Practical 7

Ethylene production in *Cupriavidus necator* H16

In current research into biofuels, there is interest in investigating the feasibility of using microorganisms to produce biofuels and biochemcials from renewable carbon sources. In this practical we will use a GSM to investigate the efficiency of making ethylene in *C. necator* H16 (also known as *Ralstonia eutropha*) for growth under both heterotrophic and autotrophic conditions. We will also use this model to identify strategies for coupling ethylene production to growth. More details about *Cupriavidus necator* will be provided in the talk on Friday by Nicole Pearcy.

Before you start, read the documentation of the ScrumPy LP module, here. It would be a good idea to open this page in a separate tab so that you can refer to it as you work on the practical.

1 Download the archive containing the model and extract the files.

1.a Start ScrumPy in the folder containing the .spy files and load the top-level model file MetaReutro.spy to create a model object. Note that the model is created in a modular fashion, and the top-level file will load the different components of the model and open a window for each.

1.b Examine how the model is created. Note that the reactions that you need to manipulate for this practical are those contained in Transports.spy.

1.c If you want to check the details of any of the reactions or metabolites in the database-derived component of the model - AutoReutro.spy - you can paste the reaction or metabolite identifier into the search box at MetaCyc.

2 Add a new submodule (named 'ethylene_biosynthesis_pathway.spy') that contains the ethylene-forming enzyme (see the reaction stoichiometry here: https://metacyc.org/META/NEW-IMAGE?type=REACTION&object=RXN-12535) and a transport reaction for ethylene and guanidium. Note that guanidium is not recycled by any reactions native to *C. necator* so we need to add a transporter for this by-product to allow its excretion. Without this transporter a feasible solution will not be possible. Also, note that the metabolite identifiers you use must match those already in the model (i.e. arginine is 'ARG' in the model so you must use this identifier when adding your reaction. Most metabolites in the model use the ID given by BioCyc).

3 Now that you have the ethylene route added to the model, predict what the maximum theoretical yield is for this product. To do this, set up and solve an LP problem where the objective is to maximise ethylene production, whilst growing on 2.0 mmol/gDCW/h of fructose.

3.a Create the LP object with the model as argument (hint: lp = m.getLP())

3.b Constrain the flux of the fructose reaction to 2.0 using the SetFixedFlux function of the LP object.

3.c As can be seen from Transport.spy, the model can allow both uptake and production of a number of carbon compounds. To make sure that fructose is used as the only substrate, the import of the other carbon substrates must be constrained to zero, though their export can be allowed. This can be done with the SetFluxBounds function, which constrains fluxes within a specified range. For example, to prevent uptake of any other nutrients not in fructose minimal media, we could write the following:

```
media = [“FRU_tx”, “O2_tx”, “Pi_tx”, “FE2_tx”, “SULFATE_tx”, “NH4_tx”]
for i in m.sm.cnames:
    if i not in media:
        if '_tx' in i:
            lp.SetFluxBounds({i : (None, 0)})
```

3.d Set the optimisation direction to maximisation

3.e Next, using the name of the ethylene transporter that you defined earlier that should be maximised and enter this as an argument to the objective function. (Note that the argument must be a list of reaction(s), i.e. your argument should be a list with one reaction).

3.f Solve the LP. (ie., lp.Solve()) The message 'Optimal solution' should appear. To obtain the solution use the LP method GetPrimSol(). This method returns a dictionary object of reactions in the solution as keys and flux values as values, so for convenience assign a name to this solution, e.g. Examine the ethylene production rate as follows:

3.f.a print(sol[‘ETHYLENE_tx’])

3.g Calculate the maximum theoretical ethylene yield from these results?

3.h What is the oxygen requirement when the ethylene flux is at its maximum rate?

4 The maximum theoretical yield gives us an idea of the capabilities of the bacteria as a host for producing ethylene. However, *in vivo* the bacteria do not natively produce ethylene (without over-expressing genes or knocking genes out). One strategy for redirecting flux towards a product of interest is by blocking reactions that result in the target product becoming an essential byproduct for growth. This is called a growth-coupling strategy. We can create such a coupling by blocking pathways to essential biomass components, which are then restored via the EFE reaction. EFE produces L-DELTA1-PYRROLINE_5-CARBOXYLATE as a byproduct, which is a precursor to proline biosynthesis.

4.a Identify the reactions in the model involved in proline biosythesis either using the model or by searching the pathways in BioCyc to find their IDs (https://biocyc.org/GCF_004798725/substring-search?type=NIL&object=proline+biosynthesis).

4.b Identify the reaction(s) to block in the model that couples EFE to proline biosynthesis (look back at the diagram in the presentation). Make sure that any reaction you block is not a spontaneous reaction. For this simulation set the biomass reaction as the objective function and maximise as follows:

```
lp = m.GetLP()
for i in m.sm.cnames:
        if '_tx' in i:
                if I not in media:
                        lp.SetFluxBounds({i : (None, 0.0)})
lp.SetFixedFlux({'FRU_tx': 2.0})
lp.SetObjective(['Biomass'])
lp.SetObjDirec(direc = 'Max')
lp.SetFixedFlux({*insert reaction KO name here* : 0.0}) ← this here is the reaction you want to block
lp.Solve()
sol = lp.GetPrimSol()
```

Is ethylene in the solution (i.e., print(sol['ETHYLENE_tx'])? What is the difference in the growth rate from the wild type? Note: to simulate for the wild type flux, run the above code but without the last line. To check the growth rate flux >> print(sol['Biomass']).

4.c Repeat using the minimisation of the sum of fluxes as the objective function. Is ethylene still in the solution?

```
lp = m.GetLP()
for i in m.sm.cnames:
        if '_tx' in i:
                if I not in media:
                        lp.SetFluxBounds({i : (None, 0.0)})
lp.SetFixedFlux({'FRU_tx': 2.0})
lp.SetObjective(m.sm.cnames)
lp.SetFixedFlux({*insert reaction KO name here* : 0.0}) ← this here is the reaction you want to block
lp.Solve()
sol = lp.GetPrimSol()
```

Bonus task: Repeat the above steps from 3 onwards for growth on carbon dioxide and hydrogen. To do this, fix the carbon dioxide uptake rate to 5.0 mmol/gDCW/h and leave the hydrogen uptake rate unconstrained. What does the model predict the hydrogen requirement is when ethylene is at its maximum production rate? Does the growth-couple found in 4 work on carbon dioxide? How does the yield compare?