

Table of Contents

- [1 Goals for Fluxomics Tutorial](#)
- [2 Simple metabolic model](#)
 - [2.1 Simple Metabolic Reaction Network](#)
 - [2.1.1 Elementary modes of Simple Metabolic Reaction Network](#)
 - [2.2 Definitions](#)
 - [2.3 Solving for the MIDs of the EMU network](#)
 - [2.3.1 EMU network size 1](#)
 - [2.3.1.1 Isotopic labeling state](#)
 - [2.3.1.2 EMU size 1 balance equations](#)
 - [2.3.1.3 EMU size 1 matrix equation](#)
 - [2.3.1.4 EMU size 1 solution](#)
 - [2.3.2 EMU Network size 2](#)
 - [2.3.2.1 Isotopic labeling state](#)
 - [2.3.2.2 EMU size 2 balance equations](#)
 - [2.3.2.3 EMU size 2 matrix equation](#)
 - [2.3.2.4 EMU size 2 solution](#)
 - [2.3.3 EMU Network size 3](#)
 - [2.3.3.1 Isotopic labeling state](#)
 - [2.3.3.2 EMU size 3 balance equations](#)
 - [2.3.3.3 EMU size 3 matrix equation](#)
 - [2.3.3.4 EMU size 3 solution](#)
 - [2.4 Studying the effect of good and poor tracers](#)
 - [2.4.1 Combined EMU Network](#)
 - [2.4.2 EMU basis vectors](#)
 - [2.4.3 EMU pathways](#)
 - [2.4.4 Tracer Compositions for Simple EMU network](#)
 - [2.4.4.1 MID plots of EMU basis vectors for each tracer](#)
 - [2.4.4.2 MID plots of measured EMUs](#)
- [3 Extended metabolic model](#)
 - [3.1 Reaction network for EMU size 1](#)
 - [3.1.1 EMU size 1 balance equations](#)
 - [3.1.2 EMU size 1 matrix equation](#)
 - [3.1.3 EMU size 1 solution](#)
 - [3.2 Reaction network for EMU size 2](#)
 - [3.2.1 EMU size 2 balance equations](#)
 - [3.2.2 EMU size 2 matrix equation](#)
 - [3.2.3 Expansion of \$D_{2,3}\$ and \$B_{2,3}\$](#)
 - [3.2.4 Expansion of \$B_3 \times C_1\$ and \$A_{2,3}\$](#)
 - [3.2.5 EMU size 2 solution](#)
 - [3.3 Reaction network for EMU size 3](#)
 - [3.3.1 EMU size 3 balance equations](#)
 - [3.3.2 EMU size 3 Matrix equation](#)
 - [3.3.3 EMU size 3 solution](#)
 - [3.4 Combined EMU transition network of extended metabolic reaction network](#)
 - [3.5 Tracer Compositions for Extended EMU network](#)

4 TCA Cycle metabolic model

- [4.1 TCA Cycle metabolic reaction network](#)
- [4.2 Combined EMU network for TCA Cycle](#)
- [4.3 Tracer Compositions for TCA EMU Network](#)

In [47]:

```

1 %load_ext nb_black
2
3
4 from IPython.display import display, Image, Markdown, Latex, HTML
5 import pandas as pd
6 import numpy as np
7 from cobra import Metabolite, Reaction, Model
8 from cobra.util import create_stoichiometric_matrix
9 import cobra
10 cobra_config = cobra.Configuration()
11 cobra_config.solver='glpk_exact'
12 import sys, re,os
13 #sys.path.append(os.path.join(os.environ['HOME'], 'Projects/src'))
14 from pyefm import calculate_elementary_modes, calculate_minimum_cut_set
15 from d3flux import flux_map
16 from IPython.display import display, Image, SVG, Markdown, Latex, HTML
17 from cobra.flux_analysis.geometric import geometric_fba
18 from pyemu import *
19 from plotnine import (
20     position_dodge,
21     ggplot,
22     geom_bar,
23     aes,
24     ggtile,
25     theme,
26     element_rect,
27     element_text,
28     geom_col,
29     facet_wrap,
30     save_as_pdf_pages,
31 )
32
33 def plot_mid(mid):
34     mid_gg = mid.stack().reset_index().rename(columns={0: "MID"})
35     plot = (
36         ggplot(
37             mid_gg,
38             aes(x="Isotopologue", y="MID"),
39         )
40         + ggtile("MID")
41         + theme(
42             panel_background=element_rect(fill="white"),
43             axis_text_x=element_text(angle=90),
44         )
45         + geom_col()
46         + facet_wrap("EMU")
47     )
48     return plot
49
50

```

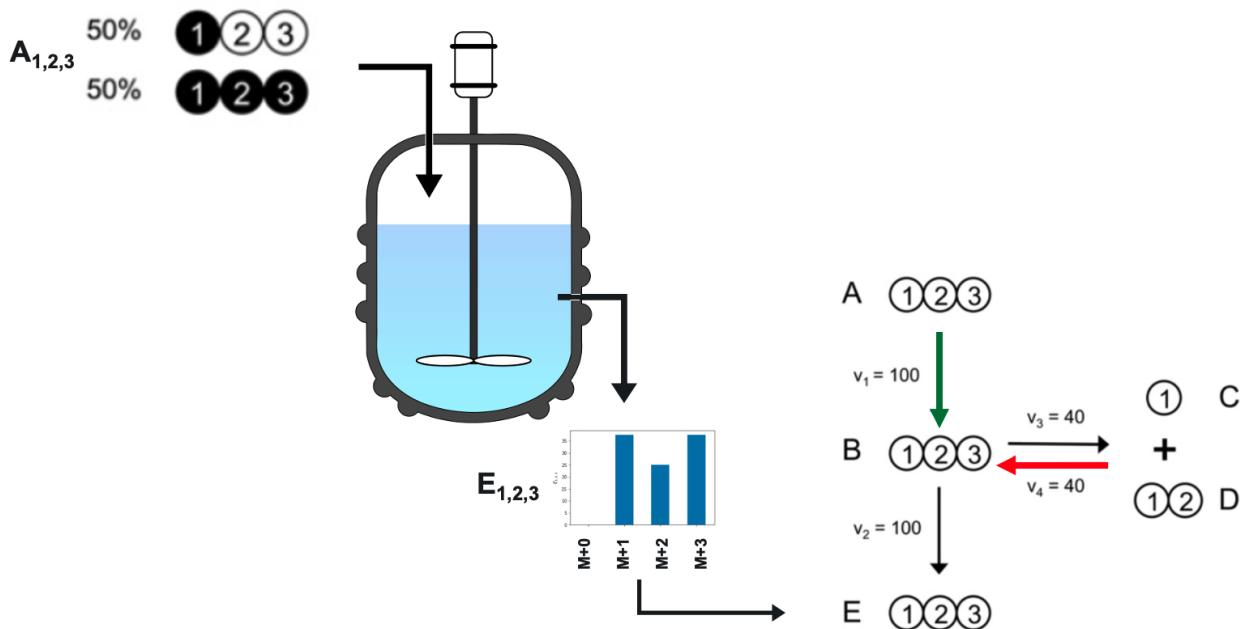
The `nb_black` extension is already loaded. To reload it, use:

```
%reload_ext nb_black
```

1 Goals for Fluxomics Tutorial

In a ^{13}C fluxomics experiment, an organism is fed a substrate with a mixture of isotopic labels where the location of each labeled atom is known. To fully specify the **isotopic labeling state** of the mixture, both the percentage of each component of the mixture and which atoms are labeled in each component must be described.

For today's fluxomics tutorial, we will assume all samples are taken when the organism is in both **metabolic steady state**, where the concentration of each metabolite does not change, and **isotopic stationary state**, where the isotopic labeling state of each metabolite does not change.



If these assumptions are met, mass spectrometry still cannot determine the isotopic labeling state of each metabolite it measures. The best it can do is measure the **Mass Isotopomer Distribution (MID)**, which represents a substrate with a mixture of labels where the location of each labeled atom is not known. For a molecule with n atoms, M_0, M_1, \dots, M_n specifies the percentage of atoms that are unlabeled, 1-labeled, up to fully labeled.

Metabolic flux analysis (MFA) takes as input:

1. the isotopic labeling state of the substrate,
2. the MID of a set of measured metabolites,
3. a metabolic model of the organism,
4. knowledge of atom transitions
5. and a known uptake rate

and uses this information to calculate the material and energy flows through the organism.

The goal of today's tutorial is to understand how this is done. We will start with a simple model, introduce some new concepts that build upon what we learned this morning about flux balance analysis and elementary modes. We will then use a simple model to understand:

1. Step-by-step how steady-state fluxes can be used to calculate mass isotopomer distributions (MIDs).
2. How measurements of stationary-state MIDs can be used to solve for steady-state fluxes.

3. Why the choice of tracers is critical for identifying flux distributions
4. and how to distinguish good tracers from bad ones.

2 Simple metabolic model

In this system, metabolite A is the substrate, metabolite E is the product, and the intermediate metabolites B , C and D are assumed to be at metabolic and isotopic steady state. This network has two free fluxes, namely the substrate consumption rate v_1 and the flux of the reversible cleavage/condensation reaction v_4 . We would like to determine flux v_4 based solely on the measurement of the Mass Isotope Distribution (MID) of E and a known uptake rate of A .

2.1 Simple Metabolic Reaction Network

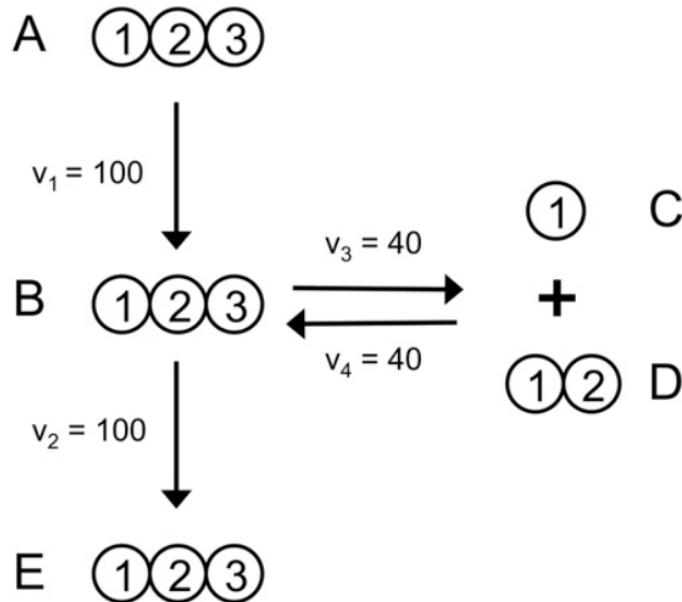


Fig. 1. Simple metabolic network model used to demonstrate the need for a more rational approach to tracer design. The assumed fluxes have arbitrary units.

Table 1
Stoichiometry and atom transitions for the reactions in the example network model.

Reaction number	Reaction stoichiometry	Atom transitions
1	$A \rightarrow B$	$abc \rightarrow abc$
2	$B \rightarrow E$	$abc \rightarrow abc$
3 and 4	$B \leftrightarrow C + D$	$abc \leftrightarrow c + ab$

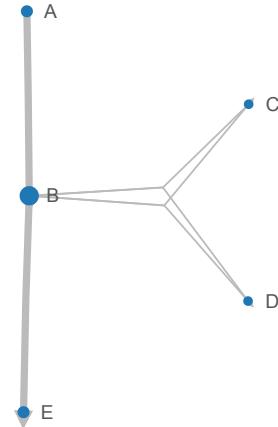
In [95]:

```

1 from IPython.display import Image, display, Markdown
2
3 simple_metabolic_network = Model("figure1_simple_metabolic_network")
4 v1, v2, v3, v4 = 100, 100, 40, 40
5 A = Metabolite("A")
6 B = Metabolite("B")
7 C = Metabolite("C")
8 D = Metabolite("D")
9 E = Metabolite("E")
10 R1 = Reaction("R1")
11 R2 = Reaction("R2")
12 R3 = Reaction("R3")
13 R4 = Reaction("R4")
14 B1 = Reaction("B1")
15 B2 = Reaction("B2")
16 simple_metabolic_network.add_metabolites([A, B, C, D, E])
17 simple_metabolic_network.add_reactions([R1, R2, R3, R4, B1, B2])
18 simple_metabolic_network.reactions.R1.build_reaction_from_string("A --> B")
19 simple_metabolic_network.reactions.R2.build_reaction_from_string("B --> C")
20 simple_metabolic_network.reactions.R3.build_reaction_from_string("B --> D")
21 simple_metabolic_network.reactions.R4.build_reaction_from_string("C + D --> E")
22 simple_metabolic_network.reactions.B1.build_reaction_from_string("--> B")
23 simple_metabolic_network.reactions.B2.build_reaction_from_string("E --> B")
24
25 simple_metabolic_network.reactions.R1.notes["CARBON TRANSITIONS"] = "A"
26 simple_metabolic_network.reactions.R2.notes["CARBON TRANSITIONS"] = "B"
27 simple_metabolic_network.reactions.R3.notes[
28     "CARBON TRANSITIONS"
29 ] = "B -- C + D\ta bc : c + ab"
30 simple_metabolic_network.reactions.R4.notes[
31     "CARBON TRANSITIONS"
32 ] = "C + D -- B\tc + ab : abc"
33
34 S = get_stoichiometric_matrix(simple_metabolic_network)
35 rxns, mets = S.columns, S.index
36 display(S.astype(int))
37 display(
38     flux_map(
39         cobra.io.load_json_model("figure1_simple_metabolic_network.json",
40             # display_name_format=lambda x: str(x.id),
41             flux_dict=dict(R1=v1, R2=v2, R3=v3, R4=v4, B1=100, B2=100),
42             default_flux_width=1,
43         )
44 )

```

	R1	R2	R3	R4	B1	B2
A	-1	0	0	0	1	0
B	1	-1	-1	1	0	0
C	0	0	1	-1	0	0
D	0	0	1	-1	0	0
E	0	1	0	0	0	-1

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)[simple network with fluxes \(simple-network-with-fluxes.png\)](#)

2.1.1 Elementary modes of Simple Metabolic Reaction Network

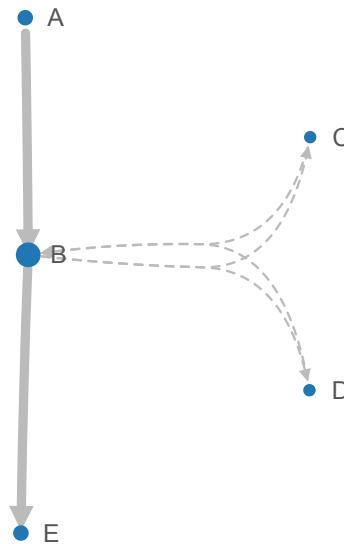
In [97]:

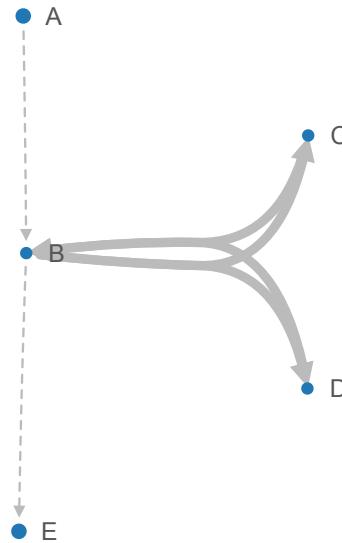
```
1 elmo = calculate_elementary_modes(simple_metabolic_network, verbose=False)
2 display(elmo)
```

/Users/zuck016/.pyenv/versions/anaconda3-2020.11/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering (buffering=1) isn't supported in binary mode, the default buffer size will be used

	R1	R2	R3	R4	B1	B2
EM1	1.0	1.0	0.0	0.0	1.0	1.0
EM2	0.0	0.0	1.0	1.0	0.0	0.0

```
In [98]: 1 display(
  2     flux_map(
  3         cobra.io.load_json_model("figure1_simple_metabolic_network.json"),
  4         display_name_format=lambda x: str(x.id),
  5         flux_dict={
  6             rxn.id: elmo.loc["EM1", rxn.id]
  7             for rxn in simple_metabolic_network.reactions
  8         },
  9         figsize=(800, 600),
10     )
11 )
12 display(
13     flux_map(
14         cobra.io.load_json_model("figure1_simple_metabolic_network.json"),
15         display_name_format=lambda x: str(x.id),
16         flux_dict={
17             rxn.id: elmo.loc["EM2", rxn.id]
18             for rxn in simple_metabolic_network.reactions
19         },
20         figsize=(800, 600),
21     )
22 )
```

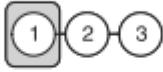
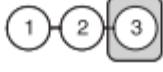
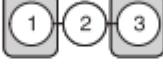
[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

[simple network elmo 1 \(simple-network-elmo-1.png\)](#) [simple network elmo 2 \(simple-network-elmo-2.png\)](#)

2.2 Definitions

- **Elementary Metabolite Units (EMU)** are distinct subsets of the compound's atoms. There are 7 EMU's for a 3-atom metabolite A shown in the figure from [Antoniewicz 2007 (<http://linkinghub.elsevier.com/retrieve/pii/S109671760600084X>) below. The subscript in the first column and the shaded areas in the second column denote the atoms that are included in the EMU. The EMU size is the number of atoms included in the EMU.

