Selection of modularity in genome-scale metabolic networks

Fengdan Ye
April, 2018

Please cite my GitHub page https://github.com/Fengdan92/KEGG-database for ANY use of my codes

Introduction

Flux Balance Analysis (FBA)

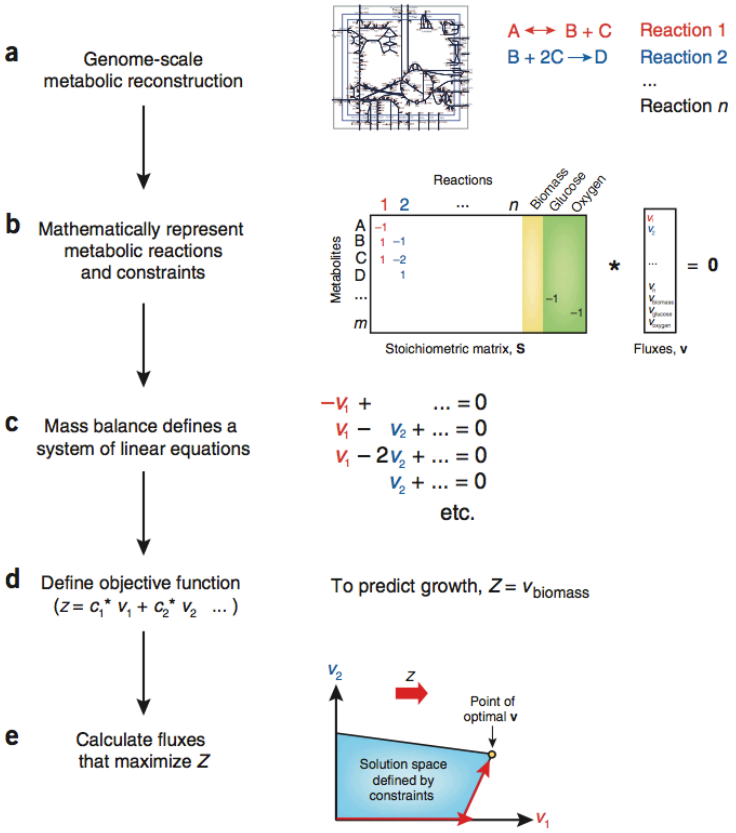
[Orth and Palsson. Nature Biotechnology **28**, 245-248 (2010)]

Toolbox: COBRA and linear programming solver

Biomass production:
The rate at which metabolic compounds are converted into biomass constituents such as nucleic acids, proteins and lipids.

To predict growth in FBA, a **biomass reaction** is added. The reaction consumes precursor metabolites at stoichiometries that simulate biomass production.

Stoichiometries are decided based on experimental measurements, and are scaled so that **the flux through biomass reaction $v_{biomass}$ is the exponential growth rate of the organism (h^{-1}).**



*i*ML1515: the newest knowledgebase of *E. coli* str. K-12 substr. MG1655

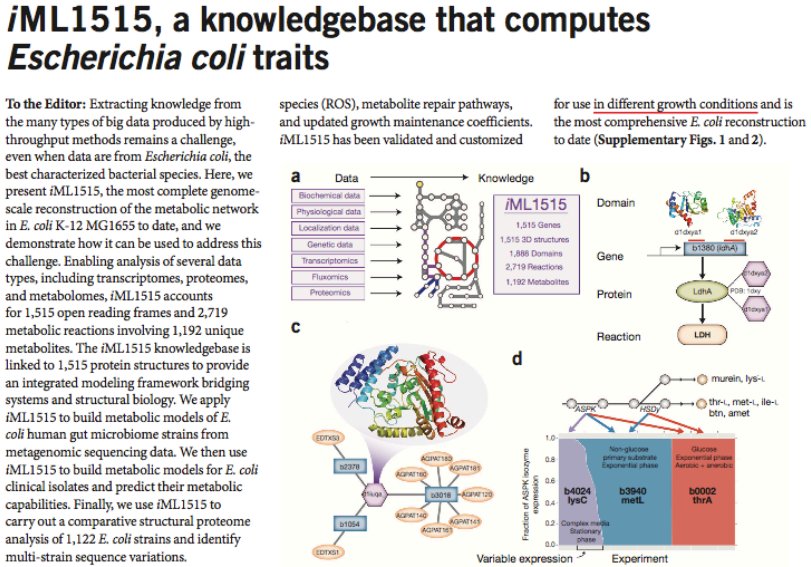
[Monk et al. Nature Biotechnology **35**, 904-908 (2017)]

2712 reactions, including two biomass reactions

1516 genes

1877 metabolites

- Manually constructed based on database such as KEGG, Pubmed, EcoCyc, PDB and CHEBI.
- The model is FBA-ready with all necessary information.



***i*ML1515, a knowledgebase that computes *Escherichia coli* traits**

To the Editor: Extracting knowledge from the many types of big data produced by high-throughput methods remains a challenge, even when data are from *Escherichia coli*, the best characterized bacterial species. Here, we present *i*ML1515, the most complete genome-scale reconstruction of the metabolic network in *E. coli* K-12 MG1655 to date, and we demonstrate how it can be used to address this challenge. Enabling analysis of several data types, including transcriptomes, proteomes, and metabolomes, *i*ML1515 accounts for 1,515 open reading frames and 2,719 metabolic reactions involving 1,192 unique metabolites. The *i*ML1515 knowledgebase is linked to 1,515 protein structures to provide an integrated modeling framework bridging systems and structural biology. We apply *i*ML1515 to build metabolic models of *E. coli* human gut microbiome strains from metagenomic sequencing data. We then use *i*ML1515 to build metabolic models for *E. coli* clinical isolates and predict their metabolic capabilities. Finally, we use *i*ML1515 to carry out a comparative structural proteome analysis of 1,122 *E. coli* strains and identify multi-strain sequence variations.

species (ROS), metabolite repair pathways, and updated growth maintenance coefficients. *i*ML1515 has been validated and customized for use in different growth conditions and is the most comprehensive *E. coli* reconstruction to date (**Supplementary Figs. 1 and 2**).

Model obtained from: http://bigg.ucsd.edu/models/iML1515

*i*ML1515: Metabolites

1877 metabolites

Three different compartments:

e: extracellular

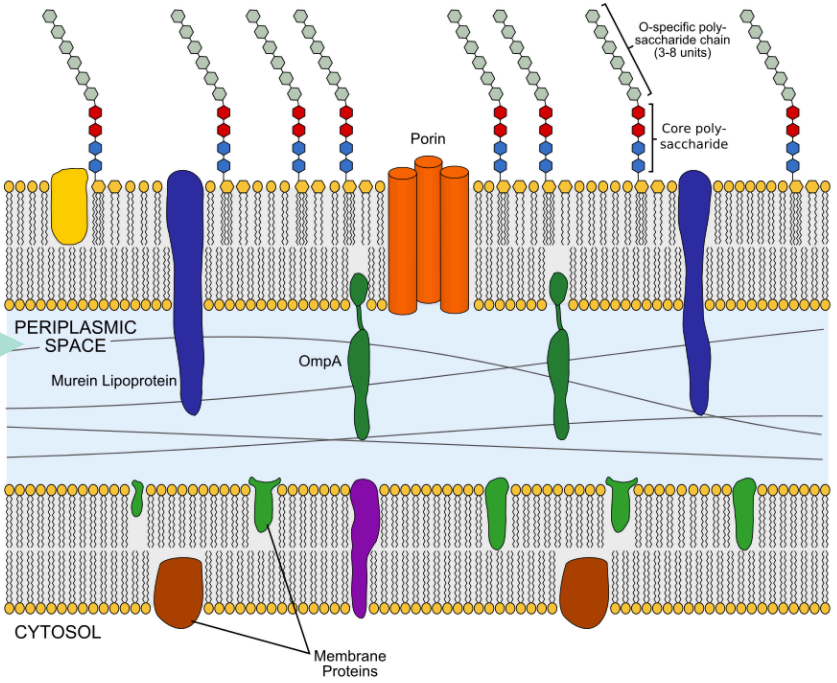
e.g. glc__D_e

p: periplasm

e.g. glc__D_p

c: cellular

e.g. glc__D_c



*i*ML1515: Reactions

2712 reactions

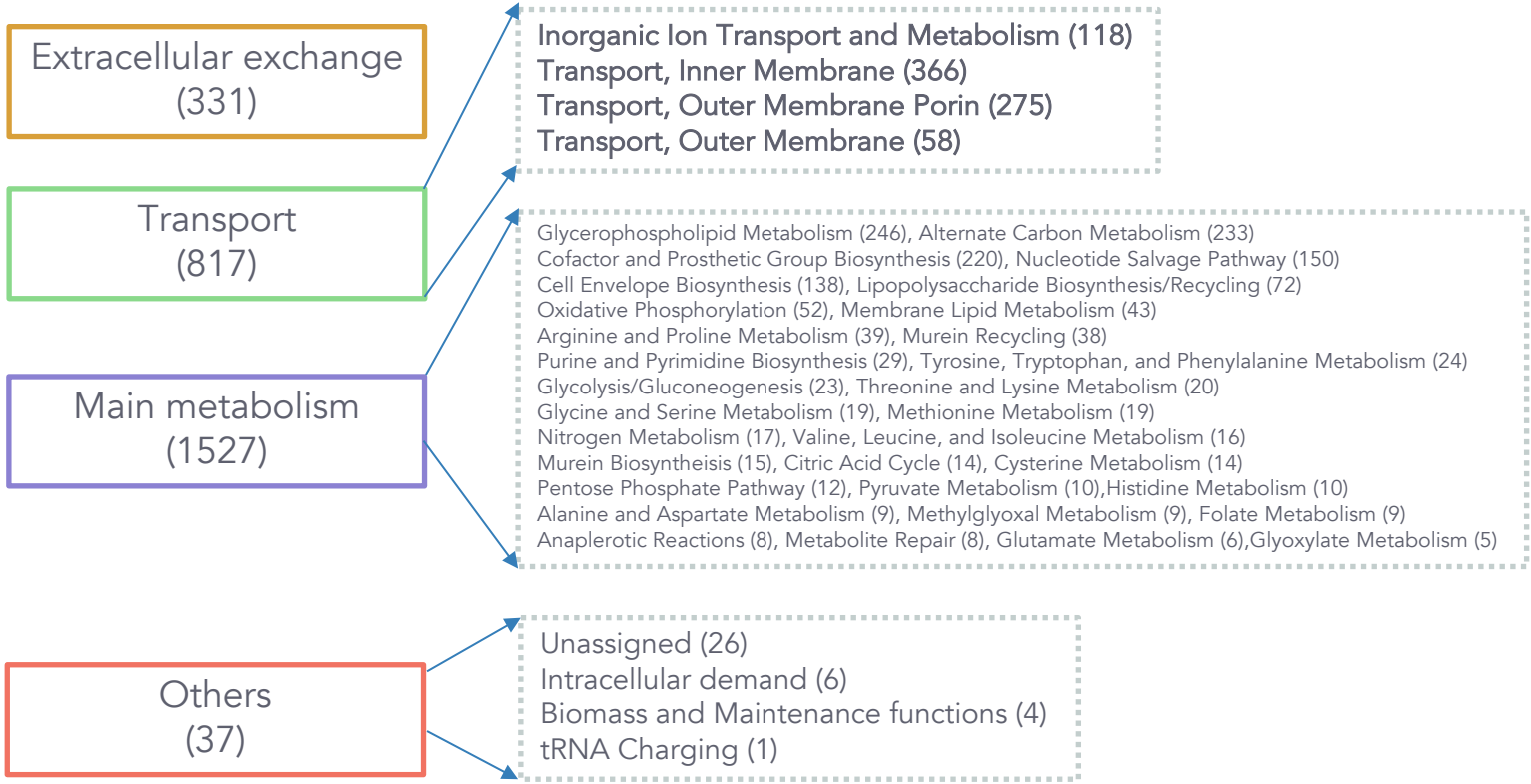
Each reaction has the following information:

Items	Comments
Abbreviation	Abbreviation of the reaction, e.g. ALATA_D2.
Description	Description of the reaction, e.g. D-alanine transaminase.
Reaction	Direction and stoichiometry. e.g. pydx5p_c + ala__D_c -> pyam5p_c + pyr_c
GPR	Gene-protein-reaction (GPR) relationships, e.g. b2551 (gene ID).
Lower bound	Lower bound of flux for this reaction in FBA, unit: mmol gDW^{-1} h^{-1}
Upper bound	Upper bound of flux for this reaction in FBA, unit: mmol gDW^{-1} h^{-1}
Objective	Objective coefficient of this reaction.
Subsystem	Subsystem this reaction belongs to, e.g. Cofactor and Prosthetic Group Biosynthesis.

Reversible reactions have an upper bound of 1000 mmol gDW^{-1} h^{-1} and a lower bound of -1000 mmol gDW^{-1} h^{-1}

Irreversible reactions have a lower bound of zero.

*i*ML1515: Reactions



*i*ML1515: Reactions

Extracellular exchange (331)

co2_e <=>
lb=-1000, ub = 1000
lb < 0 means uptake possible

Transport (817)

co2_e <=> co2_p
lb=-1000, ub = 1000
CO2 transport via diffusion (extracellular to periplasm)

Others (37)

Biomass and Maintenance functions
1. ATP maintenance requirement. Lb= Non-growth associated maintenance (NGAM), ub=1000.
2. Biomass – core/WT. Includes growth associated maintenance (GAM) as cost of energy.
Both NGAM and GAM are determined by fitting experimental data (with varying growth conditions).

FBA on *i*ML1515: setting parameters for growth media

[Supplementary Methods of Monk et al. Nature Biotechnology **35**, 904-908 (2017)]

Objective: **Biomass_core** ($Z=v_{biomass\_core}$)

	Growth sources	Oxygen	Comments	Variations
Glucose aerobic	Glucose (lb=-10)	lb=-1000	Default	Change glucose or oxygen lbs
Glucose anaerobic	Glucose (lb=-10)	lb=0	No oxygen uptake	Change glucose lb.
Alternative sources (C, N, P or S)	Source (lb=-10)	lb=0 or negative values	Default carbon, nitrogen, phosphorus, and sulfur sources **glc__D, nh4, pi and so4**, lb=-1000. If studying alternative C sources, then glc__D lb=0, similar for N, P and S.	Change source and oxygen lbs.

- **Default:** exchange reactions have a lower bound of **zero**, except for glucose (-10), oxygen (-1000) and all inorganic ions required by the biomass reactions (-1000).
- If not specified, follow default values.

