Bioinformatics III

Ninth Assignment

Thibault Schowing (2571837)
Wiebke Schmitt (2543675)

September 19, 2018

Exercise 9.1: Extreme Pathways and Steady State Flux Distribution. Paper-based

(a) **Construct the stoichiometric matrix:**

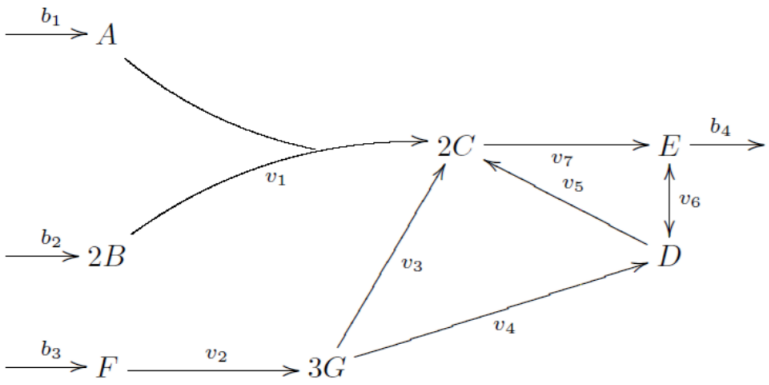


Figure 1: Reaction network to derive extreme pathways from.

Table 1: Stochiometric Matrix

	v1	v2	v3	v4	v5	v6	v6	v7	b1	b2	b3	b4
A	-1	0	0	0	0	0	0	0	1	0	0	0
B	-2	0	0	0	0	0	0	0	0	1	0	0
C	2	0	2	0	2	0	0	-2	0	0	0	0
D	0	0	0	1	-1	1	-1	0	0	0	0	0
E	0	0	0	0	0	-1	1	1	0	0	0	-1
F	0	-1	0	0	0	0	0	0	0	0	1	0
G	0	3	-3	-3	0	0	0	0	0	0	0	0

(b) **Calculate from the stoichiometric matrix the extreme pathways. Give pathways as formulas.**

(c) **Formulate the pathway length matrix. Which information does it provide (diagonal vs off-diagonal entries?**

(d) **Formulate the reaction particiaption matrix. Which information does it provide?**

(e) **Cut-set.** *A reaction or a set of reactions are essential for the network, when there is no output if this reactions are blocked. List all those reactions.*

(f) **Biomass producation.***Now assume that the potential input into the network through b1, b2, and b3, i.e., the sum of the fluxes through these reactions is limited to 5 units. How must this input be distributed onto these reactions to give the highest output through b4?*

Exercise 9.2: Hands-on with COnstraint-Based Reconstruction and Analysis (COBRA) in Python.

(a) **Provide formulas of the reactions participating in the chain.**

```
Reactions
PYK : adp_c + h_c + pep_c --> atp_c + pyr_c
PFK : atp_c + f6p_c --> adp_c + fdp_c + h_c
FBA : fdp_c <=> dhap_c + g3p_c
PGK : 3pg_c + atp_c <=> 13dpg_c + adp_c
HEX1 : atp_c + glc__D_c --> adp_c + g6p_c + h_c
ENO : 2pg_c <=> h2o_c + pep_c
PGM : 2pg_c <=> 3pg_c
GAPD : g3p_c + nad_c + pi_c <=> 13dpg_c + h_c + nadh_c
PGI : g6p_c <=> f6p_c
TPI : dhap_c <=> g3p_c

Metabolites
pep_c : C3H2O6P
h_c : H
adp_c : C10H12N5O10P2
pyr_c : C3H3O3
atp_c : C10H12N5O13P3
f6p_c : C6H11O9P
fdp_c : C6H10O12P2
dhap_c : C3H5O6P
g3p_c : C3H5O6P
3pg_c : C3H4O7P
13dpg_c : C3H4O10P2
glc__D_c : C6H12O6
g6p_c : C6H11O9P
2pg_c : C3H4O7P
h2o_c : H2O
pi_c : HO4P
nad_c : C21H26N7O14P2
nadh_c : C21H27N7O14P2

Genes
b1854 is associated with reactions: {PYK}
b1676 is associated with reactions: {PYK}
b1723 is associated with reactions: {PFK}
b3916 is associated with reactions: {PFK_2, PFK, PFK_3}
b1773 is associated with reactions: {FBA}
b2097 is associated with reactions: {FBA}
b2925 is associated with reactions: {FBA3, FBA}
b2926 is associated with reactions: {PGK}
b2388 is associated with reactions: {HEX1}
b2779 is associated with reactions: {ENO}
b0755 is associated with reactions: {PGM}
b4395 is associated with reactions: {PGM}
b3612 is associated with reactions: {PGM}
b1779 is associated with reactions: {GAPD, E4PD}
b4025 is associated with reactions: {PGI}
b3919 is associated with reactions: {TPI}
```