Elementary metabolite unit	Atoms included in the EMU	EMU size
A_1		1
A_2		1
A_3		1
A_{12}		2
A_{13}		2
A_{23}		2
A_{123}		3

First, we need to introduce the concept of an **EMU reaction**. The figure below shows three types of **EMU reactions** that we distinguish:

1. a condensation reaction,
2. a cleavage reaction,
3. and a unimolecular reaction.

For each reaction type, we would like to determine the minimum amount of information that is needed to determine the mass isotopomer distribution (MID) of product C , i.e., EMU $C_{1,2,3}$. For the cleavage and the unimolecular reactions, the MID of $C_{1,2,3}$ is equal to the MID of $A_{1,2,3}$. Note that for the cleavage reaction, atoms of A that are not transferred to $C_{1,2,3}$ are not considered in the EMU reaction, i.e., their labeling does not affect the labeling state of C .

For the condensation reaction, the MID of $C_{1,2,3}$ is obtained from the convolution (denoted by \times) of MIDs of EMUs $A_{1,2}$ and B_1 .

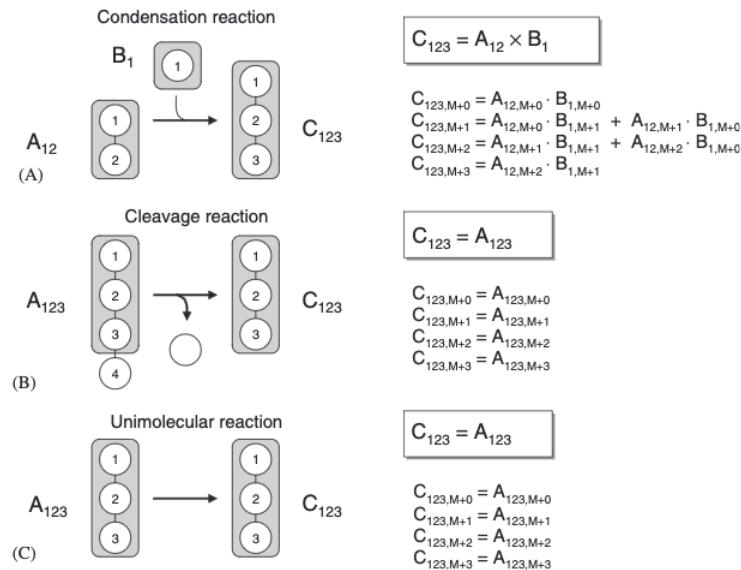


Fig. 2. Three types of biochemical reactions with the corresponding EMU reactions. Shaded areas indicate atoms included in the EMUs. The MID of product C is fully determined by MIDs of substrate EMUs. For the condensation reaction, MID of C_{123} is obtained from the convolution (or Cauchy product, denoted by ' \times ') of MIDs of A_{12} and B_1 . For the cleavage reaction and unimolecular reaction MID of C_{123} equals MID of A_{123} .

- An **EMU network** is a set of EMU's and the transitions between them. The network can be restricted to EMUs of a particular size (Figure 3A), or it can be combined into a larger network composed of EMU's of all sizes (Figure 3B). Note that not all possible EMUs participate in a transition. This insight allows us to significantly reduce the size of the problem.

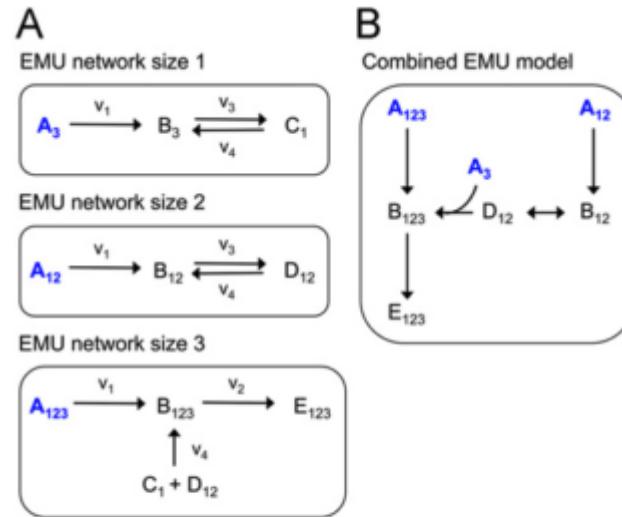


Fig. 3. (A) Decoupled EMU reaction networks for the example network model. (B) Combined EMU reaction network showing the flow of isotopic labeling from the substrate A to the network product E . The two EMU basis vectors in the model were A_{123} and $A_{12} \times A_3$.

Now that we understand EMUs, EMU reactions and EMU networks, our first goal is to calculate the Mass Isotopic Distribution (MID) from the steady state fluxes given the isotopic labeling state of the substrate.

2.3 Solving for the MIDs of the EMU network

2.3.1 EMU network size 1

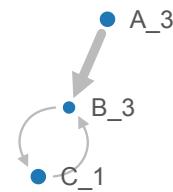
We first solve for the MIDs of all EMUs of size 1 from the steady-state fluxes given the isotopic labeling state of the substrate.

In [80]:

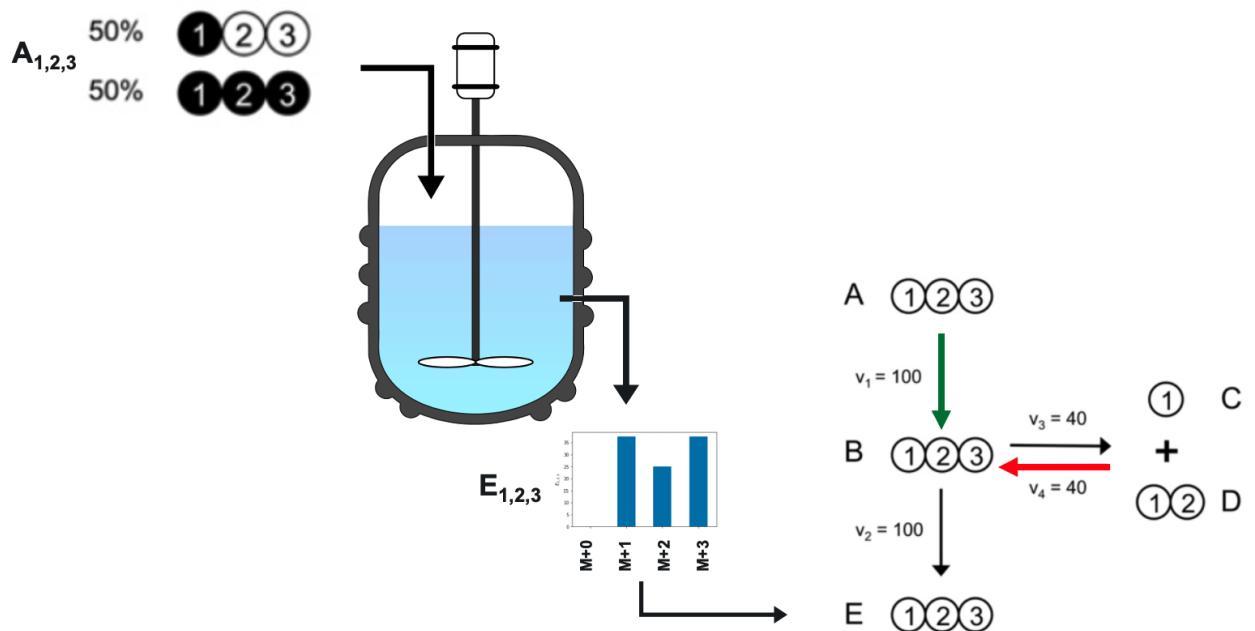
```

1 emu_network_size_1 = Model("figure3a_EMU_network_size_1")
2
3 A_3 = Metabolite("A_3")
4 B_3 = Metabolite("B_3")
5 C_1 = Metabolite("C_1")
6
7 R1 = Reaction("R1")
8 R3 = Reaction("R3")
9 R4 = Reaction("R4")
10 B1 = Reaction("B1")
11 B2 = Reaction("B2")
12
13 emu_network_size_1.add_metabolites([A_3, B_3, C_1])
14 emu_network_size_1.add_reactions([R1, R3, R4, B1, B2])
15
16 emu_network_size_1.reactions.R1.build_reaction_from_string("A_3 --> B_3")
17 emu_network_size_1.reactions.R3.build_reaction_from_string("B_3 --> C_1")
18 emu_network_size_1.reactions.R4.build_reaction_from_string("C_1 <--> E")
19 emu_network_size_1.reactions.B1.build_reaction_from_string("--> A_3")
20 emu_network_size_1.reactions.B2.build_reaction_from_string("C_1 --> E")
21
22 # cobra.io.save_json_model(emu_network, "figure3_combined_emu_network.json")
23 EMU1 = get_stoichiometric_matrix(emu_network_size_1).astype(int)
24 display(
25     flux_map(
26         # cobra.io.load_json_model("figure1_simple_metabolic_network.json")
27         emu_network_size_1,
28         display_name_format=lambda x: str(x.id),
29         figsize=(800, 600),
30         flux_dict=dict(R1=v1, R3=v3, R4=v4),
31     )
32 )

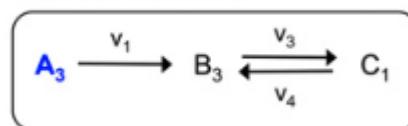
```



[Save JSON](#) [Show Reaction Nodes](#) [Download SVG](#)



EMU network size 1



2.3.1.1 Isotopic labeling state

$$A_3 = \begin{bmatrix} M+0 & M+1 \\ 0.5 & 0.5 \end{bmatrix} \quad (1)$$

2.3.1.2 EMU size 1 balance equations

$$(v_1 + v_4)B_3 = v_1 A_3 + v_4 C_1 \quad (2)$$

$$v_3 D_3 = v_3 B_3 \quad (3)$$

2.3.1.3 EMU size 1 matrix equation

$$\begin{bmatrix} v_1 + v_4 & -v_4 \\ v_3 & -v_3 \end{bmatrix} \begin{bmatrix} B_3 \\ C_1 \end{bmatrix} = \begin{bmatrix} v_1 \\ 0 \end{bmatrix} \begin{bmatrix} A_3 \end{bmatrix} \quad (4)$$

2.3.1.4 EMU size 1 solution

$$\begin{bmatrix} B_3 \\ C_1 \end{bmatrix} = \begin{bmatrix} 140 & -40 \\ 40 & -40 \end{bmatrix}^{-1} \begin{bmatrix} 100 \\ 0 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} = [] \quad (5)$$

In [49]:

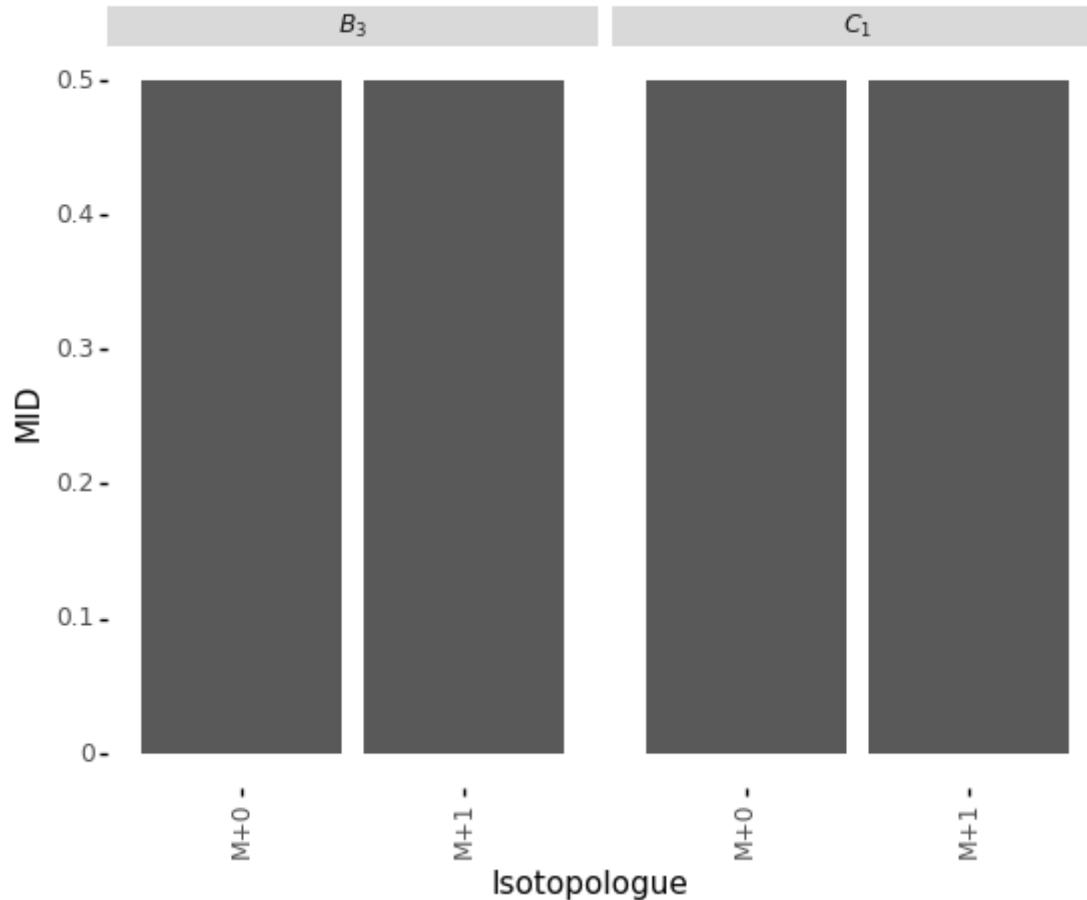
```

1 import numpy as np
2 import pandas as pd
3
4 v1, v2, v3, v4 = 100, 100, 40, 40
5
6 A_3 = pd.DataFrame([[0.5, 0.5]], columns=["M+0", "M+1"], index=["$A_3"])
7 A = np.array([[v1 + v4, -v4], [v3, -v3]])
8 B = np.array([[v1], [0]])
9 Y = A_3.values
10
11 X = pd.DataFrame(
12     np.linalg.pinv(A).dot(B).dot(Y), columns=["M+0", "M+1"], index=[r'$X$'])
13 ).rename_axis(index="EMU", columns="Isotopologue")
14 display(X)
15 display(plot_mid(X))

```

Isotopologue M+0 M+1**EMU**

B_3	0.5	0.5
C_1	0.5	0.5

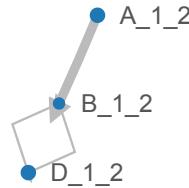
MID

<ggplot: (8767530851871)>

2.3.2 EMU Network size 2

In [100]:

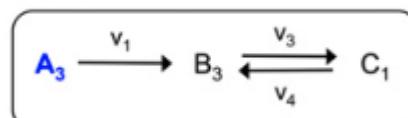
```
1 emu_network_size_2 = Model("figure3a_EMU_network_size_2")
2
3 A_1_2 = Metabolite("A_1_2")
4 B_1_2 = Metabolite("B_1_2")
5 D_1_2 = Metabolite("D_1_2")
6
7 R1 = Reaction("R1")
8 R3 = Reaction("R3")
9 R4 = Reaction("R4")
10 B1 = Reaction("B1")
11 B2 = Reaction("B2")
12
13 emu_network_size_2.add_metabolites([A_1_2, B_1_2, D_1_2])
14 emu_network_size_2.add_reactions([R1, R3, R4, B1, B2])
15
16 emu_network_size_2.reactions.R1.build_reaction_from_string("A_1_2 -->")
17 emu_network_size_2.reactions.R3.build_reaction_from_string("B_1_2 -->")
18 emu_network_size_2.reactions.R4.build_reaction_from_string("D_1_2 <-->")
19 emu_network_size_2.reactions.B1.build_reaction_from_string("--> A_1_2")
20 emu_network_size_2.reactions.B2.build_reaction_from_string("D_1_2 -->")
21
22 # cobra.io.save_json_model(emu_network, "figure3_combined_emu_network")
23 EMU2 = get_stoichiometric_matrix(emu_network_size_2).astype(int)
24 display(
25     flux_map(
26         # cobra.io.load_json_model("figure1_simple_metabolic_network.json")
27         emu_network_size_2,
28         display_name_format=lambda x: str(x.id),
29         figsize=(800, 600),
30         flux_dict=dict(R1=v1, R3=v3, R4=v4),
31     )
32 )
```