FBA on *i*ML1515: Biomass_core

BIOMASS_Ec_iML1515_core_75p37M

(E. coli biomass objective function (iML1515) - core - with 75.37 GAM estimate)

0.000691 mn2_c + 0.000223 10fthf_c + 0.000223 pydx5p_c + 0.25369 thr__L_c + 0.000223 thmpp_c + 0.000323 ni2_c + 0.21579 ser__L_c + 0.000223 sheme_c + 0.13351 ctp_c + 0.29579 arg__L_c + 0.001831 nad_c + 0.008675 mg2_c + 0.24105 asp__L_c + 0.15369 met__L_c + 0.000223 amet_c + 0.013013 nh4_c + 0.000341 zn2_c + 0.1379 tyr__L_c + 0.09158 cys__L_c + 75.5522 atp_c + 0.000223 thf_c + 0.063814 pe160_p + 0.026166 dttp_c + 0.056843 trp__L_c + 0.34316 lys__L_c + 0.027017 dctp_c + 0.1441 utp_c + 0.19519 k_c + 0.026166 datp_c + 2.5e-05 cobalt2_c + 0.006715 fe2_c + 0.000447 nadp_c + 0.26316 glu__L_c + 0.29053 ile__L_c + 0.26316 gln__L_c + 0.007808 fe3_c + 5.5e-05 udcpdp_c + 0.51369 ala__L_c + 0.18527 phe__L_c + 0.000223 mlthf_c + 0.019456 kdo2lipid4_e + 70.0288 h2o_c + 0.61264 gly_c + 0.000709 cu2_c + 0.005205 ca2_c + 0.005205 cl_c + 0.013894 murein5px4p_p + 0.000223 pheme_c + 0.000223 2ohph_c + 7e-06 mobd_c + 0.094738 his__L_c + 0.000223 ribflv_c + 0.004338 so4_c + 0.075214 pe161_p + 2.6e-05 2fe2s_c + 0.2151 gtp_c + 2e-06 btn_c + 0.42316 val__L_c + 0.22106 pro__L_c + 0.000576 coa_c + 0.24105 asn__L_c + 0.45053 leu__L_c + 0.00026 4fe4s_c + 0.027017 dgtp_c + 0.000223 fad_c + 9.8e-05 succoa_c

-> 75.3772 h_c + 75.3772 adp_c + 75.3732 pi_c + 0.7739 ppi_c

iML1515.xlsx

Sidenote: Biomass_core vs Biomass_WT (wild type)

The "**wild-type**" biomass reaction contains the precursors to all the typical wild-type cellular components of E. coli, while the "**core**" biomass reaction contains the precursors only to essential components.

[Supplementary Methods of Monk et al. Nature Biotechnology **35**, 904-908 (2017)]

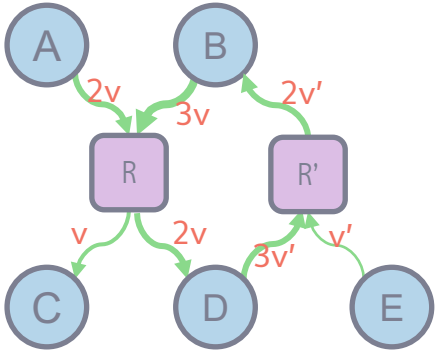
Biomass_core can result in increased accuracy when predicting gene, reaction, and metabolite essentiality and is formulated using experimental data from genetic mutants and knockout strains.

[Feist and Palsson, Curr Opin Microbiol **13**, 344-349 (2010)]

A common problem that arises when using a **BOF based on wild- type** measurements is that potential false positives can be generated when conditionally essential metabolites are inappropriately included in the BOF.

[Feist et al. Molecular Systems Biology **3**, 121 (2007)]

An example of a bipartite metabolic network



R: 2A+3B => C + 2D
Flux through R: v

R′: 3D + E => 2B
Flux through R′: v′

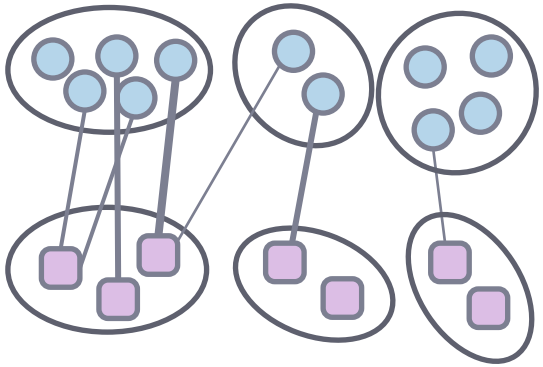
Metabolites

Reactions

There is NO link between same type of nodes – a bipartite network.

The model considers both stoichiometries and fluxes through each reaction and is therefore consistent with flux balance analysis.

Bipartite network and its modularity I



An example of a clustered bipartite network

Undirected networks

For definition of modularity, adopt Murata+, an improvement based on Murata bipartite modularity:

$Graph:\ G(V_1 \cup V_2, E, \omega)$

V_1 and V_2 : the two sets of vertices
E: set of edges e = (i, j).
ω: weights of edges

$$M = \sum_{e \in E} \omega(e)$$

Let P1={C1, C2,…} denote a set of communities in V1, and P2={D1, D2,…} in V2

$Let\ C \in P_1, D \in P_2$

$E_{C,D}$ the set of all edges that connect a vertex in C with a vertex in D

$$e_{C,D} = \frac{1}{2M} \sum_{e \in E_{C,D}} \omega(e)$$

$$a_C = \sum_{D \in P_2} e_{C,D}$$

$$a_D = \sum_{C \in P_1} e_{D,C}$$

Co-cluster of C:

$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{C,D} - a_C a_D)$$

Co-cluster of D:

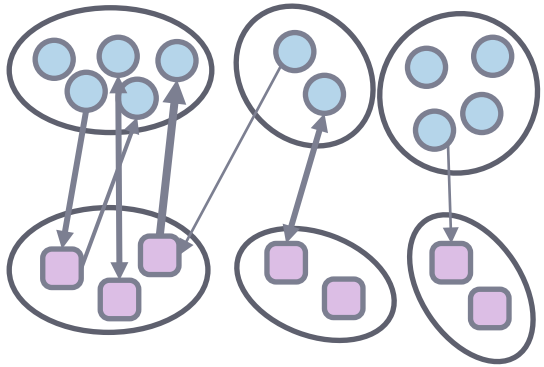
$$C_D = \underset{C \in P_1}{\operatorname{argmax}}(e_{D,C} - a_D a_C)$$

Modularity:

$$Q_{B,undirected} = \sum_{C \in P_1} (e_{C,D_C} - a_C a_{D_C}) + \sum_{D \in P_2} (e_{D,C_D} - a_D a_{C_D})$$

[Murata, "Detecting Communities from Bipartite Networks based on Bipartite Modularities" 2009]
[Pesantez-Cabrera and Kalyanaraman, "Detecting Communitites in Biological Bipartite Networks" 2016]

Bipartite network and its modularity II



An example of a clustered bipartite network

Directed networks

For definition of modularity, adopt Murata+, an improvement based on Murata bipartite modularity:

$Graph:\ G(V_1 \cup V_2, E, \omega)$

V_1 and V_2 : the two sets of vertices
E: set of edges e = (i, j).
ω: weights of edges

$$M = \sum_{e \in E} \omega(e)$$

Let P1={C1, C2,…} denote a set of communities in V1, and P2={D1, D2,…} in V2

$Let\ C \in P_1, D \in P_2$

$E_{C \to D}$ the set of all edges that connect a vertex in C *to* a vertex in D

$$e_{C \to D} = \frac{1}{M} \sum_{e \in E_{C \to D}} \omega(e)$$

$$a_C^{out} = \sum_{D \in P_2} e_{C \to D}$$

$$a_D^{out} = \sum_{C \in P_1} e_{D \to C}$$

Co-cluster of C:

$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{C \to D} - a_C^{out} a_D^{in})$$

Co-cluster of D:

$$C_D = \underset{C \in P_1}{\operatorname{argmax}}(e_{D \to C} - a_D^{out} a_C^{in})$$

Modularity:

$$Q_{B,directed} = \sum_{C \in P_1} (e_{C \to D_C} - a_C^{out} a_{D_C}^{in}) + \sum_{D \in P_2} (e_{D \to C_D} - a_D^{out} a_{C_D}^{in})$$

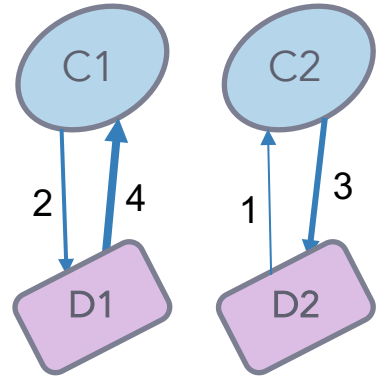
Problem: asymmetric, can define:

$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{D \to C} - a_D^{out} a_C^{in})$$

And Q will be different

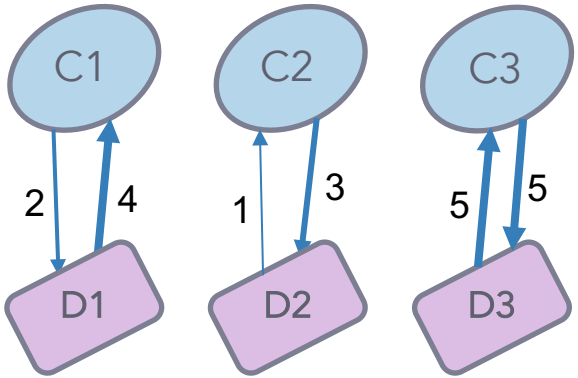
Bipartite network and its modularity – examples of definition II

$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{C \to D} - a_C^{out} a_D^{in})$$



Example 1

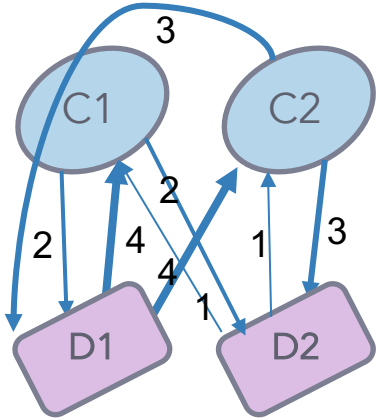
$$Q_{B,directed} = (0.2 - 0.2 * 0.2) + (0.3 - 0.3 * 0.3) + (0.4 - 0.4 * 0.4) + (0.1 - 0.1 * 0.1) = 0.7$$



Example 2

$$Q_{B,directed} = (0.1 - 0.1 * 0.1) + (0.15 - 0.15 * 0.15) + (0.25 - 0.25 * 0.25) + (0.2 - 0.2 * 0.2) + (0.05 - 0.05 * 0.05) + (0.25 - 0.25 * 0.25) = 0.8$$

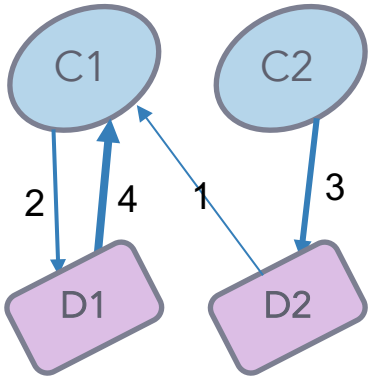
Bipartite network and its modularity – examples of definition II



$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{C\to D} - a_C^{out} a_D^{in})$$

Example 3

$$Q_{B,directed} = (0.1 - 0.2 * 0.25) + (0.15 - 0.3 * 0.25) + (0.05 - 0.1 * 0.25) + (0.2 - 0.4 * 0.25) = 0.25$$

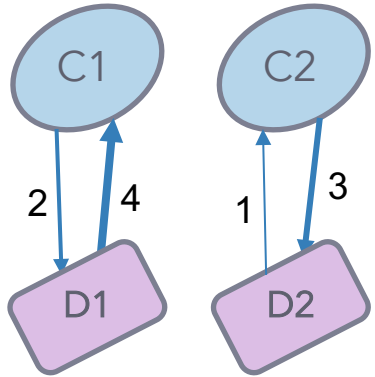


Example 4

$$Q_{B,directed} = (0.2 - 0.2 * 0.2) + (0.3 - 0.3 * 0.3) + (0.4 - 0.4 * 0.5) + (0.1 - 0.1 * 0.5) = 0.62$$

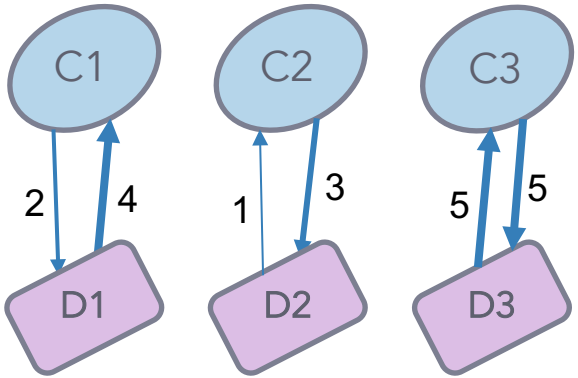
Bipartite network and its modularity – examples of definition II reversed

$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{D \to C} - a_D^{out} a_C^{in})$$



Example 1

$$Q_{B,directed} = (0.4 - 0.4 * 0.4) + (0.1 - 0.1 * 0.1) + (0.2 - 0.2 * 0.2) + (0.3 - 0.3 * 0.3) = 0.7$$

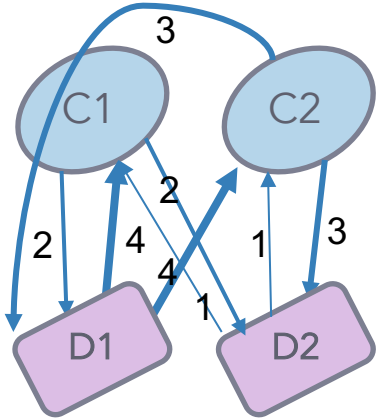


Example 2

$$Q_{B,directed} = (0.1 - 0.1 * 0.1) + (0.15 - 0.15 * 0.15) + (0.25 - 0.25 * 0.25) + (0.2 - 0.2 * 0.2) + (0.05 - 0.05 * 0.05) + (0.25 - 0.25 * 0.25) = 0.8$$

Bipartite network and its modularity – examples of definition II reversed

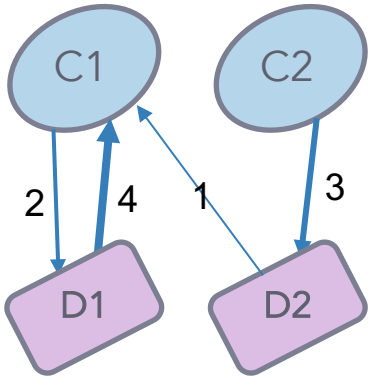
$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{D \to C} - a_D^{out} a_C^{in})$$



Example 3

$$Q_{B,directed} = (0.15 - 0.3 * 0.25) + (0.15 - 0.3 * 0.25) + (0.2 - 0.4 * 0.25) + (0.2 - 0.4 * 0.25) = 0.35$$

Note that it is different from non-reversed definition

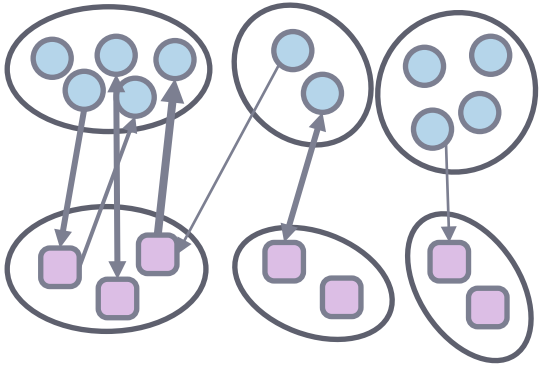


Example 4

$$Q_{B,directed} = (0.2 - 0.2 * 0.2) + (0.3 - 0.3 * 0.3) + (0.4 - 0.4 * 0.5) + (0) = 0.57$$

Note that it is different from non-reversed definition

Bipartite network and its modularity III



An example of a clustered bipartite network

Directed networks

For definition of modularity, adopt Murata+, an improvement based on Murata bipartite modularity:

$Graph:\ G(V_1 \cup V_2, E, \omega)$

V_1 and V_2 : the two sets of vertices
E: set of edges e = (i, j).
ω: weights of edges

$$M = \sum_{e \in E} \omega(e)$$

Let P1={C1, C2,…} denote a set of communities in V1, and P2={D1, D2,…} in V2

$Let\ C \in P_1, D \in P_2$

$E_{C \to D}$ the set of all edges that connect a vertex in C *to* a vertex in D

$$e_{C \to D} = \frac{1}{M} \sum_{e \in E_{C \to D}} \omega(e)$$

$$a_C^{out} = \sum_{D \in P_2} e_{C \to D}$$

$$a_D^{out} = \sum_{C \in P_1} e_{D \to C}$$

Co-cluster of C:

$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{C \to D} - a_C^{out} a_D^{in} + e_{D \to C} - a_D^{out} a_C^{in})$$

Co-cluster of D:

$$C_D = \underset{C \in P_1}{\operatorname{argmax}}(e_{D \to C} - a_D^{out} a_C^{in} + e_{C \to D} - a_C^{out} a_D^{in})$$

Modularity:

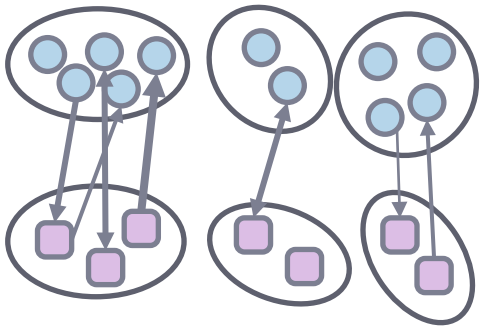
$$Q_{B,directed} = \sum_{C \in P_1} (e_{C \to D_C} - a_C^{out} a_{D_C}^{in} + e_{D_C \to C} - a_{D_C}^{out} a_C^{in}) + \sum_{D \in P_2} (e_{D \to C_D} - a_D^{out} a_{C_D}^{in} + e_{C_D \to D} - a_{C_D}^{out} a_D^{in})$$

Might be problematic

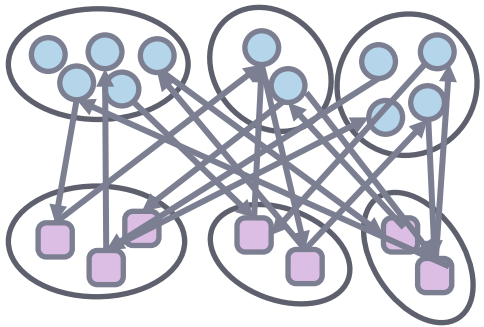
Bipartite network and its modularity

The null model $a_C a_D$ might be inappropriate (that there is a random chance each pair of cluster is linked), as it implies that there is a chance C_1 and C_2 can be linked (P= $a_{C1} a_{C2}$). That is, the bipartite nature of the network is nowhere to be seen in the null model.

Qualitatively, we have the following intuition:



$Q \to 1$



$Q \to 0$

Bipartite network and its modularity IV

Based on previous arguments, an attempt to define bipartite directed modularity is as follows:

$Graph$: $G(V_1 \cup V_2, E, \omega)$

V_1 and V_2 : the two sets of vertices
E: set of edges e = (i, j).
ω: weights of edges

$$M = \sum_{e \in E} \omega(e)$$

Let P1={C1, C2,…} denote a set of communities in V1, and P2={D1, D2,…} in V2

$Let\ C \in P_1, D \in P_2$

$E_{C \to D}$ the set of all edges that connect a vertex in C *to* a vertex in D

Null model:
The ***out*** degrees of each cluster are evenly spilt between ***all*** clusters of the other type.

$$e_{C \to D} = \frac{1}{M} \sum_{e \in E_{C \to D}} \omega(e)$$

$$a_C^{out} = \sum_{D \in P_2} e_{C \to D}$$

$$a_D^{out} = \sum_{C \in P_1} e_{D \to C}$$

Co-cluster of C:

$$D_C = \underset{D \in P_2}{\operatorname{argmax}}(e_{C \to D})$$

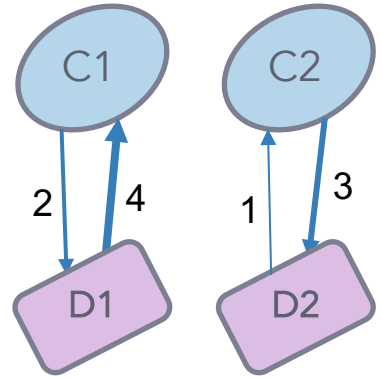
Co-cluster of D:

$$C_D = \underset{C \in P_1}{\operatorname{argmax}}(e_{D \to C})$$

Modularity:

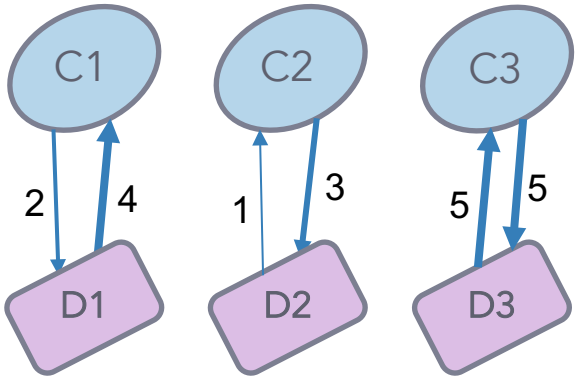
$$Q_{B,directed} = \sum_{C \in P_1}\left(e_{C \to D_C} - \frac{a_C^{out}}{total\ \#\ of\ Ds}\right) + \sum_{D \in P_2}\left(e_{D \to C_D} - \frac{a_D^{out}}{total\ \#\ of\ Cs}\right)$$

Bipartite network and its modularity – examples of definition IV



Example 1

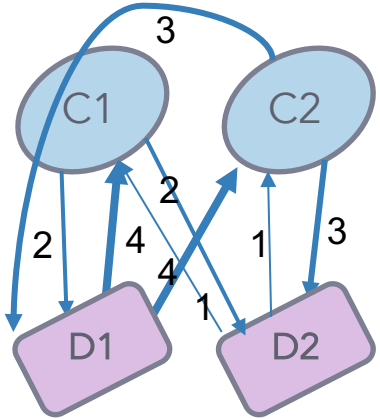
$$Q_{B,directed} = (0.2 - 0.1) + (0.3 - 0.15) + (0.4 - 0.2) + (0.1 - 0.05) = 0.5$$



Example 2

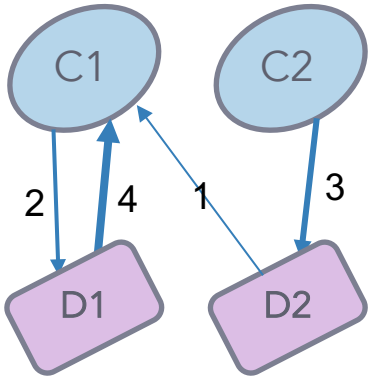
$$Q_{B,directed} = (0.1 - 0.1/3) + (0.15 - 0.15/3) + (0.25 - 0.25/3) + (0.2 - 0.2/3) + (0.05 - 0.05/3) + (0.25 - 0.25/3) = 2/3$$

Bipartite network and its modularity – examples of definition IV



Example 3

$$Q_{B,directed} = (0.1 - 0.2/2) + (0.15 - 0.3/2) + (0.05 - 0.1/2) + (0.2 - 0.4/2) = 0$$



Example 4

$$Q_{B,directed} = (0.2 - 0.2/2) + (0.3 - 0.3/2) + (0.4 - 0.4/2) + (0.1 - 0.1/2) = 0.5$$

This is not good, it gets the same modularity as Example 1.

Bipartite network and its modularity

After using the above examples to compare definition *II*, *II Reversed* and *IV*. It looks like *II* is better than *IV*.

The reason that *II* might be better *II Reversed* is that we are interested in downstream flux: from media to growth.

Therefore, the following growth will be used:

Definition II

$$e_{C\to D} = \frac{1}{M}\sum_{e\in E_{C\to D}} \omega(e)$$

$$a_C^{out} = \sum_{D\in P_2} e_{C\to D}$$

$$a_D^{out} = \sum_{C\in P_1} e_{D\to C}$$

Co-cluster of C:

$$D_C = \underset{D\in P_2}{\operatorname{argmax}}(e_{C\to D} - a_C^{out}a_D^{in})$$

Co-cluster of D:

$$C_D = \underset{C\in P_1}{\operatorname{argmax}}(e_{D\to C} - a_D^{out}a_C^{in})$$

Modularity:

$$Q_{B,directed} = \sum_{C\in P_1}(e_{C\to D_C} - a_C^{out}a_{D_C}^{in}) + \sum_{D\in P_2}(e_{D\to C_D} - a_D^{out}a_{C_D}^{in})$$

Programming

Bipartite directed modularity definition

Definition II

$$e_{C\to D} = \frac{1}{M}\sum_{e\in E_{C\to D}} \omega(e)$$

$$a_C^{out} = \sum_{D\in P_2} e_{C\to D}$$

$$a_D^{out} = \sum_{C\in P_1} e_{D\to C}$$

Co-cluster of C:

$$D_C = \underset{D\in P_2}{\operatorname{argmax}}(e_{C\to D} - a_C^{out} a_D^{in})$$

Co-cluster of D:

$$C_D = \underset{C\in P_1}{\operatorname{argmax}}(e_{D\to C} - a_D^{out} a_C^{in})$$

Modularity:

$$Q_{B,directed} = \sum_{C\in P_1}(e_{C\to D_C} - a_C^{out} a_{D_C}^{in}) + \sum_{D\in P_2}(e_{D\to C_D} - a_D^{out} a_{C_D}^{in})$$

$Graph:\ G(V_1 \cup V_2, E, \omega)$

V_1 and V_2 : the two sets of vertices
E: set of edges e = (i, j).
ω: weights of edges

$$M = \sum_{e\in E} \omega(e)$$

Let P1={C1, C2,…} denote a set of communities in V1, and P2={D1, D2,…} in V2

$Let\ C \in P_1, D \in P_2$

$E_{C\to D}$ the set of all edges that connect a vertex in C *to* a vertex in D

Bipartite modularity algorithm - steps

1. Given an input bipartite graph G(V1 ∪ V2, E, ω), initialize a set of n1 + n2 communities, where n1 = |V1| and n2 = |V2|, by placing each vertex in its own community.
2. At every iteration, all vertices in V1 and V2 are scanned linearly (in an arbitrary order, here we use random order). For each vertex i:

    a) Acquire a list of its candidate communities to which it can potentially migrate to; (here, we iterate through all same-type communities)
    b) Evaluate the modularity gain that would result from the scenario of migrating i to each of the candidate communities.
        1) Identify affected communities S
        2) Loop through each to calculate modularity gain
    c) Finally, migrate vertex i to a candidate community that maximizes the modularity gain, only if such gain is positive (otherwise, no change).

3. A phase ends when the net modularity gain achieved between two consecutive iterations is negligible — i.e., below a certain threshold τi, which we refer to as the iteration cutoff.

Bipartite modularity algorithm - steps

4. Once a phase terminates, a new graph G′(V1′ ∪ V2′,E′,ω′) is generated through a compaction step, which collapses each community to a vertex, and edges and their weights in the new graph corresponds to the strength of edges connecting any two communities.
5. The new compacted graph is input to the next phase (step 1). The algorithm terminates when any two con- secutive phases result in a negligible modularity gain, defined by a threshold τp, which we call phase cutoff.

Reference:
Pesantez-Cabrera Paola, and Ananth Kalyanaraman."Efficient Detection of Communities in Biological Bipartite Networks, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2017.

Node

Attributes:

id
type
inNeighbors
outNeighbors
inMetanodes
outMetanodes
Metanode

Methods:

setMetanode

MetaNode

Attributes:

members
type
inNeighbors
outNeighbors
inComm
outComm
inDegree
outDegree
community

Methods:

updateInComm
updateOutComm
setCommunity

Community

Attributes:

id
members
type
inNeighbors
outNeighbors
inDegree
outDegree
coCluster
modContribution

Methods:

addMetaNode
removeMetaNode
updateIn
updateOut
setCoCloster
setModContribution

Graph

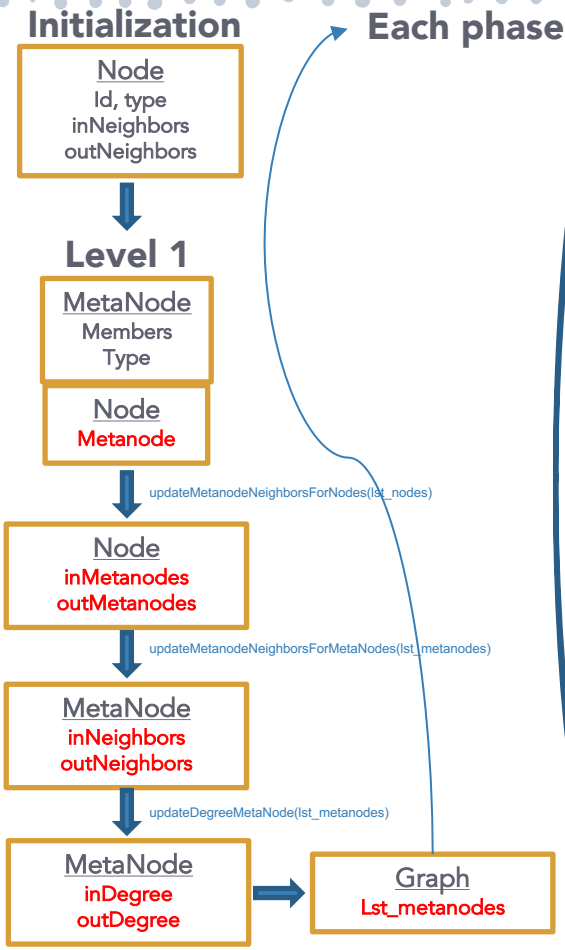
Attributes:

Metanodes
communities

Methods:

updateCommunity

bidiLouvain.py
bidiLouvain_utils.py
bidiLouvain_structs.py



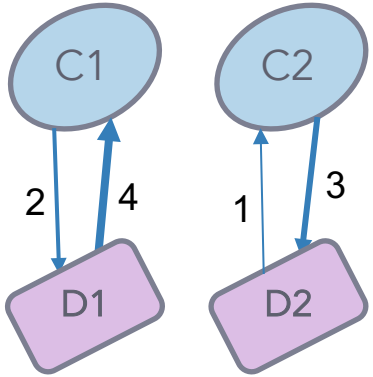
Start of phase
Update metanode communities 1-to-1 (MetaNode.community, comm.id, comm.members, comm.type)
set graph communities (graph.communites)
set inComm and outComm for metanodes (MetaNode.inComm/outComm)
set community neighbors (Community.inNeighbors/outNeighbors)
set community degrees (Community.inDegree/outDegree)
set community co-clusters and modcontribution (Community.coCluster, Community.modContribution)
Start of Iteration
Start of each metanode
Testmove… (keep graph the same!)
Move metanode
Update moved metanode community allegiance (metanoteMoved.community)
Update the inComm and outComm of metanode neighbors of the moved metanode (metanoteMoved.in/outNeighbors. In/outComm)
Update fromComm (fromComm.members/inNeighbors/outNeighbors/inDgree/outDegree)
Update toComm (inComm.members/inNeighbors/outNeighbors/inDgree/outDegree)
Update communities that are connected to the moved metanode. Their in/outNeighbours changed. (affectedCommunity.in/outNeighbours)
Update whole graph's communities (graph.communites)
set all affected communities' co-clusters and modcontribution (Community.coCluster, Community.modContribution)
End of each metanode

Return iterGain
End of iteration
Append g to graphs
g compaction:
New lst_metanodes with members and type (metanode.members/type)
Know metanode in and out neighbors directly (metanode.inNeighbors/outNeighbors)
Know metanode in and out degrees directly (metanode.inDegree/outDegree)
Update node allegiance (node.metanode, node.inMetanodes/outMetanodes)
g=graph(new lst_metanodes)
End of phase

Affected communities (in terms of cocluster and modcontribution) is: fromComm, toComm, and their neighbors.

