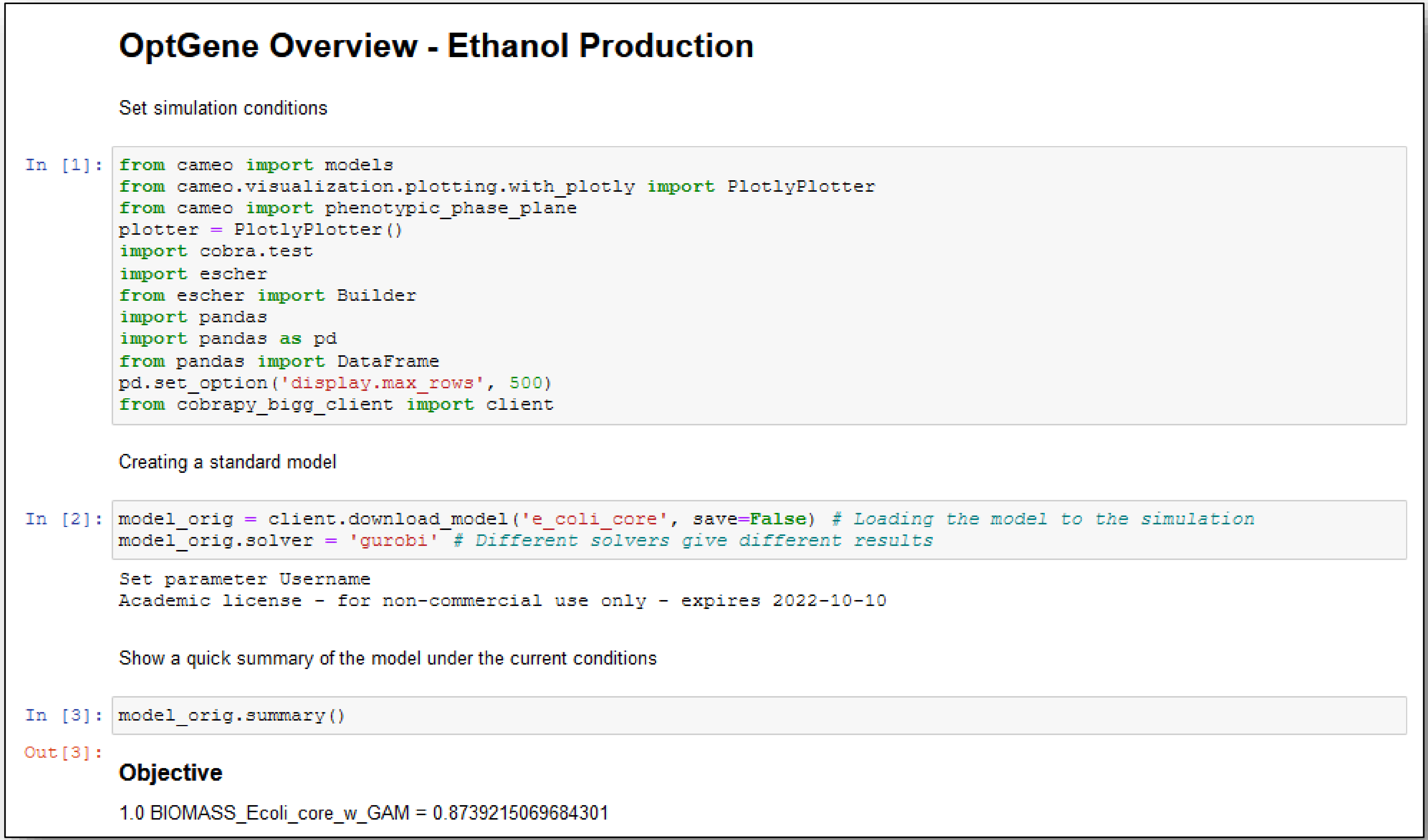
Methods

run(target=None, biomass=None, substrate=None, max_knockouts=5, variable_size=True, simulation_method=<function fba>, growth_coupled=False, max_evaluations=20000, population_size=200, max_results=50, use_nullspace_simplification=True, seed=None, **kwargs)

Parameters

- ***target*** (str, Metabolite or Reaction) – The design target
- ***biomass*** (str, Metabolite or Reaction) – The biomass definition in the model
- ***substrate*** (str, Metabolite or Reaction) – The main carbon source
- ***max_knockouts*** (int) – Max number of knockouts allowed
- ***variable_size*** (bool) – If true, all candidates have the same size. Otherwise the candidate size can be from 1 to max_knockouts.
- ***simulation_method*** (function) – Any method from cameo.flux_analysis.simulation or equivalent
- ***growth_coupled*** (bool) – If true will use the minimum flux rate to compute the fitness
- ***max_evaluations*** (int) – Number of evaluations before stop
- ***population_size*** (int) – Number of individuals in each generation
- ***max_results*** (int) – Max number of different designs to return if found.
- ***kwargs*** (dict) – Arguments for the simulation method.
- ***seed*** (int) – A seed for random.
- ***use_nullspace_simplification*** (Boolean (default True)) – Use a basis for the nullspace to find groups of reactions whose fluxes are multiples of each other and dead end reactions. From each of these groups only 1 reaction will be included as a possible knockout.