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

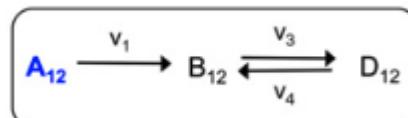
1

[\[EMU_size_2\].\(emu-size-2.png\)](#)[EMU size 2 \(emu-size-2.png\)](#)

EMU network size 1



EMU network size 2



2.3.2.1 Isotopic labeling state

$$A_{1,2} = \begin{bmatrix} M+0 & M+1 & M+2 \\ 0 & 0.5 & 0.5 \end{bmatrix} \quad (1)$$

2.3.2.2 EMU size 2 balance equations

$$(v_1 + v_4)B_{1,2} = v_1 A_{1,2} + v_4 D_{1,2} \quad (2)$$

$$v_3 D_{1,2} = v_3 B_{1,2} \quad (3)$$

2.3.2.3 EMU size 2 matrix equation

$$\begin{bmatrix} v_1 + v_4 & -v_4 \\ v_3 & -v_3 \end{bmatrix} \begin{bmatrix} B_{1,2} \\ D_{1,2} \end{bmatrix} = \begin{bmatrix} v_1 \\ 0 \end{bmatrix} \begin{bmatrix} A_{1,2} \end{bmatrix} \quad (4)$$

2.3.2.4 EMU size 2 solution

$$\begin{bmatrix} B_3 \\ C_1 \end{bmatrix} = \begin{bmatrix} 140 & -40 \\ 40 & -40 \end{bmatrix}^{-1} \begin{bmatrix} 100 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 \end{bmatrix} \quad (5)$$

In [50]:

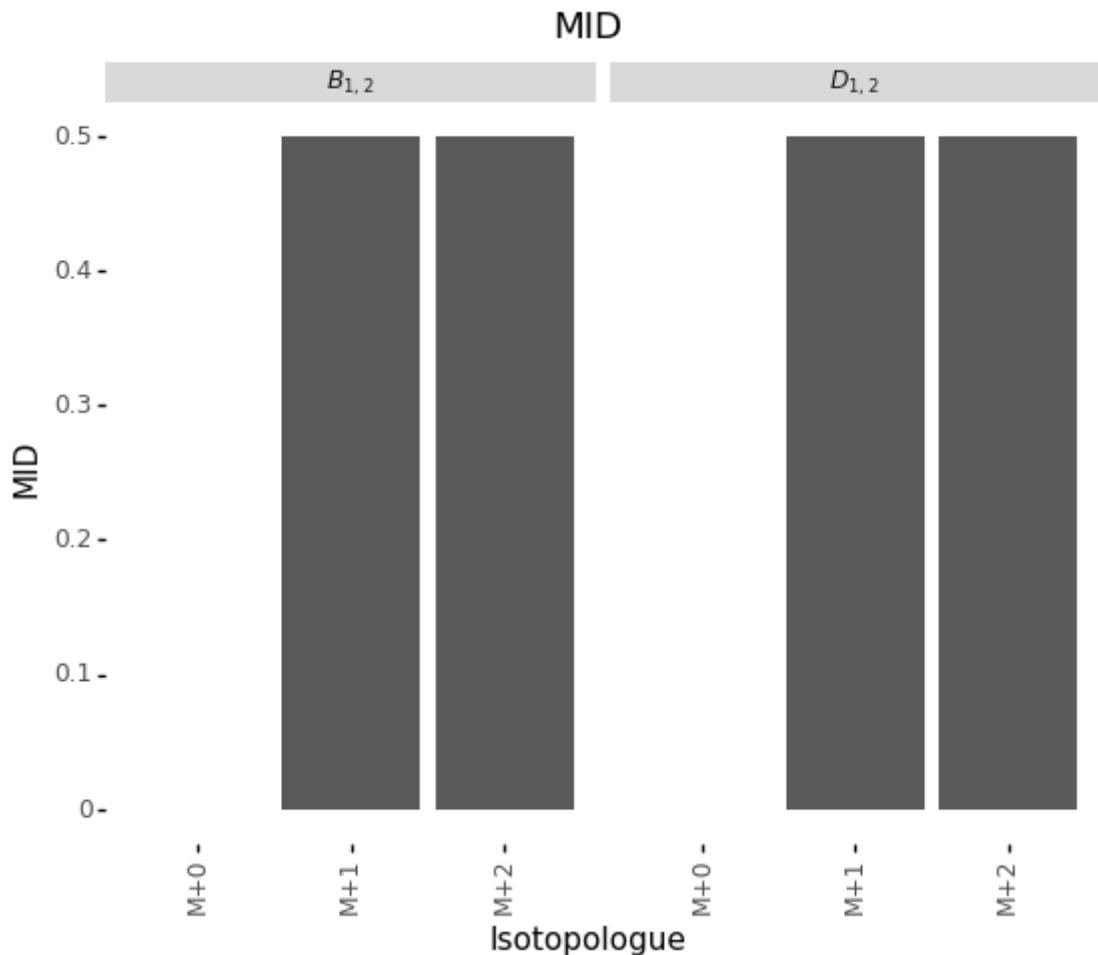
```

1 import numpy as np
2 import pandas as pd
3
4 v1, v2, v3, v4 = 100, 100, 40, 40
5
6 A_1_2 = pd.DataFrame(
7     [[0, 0.5, 0.5]], columns=["M+0", "M+1", "M+2"], index=[ "$A_{1,2}$" ]
8 )
9 display(A_1_2)
10 A = np.array([[v1 + v4, -v4], [v3, -v3]])
11 B = np.array([[v1], [0]])
12 Y = A_1_2.values
13 X = pd.DataFrame(
14     np.linalg.pinv(A).dot(B).dot(Y),
15     columns=["M+0", "M+1", "M+2"],
16     index=[r"$B_{1,2}$", "$D_{1,2}$"],
17 ).rename_axis(index="EMU", columns="Isotopologue")
18 display(X)
19 plot_mid(X)

```

	M+0	M+1	M+2
$A_{1,2}$	0	0.5	0.5

Isotopologue	M+0	M+1	M+2
EMU			
$B_{1,2}$	0.0	0.5	0.5
$D_{1,2}$	0.0	0.5	0.5



Out[50]: <ggplot: (8767548086516)>

In []: 1

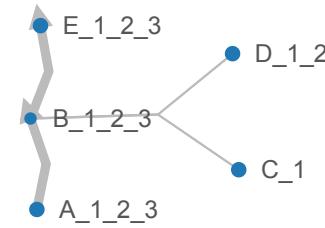
2.3.3 EMU Network size 3

In [101]:

```

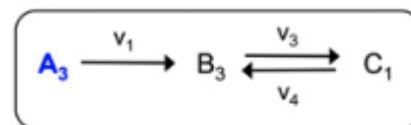
1 emu_network_size_3 = Model("figure3a_EMU_network_size_3")
2
3 A_1_2_3 = Metabolite("A_1_2_3")
4 B_1_2_3 = Metabolite("B_1_2_3")
5 C_1 = Metabolite("C_1")
6 D_1_2 = Metabolite("D_1_2")
7 E_1_2_3 = Metabolite("E_1_2_3")
8
9 R1 = Reaction("R1")
10 R2 = Reaction("R2")
11 R4 = Reaction("R4")
12 B1 = Reaction("B1")
13 B2 = Reaction("B2")
14 B3 = Reaction("B3")
15 B4 = Reaction("B4")
16
17 emu_network_size_3.add_metabolites([A_1_2_3, B_1_2_3, D_1_2, C_1, E_1_2_3])
18 emu_network_size_3.add_reactions([R1, R2, R4, B1, B2, B3, B4])
19
20 emu_network_size_3.reactions.R1.build_reaction_from_string("A_1_2_3 --> B_1_2_3")
21 emu_network_size_3.reactions.R2.build_reaction_from_string("B_1_2_3 --> D_1_2 + C_1")
22 emu_network_size_3.reactions.R4.build_reaction_from_string("D_1_2 + C_1 --> A_1_2_3")
23 emu_network_size_3.reactions.B1.build_reaction_from_string("--> A_1_2_3")
24 emu_network_size_3.reactions.B2.build_reaction_from_string("E_1_2_3 --> D_1_2")
25 emu_network_size_3.reactions.B3.build_reaction_from_string("--> D_1_2")
26 emu_network_size_3.reactions.B4.build_reaction_from_string("--> C_1")
27
28
29 # cobra.io.save_json_model(emu_network, "figure3_combined_emu_network.json")
30 EMU3 = get_stoichiometric_matrix(emu_network_size_3).astype(int)
31 display(
32     flux_map(
33         cobra.io.load_json_model(f"{emu_network_size_3}.json"),
34         display_name_format=lambda x: str(x.id),
35         figsize=(800, 600),
36         flux_dict=dict(R1=v1, R2=v2, R4=v4),
37     )
38 )

```

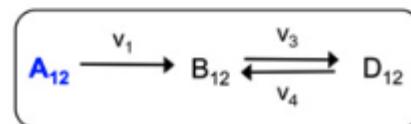
[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

[emu-size-3 \(emu-size-3.png\)](#)

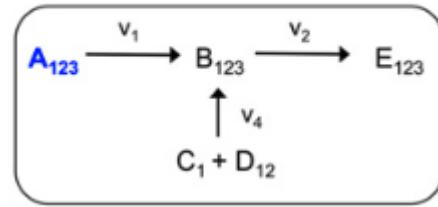
EMU network size 1



EMU network size 2



EMU network size 3



2.3.3.1 Isotopic labeling state

$$\begin{bmatrix} A_{1,2,3} \\ C_1 \times D_{1,2} \end{bmatrix} = \begin{bmatrix} M+0 & M+1 & M+2 & M+3 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0.25 & 0.5 & 0.25 \end{bmatrix} \quad (1)$$

2.3.3.2 EMU size 3 balance equations

$$(v_1 + v_4)B_{1,2,3} = v_1 A_{1,2,3} + v_4 C_1 \times D_{1,2} \quad (2)$$

$$v_2 E_{1,2,3} = v_2 B_{1,2,3} \quad (3)$$

2.3.3.3 EMU size 3 matrix equation

$$\begin{bmatrix} v_1 + v_4 & 0 \\ v_2 & -v_2 \end{bmatrix} \begin{bmatrix} B_{1,2,3} \\ E_{1,2,3} \end{bmatrix} = \begin{bmatrix} v_1 & v_4 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} A_{1,2,3} \\ C_1 \times D_{1,2} \end{bmatrix} \quad (4)$$

2.3.3.4 EMU size 3 solution

$$\begin{bmatrix} B_{1,2,3} \\ E_{1,2,3} \end{bmatrix} = \begin{bmatrix} 140 & 0 \\ 100 & -100 \end{bmatrix}^{-1} \begin{bmatrix} 100 & 40 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.43 & 0.14 & 0.43 \\ 0 & 0.43 & 0.14 & 0.43 \end{bmatrix}$$

In [77]:

```

1 import numpy as np
2 import pandas as pd
3
4 v1, v2, v3, v4 = 100, 100, 40, 40
5 A123 = pd.DataFrame(
6     [[0, 0.5, 0, 0.5]], columns=[f"M{i}" for i in range(4)], index=[r"$A_{1,2,3}$"]
7 )
8 display(A123)
9 C1 = pd.DataFrame([[0.5, 0.5]], columns=["M+0", "M+1"], index=[r"$C_1$"])
10 display(C1)
11 D12 = pd.DataFrame([[0, 0.5, 0.5]], columns=["M+0", "M+1", "M+2"], index=[r"$D_{1,2}$"])
12 display(D12)
13
14 display()
15 C1_X_D12 = pd.DataFrame(
16     [np.convolve(C1.loc[r"$C_1$"].values, D12.loc[r"$D_{1,2}$"].values,
17                  columns=[f"M{i}" for i in range(4)],
18                  index=[r"$C_1 \times D_{1,2}$"]),
19      )
20 )
21 display(C1_X_D12)
22 A_1_2_3_and_C1_times_D_1_2 = pd.DataFrame(
23     [[0, 0.5, 0, 0.5], [0, 0.25, 0.5, 0.25]],
24     columns=["M+0", "M+1", "M+2", "M+3"],
25     index=[r"$A_{1,2,3}$", r"$C_1 \times D_{1,2}$"],
26 )
27 A = np.array([[v1 + v4, 0], [v2, -v2]])
28 B = np.array([[v1, v4], [0, 0]])
29 Y = pd.concat(
30     [A123, C1_X_D12],
31 )
32 display(Y)
33 X = pd.DataFrame(
34     np.linalg.pinv(A).dot(B).dot(Y),
35     columns=["M+0", "M+1", "M+2", "M+3"],
36     index=[r"$B_{1,2,3}$", "$E_{1,2,3}$"],
37 ).rename_axis(index="EMU", columns="Isotopologue")
38 display(X)
39 display(plot_mid(X))

```

	M+0	M+1	M+2	M+3
A _{1,2,3}	0	0.5	0	0.5

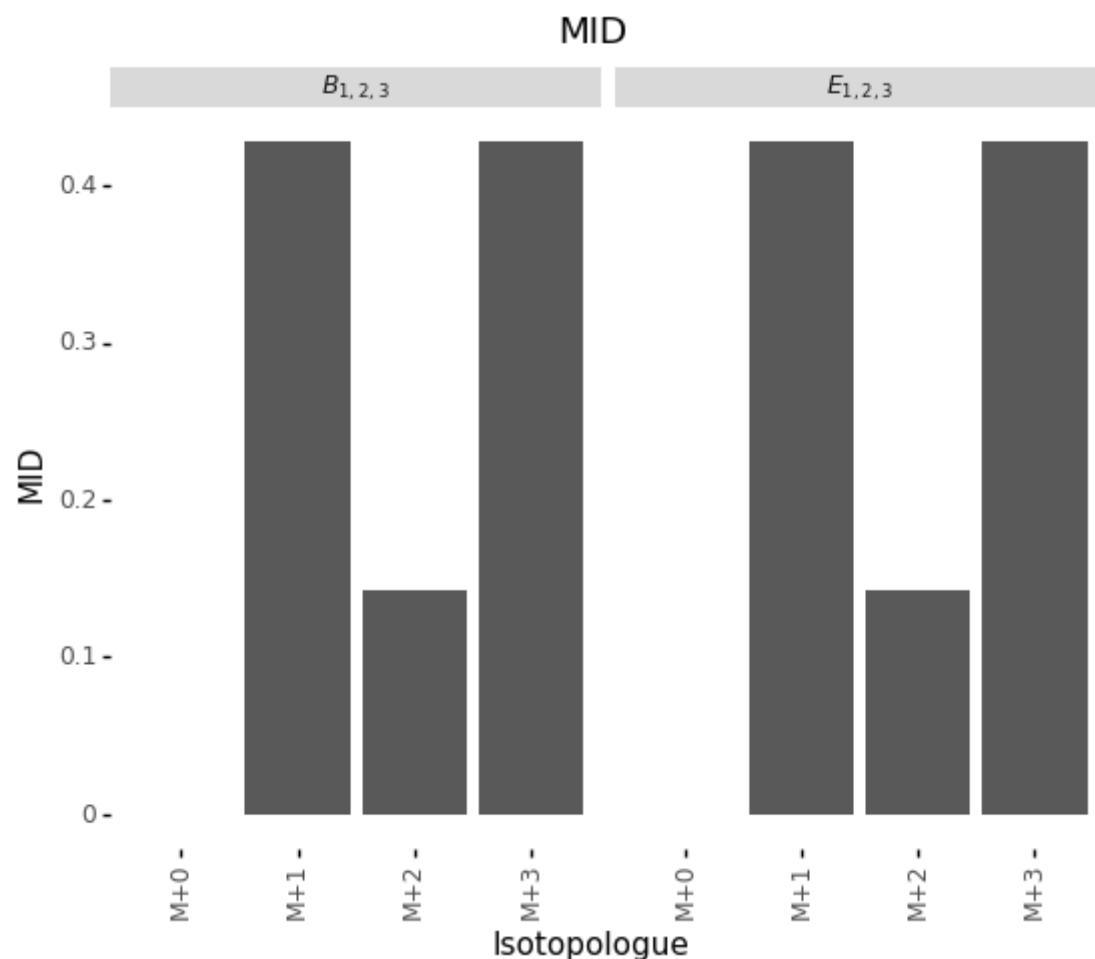
	M+0	M+1
C ₁	0.5	0.5

	M+0	M+1	M+2
D _{1,2}	0	0.5	0.5

	M+0	M+1	M+2	M+3
$C_1 \times D_{1,2}$	0.0	0.25	0.5	0.25

	M+0	M+1	M+2	M+3
$A_{1,2,3}$	0.0	0.50	0.0	0.50
$C_1 \times D_{1,2}$	0.0	0.25	0.5	0.25

Isotopologue	M+0	M+1	M+2	M+3
EMU				
$B_{1,2,3}$	0.0	0.428571	0.142857	0.428571
$E_{1,2,3}$	0.0	0.428571	0.142857	0.428571



```
<ggplot: (8767530711796)>
```

2.4 Studying the effect of good and poor tracers

Normally, the process for calculating the steady-state fluxes given the measured MIDs is as follows:

1. Guess a plausible flux distribution
2. Solve for the MIDs that this flux distribution entails.
3. Compare the measured MIDs with the calculated MIDs
4. If they are close enough, return the flux distribution.
5. Otherwise, repeat step 1.