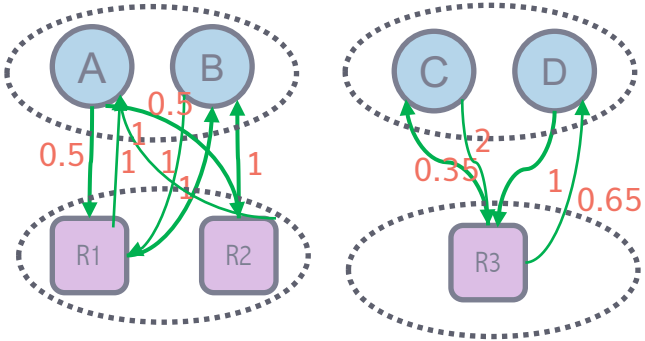
To call the code, use:

```
from bidiLouvain_structs import *
from bidiLouvain_utils import *
from bidiLouvain import *
# maxM is the maximized modularity, graphs is a list of graphs
[maxM, graphs]=bidilouvain(lst_metabolites, lst_reactions, edges)
# s is the final community structure
s = getCommStruc(graphs[-1])
```



Test on simple graph

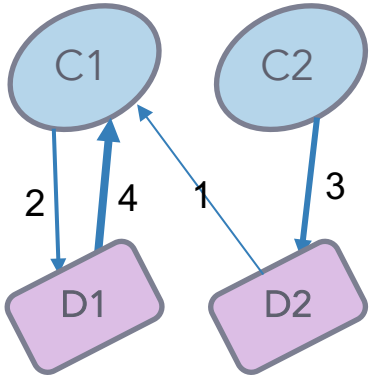
$$Q_{B,directed}$$
$$= (0.2 - 0.2 * 0.2) + (0.3 - 0.3 * 0.3) + (0.4 - 0.4 * 0.4)$$
$$+ (0.1 - 0.1 * 0.1) = 0.7$$



Test2.ipynb – expected result verified

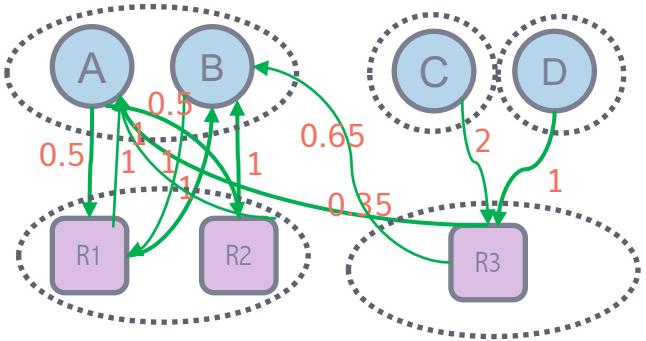
lst_metabolites=['A','B','C','D']
lst_reactions=['R1','R2','R3']
edges={('A','R1'):0.5, ('A','R2'):0.5, ('B','R1'):1.0, ('C','R3'):2.0, ('D','R3'):1.0, ('R1','A'):1.0, ('R1','B'):1.0, ('R2','B'):1.0, ('R2', 'A'):1.0, ('R3','C'):0.35, ('R3', 'D'): 0.65}

maxM = 0.7, community structure verified



Example 4

$$Q_{B,directed}$$
$$= (0.2 - 0.2 * 0.2) + (0.3 - 0.3 * 0.3) + (0.4 - 0.4 * 0.5)$$
$$+ (0.1 - 0.1 * 0.5) = 0.62$$



maxM=0.62. Algorithm did not combine C and D into 1 community, but that's fine.

Test on random, more complex graph

Here we found a problem: compare floats to zero is not feasible, and the program never really remove and neighbors because it thinks 1e-16 != 0.

To solve this problem, I have changed all == 0 to < 1e-5. The choice of threshold is *arbitrary* and might affect performance.

```
from randomSample import *
import string
list1=list(string.ascii_lowercase)
list2 = list(map(str, range(10 + 1)))
# randomly generated edges
edges=rndSampleGenerator(list1, list2)
from bidiLouvain_structs import *
from bidiLouvain_utils import *
from bidiLouvain import *
[maxM, graphs]=bidilouvain(list1, list2, edges)
s=getCommStruc(graphs[-1])
print(s)
```

The program was tested on 10,000 random samples using the code on the left, without reporting errors.

Based on the successful performance of the program on:
(i) Intuitive but simple model (correct M is known)
(ii) Unintuitive but random structure (correct M is unknown)
It is concluded that the program is working correctly.

Flux Balance Analysis

Reactions in *i*ML1515.xlsx

iML1515.mat downloaded from http://bigg.ucsd.edu/models/iML1515

In MATLAB:

```
initCobraToolbox
model=readCbModel('iML1515.mat');
writeCbModel(model, 'fileName', 'iML1515.xls','format','xls')
```

On page 9, it is mentioned that "Default: exchange reactions have a lower bound of zero, except for glucose (-10), oxygen (-1000) and all inorganic ions required by the biomass reactions (-1000)."

The following inorganic ions are directly in **biomass_core's** equation and have lb=-1000:
Mn2, Ni2, Mg2, NH4, Zn2, K, Co2+, Fe2, Fe3+, Cu2, Ca2, Cl, mobd, SO4, H, pi

In addition, the following exchange has negative lower bound:
H2O(-1000), CO2 (-1000), Na(-1000), sel(-1000), slnt(-1000), tungs(-1000), glc__D (-10), O2(-1000)

FBA on *i*ML1515: Biomass_core

BIOMASS_Ec_iML1515_core_75p37M

(E. coli biomass objective function (iML1515) - core - with 75.37 GAM estimate)

0.000691 mn2_c + 0.000223 10fthf_c + 0.000223 pydx5p_c + 0.25369 thr__L_c + 0.000223 thmpp_c + 0.000323 ni2_c + 0.21579 ser__L_c + 0.000223 sheme_c + 0.13351 ctp_c + 0.29579 arg__L_c + 0.001831 nad_c + 0.008675 mg2_c + 0.24105 asp__L_c + 0.15369 met__L_c + 0.000223 amet_c + 0.013013 nh4_c + 0.000341 zn2_c + 0.1379 tyr__L_c + 0.09158 cys__L_c + 75.5522 atp_c + 0.000223 thf_c + 0.063814 pe160_p + 0.026166 dttp_c + 0.056843 trp__L_c + 0.34316 lys__L_c + 0.027017 dctp_c + 0.1441 utp_c + 0.19519 k_c + 0.026166 datp_c + 2.5e-05 cobalt2_c + 0.006715 fe2_c + 0.000447 nadp_c + 0.26316 glu__L_c + 0.29053 ile__L_c + 0.26316 gln__L_c + 0.007808 fe3_c + 5.5e-05 udcpdp_c + 0.51369 ala__L_c + 0.18527 phe__L_c + 0.000223 mlthf_c + 0.019456 kdo2lipid4_e + 70.0288 h2o_c + 0.61264 gly_c + 0.000709 cu2_c + 0.005205 ca2_c + 0.005205 cl_c + 0.013894 murein5px4p_p + 0.000223 pheme_c + 0.000223 2ohph_c + 7e-06 mobd_c + 0.094738 his__L_c + 0.000223 ribflv_c + 0.004338 so4_c + 0.075214 pe161_p + 2.6e-05 2fe2s_c + 0.2151 gtp_c + 2e-06 btn_c + 0.42316 val__L_c + 0.22106 pro__L_c + 0.000576 coa_c + 0.24105 asn__L_c + 0.45053 leu__L_c + 0.00026 4fe4s_c + 0.027017 dgtp_c + 0.000223 fad_c + 9.8e-05 succoa_c

-> 75.3772 h_c + 75.3772 adp_c + 75.3732 pi_c + 0.7739 ppi_c

iML1515.xlsx

Reactions in *i*ML1515.xlsx

It can be concluded that iML1515.xlsx/iML1515.mat contain the default parameters of E.coli model.
The exchange functions' lower bounds can be changed to study different growth media conditions, and the objective function can be changed between Biomass_core and Biomass_WT. All other parameters should be kept unchanged.

Default

Exchange	Lower bound	Exchange	Lower bound
Mn2	-1000	Modb (molybdate)	-1000
Ni2	-1000	SO4 (default S source)	-1000
Mg2	-1000	H	-1000
NH4 (default N source)	-1000	Pi (default P source)	-1000
Zn2	-1000	H2O	-1000
K	-1000	CO2	-1000
Cobalt2	-1000	Na	-1000
Fe2	-1000	Sel (selenate)	-1000
Fe3	-1000	Slnt (selenite)	-1000
Cu2	-1000	Tungs (tungstate)	-1000
Ca2	-1000	Glc__D (default C source)	-10
Cl	-1000	O2	-1000

Parameters: glucose aerobic (= default)

Exchange	Lower bound
Mn2	-1000
Ni2	-1000
Mg2	-1000
NH4 (default N source)	-1000
Zn2	-1000
K	-1000
Cobalt2	-1000
Fe2	-1000
Fe3	-1000
Cu2	-1000
Ca2	-1000
Cl	-1000

Exchange	Lower bound
Modb (molybdate)	-1000
SO4 (default S source)	-1000
H	-1000
Pi (default P source)	-1000
H2O	-1000
CO2	-1000
Na	-1000
Sel (selenate)	-1000
Slnt (selenite)	-1000
Tungs (tungstate)	-1000
Glc__D (default C source)	-10
O2	-1000

Objective function: Biomass_core or Biomass_WT

```
model=readCbModel('iML1515.mat');
modelGluAerobic = model;
modelGluAerobic = changeObjective (modelGluAerobic, 'BIOMASS_Ec_iML1515_core_75p37M');
FBAGluAerobic = optimizeCbModel(modelGluAerobic,'max');
```

Save fluxes to file (matlab, COBRA)

```
T = table(modelGluAerobic.rxns, modelGluAerobic.rxnNames, FBAGluAerobic.v);
writetable(T,'Flux_GluAerobic.xlsx','FileType','spreadsheet');
```

Convert reaction equation to vertex pairs (python)

```
conda install openpyxl
conda install xlsxwriter
```

Then run Parse_vertex_pairs.ipynb or Parse_vertex_pairs.py

This gives us an xlsx file iML1515_pairs.xlsx and a pickle file dict_all.pkl. The latter can be loaded and used in Python:

```
file = open(r'dict_all.pkl', 'rb')
dict_all = pickle.load(file)
file.close()
```

Dict_all[reac_abbr] gives {(src, dest): stoichiometry}

FCLT: ppp9_c + fe2_c -> 2 h_c + pheme_c

Dict_all['FCLT'] gives:
{('FCLT', 'h_c'): 2.0, ('FCLT', 'pheme_c'): 1.0, ('fe2_c', 'FCLT'): 1.0, ('ppp9_c', 'FCLT'): 1.0}

Save flux info to pickle: run Save_flux.py or Save_flux.ipynb.
The result is saved to dict_flux.pkl. Reaction with flux < 1e-5 and the BIOMASS reactions are removed. BIOMASS were removed, because the flux through them has special unit h-1 and they are not really a reaction.

Now, to calculate modularity: run run_bidiLouvain.py or run_bidiLouvain.ipynb.
Obtain modularity and community structure.

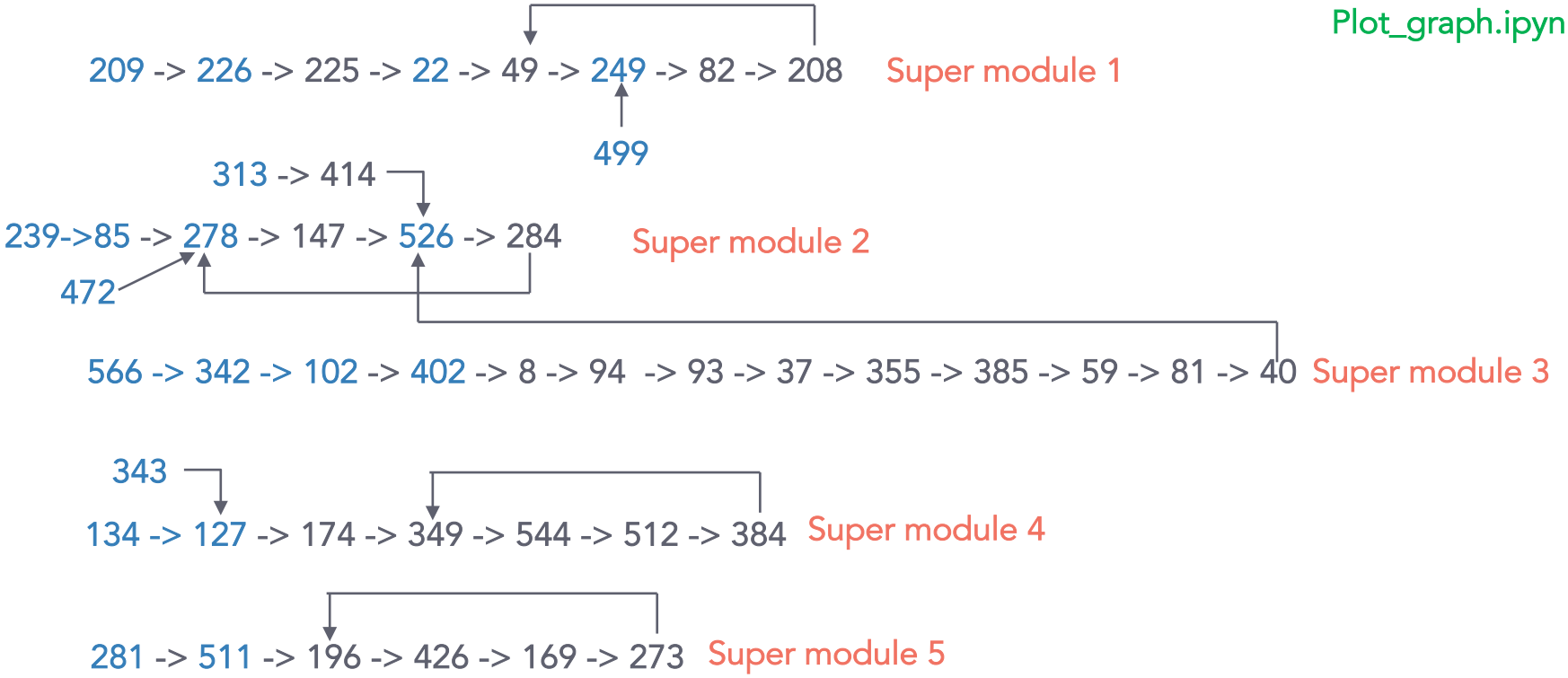
After changing back affected communities to C, C' and their neighbors, runtime becomes ***3 hours.***
Results is saved to: commStruc_gluAerobic.pkl and cocluster_gluAerobic.pkl.