Return type: OptGeneResult



OptGene Overview - Ethanol Production

Set simulation conditions

```
In [1]: from cameo import models
        from cameo.visualization.plotting.with_plotly import PlotlyPlotter
        from cameo import phenotypic_phase_plane
        plotter = PlotlyPlotter()
        import cobra.test
        import escher
        from escher import Builder
        import pandas
        import pandas as pd
        from pandas import DataFrame
        pd.set_option('display.max_rows', 500)
        from cobrapy_bigg_client import client
```

Creating a standard model

```
In [2]: model_orig = client.download_model('e_coli_core', save=False) # Loading the model to the simulation
        model_orig.solver = 'gurobi' # Different solvers give different results

        Set parameter Username
        Academic license - for non-commercial use only - expires 2022-10-10
```

Show a quick summary of the model under the current conditions

```
In [3]: model_orig.summary()
Out[3]:
```

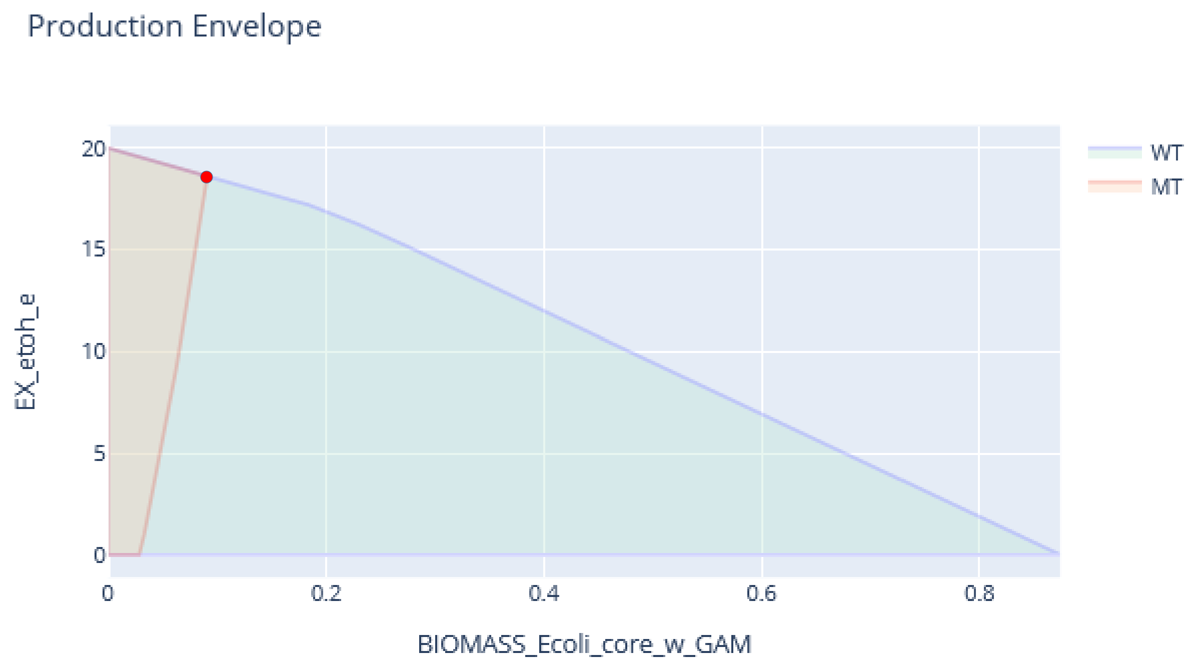
Objective

1.0 BIOMASS_Ecoli_core_w_GAM = 0.8739215069684301

OptGene_overview_ethanol.ipynb

Lesson Outline

- Overview
- Yields
- OptKnock
- OptGene



Review Questions

- What is a molar yield?
- What is a mass yield?
- What is a carbon yield?
- What is a biomass yield?
- What is OptKnock?
- Why should the number of potential knockout reactions be limited?
- How do you knockout a reaction using the COBRApy?
- What does it mean to couple the growth and metabolite production?
- Why is there a trade-off between biomass growth and bioproduct production?
- How can you simulate the engineered mutant cell using the knockouts identified by OptKnock?
- What are the limitations of OptKnock?
- What is OptGene?
- What is the difference between OptKnock and OptGene?
- What are the limitations of OptGene?

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.

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.