However, our network is small enough that we can calculate an analytical solution. We use this solution to study the effect of good and bad tracers on our ability to calculate steady-state fluxes.

2.4.1 Combined EMU Network

[Combined emu \(emu-combined.png\)](#)

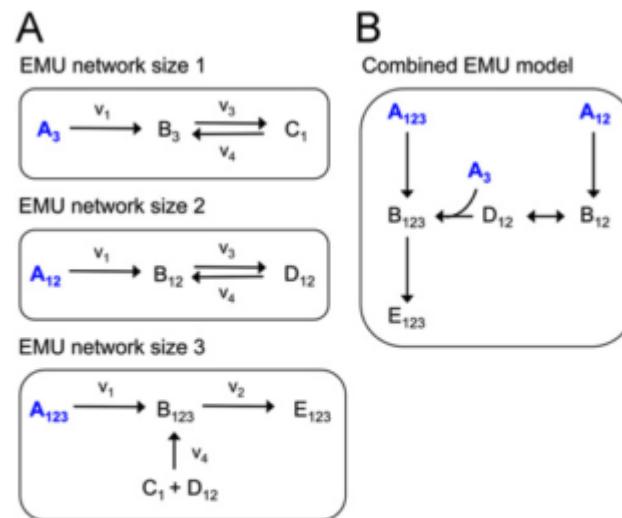


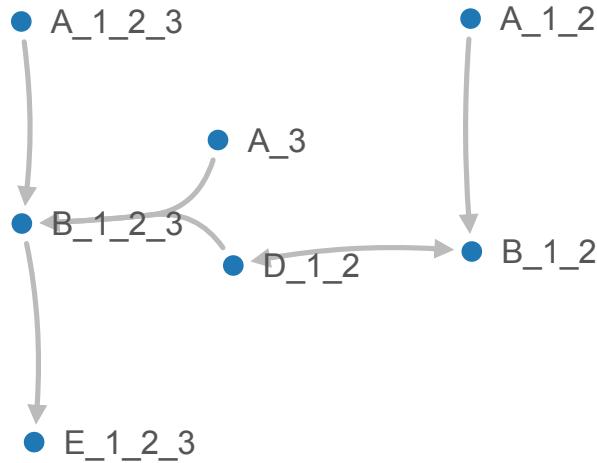
Fig. 3. (A) Decoupled EMU reaction networks for the example network model. (B) Combined EMU reaction network showing the flow of isotopic labeling from the substrate A to the network product E . The two EMU basis vectors in the model were A_{123} and $A_{12} \times A_3$.

In [52]:

```

1 emu_network = Model("figure3_combined_EMU_network")
2 A_1_2_3 = Metabolite("A_1_2_3")
3 A_1_2 = Metabolite("A_1_2")
4 A_3 = Metabolite("A_3")
5 B_1_2_3 = Metabolite("B_1_2_3")
6 D_1_2 = Metabolite("D_1_2")
7 B_1_2 = Metabolite("B_1_2")
8 E_1_2_3 = Metabolite("E_1_2_3")
9 R1 = Reaction("R1")
10 R2 = Reaction("R2")
11 R3 = Reaction("R3")
12 R4 = Reaction("R4")
13 R5 = Reaction("R5")
14 B1 = Reaction("B1")
15 B2 = Reaction("B2")
16 B3 = Reaction("B3")
17 B4 = Reaction("B4")
18 emu_network.add_metabolites([A_1_2_3, A_1_2, A_3, B_1_2_3, B_1_2, D_1_2])
19 emu_network.add_reactions([R1, R2, R3, R4, R5, B1, B2, B3, B4])
20
21 emu_network.reactions.R1.build_reaction_from_string("A_1_2_3 --> B_1_2")
22 emu_network.reactions.R2.build_reaction_from_string("B_1_2_3 --> E_1_2")
23 emu_network.reactions.R3.build_reaction_from_string("A_3 + D_1_2 --> E_1_2")
24 emu_network.reactions.R4.build_reaction_from_string("B_1_2 <--> D_1_2")
25 emu_network.reactions.R5.build_reaction_from_string("A_1_2 --> B_1_2")
26 emu_network.reactions.B1.build_reaction_from_string("--> A_1_2_3")
27 emu_network.reactions.B2.build_reaction_from_string("--> A_1_2")
28 emu_network.reactions.B3.build_reaction_from_string("--> A_3")
29 emu_network.reactions.B4.build_reaction_from_string("E_1_2_3 -->")
30
31 # cobra.io.save_json_model(emu_network, "figure3_combined_emu_network.json")
32 EMU_Combined = get_stoichiometric_matrix(emu_network).astype(int)
33
34 display(
35     flux_map(
36         cobra.io.load_json_model("figure3_combined_emu_network.json"),
37         display_name_format=lambda x: str(x.id),
38         figsize=(600, 400),
39     )
40 )

```

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

2.4.2 EMU basis vectors

Each **EMU basis vector** can be viewed as a unique way for assembling substrate EMUs to form the product. By definition, EMU basis vectors have the same EMU size as the product. Importantly, given a network and a labeled substrate, one can associate a Mass Isotopic Distribution (MID) with each EMU basis vector. Ultimately, one can interpret the MID of the product as a linear combination of EMU basis vector MIDs, where the coefficients are solely a function of free fluxes. The coefficients quantify the fractional contribution of each EMU basis vector to the product. Thus, by definition, the sum of the coefficients must equal one.

For the simple model, an analytic solution exists that can be written in terms of the EMU basis vectors and the free fluxes:

$$E_{1,2,3} = \frac{v_1}{v_1 + v_4} A_{1,2,3} + \frac{v_4}{v_1 + v_4} A_{1,2} \times A_3 \quad (1)$$

In [81]:

```

1 display(Markdown("#### EMU basis vectors for $E_{1,2,3}\$\n"))
2 emu_transitions, emus = EMU_Combined.columns, EMU_Combined.index
3 external_emus = ["A_1_2_3", "A_1_2", "A_3", "E_1_2_3"]
4 external_emu_transitions = ["B1", "B2", "B3", "B4"]
5 output_emu = "E_1_2_3"
6 output_emu_transition = "B4"
7 ebv, emu_elmo = get_EMU_basis_vectors_and_pathways(
8     emu_network,
9     external_emu_transitions,
10    external_emus,
11    output_emu,
12    output_emu_transition,
13 )
14 ebv

```

EMU basis vectors for $E_{1,2,3}$

```
/Users/zuck016/.pyenv/versions/anaconda3-2020.11/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering (buffering=1) isn't supported in binary mode, the default buffer size will be used
```

Out[81]:

	A_1_2_3	A_1_2	A_3	E_1_2_3	EMU Basis Vector
EM1	-1.0	-0.0	-0.0	1.0	$A_{1,2,3}$
EM2	-0.0	-1.0	-1.0	1.0	$A_{1,2} \times A_3$

2.4.3 EMU pathways

In addition to the EMU basis vector, Crown et al also define an **EMU pathway**. Each EMU basis vector specifies a unique EMU pathway from substrate(s) to product.

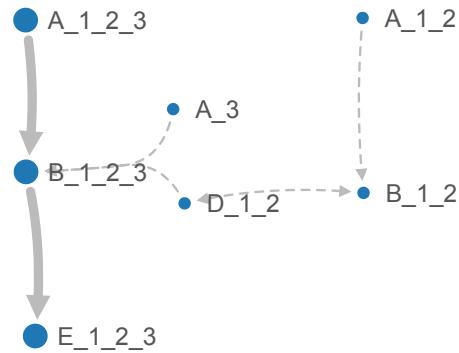
- An **Elementary mode (ELMO)** of a network is a minimal cardinality set of reactions that form a steady-state. Minimal cardinality means that no steady-state flux distribution can be a subset of an elementary mode. The set of elementary modes forms a basis for steady-state flux space.
- Assertion: The elementary modes of the EMU network are the EMU pathways. Evidence for this assertion is provided below.

```
In [78]: 1 display(Markdown("### EMU Pathways for $E_{1,2,3}\$\n"))
2 emu_elmo[ "EMU Basis Vector" ] = ebv[ "EMU Basis Vector" ]
3
4 display(emu_elmo)
5
6 for em in emu_elmo.index:
7     display(
8         Markdown(
9             "#### EMU Pathway {} ({.format(em)} + ebv.loc[em, "EMU Basi
10            )
11        )
12    display(
13        flux_map(
14            cobra.io.load_json_model("figure3_combined_emu_network.json"),
15            display_name_format=lambda x: str(x.id),
16            flux_dict={
17                emu_transition.id: emu_elmo.loc[em, emu_transition.id]
18                for emu_transition in emu_network.reactions
19            },
20            figsize=(800, 600),
21        )
22    )
```

EMU Pathways for $E_{1,2,3}$

	R1	R2	R3	R4	R5	B1	B2	B3	B4	EMU Basis Vector
EM1	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	$A_{1,2,3}$
EM2	0.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	$A_{1,2} \times A_3$

EMU Pathway EM1 ($A_{1,2,3}$)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

EMU Pathway EM2 ($A_{1,2} \times A_3$)

[EMU pathway 1 \(emu-pathway-1.png\)](#)

[EMU pathway 2 \(emu-pathway-2.png\)](#)

2.4.4 Tracer Compositions for Simple EMU network

Given 4 different tracer compositions of:

- 100% A_{100} ,
- 100% A_{111} ,
- 50% A_{101} and 50% A_{111} ,
- 50% A_{100} and 50% A_{111} ,

what are the isotopic labeling states and mass isotopomer distributions for each EMU Basis Vector?

Compare the results with Figure 4.

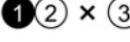
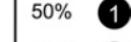
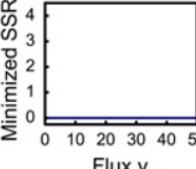
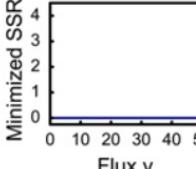
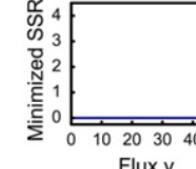
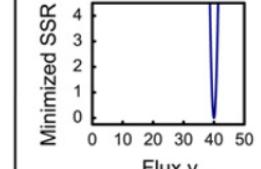
In [79]:

```

1 display(Image("TracerCompositions.png"))
2 isotopic_labeling_states = [
3     pd.DataFrame([{"%": 100, 1: 1, 2: 0, 3: 0}], index=["$100%\ A_{100}"]),
4     pd.DataFrame([{"%": 100, 1: 1, 2: 1, 3: 1}], index=["$100%\ A_{111}"]),
5     pd.DataFrame(
6         [{"%": 50, 1: 1, 2: 0, 3: 1}, {"%": 50, 1: 1, 2: 1, 3: 1}],
7         index=["$50%\ A_{101}$", "$50%\ A_{111}$"]),
8     ),
9     pd.DataFrame(
10        [{"%": 50, 1: 1, 2: 0, 3: 0}, {"%": 50, 1: 1, 2: 1, 3: 1}],
11        index=["$50%\ A_{100}$", "$50%\ A_{111}$"]),
12    ),
13 ]
14 tracer_mixtures = [dict(A=Tracer("A", 3, ils)) for ils in isotopic_labeling_states]
15 emu_basis_vectors = [
16     EMU_Basis_Vector(ebv.iloc[0, :-1]),
17     EMU_Basis_Vector(ebv.iloc[1, :-1]),
18 ]
19 mid_comparison = {}
20 isl_comparison = {}
21 for tracers in tracer_mixtures:
22     for t in tracers:
23         for emu_basis_vector in emu_basis_vectors:
24             for emu in emu_basis_vector.get_input_emos():
25                 emu_isl = emu.get_isotopic_labeling_state(tracers)
26                 emu_isl.index = pd.MultiIndex.from_tuples(
27                     [(tracers[t].get_name(), a) for a in emu_isl.index])
28             for atom in emu_isl.columns:
29                 key = (emu_basis_vector.get_name(), emu.get_name())
30                 if key in isl_comparison:
31                     isl_comparison[key] = isl_comparison[key].append(emu_isl[atom])
32                 else:
33                     isl_comparison[key] = emu_isl[atom]
34     mid_comparison[
35         tracers[t].get_name(), emu_basis_vector.get_name()
36     ] = emu_basis_vector.get_mid(tracers)
37
38 display(Markdown("### Isotopomer labeling states"))
39 isl_df = pd.DataFrame(isl_comparison)
40 isl_df.columns = isl_df.columns.set_names(["EMU Basis Vectors", "EMUs"])
41 isl_df.index = isl_df.index.set_names(["Tracer Compositions", "Tracers"])
42 isl_df.style.set_table_styles([dict(selector="th", props=[("text-align", "center")])])
43 isl_df.style.set_properties(**{"text-align": "left"})
44 display(isl_df.T)
45 display(Markdown("### Mass Isotopomer Distribution"))
46
47
48 mid = pd.DataFrame(mid_comparison).T
49 mid.index.set_names(["Tracer compositions", "Tracers"], inplace=True)
50 display(mid)

```

Tracer Composition

	100% 	100% 	50%  50% 	50%  50% 
A_{123}	MID = [0 1 0 0]	MID = [0 0 0 1]	MID = [0 0 0.5 0.5]	MID = [0 0.5 0 0.5]
$A_{12} \times A_3$	100%  	100%  	50%   50%  	50%   50%  
Confidence Intervals	Minimized SSR 	Minimized SSR 	Minimized SSR 	Minimized SSR 

Isotopomer labeling states

Tracer Compositions	100% A_{100}	100% A_{111}	50% A_{101} and 50% A_{111}		50% A_{100} and 50% A_{111}	
Tracers	100% A_{100}	100% A_{111}	50% A_{101}	50% A_{111}	50% A_{100}	50% A_{111}
EMU Basis Vectors						
$A_{1,2,3}$	$A_{1,2,3}$	1	1	1	1	1
		2	0	1	0	1
		3	0	1	1	0
$A_{1,2} \times A_3$	$A_{1,2}$	1	1	1	1	1
		2	0	1	0	1
	A_3	3	0	1	1	0

Mass Isotopomer Distribution

Tracer compositions		M+0	M+1	M+2	M+3
Tracers					
100% A_{100}	$A_{1,2,3}$	0.0	100.0	0.0	0.0
$A_{1,2} \times A_3$		0.0	100.0	0.0	0.0
100% A_{111}	$A_{1,2,3}$	0.0	0.0	0.0	100.0

M+0 M+1 M+2 M+3

Tracer compositions		Tracers	0.0	0.0	0.0	100.0
	A _{1,2} × A ₃		0.0	0.0	0.0	100.0
50% A ₁₀₁ and 50% A ₁₁₁	A _{1,2} × A ₃	A _{1,2,3}	0.0	0.0	50.0	50.0
50% A ₁₀₀ and 50% A ₁₁₁	A _{1,2} × A ₃	A _{1,2,3}	0.0	50.0	0.0	50.0
			0.0	25.0	50.0	25.0