M = 0.712, # of communities: 567, growth rate = 0.8770 h-1

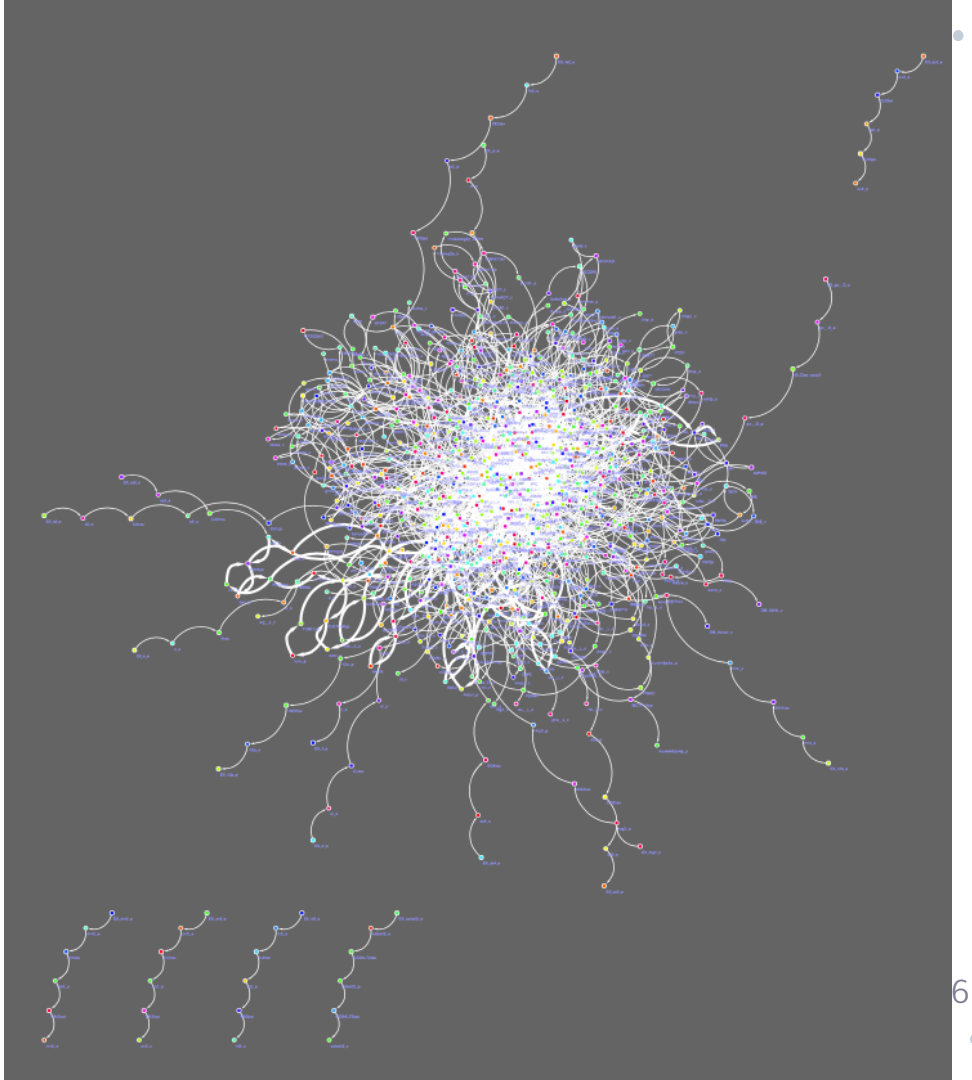
Community structure: focus on big modules

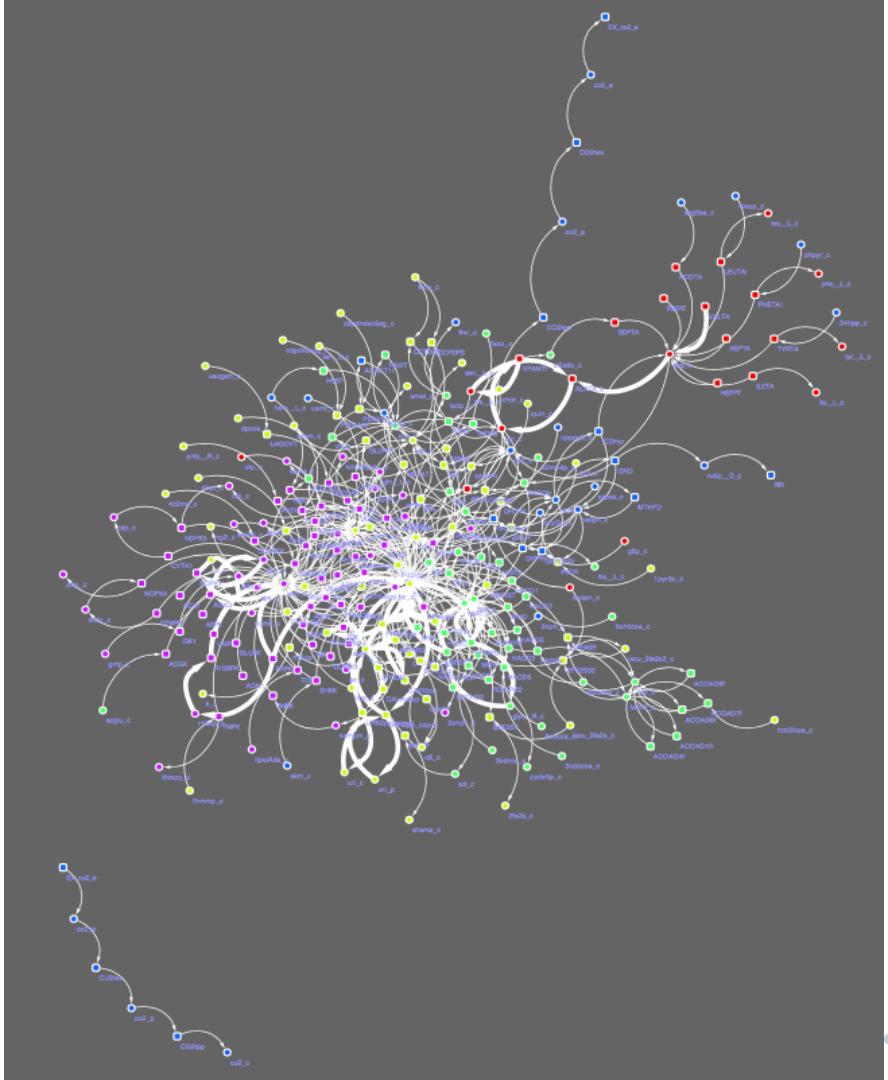
Each number represents one community

Plot_graph.ipynb



Full Network



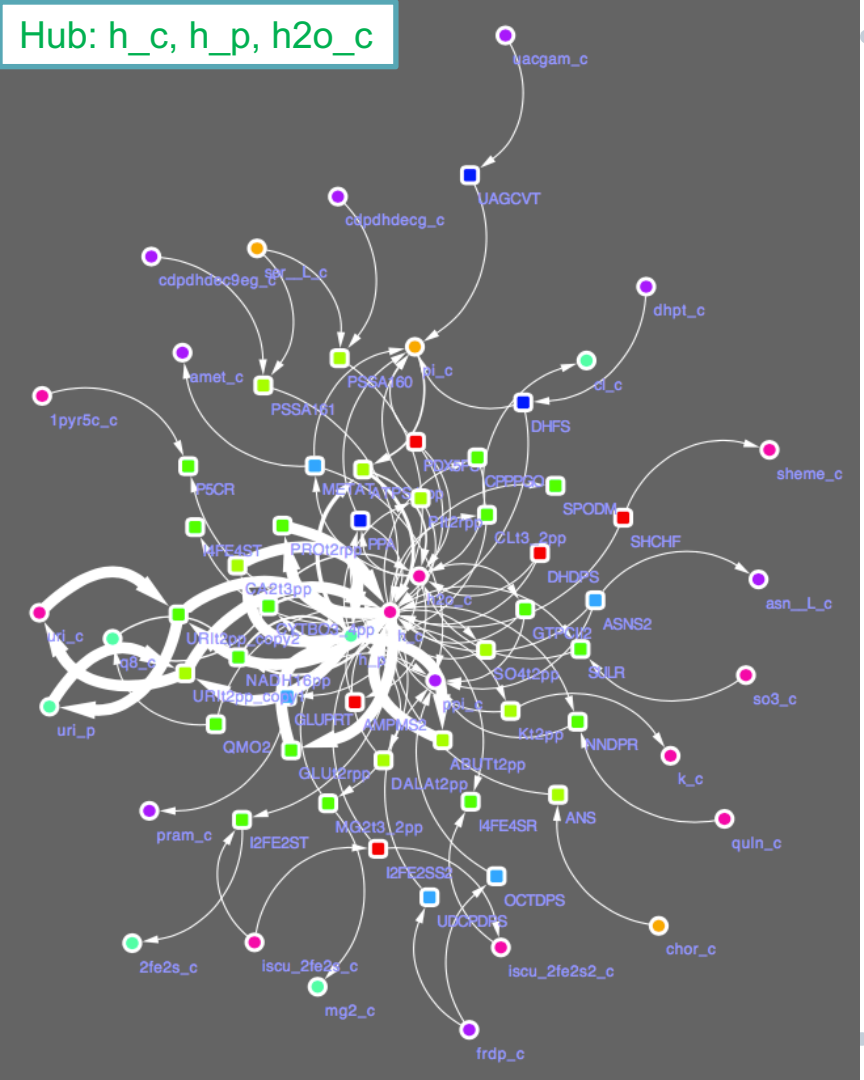


Big modules only

- Super modules 1-5 plotted (Cytoscape). Different colors represent different super modules.

Super module 1
Super module 2
Super module 3
Super module 4
Super module 5

- This is a subset of all nodes, and only edges between subset nodes are kept.



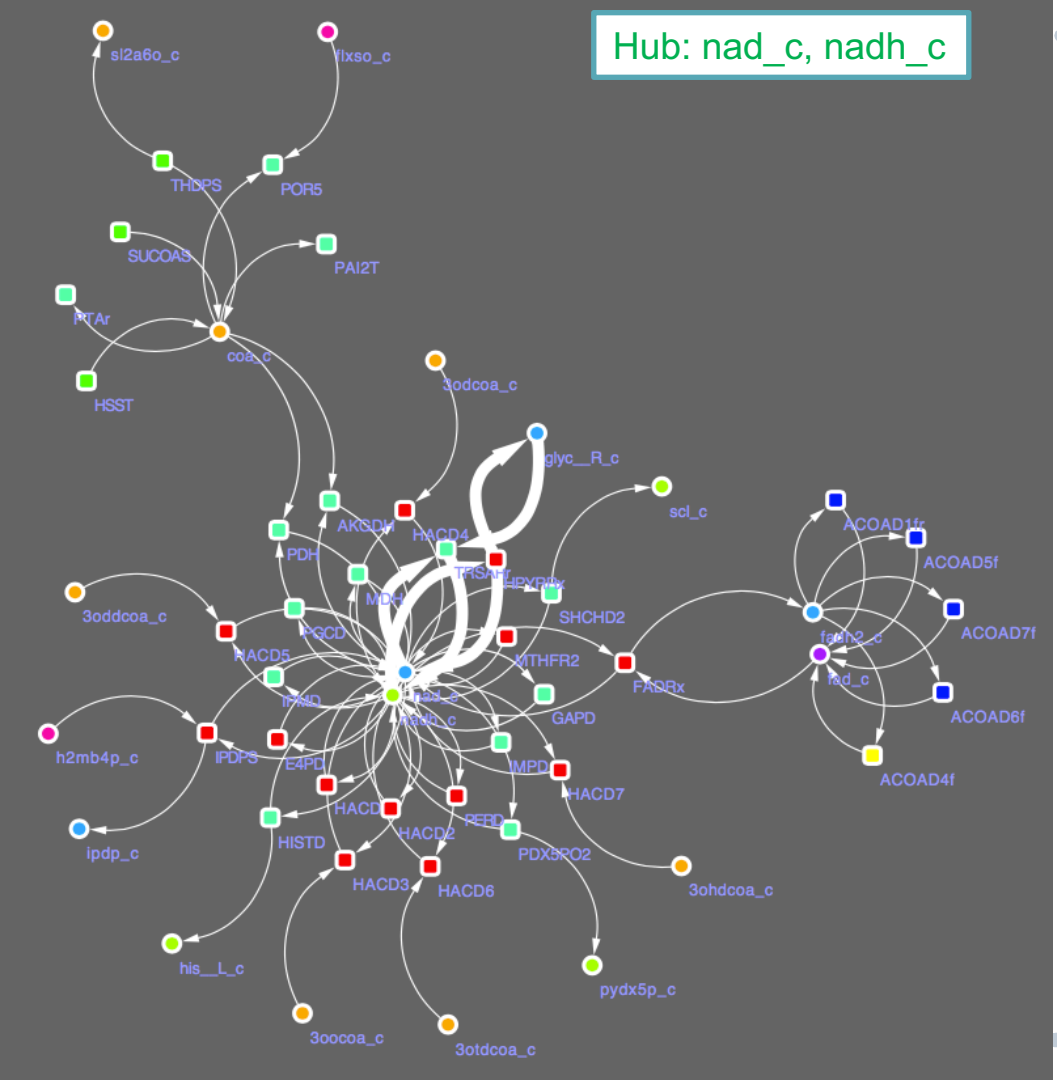
Super Module 1

209 -> 226 -> 225 -> 22 -> 49 -> 249 -> 82 -> 208

499

All percentage = # in this super module/total # of active (flux !=0)

80% of Oxidative Phosphorylation (4/5), 3 in module 82, 1in 49
50% of Anaplerotic Reactions (1/2)
40% of Inorganic Ion Transport and Metabolism (6/15)
33.3% of Unassigned (1/3)
33.3% of Transport, Inner Membrane (6/18)
25% of Alanine and Aspartate Metabolism (1/4)
20% of Glycerophospholipid Metabolism (2/10)
15.7% of Cofactor and Prosthetic Group Biosynthesis (13/83)
14.2% of Cysteine Metabolism (1/7)
10% of Methionine Metabolism (1/10)
8.3% of Threonine and Lysine Metabolism (1/12)
8.3% of Arginine and Proline Metabolism (1/12)
5.6% of Tyrosine, Tryptophan, and Phenylalanine Metabolism (1/18)
4.5% of Purine and Pyrimidine Biosynthesis (1/22)
4.3% of Cell Envelope Biosynthesis (1/23)