(b) **Fill in the stoichiometry matrix.**

Table 2: Stochiometry matrix

	HEX1	PGI	PFK	FBA	TPI	GAPD	PGK	PGM	ENO	PYK
ATP	-1		-1				-1			1
GLC	-1									
ADP	1		1				1			-1
G6P	1	-1								
H	1		1			1				-1
F6P		1	-1							
FDP			1	-1						
DHAP				1	-1					
G3P				1	1	-1				
NAD						-1				
PI						-1				
13DPG						1	1			
NADH						1				
3PG							-1	1		
2PG								-1	-1	
PEP									1	-1
H2O									1	
PYR										1

(c) **Create the model for the given chain of reactions. Provide the number of reactions, metabolites and genes in it. (Python)**

We obtain 10 reactions implying 16 genes and 18 metabolites. Code below.

Listing 1: Correlation network

```
from __future__ import print_function
import cobra.test
from cobra import Model, Reaction, Metabolite

# "textbook" and "salmonella" are also valid arguments
model = cobra.test.create_test_model("ecoli")

# print(len(model.reactions))
# print(len(model.metabolites))
# print(len(model.genes))

reactions = {"HEX1", "PGI", "PFK", "FBA", "TPI", "GAPD", "PGK", "PGM", "ENO",
    "PYK"}

for r in reactions:
    print(r)

cobra_model = Model("Model_assignment9")

fo = open("Output.txt", "w+")


# Here we add the specific reactions we need in our new model
# add_reaction is deprecated but it doesn't seem to work with add_reactions
    and the documentation is hard to read
#cobra_model.add_reactions(model.reactions.get_by_id(reactions))

for r in reactions:
    cobra_model.add_reaction(model.reactions.get_by_id(r))
```

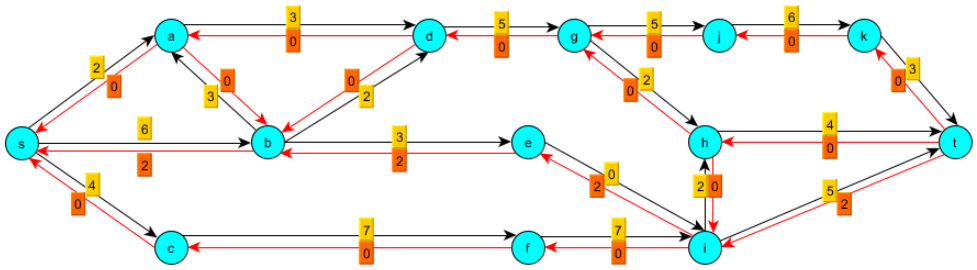
```
print("\n-----------------------------------------\nReactions\n
    -----------------------------------------")

fo.write("Reactions\n")

for reaction in cobra_model.reactions:
    s = "%s : %s" % (reaction.id, reaction.reaction)
    print(s)
    fo.writelines(s + "\n")

fo.write("\nMetabolites\n")
print("\n-----------------------------------------\nMetabolites\n
    -----------------------------------------")
for x in cobra_model.metabolites:
    s = '%s : %s' % (x.id, x.formula)
    print(s)
    fo.writelines(s + "\n")


fo.write("\nGenes\n")
print("\n-----------------------------------------\nGenes\n
    -----------------------------------------")

for gene in cobra_model.genes:
    reactions_list_str = "{" + ", ".join((i.id for i in gene.reactions)) + "}"
    print("%s is associated with reactions: %s" % (gene.id, reactions_list_str
        ))
    fo.writelines("%s is associated with reactions: %s" % (gene.id,
        reactions_list_str) + "\n")

fo.close()

print("Number of reactions")
print(len(cobra_model.reactions))

print("Number of genes")
print(len(cobra_model.genes))

print("Number of metabolites")
print(len(cobra_model.metabolites))
```