2.4.4.1 MID plots of EMU basis vectors for each tracer

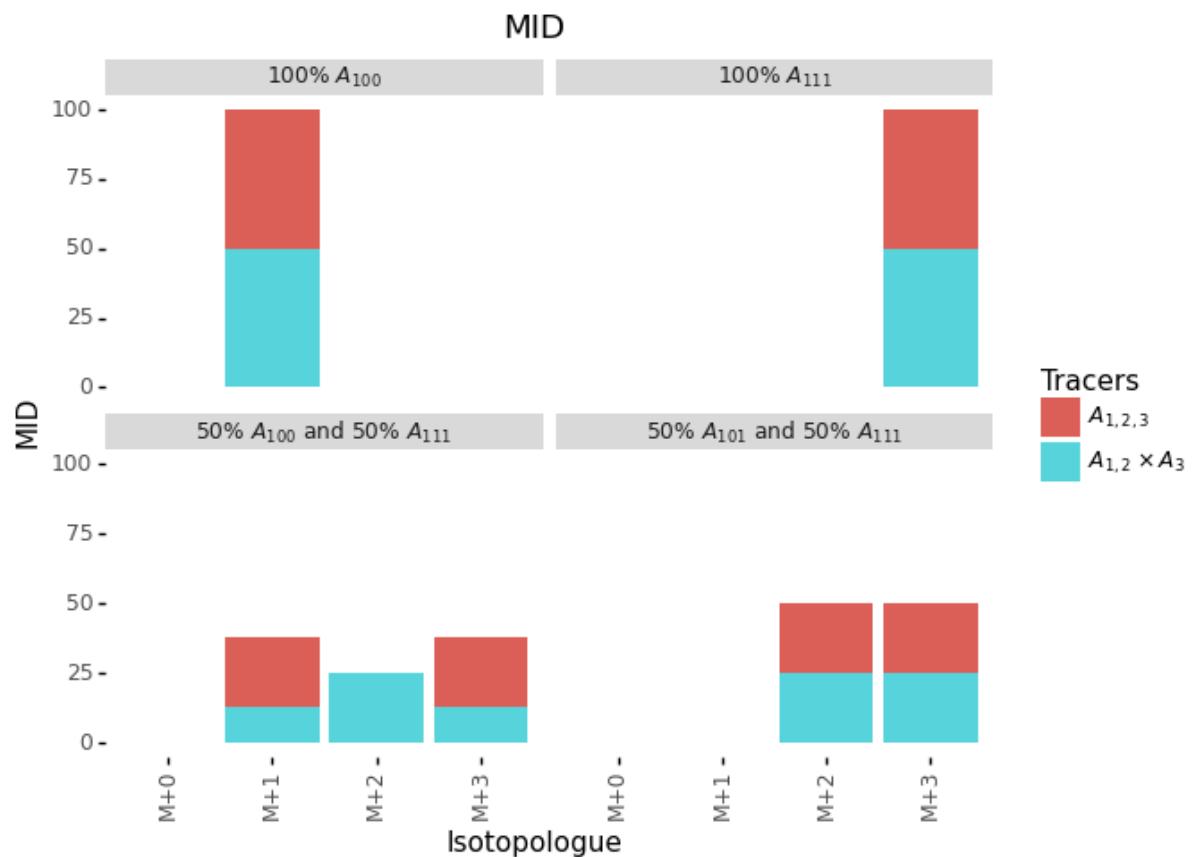
In [109]:

```

1 mid.columns.name = "Isotopologue"
2 mid_gg = mid.stack().reset_index().rename(columns={0: "MID"})
3 mid_gg.Tracers = mid_gg.Tracers.str.replace(r"\text{A}", "A")
4 mid_gg.MID = mid_gg.MID / 2
5 mid_gg
6 plot = (
7     ggplot(
8         mid_gg,
9         aes(x="Isotopologue", y="MID", fill="Tracers"),
10    )
11    + ggtitle("MID")
12    + theme(
13        panel_background=element_rect(fill="white"),
14        axis_text_x=element_text(angle=90),
15    )
16    + geom_col()
17    + facet_wrap("Tracer compositions")
18 )
19 display(plot)

```

<ipython-input-109-37c1e14e5847>:3: FutureWarning: The default value of regex will change from True to False in a future version.



<ggplot: (8791942336506)>

```
In [110]: 1 observed_mid = mid.groupby(level="Tracer compositions").sum() / 2
           2 observed_mid
```

Out[110]:

Isotopologue	M+0	M+1	M+2	M+3
Tracer compositions				
100% A_{100}	0.0	100.0	0.0	0.0
100% A_{111}	0.0	0.0	0.0	100.0
50% A_{100} and 50% A_{111}	0.0	37.5	25.0	37.5
50% A_{101} and 50% A_{111}	0.0	0.0	50.0	50.0

2.4.4.2 MID plots of measured EMUs

```
In [111]: 1 observed_mid.columns.name = "Isotopologue"
           2 observed_mid_gg = observed_mid.stack().reset_index().rename(columns={0: "Tracer compositions", 1: "Isotopologue", 2: "MID"})
           3 observed_mid_gg
```

Out[111]:

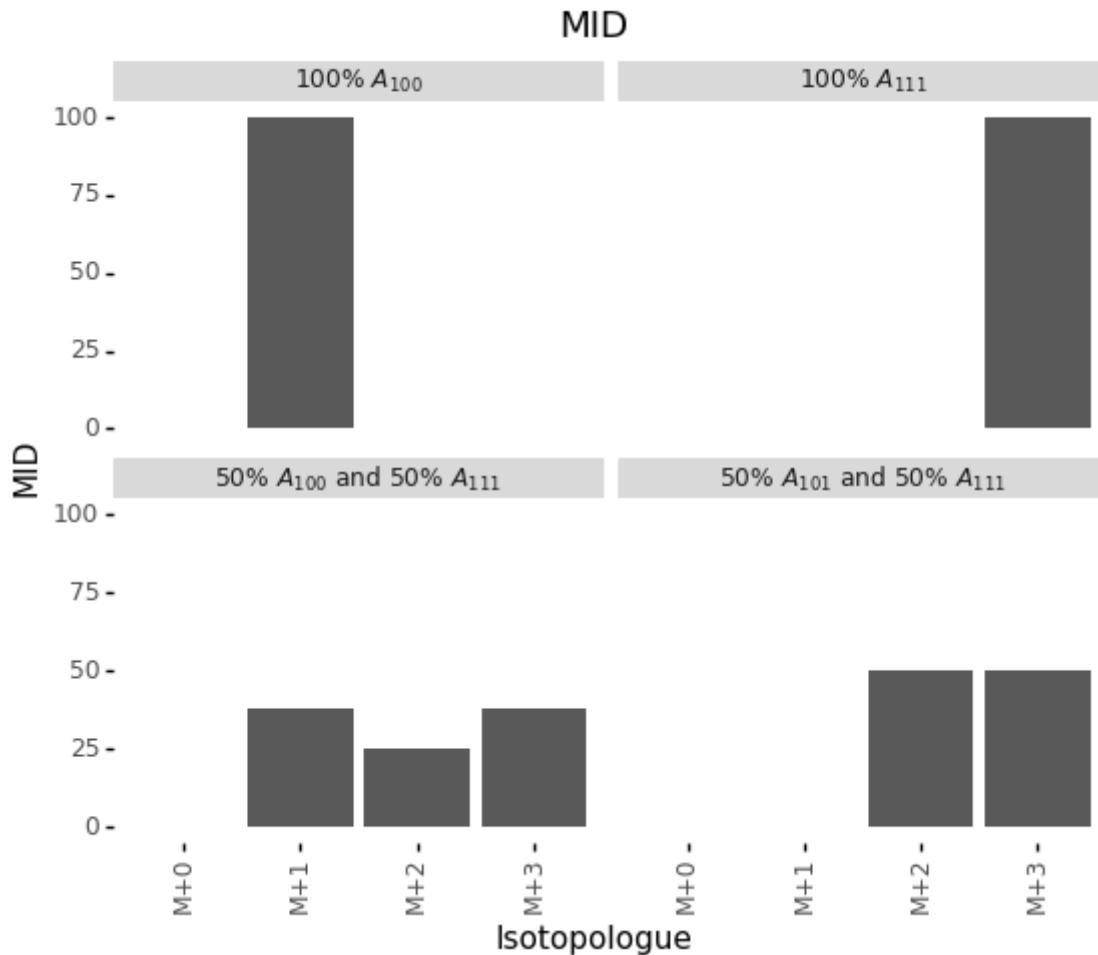
	Tracer compositions	Isotopologue	MID
0	100% A_{100}	M+0	0.0
1	100% A_{100}	M+1	100.0
2	100% A_{100}	M+2	0.0
3	100% A_{100}	M+3	0.0
4	100% A_{111}	M+0	0.0
5	100% A_{111}	M+1	0.0
6	100% A_{111}	M+2	0.0
7	100% A_{111}	M+3	100.0
8	50% A_{100} and 50% A_{111}	M+0	0.0
9	50% A_{100} and 50% A_{111}	M+1	37.5
10	50% A_{100} and 50% A_{111}	M+2	25.0
11	50% A_{100} and 50% A_{111}	M+3	37.5
12	50% A_{101} and 50% A_{111}	M+0	0.0
13	50% A_{101} and 50% A_{111}	M+1	0.0
14	50% A_{101} and 50% A_{111}	M+2	50.0
15	50% A_{101} and 50% A_{111}	M+3	50.0

In [112]:

```

1 plot = (
2     ggplot(
3         observed_mid_gg,
4         aes(x="Isotopologue", y="MID"),
5     )
6     + ggtitle("MID")
7     + theme(
8         panel_background=element_rect(fill="white"),
9         axis_text_x=element_text(angle=90),
10    )
11    + geom_col()
12    + facet_wrap("Tracer compositions")
13)
14 display(plot)

```



<ggplot: (8791840034262)>

3 Extended metabolic model

The Simple metabolic model is extended with three new reactions, and the atom transitions are shown below

In [17]:

```

1 display(Markdown("## Extended Metabolic Reaction Network"))
2 display(Image("Extended_EMU_network.png"))
3 display(Image("Table2.png"))

```

Extended Metabolic Reaction Network

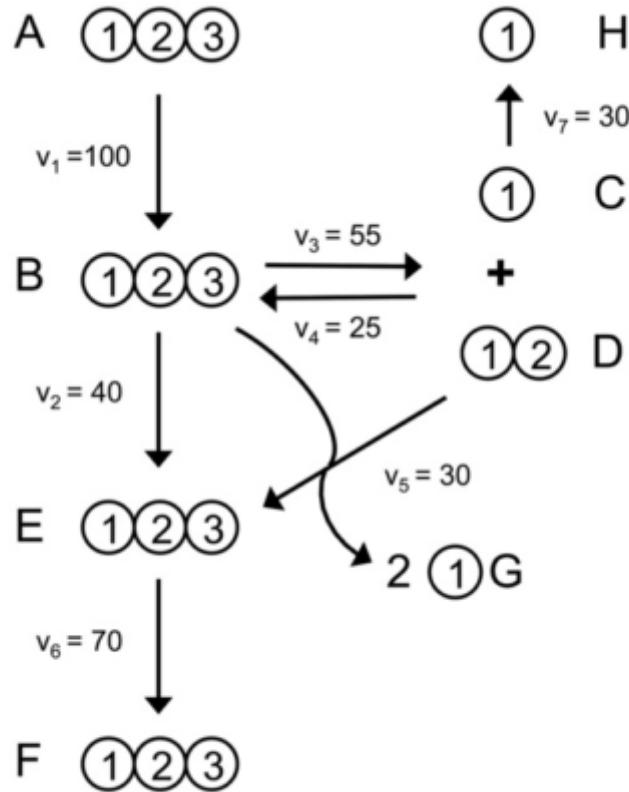


Fig. 5. Extended metabolic network model. The assumed fluxes have arbitrary units.

Table 2

Stoichiometry and atom transitions for the reactions in the extended network model.

Reaction number	Reaction stoichiometry	Atom transitions
1	$A \rightarrow B$	$abc \rightarrow abc$
2	$B \rightarrow E$	$abc \rightarrow abc$
3 and 4	$B \leftrightarrow C + D$	$abc \leftrightarrow c + ab$
5	$B + D \rightarrow E + 2G$	$abc + de \rightarrow abd + c + e$
6	$E \rightarrow F$	$abc \rightarrow abc$
7	$C \rightarrow H$	$a \rightarrow a$

In [114]:

```

1 metabolic_network = Model("extended_metabolic_network")
2 A = Metabolite("A")
3 B = Metabolite("B")
4 C = Metabolite("C")
5 D = Metabolite("D")
6 E = Metabolite("E")
7 F = Metabolite("F")
8 G = Metabolite("G")
9 H = Metabolite("H")
10 v1 = Reaction("v1")
11 v2 = Reaction("v2")
12 v3 = Reaction("v3")
13 v4 = Reaction("v4")
14 v5 = Reaction("v5")
15 v6 = Reaction("v6")
16 v7 = Reaction("v7")
17 b1 = Reaction("b1")
18 b2 = Reaction("b2")
19 b3 = Reaction("b3")
20 b4 = Reaction("b4")
21 metabolic_network.add_metabolites([A, B, C, D, E, F, G, H])
22 metabolic_network.add_reactions([v1, v2, v3, v4, v5, v6, v7, b1, b2, b3, b4])
23 metabolic_network.reactions.v1.build_reaction_from_string("A --> B")
24 metabolic_network.reactions.v2.build_reaction_from_string("B --> E")
25 metabolic_network.reactions.v3.build_reaction_from_string("B --> C + I")
26 metabolic_network.reactions.v4.build_reaction_from_string("C + D --> E")
27 metabolic_network.reactions.v5.build_reaction_from_string("B + D --> F")
28 metabolic_network.reactions.v6.build_reaction_from_string("E --> F")
29 metabolic_network.reactions.v7.build_reaction_from_string("C --> H")
30 metabolic_network.reactions.b1.build_reaction_from_string("--> A")
31 metabolic_network.reactions.b2.build_reaction_from_string("F -->")
32 metabolic_network.reactions.b3.build_reaction_from_string("G -->")
33 metabolic_network.reactions.b4.build_reaction_from_string("H -->")
34
35 metabolic_network.reactions.v1.notes["CARBON TRANSITIONS"] = "A --> B\t"
36 metabolic_network.reactions.v2.notes["CARBON TRANSITIONS"] = "B --> E\t"
37 metabolic_network.reactions.v3.notes["CARBON TRANSITIONS"] = "B --> C + I\t"
38 metabolic_network.reactions.v4.notes["CARBON TRANSITIONS"] = "C + D --> E\t"
39 metabolic_network.reactions.v5.notes[
40     "CARBON TRANSITIONS"
41 ] = "B + D --> E + 2 G\tabc + de : abd + c + e"
42 metabolic_network.reactions.v6.notes["CARBON TRANSITIONS"] = "E --> F\t"
43 metabolic_network.reactions.v7.notes["CARBON TRANSITIONS"] = "C --> H\t"
44
45 S = get_stoichiometric_matrix(metabolic_network)
46 rxns, mets = S.columns, S.index
47
48
49 # display(Markdown('### Nullspace of Extended Metabolic Reaction Network'))
50 # display(get_nullspace(S).T)
51 display(Markdown("### Elementary modes of Extended Metabolic Reaction Network"))
52 elmo = calculate_elementary_modes(metabolic_network, verbose=False)
53 display(elmo)
54
55 for em in elmo.index:
56     display(

```

```

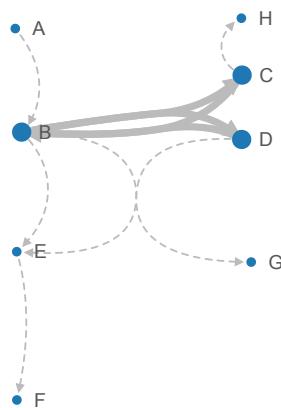
57     flux_map(
58         cobra.io.load_json_model("extended_metabolic_network.json",
59         display_name_format=lambda x: str(x.id),
60         flux_dict={
61             rxn.id: elmo.loc[em, rxn.id] for rxn in metabolic_netw
62         },
63     )
64

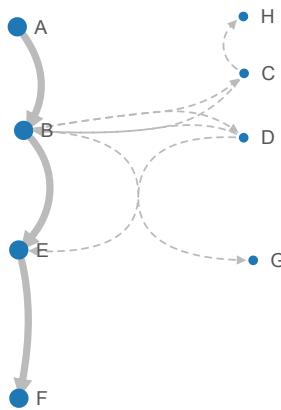
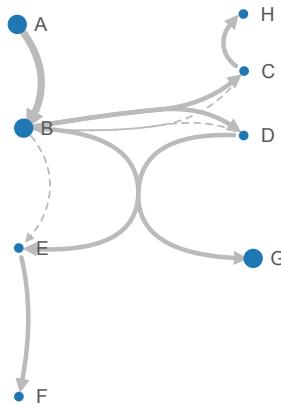
```

Elementary modes of Extended Metabolic Reaction Network

/Users/zuck016/.pyenv/versions/anaconda3-2020.11/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering (buffering=1) isn't supported in binary mode, the default buffer size will be used

	v1	v2	v3	v4	v5	v6	v7	b1	b2	b3	b4
EM1	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
EM2	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0
EM3	1.0	0.0	0.5	0.0	0.5	0.5	0.5	1.0	0.5	1.0	0.5