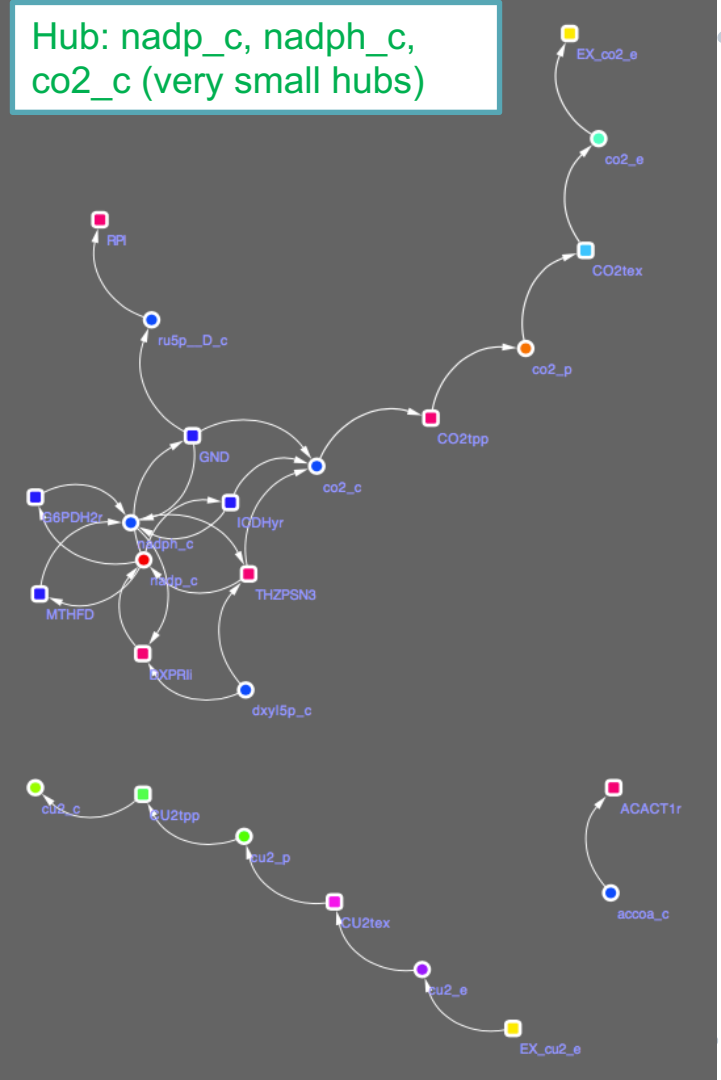
Super Module 2

313 -> 414

239->85 -> 278 -> 147 -> 526 -> 284

472

66.7% of Pyruvate Metabolism: 2/3, both in module 278
38.7% of Membrane Lipid Metabolism: 12/31
37.5% of Citric Acid Cycle: 3/8
25% of Alternate Carbon Metabolism: 2/8
25% of Folate Metabolism: 1/4
22.2% of Glycolysis/Gluconeogenesis: 2/9
20% of Methionine Metabolism: 2/10
16.7% of Glycine and Serine Metabolism: 1/6
10.0% of Histidine Metabolism: 1/10
8.3% of Threonine and Lysine Metabolism: 1/12
7.2% of Cofactor and Prosthetic Group Biosynthesis: 6/83
6.25% of Valine, Leucine, and Isoleucine Metabolism: 1/16
4.5% of Purine and Pyrimidine Biosynthesis: 1/22

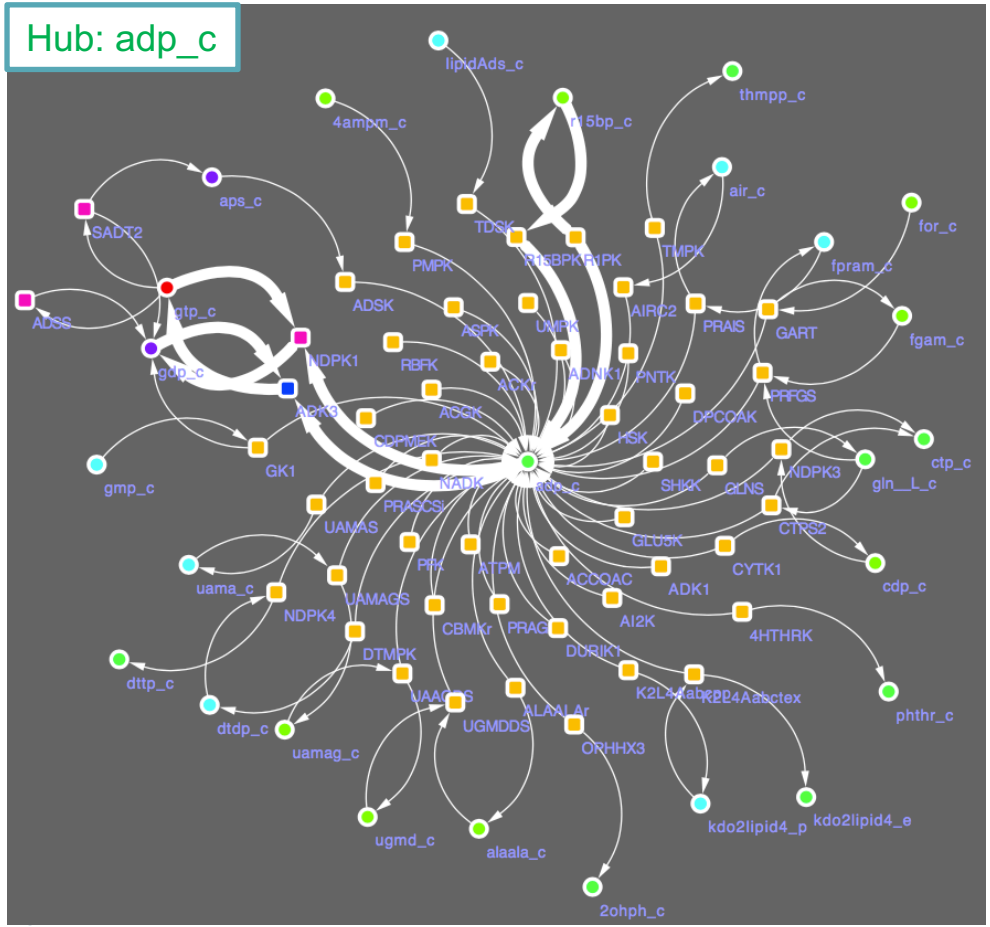


Super Module 3

566 -> 342 -> 102 -> 402 -> 8 -> 94 -> 93 -> 37 -> 355 -> 385 -> 59 -> 81 -> 40

50% of 'Intracellular demand': 1/2
37.5% of 'Pentose Phosphate Pathway': 3/8, 2 in 342, 1in 402
25% of 'Folate Metabolism': 1/4
12.5% of 'Citric Acid Cycle': 1/8
11.1% of 'Transport, Outer Membrane Porin': 2/18
11.1% of 'Extracellular exchange': 2/18
6.7% of 'Inorganic Ion Transport and Metabolism': 1/15
5.6% of 'Transport, Inner Membrane': 1/18
3.2% of 'Membrane Lipid Metabolism': 1/31
2.4% of 'Cofactor and Prosthetic Group Biosynthesis': 2/83

Hub: adp_c



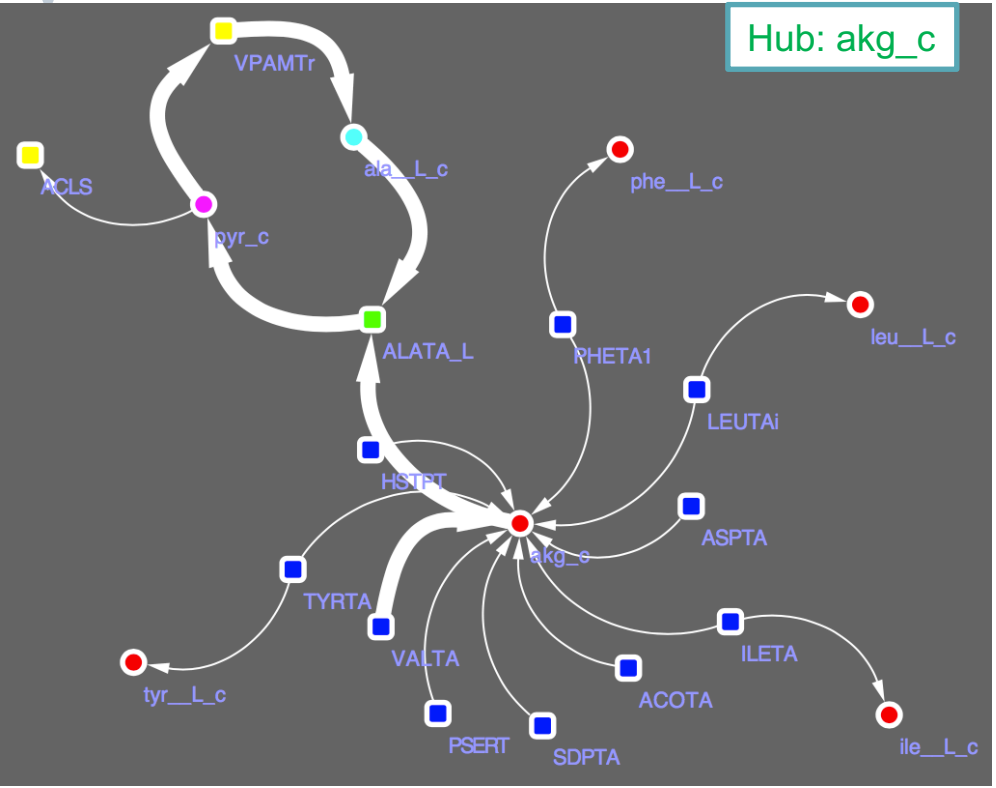
Super Module 4

343

134 -> 127 -> 174 -> 349 -> 544 -> 512 -> 384

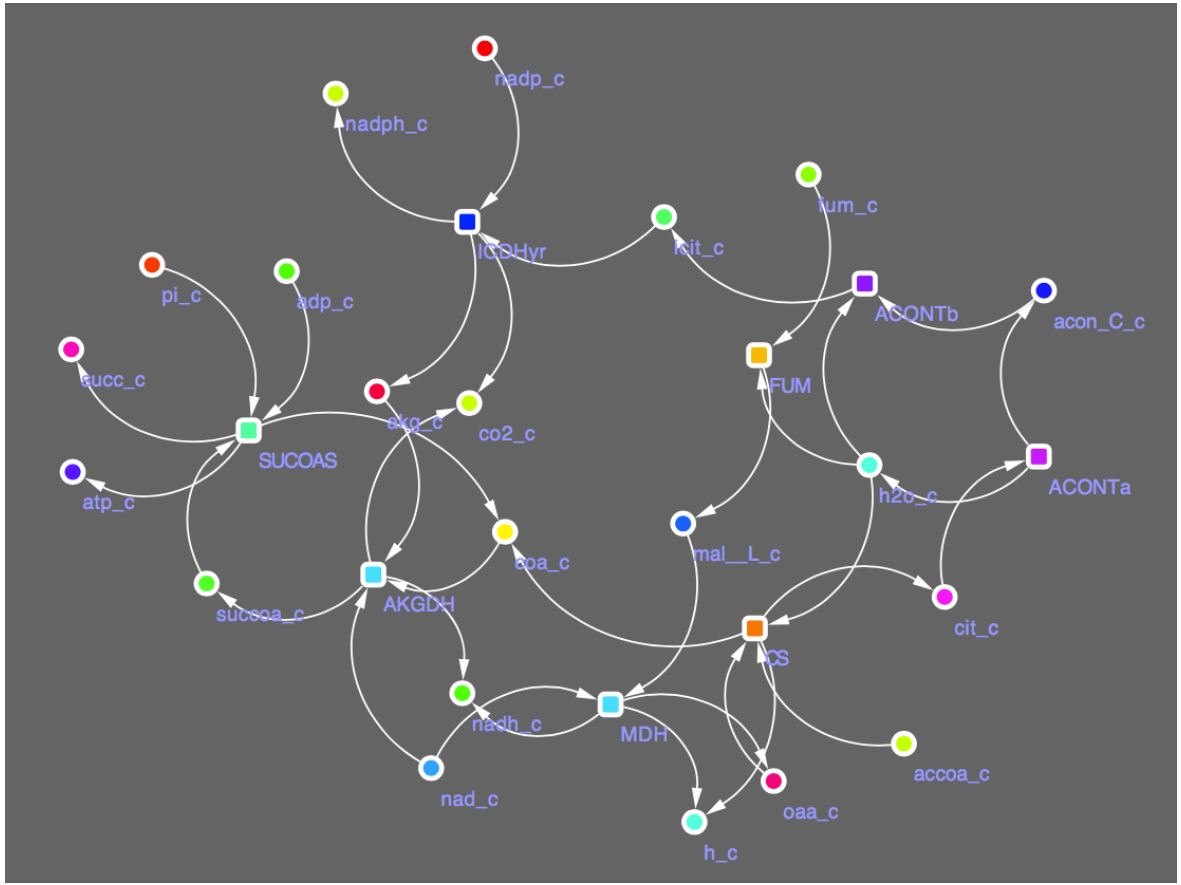
100% of 'Biomass and maintenance functions': 1/1
100% of 'Nitrogen Metabolism': 1/1
50% of 'Glutamate Metabolism': 1/2
42.3% of 'Nucleotide Salvage Pathway': 11/26, 9 in 127
36.4% of 'Purine and Pyrimidine Biosynthesis': 8/22
33.3% of 'Pyruvate Metabolism': 1/3
28.6% of 'Cysteine Metabolism': 2/7
25% of 'Alternate Carbon Metabolism': 2/8
21.7% of 'Cell Envelope Biosynthesis': 5/23
16.7% of 'Glycine and Serine Metabolism': 1/6
16.7% of 'Threonine and Lysine Metabolism': 2/12
16.7% of 'Arginine and Proline Metabolism': 2/12
16.7% of 'Lipopolysaccharide Biosynthesis / Recycling': 2/12
11.1% of 'Glycolysis/Gluconeogenesis': 1/9
10% of 'Methionine Metabolism': 1/10
9.6% of 'Cofactor and Prosthetic Group Biosynthesis': 8/83
5.6% of 'Transport, Inner Membrane': 1/18
5.6% of 'Tyrosine, Tryptophan, and Phenylalanine Metabolism': 1/18
3.2% of 'Membrane Lipid Metabolism': 1/31

Super Module 5



281 -> 511 -> 196 -> 426 -> 169 -> 273

50% of 'Alanine and Aspartate Metabolism': 2/4
31.3% of 'Valine, Leucine, and Isoleucine Metabolism': 5/16
16.7% of 'Glycine and Serine Metabolism': 1/6
11.1% of 'Tyrosine, Tryptophan, and Phenylalanine Metabolism': 2/18
11.1% of 'Tyrosine, Tryptophan, and Phenylalanine Metabolism': 2/18
8.3% of 'Arginine and Proline Metabolism': 1/12
8.3% of 'Threonine and Lysine Metabolism': 1/12



Citric Acid Cycle

Parameters: glucose anaerobic

Exchange	Lower bound
Mn2	-1000
Ni2	-1000
Mg2	-1000
NH4 (default N source)	-1000
Zn2	-1000
K	-1000
Cobalt2	-1000
Fe2	-1000
Fe3	-1000
Cu2	-1000
Ca2	-1000
Cl	-1000

Exchange	Lower bound
Modb (molybdate)	-1000
SO4 (default S source)	-1000
H	-1000
Pi (default P source)	-1000
H2O	-1000
CO2	-1000
Na	-1000
Sel (selenate)	-1000
Slnt (selenite)	-1000
Tungs (tungstate)	-1000
Glc__D (default C source)	-10
O2	0

Objective function: Biomass_core or Biomass_WT

```
model=readCbModel('iML1515.mat');
modelGluAnaerobic = model;
modelGluAnaerobic = changeRxnBounds(modelGluAnaerobic,'EX_o2_e',0,'l');
FBAGluAnaerobic = optimizeCbModel(modelGluAnaerobic,'max');

T = table(modelGluAnaerobic.rxns, modelGluAnaerobic.rxnNames, FBAGluAnaerobic.v);
writetable(T,'Flux_GluAnaerobic.xlsx','FileType','spreadsheet');
```