Exercise 9.3: FFEK Algorithm

(a) **Apply the Ford, Fulkerson, Edmonds, and Karp (FFEK) algorithm explained in the lecture to determine the s-t-cut and the capacity of the network given below. For each iteration, give the indices of the nodes, the resulting f-augmenting path with its capacity, and the updated val(f). Sketch the newly found f-augmenting paths. Also update the currents through the arcs. If you find multiple possible paths from s to t with the same length, then choose the one with the highest Δq**

FFEK iterations[1] For each iteration, the displayed graph is the result after applying the FFEK algorithm. The traces of the algorithm are stored in the queue (net nodes to visit), the Visited nodes and the trace which indicates from which node a node has been discovered. The "current" list store the order of the nodes on which the algorithm is applied and the resulting path and flow are also marked in the table. The figure 7 shows the final flows and capacity.

The max flow here is 11 and so is the min-cut. For example, we can split the vertex as {S, a, b, c, d, e, f } and {g, h, i, j, k, t} and the value of the cut is $5 + 2 + 4 = 11$.



Iteration:		
Queue	Visited	Trace
Sx Ax Bx Cx Dx Ex Fx Gx Ix H J T	S A B C D E F G I H J T	a -> s b -> s c -> s d -> a e -> b f -> e g -> d i -> e h -> g j -> g t -> i

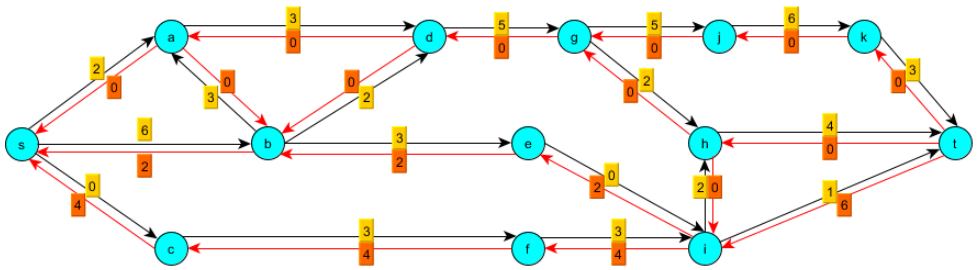
Current: S - A - B - C - D - E - F - G - I - T

Path: S - B - E - I - T

Flow: 2

Figure 2: Iteration 1

[1] Helpful video: youtube.com/watch?v=GiN3jRdgxU4



Iteration:

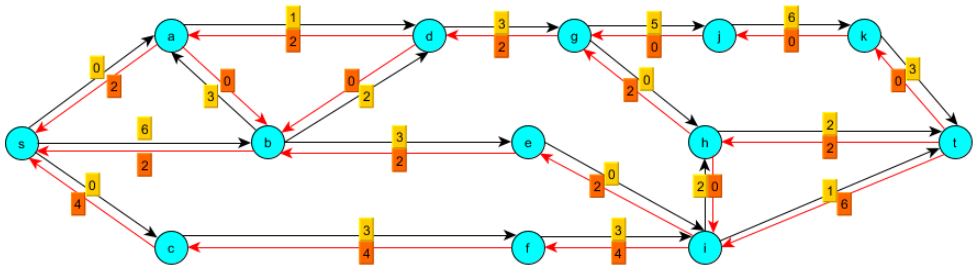
Queue	Visited	Trace
Sx Ax Bx Cx Dx Ex Fx Gx I H J T	S A B C D E F G I H J T	A - S B - S C - S D - A E - B F - C G - D I - F H - G J - G T - I