[Save JSON](#)
[Show Reaction Nodes](#)
[Download SVG](#)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

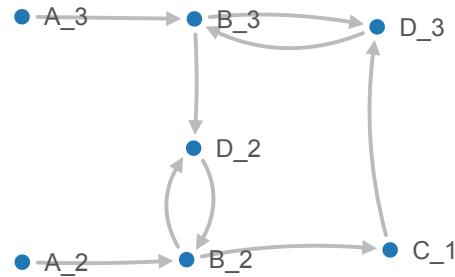
3.1 Reaction network for EMU size 1

In [115]:

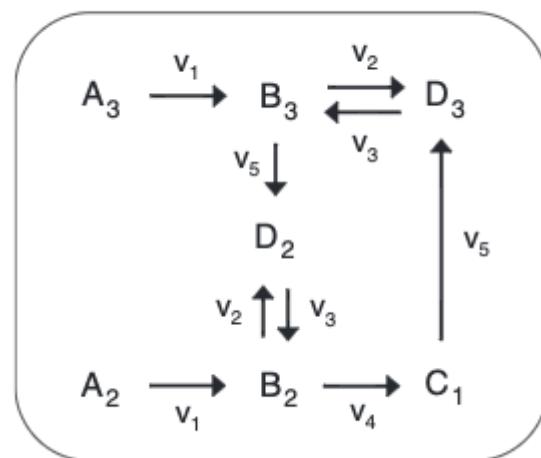
```

1 extended_emu_network_size_1 = Model("extended_emu_network_size_1")
2
3 A_2 = Metabolite("A_2")
4 A_3 = Metabolite("A_3")
5 B_2 = Metabolite("B_2")
6 B_3 = Metabolite("B_3")
7 C_1 = Metabolite("C_1")
8 D_2 = Metabolite("D_2")
9 D_3 = Metabolite("D_3")
10
11 rxns = [Reaction(f"R{i}") for i in range(1, 10)]
12
13
14
15 extended_emu_network_size_1.add_metabolites([A_2, A_3, B_2, B_3, C_1,
16 extended_emu_network_size_1.add_reactions(rxns)
17
18 extended_emu_network_size_1.reactions.R1.build_reaction_from_string("A_2 + B_2 -> R1")
19 extended_emu_network_size_1.reactions.R2.build_reaction_from_string("B_3 + C_1 -> R2")
20 extended_emu_network_size_1.reactions.R3.build_reaction_from_string("B_2 + D_2 -> R3")
21 extended_emu_network_size_1.reactions.R4.build_reaction_from_string("D_3 + C_1 -> R4")
22 extended_emu_network_size_1.reactions.R5.build_reaction_from_string("B_2 + A_2 -> R5")
23 extended_emu_network_size_1.reactions.R6.build_reaction_from_string("B_3 + D_2 -> R6")
24 extended_emu_network_size_1.reactions.R7.build_reaction_from_string("D_3 + A_2 -> R7")
25 extended_emu_network_size_1.reactions.R8.build_reaction_from_string("A_3 + C_1 -> R8")
26 extended_emu_network_size_1.reactions.R9.build_reaction_from_string("C_1 + D_2 -> R9")
27
28
29 # cobra.io.save_json_model(emu_network, "figure3_combined_emu_network.json")
30 EMU1 = get_stoichiometric_matrix(extended_emu_network_size_1).astype(int)
31 display(
32     flux_map(
33         cobra.io.load_json_model("extended_emu_network_size_1.json"),
34         # extended_emu_network_size_1,
35         display_name_format=lambda x: str(x.id),
36         figsize=(800, 600),
37     )
38 )

```

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

Reaction network for EMU size 1



$$A_3 = [] \quad (1)$$

3.1.1 EMU size 1 balance equations

$$(v_1 + v_3)B_3 = v_1A_3 + D_3v_3 \quad (2)$$

$$(v_2 + v_5)D_3 = v_2B_3 + v_5C_1 \quad (3)$$

$$(v_5 + v_2)D_2 = v_5B_3 + v_2B_2 \quad (4)$$

$$v_4C_1 = v_4B_2 \quad (5)$$

$$(v_1 + v_3)B_2 = v_3D_2 + v_1A_2 \quad (6)$$

3.1.2 EMU size 1 matrix equation

$$\begin{bmatrix} -v_4 & v_4 & 0 & 0 & 0 \\ 0 & -v_1 - v_3 & v_3 & 0 & 0 \\ 0 & v_2 & -v_2 - v_5 & v_5 & 0 \\ 0 & 0 & 0 & -v_1 - v_3 & v_3 \\ v_5 & 0 & 0 & v_2 & -v_2 - v_5 \end{bmatrix} \begin{bmatrix} C_1 \\ B_2 \\ D_2 \\ B_3 \\ D_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -v_1 & 0 \\ 0 & 0 \\ 0 & -v_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} A_2 \\ A_3 \end{bmatrix}, \quad (2a)$$

3.1.3 EMU size 1 solution

Solution of EMU balances for reaction network of EMU size 1:

$$\begin{bmatrix} C_1 \\ B_2 \\ D_2 \\ B_3 \\ D_3 \end{bmatrix} = \begin{bmatrix} -20 & 20 & 0 & 0 & 0 \\ 0 & -150 & 50 & 0 & 0 \\ 0 & 110 & -130 & 20 & 0 \\ 0 & 0 & 0 & -150 & 50 \\ 20 & 0 & 0 & 110 & -130 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ -100 & 0 \\ 0 & 0 \\ 0 & -100 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0.0667 & 0.9333 \\ 0.0667 & 0.9333 \\ 0.2000 & 0.8000 \\ 0.9333 & 0.0667 \\ 0.8000 & 0.2000 \end{bmatrix}.$$

3.2 Reaction network for EMU size 2

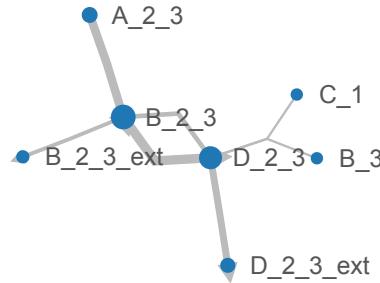
In [9]:

```

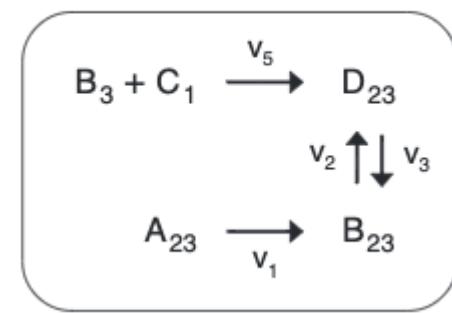
1 xextended_emu_network_size_2 = Model("figure4_extended_emu_network_size_2")
2 v1, v2, v3, v5 = 100, 110, 50, 20
3 A_2_3 = Metabolite("A_2_3")
4 B_2_3 = Metabolite("B_2_3")
5 B_3 = Metabolite("B_3")
6 C_1 = Metabolite("C_1")
7 D_2_3 = Metabolite("D_2_3")
8
9
10 rxns = [Reaction(f"R{i+1}") for i in range(6)]
11
12
13 extended_emu_network_size_2.add_metabolites([A_2_3, B_2_3, B_3, C_1, D_2_3])
14 extended_emu_network_size_2.add_reactions(rxns)
15
16 extended_emu_network_size_2.reactions.R1.build_reaction_from_string(
17     "B_3 + C_1 --> D_2_3"
18 )
19 extended_emu_network_size_2.reactions.R2.build_reaction_from_string("I")
20 extended_emu_network_size_2.reactions.R3.build_reaction_from_string("E")
21 extended_emu_network_size_2.reactions.R4.build_reaction_from_string("R")
22 extended_emu_network_size_2.reactions.R5.build_reaction_from_string(
23     "D_2_3 --> D_2_3_ext"
24 )
25 extended_emu_network_size_2.reactions.R6.build_reaction_from_string(
26     "B_2_3 --> B_2_3_ext"
27 )
28
29
30 # cobra.io.save_json_model(emu_network, "figure3_combined_emu_network.json")
31 EMU2 = get_stoichiometric_matrix(extended_emu_network_size_2).astype(int)
32 display(
33     flux_map(
34         # cobra.io.load_json_model("figure1_simple_metabolic_network.json")
35         extended_emu_network_size_2,
36         display_name_format=lambda x: str(x.id),
37         figsize=(800, 600),
38         flux_dict=dict(R1=v5, R2=v3, R3=v2, R4=v1, R5=v5 + v2 - v3, R6=0)
39     )
40 )

```

unknown metabolite 'D_2_3_ext' created
 unknown metabolite 'B_2_3_ext' created

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

Reaction network for EMU size 2



3.2.1 EMU size 2 balance equations

$$(v_5 + v_2)D_{23} = v_2 B_{23} + v_5 B_3 \times C_1 \quad (12)$$

$$(v_1 + v_3)B_{23} = v_3 D_{23} + v_1 A_{23} \quad (13)$$

3.2.2 EMU size 2 matrix equation

$$\begin{bmatrix} -v_5 - v_2 & v_2 \\ v_3 & -v_1 - v_3 \end{bmatrix} \begin{bmatrix} D_{23} \\ B_{23} \end{bmatrix} = \begin{bmatrix} -v_5 & 0 \\ 0 & -v_1 \end{bmatrix} \begin{bmatrix} B_3 \times C_1 \\ A_{23} \end{bmatrix}, \quad (2b)$$

3.2.3 Expansion of $D_{2,3}$ and $B_{2,3}$

$$\begin{bmatrix} D_{23} \\ B_{23} \end{bmatrix} = \begin{bmatrix} D_{23,M+0} & D_{23,M+1} & D_{23,M+2} \\ B_{23,M+0} & B_{23,M+1} & B_{23,M+2} \end{bmatrix}, \quad (3)$$

3.2.4 Expansion of $B_3 \times C_1$ and $A_{2,3}$

$$\begin{bmatrix} B_3 \times C_1 \\ A_{23} \end{bmatrix} = \begin{bmatrix} B_{3,M+0}C_{1,M+0} & (B_{3,M+0}C_{1,M+1} + B_{3,M+1}C_{1,M+0}) & B_{3,M+1}C_{1,M+1} \\ A_{23,M+0} & A_{23,M+1} & A_{23,M+2} \end{bmatrix}. \quad (4)$$

3.2.5 EMU size 2 solution

Solution of EMU balances for reaction network of EMU size 2:

$$\begin{bmatrix} D_{23} \\ B_{23} \end{bmatrix} = \begin{bmatrix} -130 & 110 \\ 50 & -150 \end{bmatrix}^{-1} \begin{bmatrix} -20 & 0 \\ 0 & -100 \end{bmatrix} \begin{bmatrix} 0.0622 & 0.8756 & 0.0622 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0.0133 & 0.9733 & 0.0133 \\ 0.0044 & 0.9911 & 0.0044 \end{bmatrix}.$$

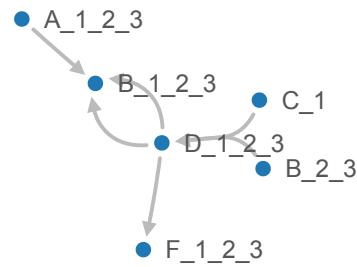
3.3 Reaction network for EMU size 3

In [93]:

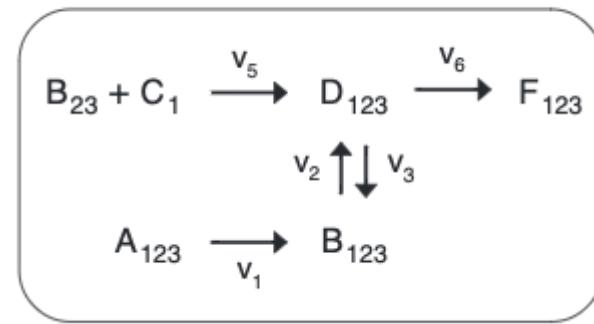
```

1 extended_emu_network_size_3 = Model("extended_emu_network_size_3")
2
3 A_1_2_3 = Metabolite("A_1_2_3")
4 B_2_3 = Metabolite("B_2_3")
5 B_1_2_3 = Metabolite("B_1_2_3")
6 C_1 = Metabolite("C_1")
7 D_1_2_3 = Metabolite("D_1_2_3")
8 F_1_2_3 = Metabolite("F_1_2_3")
9
10 rxns = [Reaction(f"R{i+1}") for i in range(5)]
11
12
13
14 extended_emu_network_size_3.add_metabolites(
15     [A_1_2_3, B_2_3, B_1_2_3, C_1, D_1_2_3, F_1_2_3]
16 )
17 extended_emu_network_size_3.add_reactions(rxns)
18
19 extended_emu_network_size_3.reactions.R1.build_reaction_from_string(
20     "B_2_3 + C_1 --> D_1_2_3"
21 )
22 extended_emu_network_size_3.reactions.R2.build_reaction_from_string(
23     "D_1_2_3 --> B_1_2_3"
24 )
25 extended_emu_network_size_3.reactions.R3.build_reaction_from_string(
26     "B_1_2_3 --> D_1_2_3"
27 )
28 extended_emu_network_size_3.reactions.R3.build_reaction_from_string(
29     "D_1_2_3 --> B_1_2_3"
30 )
31 extended_emu_network_size_3.reactions.R4.build_reaction_from_string(
32     "A_1_2_3 --> B_1_2_3"
33 )
34 extended_emu_network_size_3.reactions.R5.build_reaction_from_string(
35     "D_1_2_3 --> F_1_2_3"
36 )
37
38 # cobra.io.save_json_model(emu_network, "figure3_combined_emu_network")
39 EMU3 = get_stoichiometric_matrix(extended_emu_network_size_3).astype(:)
40 display(
41     flux_map(
42         # cobra.io.load_json_model("figure1_simple_metabolic_network"),
43         extended_emu_network_size_3,
44         display_name_format=lambda x: str(x.id),
45         figsize=(800, 600),
46     )
47 )

```

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

Reaction network for EMU size 3



3.3.1 EMU size 3 balance equations

$$(v_2 + v_5)D_{1,2,3} = v_2 B_{1,2,3} + v_5 B_{2,3} \times C_1 \quad (14)$$

$$(v_1 + v_3)B_{1,2,3} = v_1 A_{1,2,3} + v_3 D_{1,2,3} \quad (15)$$

$$v_6 F_{1,2,3} = D_{1,2,3} \quad (16)$$

3.3.2 EMU size 3 Matrix equation

$$\begin{bmatrix} -v_6 & v_6 & 0 \\ 0 & -v_5 - v_2 & v_2 \\ 0 & v_3 & -v_1 - v_3 \end{bmatrix} \begin{bmatrix} F_{123} \\ D_{123} \\ B_{123} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -v_5 & 0 \\ 0 & -v_1 \end{bmatrix} \begin{bmatrix} B_{23} \times C_1 \\ A_{123} \end{bmatrix}. \quad (2c)$$

3.3.3 EMU size 3 solution

Solution of EMU balances for reaction network of EMU size 3

$$\begin{bmatrix} F_{123} \\ D_{123} \\ B_{123} \end{bmatrix} = \begin{bmatrix} -80 & 80 & 0 \\ 0 & -130 & 110 \\ 0 & 50 & -150 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ -20 & 0 \\ 0 & -100 \end{bmatrix} \begin{bmatrix} 0.0003 & 0.0702 & 0.9253 & 0.0041 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.0001 & 0.8008 & 0.1983 & 0.0009 \\ 0.0001 & 0.8008 & 0.1983 & 0.0009 \\ 0.0000 & 0.9336 & 0.0661 & 0.0003 \end{bmatrix}.$$

3.4 Combined EMU transition network of extended metabolic reaction network

In [60]:

```

1 display(Markdown("## Figure 7\n"))
2 display(Image("Figure7.png"))

```

Figure 7

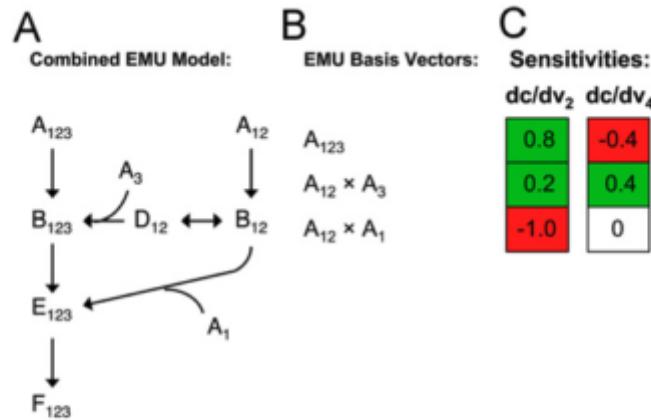


Fig. 7. (A) Combined EMU reaction network for the extended model, showing the flow of isotopic labeling from substrate A to product E. (B) EMU basis vectors for the extended network model. (C) Coefficient sensitivities ($\times 10^2$) with respect to free fluxes, v_2 and v_4 . If A was chosen such that $A_{123}=A_{12} \times A_3$, functionality with respect to v_4 was lost.