Save_flux.ipynb. → Dict_flux

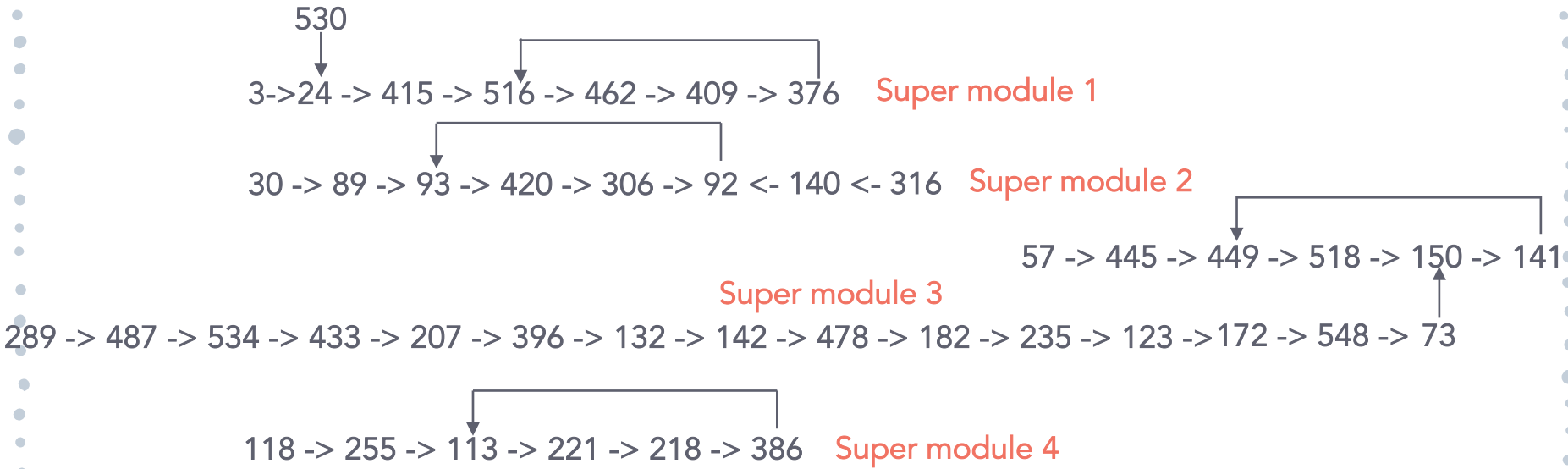
run_bidiLouvain.ipynb → commStruc_gluAnaerobic.pkl
cocluster_gluAnaerobic
nodetype_gluAnaerobic.xlsx

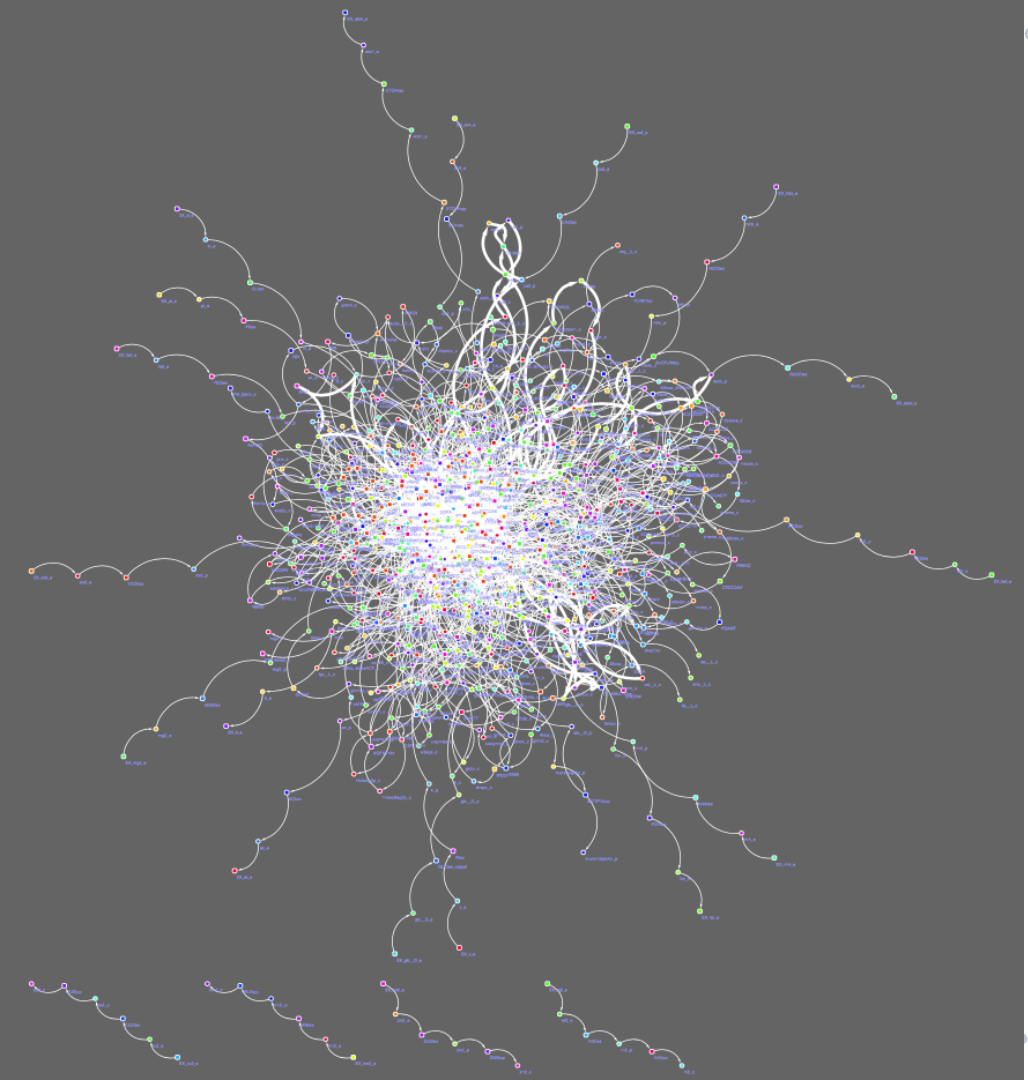
M = 0.720, # of communities: 557, growth rate = 0.1575 h-1

M is higher than aerobic case, but unclear if it is significant.

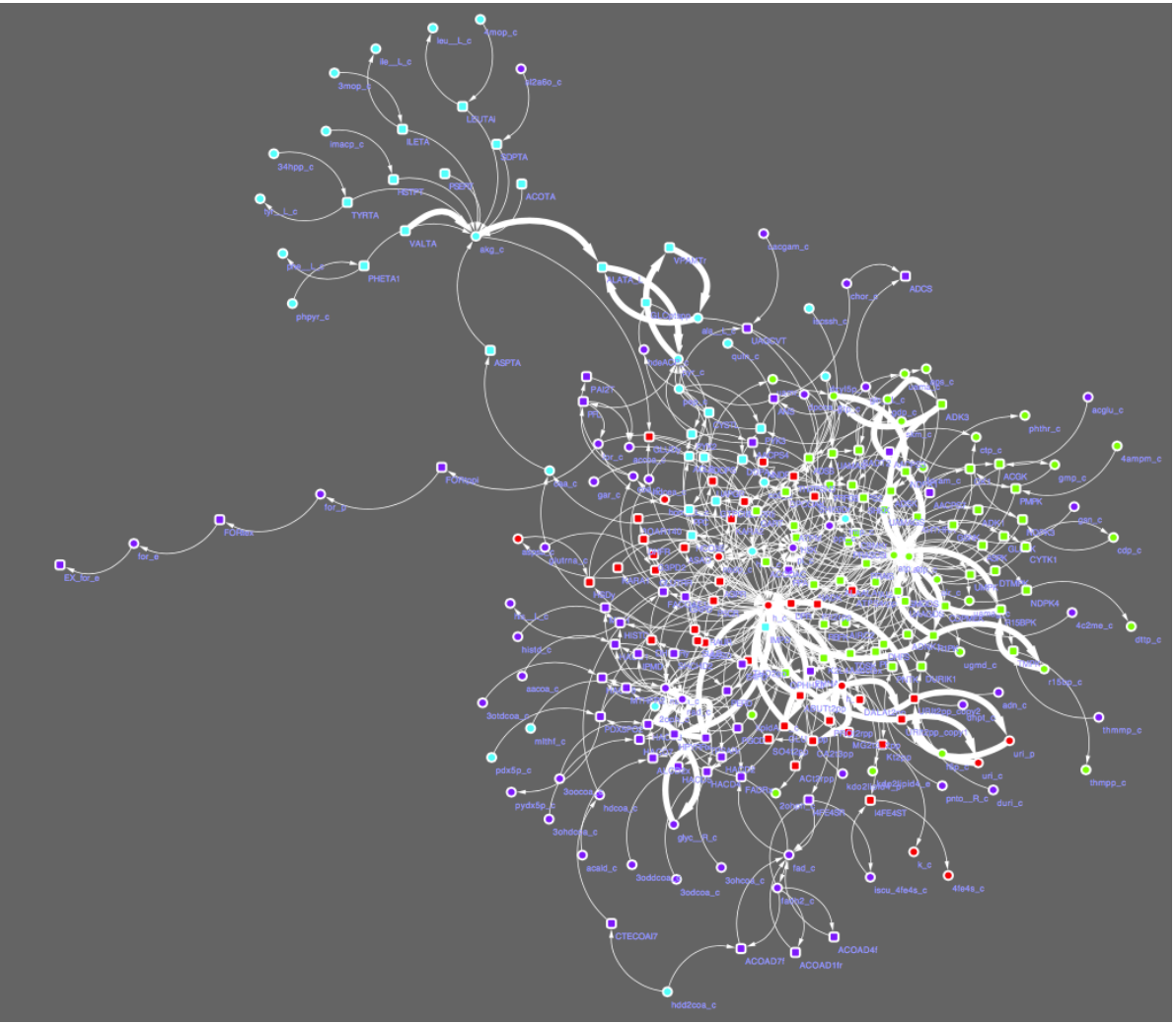
Community structure: super modules

Each number represents one community





Full Network

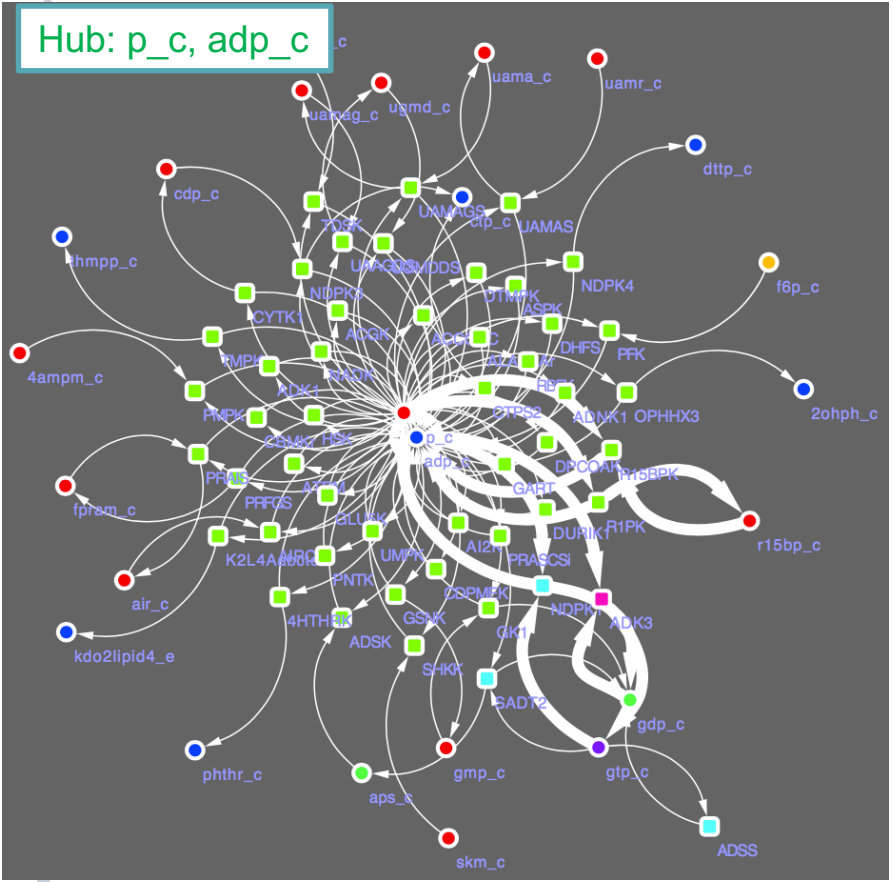


Big modules only

- Super modules 1-4 plotted (Cytoscape). Different colors represent different super modules.

Super module 1
Super module 2
Super module 3
Super module 4

- This is a subset of all nodes, and only edges between subset nodes are kept.



Super Module 1

530

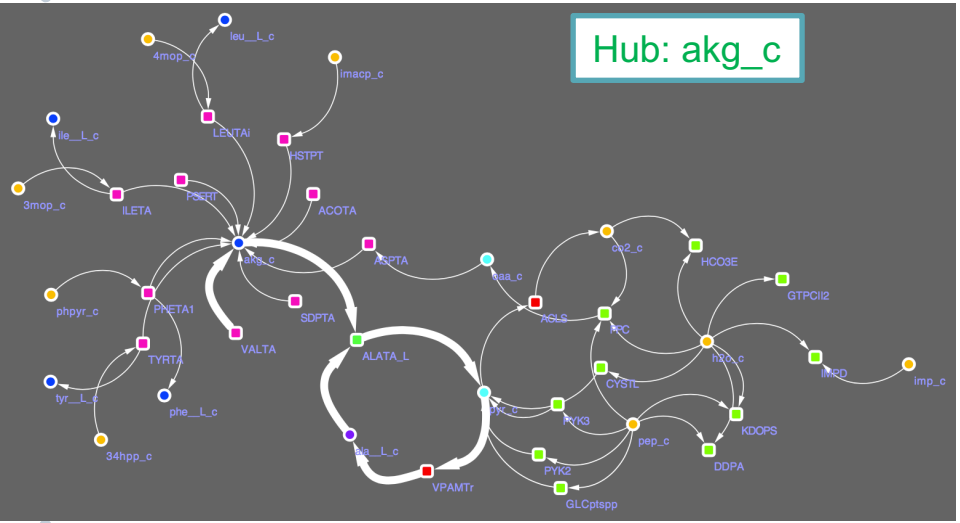
3->24 -> 415 -> 516 -> 462 -> 409 -> 376

All percentage = # in this super module/total # of active (flux !=0)

100% of 'Biomass and maintenance functions': 1/1
100% of 'Nitrogen Metabolism': 1/1
41.4% of 'Nucleotide Salvage Pathway': 12/29
31.8% of 'Purine and Pyrimidine Biosynthesis': 7/22
28.6% of 'Cysteine Metabolism': 2/7
25% of 'Alternate Carbon Metabolism': 2/8
21.7% of 'Cell Envelope Biosynthesis': 5/23
16.7% of 'Arginine and Proline Metabolism': 2/12
16.7% of 'Glycine and Serine Metabolism': 1/6
16.7% of 'Lipopolysaccharide Biosynthesis / Recycling': 2/12
16.7% of 'Threonine and Lysine Metabolism': 2/12
12.5% of 'Glycolysis/Gluconeogenesis': 1/8
11.4% of 'Cofactor and Prosthetic Group Biosynthesis': 9/79
10% of 'Methionine Metabolism': 1/10
5.6% of 'Tyrosine, Tryptophan, and Phenylalanine Metabolism': 1/18
3.2% of 'Membrane Lipid Metabolism': 1/31