Current: S - A - B - C - D - E - F - G - I - T

Path: S - C - F - I - T

Flow: 4

Figure 3: Iteration 2



Iteration:

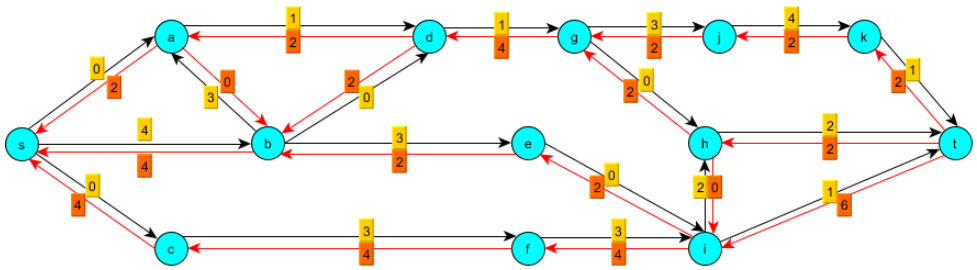
Queue	Visited	Trace
Sx Ax Bx Dx Ex Gx H J T	S A B D E G H J T	A - S B - S D - A E - B G - D H - G J - G T - H

Current: S - A - B - D - E - G - H - T

Path: A - D - G - H - T

Flow: 2

Figure 4: Iteration 3



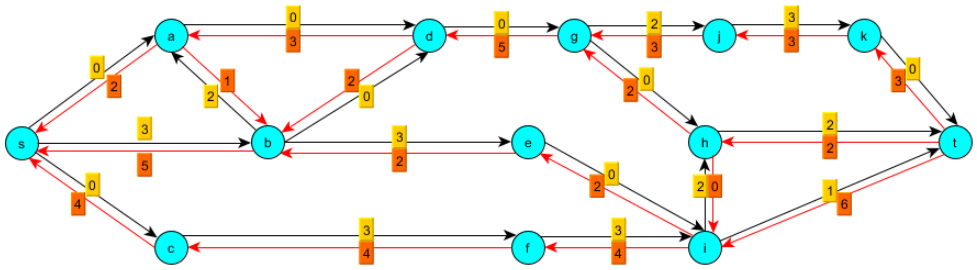
Iteration:		
Queue	Visited	Trace
Sx Bx Dx Ex Gx Ax Jx K T	S B D E G A J K T	B - S D - B E - B G - D J - G A - D K - J T - K

Current: S - B - D - E - G - A - J - K - T

Path: S - B - D - G - J - K - T

Flow: 2

Figure 5: Iteration 4



Iteration:		
Queue	Visited	Trace
Sx Bx Ax Ex Dx Gx Jx Kx T	S B A E D G J K T	B - S A - B E - B D - A G - D J - G K - J T - K

Current: S -B - A - E - D - G - J - K - T

Path: S - B - A - D - G - J - K - T

Flow: 1

Figure 6: Iteration 5 - Final iteration

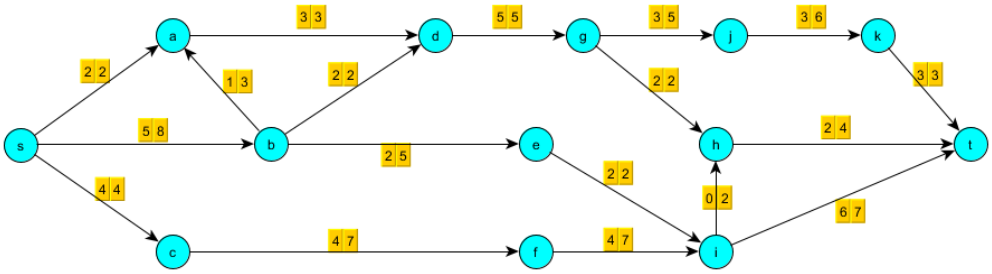


Figure 7: Resulting maximum flow