In [94]:

```

1  emu_network = Model('figure7_combined_EMU_network')
2  A_1_2_3 = Metabolite('A_1_2_3')
3  A_1_2 = Metabolite('A_1_2')
4  A_3 = Metabolite('A_3')
5  A_1 = Metabolite('A_1')
6  B_1_2_3 = Metabolite('B_1_2_3')
7  D_1_2 = Metabolite('D_1_2')
8  B_1_2 = Metabolite('B_1_2')
9  E_1_2_3 = Metabolite('E_1_2_3')
10 F_1_2_3 = Metabolite('F_1_2_3')
11
12
13 R1 = Reaction('R1')
14 R2 = Reaction('R2')
15 R3 = Reaction('R3')
16 R4 = Reaction('R4')
17 R5 = Reaction('R5')
18 R6 = Reaction('R6')
19 R7 = Reaction('R7')
20 R8 = Reaction('R8')
21 R9 = Reaction('R9')
22 R10 = Reaction('R10')
23 R11 = Reaction('R11')
24 R12 = Reaction('R12')
25
26 emu_network.add_metabolites([A_1, A_1_2_3, A_1_2, A_3, B_1_2_3, B_1_2,
27 emu_network.add_reactions([R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11,
28
29 emu_network.reactions.R1.build_reaction_from_string('A_1_2_3 --> B_1_2')
30 emu_network.reactions.R2.build_reaction_from_string('B_1_2_3 --> E_1_2')
31 emu_network.reactions.R3.build_reaction_from_string('A_3 + D_1_2 --> E_1_2')
32 emu_network.reactions.R4.build_reaction_from_string('B_1_2 <--> D_1_2')
33 emu_network.reactions.R5.build_reaction_from_string('A_1_2 --> B_1_2')
34 emu_network.reactions.R6.build_reaction_from_string('--> A_1_2_3')
35 emu_network.reactions.R7.build_reaction_from_string('--> A_1_2')
36 emu_network.reactions.R8.build_reaction_from_string('--> A_3')
37 emu_network.reactions.R9.build_reaction_from_string('E_1_2_3 --> F_1_2')
38 emu_network.reactions.R10.build_reaction_from_string('B_1_2 + A_1 --> F_1_2')
39 emu_network.reactions.R11.build_reaction_from_string('F_1_2_3 --> ')
40 emu_network.reactions.R12.build_reaction_from_string('--> A_1')
41
42 S = get_stoichiometric_matrix(emu_network)
43 rxns, mets = S.columns, S.index
44 external_mets = ['A_1', 'A_3', 'A_1_2', 'A_1_2_3', 'F_1_2_3']
45 external_rxns = ['R6', 'R7', 'R8', 'R11', 'R12']
46 output_met = 'F_1_2_3'
47 output_rxn = 'R11'
48 ebv, emu_elmo = get_EMU_basis_vectors_and_pathways(emu_network, external_mets,
49
50 #cobra.io.save_json_model(emu_network, "figure7_combined_EMU_network.json")
51
52 display(Markdown('## Combined EMU network\n'))
53 display(flux_map(cobra.io.load_json_model("figure7_combined_EMU_network.json"),
54                  flux_dict={rxn.id: emu_elmo.sum(axis=0)[rxn.id] for rxn in emu_network.reactions}))
55 display(Markdown('## EMU basis vectors for $F_{123}$\n'))
56

```

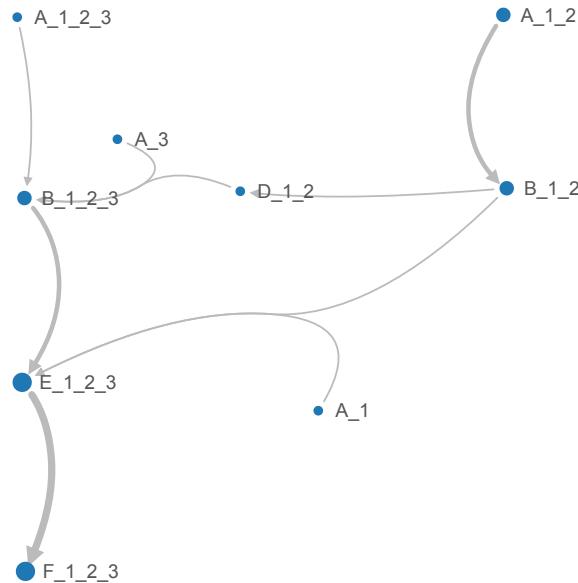
```

57 display(ebv)
58
59
60 display(Markdown('## EMU Pathways \n'))
61 emu_elmo[ 'EMU Basis Vector' ] = ebv[ 'EMU Basis Vector' ]
62
63 display(emu_elmo)
64
▼ 65 for em in emu_elmo.index:
▼ 66     with emu_network:
67         emu_network.remove_reactions([r for r in emu_elmo.columns if e
68         display(Markdown('### EMU Pathway {} ('.format(em) + ebv.loc[e
▼ 69         display(flux_map(emu_network,
70                 display_name_format=lambda x: str(x.id),
71                 flux_dict={rxn.id: emu_elmo.loc[em,rxn.id]
72                         for rxn in emu_network.reactions})))
73

```

/Users/zuck016/.pyenv/versions/anaconda3-2020.11/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering (buffering=1) isn't supported in binary mode, the default buffer size will be used

Combined EMU network


[Save JSON](#)
[Show Reaction Nodes](#)
[Download SVG](#)

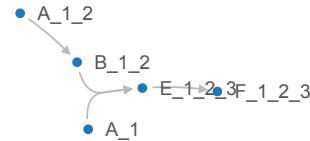
EMU basis vectors for F_{123}

	A_1	A_3	A_1_2	A_1_2_3	F_1_2_3	EMU Basis Vector
EM1	-1.0	-0.0	-1.0	-0.0	1.0	$A_1 \times A_{1,2}$
EM2	-0.0	-1.0	-1.0	-0.0	1.0	$A_3 \times A_{1,2}$
EM3	-0.0	-0.0	-0.0	-1.0	1.0	$A_{1,2,3}$

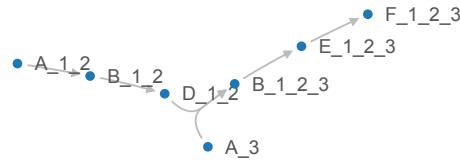
EMU Pathways

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	EMU Basis Vector
EM1	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	$A_1 \times A_{1,2}$
EM2	0.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	$A_3 \times A_{1,2}$
EM3	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	$A_{1,2,3}$

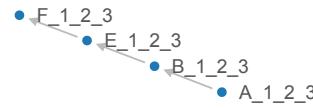
EMU Pathway EM1 ($A_1 \times A_{1,2}$)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

EMU Pathway EM2 ($A_3 \times A_{1,2}$)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

EMU Pathway EM3 ($A_{1,2,3}$)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

3.5 Tracer Compositions for Extended EMU network

Given all possible 2 tracer compositions of 3 atoms

	A_{000}	A_{100}	A_{010}	A_{001}	A_{110}	A_{101}	A_{011}	A_{111}
A_{000}								
A_{100}								
A_{010}								
A_{001}								

. A_{000} A_{100} A_{010} A_{001} A_{110} A_{101} A_{011} A_{111}

A_{110}

A_{101}

A_{011}

A_{111}

what are the isotopic labeling states and mass isotopomer distributions for each EMU Basis Vector?

Compare the results with Figure 6.

In [21]:

```

1  display(Image("Figure6.png"))
2  num_atoms = 3
3  isotopic_labeling_states = []
4  for i in range(2 ** num_atoms):
5      for j in range(i + 1):
6          if i == j:
7              isotopic_labeling_states.append(
8                  pd.DataFrame(
9                      [{"%": 100, 1: (i) & 1, 2: (i >> 1) & 1, 3: (i >>
10                         2) & 1}], index=[r"$100\%" + r"\ A_{%d%d\$}" % ((i) & 1, (i >> 1) & 1, (i >> 2) & 1)])
11      else:
12          isotopic_labeling_states.append(
13              pd.DataFrame([
14                  {"%": 50, 1: (i) & 1, 2: (i >> 1) & 1, 3: (i >> 2) & 1},
15                  {"%": 50, 1: (j) & 1, 2: (j >> 1) & 1, 3: (j >> 2) & 1}], index=[r"$50\%" + r"\ A_{%d%d\$}" % ((i) & 1, (i >> 1) & 1, (i >> 2) & 1) + r"$50\%" + r"\ A_{%d%d\$}" % ((j) & 1, (j >> 1) & 1, (j >> 2) & 1)])
16      )
17  )
18  else:
19      isotopic_labeling_states.append(
20          pd.DataFrame([
21              {"%": 50, 1: (i) & 1, 2: (i >> 1) & 1, 3: (i >> 2) & 1},
22              {"%": 50, 1: (j) & 1, 2: (j >> 1) & 1, 3: (j >> 2) & 1}], index=[r"$50\%" + r"\ A_{%d%d\$}" % ((i) & 1, (i >> 1) & 1, (i >> 2) & 1) + r"$50\%" + r"\ A_{%d%d\$}" % ((j) & 1, (j >> 1) & 1, (j >> 2) & 1)])
23  )
24  )
25  )
26  )
27  )
28  )
29  )
30  )
31  )
32  tracer_mixtures = [
33      dict(A=Tracer("A", num_atoms, ils)) for ils in isotopic_labeling_states]
34  ]
35  emu_basis_vectors = [EMU_Basis_Vector(ebv.iloc[e, :-1]) for e in range(len(emu_basis_vectors))]
36  mid_comparison = {}
37  isl_comparison = {}
38  for tracers in tracer_mixtures:
39      for t in tracers:
40          for emu_basis_vector in emu_basis_vectors:
41              for emu in emu_basis_vector.get_input_emus():
42                  emu_isl = emu.get_isotopic_labeling_state(tracers)
43                  emu_isl.index = pd.MultiIndex.from_tuples([(tracers[t].get_name(), a) for a in emu_isl.index])
44              )
45      for atom in emu_isl.columns:
46          key = (emu_basis_vector.get_name(), emu.get_name())
47          if key in isl_comparison:
48              isl_comparison[key] = isl_comparison[key].append(emu_isl[atom])
49          else:
50              isl_comparison[key] = emu_isl[atom]
51  mid_comparison[
52      tracers[t].get_name(), emu_basis_vector.get_name()
53  ] = emu_basis_vector.get_mid(tracers)
54  )
55  )
56  display(Markdown("### Isotopomer labeling states"))

```

```

57 isl_df = pd.DataFrame(isl_comparison)
58 isl_df.columns = isl_df.columns.set_names(["EMU Basis Vectors", "EMUs"])
59 isl_df.index = isl_df.index.set_names(["Tracer Compositions", "Tracers"])
60 display(HTML(isl_df.to_html()))
61 display(Markdown("### Mass Isotopomer Distribution"))
62
63 display(HTML(pd.DataFrame(mid_comparison).T.to_html()))

```

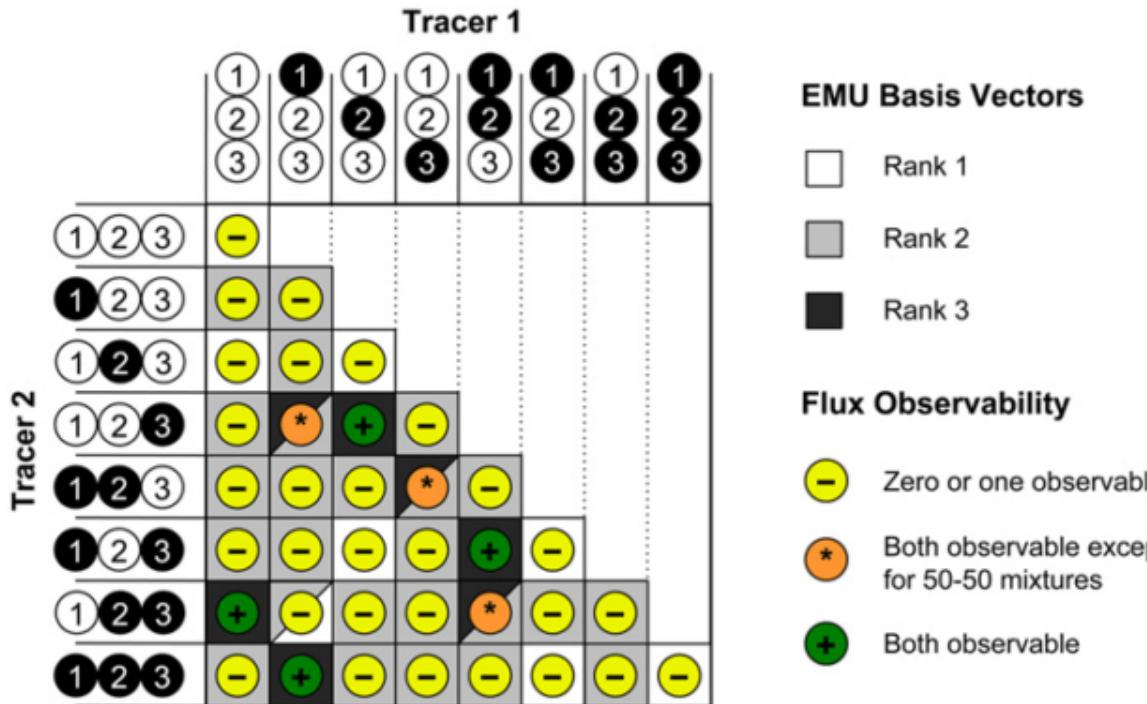


Fig. 6. Analysis of flux observability and EMU basis vector rank in the extended network model. Pure tracers are shown on the diagonal and binary mixtures of tracers are shown on the off-diagonal elements. The square colors denote the EMU basis vector rank (i.e. number of independent EMU basis vectors) and the circles correspond to flux observability. Observability of the free fluxes correlated with the independence of EMU basis vectors. Squares with upper and lower colored triangles denote cases where the EMU basis vector rank was different for 50/50 mixtures of tracers, i.e. compared to 25/75 and 75/25 mixtures.