Super Module 2

30 -> 89 -> 93 -> 420 -> 306 -> 92 <- 140 <- 316

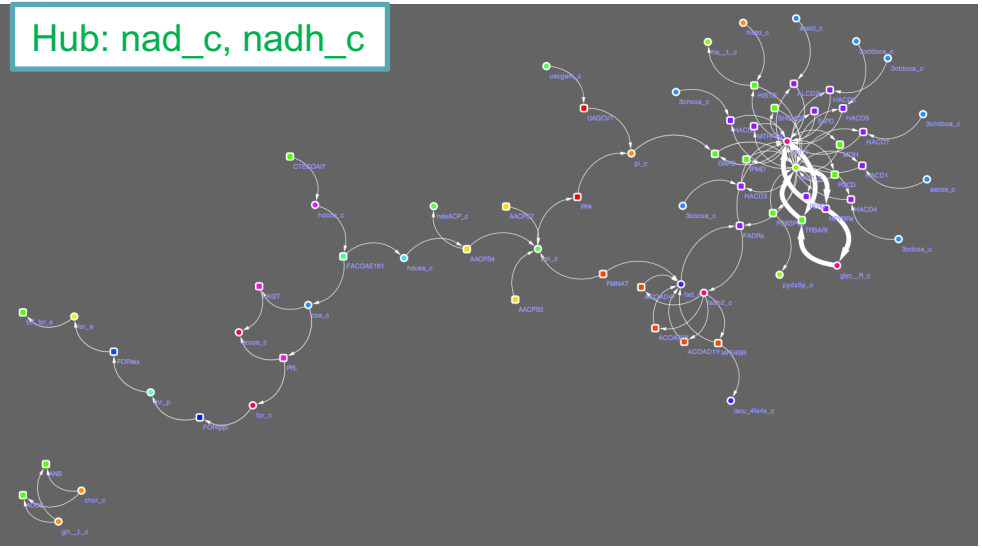


50% of 'Alanine and Aspartate Metabolism': 2/4
50% of 'Anaplerotic Reactions': ½
50% of 'Unassigned': ½
31.3% of'Valine, Leucine, and Isoleucine Metabolism': 5/16
16.7% of 'Glycine and Serine Metabolism': 1/6
16.7% of 'Tyrosine, Tryptophan, and Phenylalanine Metabolism': 3/18
10% of 'Histidine Metabolism': 1/10
10% of 'Methionine Metabolism': 1/10
8.3% of 'Arginine and Proline Metabolism': 1/12
8.3% of 'Lipopolysaccharide Biosynthesis / Recycling': 1/12
8.3% of 'Threonine and Lysine Metabolism': 1/12
6.9% of 'Nucleotide Salvage Pathway': 2/29
4.5% of 'Purine and Pyrimidine Biosynthesis': 1/22
4.5% of 'Transport, Inner Membrane': 1/22
1.3% of 'Cofactor and Prosthetic Group Biosynthesis': 1/79

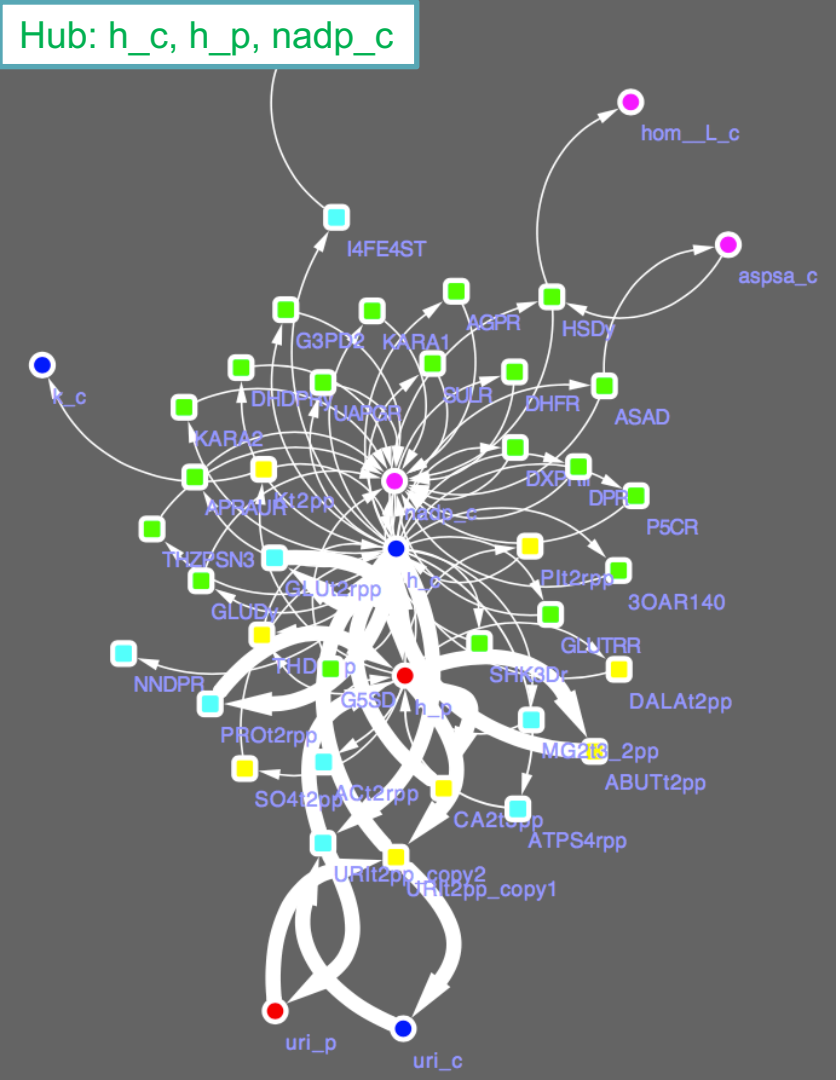
Super Module 3

57 -> 445 -> 449 -> 518 -> 150 -> 141

289 -> 487 -> 534 -> 433 -> 207 -> 396 -> 132 -> 142 -> 478 -> 182 -> 235 -> 123 ->172 -> 548 -> 73



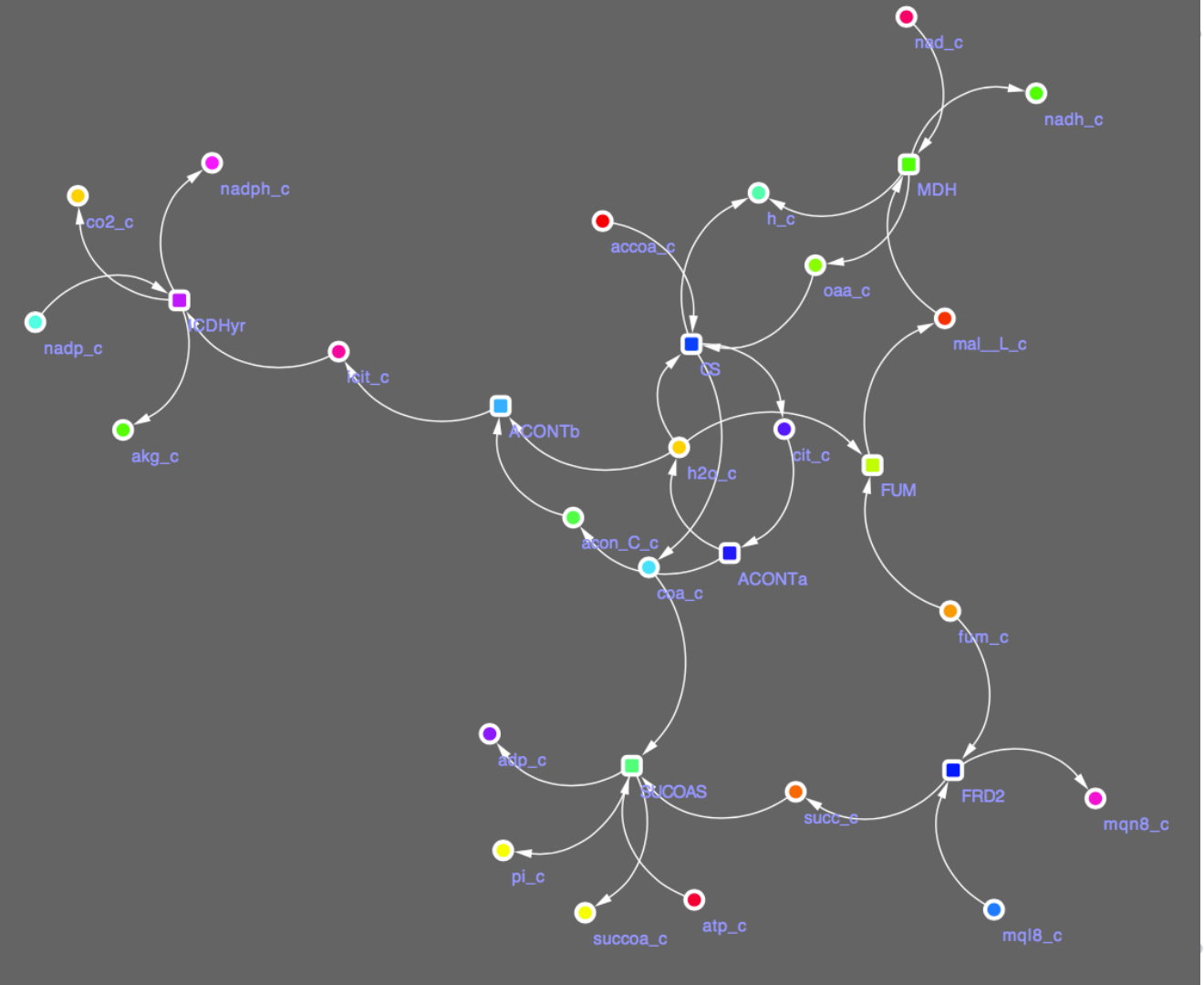
50% of 'Anaplerotic Reactions': ½
35.5% of 'Membrane Lipid Metabolism': 11/31
33.3% of 'Folate Metabolism': 1/3
33.3% of 'Pyruvate Metabolism': 2/6
25% of 'Alternate Carbon Metabolism': 2/8
21.7% of 'Cell Envelope Biosynthesis': 5/23
16.7% of 'Glycine and Serine Metabolism': 1/6
12.5% of 'Citric Acid Cycle': 1/8
12.5% of 'Glycolysis/Gluconeogenesis': 1/8
10.1% of 'Cofactor and Prosthetic Group Biosynthesis': 8/79
10% of 'Histidine Metabolism': 1/10
10% of 'Methionine Metabolism': 1/10
6.25% of 'Valine, Leucine, and Isoleucine Metabolism': 1/16
5.6% of 'Tyrosine, Tryptophan, and Phenylalanine Metabolism': 1/18
4.8% of 'Extracellular exchange': 1/21
4.8% of 'Transport, Outer Membrane Porin': 1/21
4.5% of 'Transport, Inner Membrane': 1/22



Super Module 4

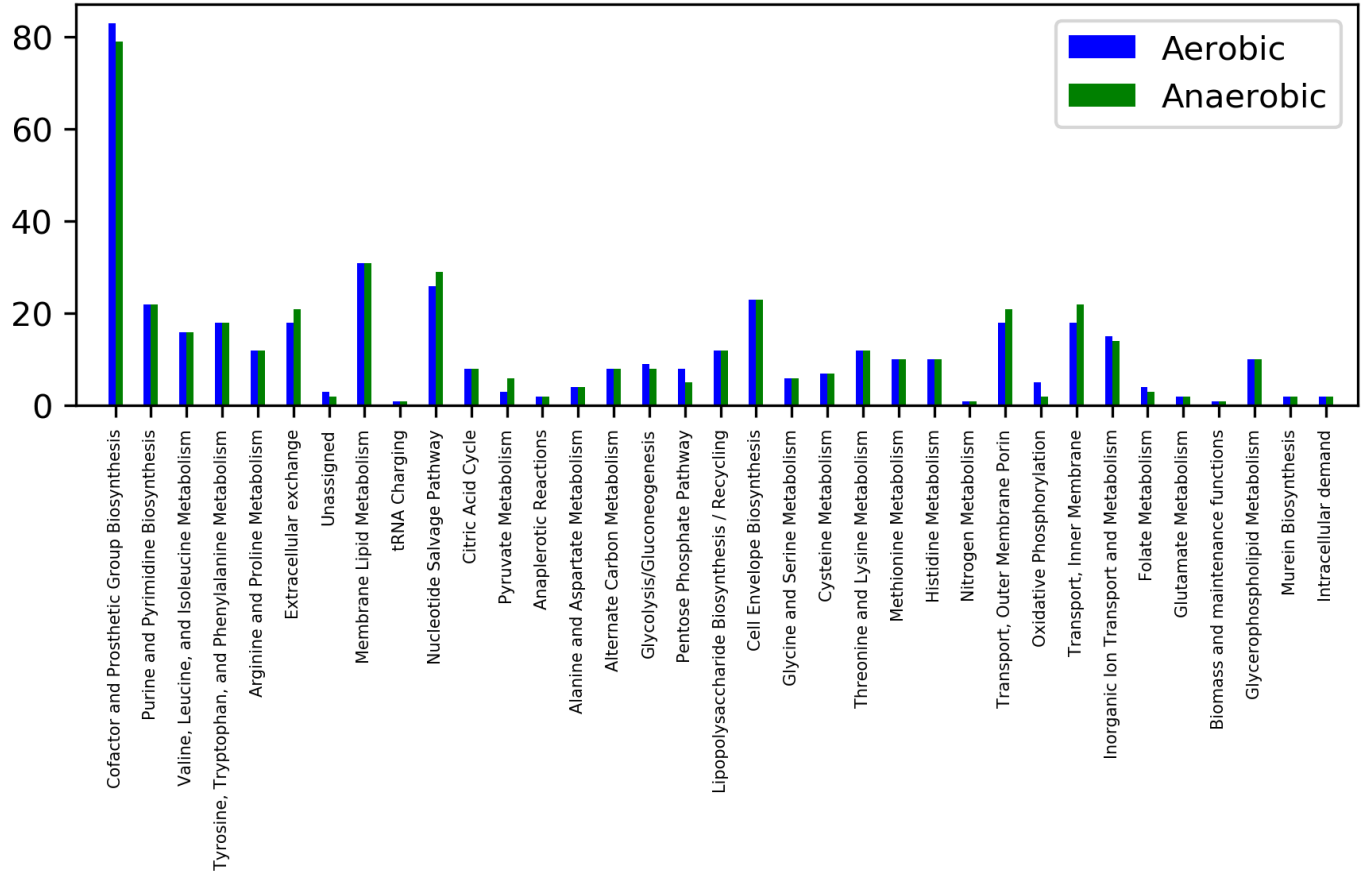
118 -> 255 -> 113 -> 221 -> 218 -> 386

100% of 'Oxidative Phosphorylation': 2/2
50% of 'Glutamate Metabolism': 1/2
35.7% of 'Inorganic Ion Transport and Metabolism': 5/14
31.8% of 'Transport, Inner Membrane': 7/22
25% of 'Threonine and Lysine Metabolism': 3/12
25% of 'Arginine and Proline Metabolism': 3/12
14.3% of 'Cysteine Metabolism': 1/7
12.5% of 'Alternate Carbon Metabolism': 1/8
12.5% of 'Valine, Leucine, and Isoleucine Metabolism': 2/16
10.1% of 'Cofactor and Prosthetic Group Biosynthesis': 8/79
8.7% of 'Cell Envelope Biosynthesis': 2/23
5.6% of 'Tyrosine, Tryptophan, and Phenylalanine Metabolism': 1/18



Citric Acid Cycle

Compare aerobic and anaerobic active pathway size



Even though the number of active reactions in the Citric Acid Cycle is the same. Anaerobic condition's flux in the cycle is an order of magnitude smaller than that in the aerobic case.