Isotopomer labeling states

EMU Basis Vectors	$A_{1,2}$ $\times A_1$			$A_{1,2}$ $\times A_3$			$A_{1,2,3}$					
EMUs	$A_{1,2}$		A_1	$A_{1,2}$		A_3	$A_{1,2,3}$					
Atoms	1	2	1				1	2	3	1	2	3
Tracer Compositions												
Tracers												
100% A_{000}	100% A_{000}		0	0		0	0	0		0	0	0
50% A_{100} and 50% A_{000}	50% A_{100}		1	0		1	1	0		0	1	0
	50% A_{000}		0	0		0	0	0		0	0	0

EMU Basis Vectors	$A_{1,2}$ $\times A_1$		$A_{1,2}$ $\times A_3$			$A_{1,2,3}$					
EMUs	$A_{1,2}$		A_1		$A_{1,2}$		A_3		$A_{1,2,3}$		
Atoms	1	2	1		1	2	3		1	2	3
Tracer Compositions											
Tracers											
100% A_{100}	100% A_{100}	1	0		1	1	0		0	1	0
50% A_{010} and	50% A_{010}	0	1		0	0	1		0	0	1
50% A_{000}	50% A_{000}	0	0		0	0	0		0	0	0
50% A_{010} and	50% A_{010}	0	1		0	0	1		0	0	1
50% A_{100}	50% A_{100}	1	0		1	1	0		0	1	0
100% A_{010}	100% A_{010}	0	1		0	0	1		0	0	1
50% A_{110} and	50% A_{110}	1	1		1	1	1		0	1	1
50% A_{000}	50% A_{000}	0	0		0	0	0		0	0	0
50% A_{110} and	50% A_{110}	1	1		1	1	1		0	1	1
50% A_{100}	50% A_{100}	1	0		1	1	0		0	1	0
50% A_{110} and	50% A_{110}	1	1		1	1	1		0	1	1
50% A_{010}	50% A_{010}	0	1		0	0	1		0	0	1
100% A_{110}	100% A_{110}	1	1		1	1	1		0	1	1
50% A_{001} and	50% A_{001}	0	0		0	0	0		1	0	0
50% A_{000}	50% A_{000}	0	0		0	0	0		0	0	0
50% A_{001} and	50% A_{001}	0	0		0	0	0		1	0	0
50% A_{100}	50% A_{100}	1	0		1	1	0		0	1	0
50% A_{001} and	50% A_{001}	0	0		0	0	0		1	0	0
50% A_{010}	50% A_{010}	0	1		0	0	1		0	0	1
50% A_{001} and	50% A_{001}	0	0		0	0	0		1	0	0
50% A_{110}	50% A_{110}	1	1		1	1	1		0	1	1
100% A_{001}	100% A_{001}	0	0		0	0	0		1	0	0
50% A_{101} and	50% A_{101}	1	0		1	1	0		1	1	0
50% A_{000}	50% A_{000}	0	0		0	0	0		0	0	0
50% A_{101} and	50% A_{101}	1	0		1	1	0		1	1	0
50% A_{100}	50% A_{100}	1	0		1	1	0		0	1	0
50% A_{101}	50% A_{101}	1	0		1	1	0		1	1	0

EMU Basis Vectors	$A_{1,2}$ $\times A_1$			$A_{1,2}$ $\times A_3$			$A_{1,2,3}$
EMUs	$A_{1,2}$		A_1	$A_{1,2}$		A_3	$A_{1,2,3}$
Atoms	1	2	1	1	2	3	1 2 3
Tracer Compositions							
Tracers							
and 50% A_{010}	50% A_{010}	0	1	0	0	1	0 0 1 0
50% A_{101} and 50% A_{110}	50% A_{101}	1	0	1	1	0	1 1 0 1
50% A_{101} and 50% A_{001}	50% A_{110}	1	1	1	1	1	0 1 1 0
100% A_{101}	50% A_{101}	1	0	1	1	0	1 1 0 1
50% A_{011} and 50% A_{000}	50% A_{001}	0	0	0	0	0	0 0 0 0
50% A_{011} and 50% A_{100}	100% A_{101}	1	0	1	1	0	1 1 0 1
50% A_{011} and 50% A_{010}	50% A_{011}	0	1	0	0	1	1 0 1 1
50% A_{011} and 50% A_{110}	50% A_{000}	0	0	0	0	0	0 0 0 0
50% A_{011} and 50% A_{100}	50% A_{011}	0	1	0	0	1	1 0 1 1
50% A_{011} and 50% A_{010}	50% A_{100}	1	0	1	1	0	0 1 0 0
50% A_{011} and 50% A_{010}	50% A_{011}	0	1	0	0	1	1 0 1 1
50% A_{011} and 50% A_{110}	50% A_{010}	0	1	0	0	1	0 0 1 0
50% A_{011} and 50% A_{110}	50% A_{011}	0	1	0	0	1	1 0 1 1
50% A_{011} and 50% A_{001}	50% A_{110}	1	1	1	1	1	0 1 1 0
50% A_{011} and 50% A_{001}	50% A_{011}	0	1	0	0	1	1 0 1 1
50% A_{011} and 50% A_{101}	50% A_{001}	0	0	0	0	0	0 0 0 0
100% A_{011}	50% A_{011}	0	1	0	0	1	1 0 1 1
50% A_{111} and 50% A_{000}	50% A_{101}	1	0	1	1	0	0 1 0 0
50% A_{111} and 50% A_{100}	50% A_{111}	1	1	1	1	1	1 1 1 1
50% A_{111} and 50% A_{010}	50% A_{100}	1	0	1	1	0	0 1 0 0
50% A_{111} and 50% A_{010}	50% A_{111}	1	1	1	1	1	1 1 1 1
50% A_{111} and 50% A_{111}	50% A_{010}	0	1	0	0	1	0 0 1 0
50% A_{111} and 50% A_{111}	50% A_{111}	1	1	1	1	1	1 1 1 1

EMU Basis Vectors	$A_{1,2}$ $\times A_1$			$A_{1,2}$ $\times A_3$			$A_{1,2,3}$			
EMUs	$A_{1,2}$		A_1	$A_{1,2}$		A_3	$A_{1,2,3}$			
Atoms	1	2	1	1	2	3	1	2	3	
Tracer Compositions										
Tracers										
50% A_{110}	50% A_{110}	1	1	1	1	1	0	1	1	
50% A_{111} and	50% A_{111}	1	1	1	1	1	1	1	1	
50% A_{001}	50% A_{001}	0	0	0	0	0	0	0	0	
50% A_{111} and	50% A_{111}	1	1	1	1	1	1	1	1	
50% A_{101}	50% A_{101}	1	0	1	1	0	0	1	0	
50% A_{111} and	50% A_{111}	1	1	1	1	1	1	1	1	
50% A_{011}	50% A_{011}	0	1	0	0	1	0	0	1	
100% A_{111}	100% A_{111}	1	1	1	1	1	1	1	1	

Mass Isotopomer Distribution

			M+0	M+1	M+2	M+3
100% A_{000}		$A_{1,2} \times A_1$	100.0	0.0	0.0	0.0
		$A_{1,2} \times A_3$	100.0	0.0	0.0	0.0
		$A_{1,2,3}$	100.0	0.0	0.0	0.0
50% A_{100}	and	$A_{1,2} \times A_1$	25.0	50.0	25.0	0.0
50% A_{000}		$A_{1,2} \times A_3$	50.0	50.0	0.0	0.0
		$A_{1,2,3}$	50.0	50.0	0.0	0.0
100% A_{100}		$A_{1,2} \times A_1$	0.0	0.0	100.0	0.0
		$A_{1,2} \times A_3$	0.0	100.0	0.0	0.0
		$A_{1,2,3}$	0.0	100.0	0.0	0.0
50% A_{010}	and	$A_{1,2} \times A_1$	50.0	50.0	0.0	0.0
50% A_{000}		$A_{1,2} \times A_3$	50.0	50.0	0.0	0.0
		$A_{1,2,3}$	50.0	50.0	0.0	0.0
50% A_{010}	and	$A_{1,2} \times A_1$	0.0	50.0	50.0	0.0
50% A_{100}		$A_{1,2} \times A_3$	0.0	100.0	0.0	0.0
		$A_{1,2,3}$	0.0	100.0	0.0	0.0
100% A_{010}		$A_{1,2} \times A_1$	0.0	100.0	0.0	0.0
		$A_{1,2} \times A_3$	0.0	100.0	0.0	0.0

			M+0	M+1	M+2	M+3
		A _{1,2,3}	0.0	100.0	0.0	0.0
50% A ₁₁₀	and	A _{1,2} × A ₁	25.0	25.0	25.0	25.0
50% A ₀₀₀		A _{1,2} × A ₃	50.0	0.0	50.0	0.0
		A _{1,2,3}	50.0	0.0	50.0	0.0
50% A ₁₁₀	and	A _{1,2} × A ₁	0.0	0.0	50.0	50.0
50% A ₁₀₀		A _{1,2} × A ₃	0.0	50.0	50.0	0.0
		A _{1,2,3}	0.0	50.0	50.0	0.0
50% A ₁₁₀	and	A _{1,2} × A ₁	0.0	25.0	50.0	25.0
50% A ₀₁₀		A _{1,2} × A ₃	0.0	50.0	50.0	0.0
		A _{1,2,3}	0.0	50.0	50.0	0.0
100% A ₁₁₀		A _{1,2} × A ₁	0.0	0.0	0.0	100.0
		A _{1,2} × A ₃	0.0	0.0	100.0	0.0
		A _{1,2,3}	0.0	0.0	100.0	0.0
50% A ₀₀₁	and	A _{1,2} × A ₁	100.0	0.0	0.0	0.0
50% A ₀₀₀		A _{1,2} × A ₃	50.0	50.0	0.0	0.0
		A _{1,2,3}	50.0	50.0	0.0	0.0
50% A ₀₀₁	and	A _{1,2} × A ₁	25.0	50.0	25.0	0.0
50% A ₁₀₀		A _{1,2} × A ₃	25.0	50.0	25.0	0.0
		A _{1,2,3}	0.0	100.0	0.0	0.0
50% A ₀₀₁	and	A _{1,2} × A ₁	50.0	50.0	0.0	0.0
50% A ₀₁₀		A _{1,2} × A ₃	25.0	50.0	25.0	0.0
		A _{1,2,3}	0.0	100.0	0.0	0.0
50% A ₀₀₁	and	A _{1,2} × A ₁	25.0	25.0	25.0	25.0
50% A ₁₁₀		A _{1,2} × A ₃	25.0	25.0	25.0	25.0
		A _{1,2,3}	0.0	50.0	50.0	0.0
100% A ₀₀₁		A _{1,2} × A ₁	100.0	0.0	0.0	0.0
		A _{1,2} × A ₃	0.0	100.0	0.0	0.0
		A _{1,2,3}	0.0	100.0	0.0	0.0
50% A ₁₀₁	and	A _{1,2} × A ₁	25.0	50.0	25.0	0.0
50% A ₀₀₀		A _{1,2} × A ₃	25.0	50.0	25.0	0.0
		A _{1,2,3}	50.0	0.0	50.0	0.0
50% A ₁₀₁	and	A _{1,2} × A ₁	0.0	0.0	100.0	0.0
50% A ₁₀₀						

			M+0	M+1	M+2	M+3
		$A_{1,2} \times A_3$	0.0	50.0	50.0	0.0
		$A_{1,2,3}$	0.0	50.0	50.0	0.0
50% A_{101}	and	$A_{1,2} \times A_1$	0.0	50.0	50.0	0.0
50% A_{010}		$A_{1,2} \times A_3$	0.0	50.0	50.0	0.0
		$A_{1,2,3}$	0.0	50.0	50.0	0.0
50% A_{101}	and	$A_{1,2} \times A_1$	0.0	0.0	50.0	50.0
50% A_{110}		$A_{1,2} \times A_3$	0.0	25.0	50.0	25.0
		$A_{1,2,3}$	0.0	0.0	100.0	0.0
50% A_{101}	and	$A_{1,2} \times A_1$	25.0	50.0	25.0	0.0
50% A_{001}		$A_{1,2} \times A_3$	25.0	50.0	25.0	0.0
		$A_{1,2,3}$	50.0	0.0	50.0	0.0
100% A_{101}		$A_{1,2} \times A_1$	0.0	0.0	100.0	0.0
		$A_{1,2} \times A_3$	0.0	0.0	100.0	0.0
		$A_{1,2,3}$	0.0	0.0	100.0	0.0
50% A_{011}	and	$A_{1,2} \times A_1$	50.0	50.0	0.0	0.0
50% A_{000}		$A_{1,2} \times A_3$	25.0	50.0	25.0	0.0
		$A_{1,2,3}$	50.0	0.0	50.0	0.0
50% A_{011}	and	$A_{1,2} \times A_1$	0.0	50.0	50.0	0.0
50% A_{100}		$A_{1,2} \times A_3$	0.0	50.0	50.0	0.0
		$A_{1,2,3}$	0.0	50.0	50.0	0.0
50% A_{011}	and	$A_{1,2} \times A_1$	0.0	100.0	0.0	0.0
50% A_{010}		$A_{1,2} \times A_3$	0.0	50.0	50.0	0.0
		$A_{1,2,3}$	0.0	50.0	50.0	0.0
50% A_{011}	and	$A_{1,2} \times A_1$	0.0	25.0	50.0	25.0
50% A_{110}		$A_{1,2} \times A_3$	0.0	25.0	50.0	25.0
		$A_{1,2,3}$	0.0	0.0	100.0	0.0
50% A_{011}	and	$A_{1,2} \times A_1$	50.0	50.0	0.0	0.0
50% A_{001}		$A_{1,2} \times A_3$	25.0	50.0	25.0	0.0
		$A_{1,2,3}$	50.0	0.0	50.0	0.0
50% A_{011}	and	$A_{1,2} \times A_1$	0.0	50.0	50.0	0.0
50% A_{101}		$A_{1,2} \times A_3$	0.0	50.0	50.0	0.0
		$A_{1,2,3}$	0.0	50.0	50.0	0.0

			M+0	M+1	M+2	M+3
100% A_{011}		$A_{1,2} \times A_1$	0.0	100.0	0.0	0.0
		$A_{1,2} \times A_3$	0.0	0.0	100.0	0.0
		$A_{1,2,3}$	0.0	0.0	100.0	0.0
50% A_{111}	and	$A_{1,2} \times A_1$	25.0	25.0	25.0	25.0
		$A_{1,2} \times A_3$	25.0	25.0	25.0	25.0
		$A_{1,2,3}$	50.0	0.0	0.0	50.0
50% A_{111}	and	$A_{1,2} \times A_1$	0.0	0.0	50.0	50.0
		$A_{1,2} \times A_3$	0.0	25.0	50.0	25.0
		$A_{1,2,3}$	0.0	50.0	0.0	50.0
50% A_{111}	and	$A_{1,2} \times A_1$	0.0	25.0	50.0	25.0
		$A_{1,2} \times A_3$	0.0	25.0	50.0	25.0
		$A_{1,2,3}$	0.0	50.0	0.0	50.0
50% A_{111}	and	$A_{1,2} \times A_1$	0.0	0.0	0.0	100.0
		$A_{1,2} \times A_3$	0.0	0.0	50.0	50.0
		$A_{1,2,3}$	0.0	0.0	50.0	50.0
50% A_{111}	and	$A_{1,2} \times A_1$	25.0	25.0	25.0	25.0
		$A_{1,2} \times A_3$	25.0	25.0	25.0	25.0
		$A_{1,2,3}$	50.0	0.0	0.0	50.0
50% A_{111}	and	$A_{1,2} \times A_1$	0.0	0.0	50.0	50.0
		$A_{1,2} \times A_3$	0.0	25.0	50.0	25.0
		$A_{1,2,3}$	0.0	50.0	0.0	50.0
50% A_{111}	and	$A_{1,2} \times A_1$	0.0	0.0	50.0	50.0
		$A_{1,2} \times A_3$	0.0	25.0	50.0	25.0
		$A_{1,2,3}$	0.0	50.0	0.0	50.0
100% A_{111}		$A_{1,2} \times A_1$	0.0	0.0	0.0	100.0
		$A_{1,2} \times A_3$	0.0	0.0	0.0	100.0
		$A_{1,2,3}$	0.0	0.0	0.0	100.0

4 TCA Cycle metabolic model

```
In [22]: 1 display(Image('TCA.png'))
2
```

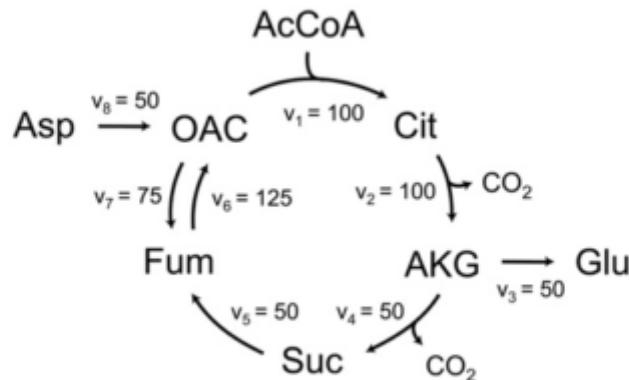


Fig. 10. Simplified model of the TCA cycle. The assumed fluxes have arbitrary units.

Table 3

Stoichiometry and atom transitions for the reactions in the TCA cycle model.

Reaction number	Reaction stoichiometry	Carbon atom transitions
1	OAC + AcCoA → Cit	abcd + ef → dcbafe
2	Cit → AKG + CO ₂	abcdef → abcde + f
3	AKG → Glu	abcde → abcde
4	AKG → Suc + CO ₂	abcde → ½ bcde + ½ edcb + a
5	Suc → Fum	½ abcd + ½ dcba → ½ abcd + ½ dcba
6	Fum → OAC	½ abcd + ½ dcba → abcd
7	OAC → Fum	abcd → ½ abcd + ½ dcba
8	Asp → OAC	abcd → abcd

4.1 TCA Cycle metabolic reaction network

In [23]:

```

1 tca_metabolic_network = Model('TCA_Metabolic_Network')
2 Asp = Metabolite('Asp')
3 OAC = Metabolite('OAC')
4 AcCoA = Metabolite('AcCoA')
5 Cit = Metabolite('Cit')
6 CO2 = Metabolite('CO2')
7 AKG = Metabolite('AKG')
8 Glu = Metabolite('Glu')
9 Suc = Metabolite('Suc')
10 Fum = Metabolite('Fum')
11
12
13 v1 = Reaction('v1')
14 v2 = Reaction('v2')
15 v3 = Reaction('v3')
16 v4 = Reaction('v4')
17 v5 = Reaction('v5')
18 v6 = Reaction('v6')
19 v7 = Reaction('v7')
20 v8 = Reaction('v8')
21 b1 = Reaction('b1')
22 b2 = Reaction('b2')
23 b3 = Reaction('b3')
24 tca_metabolic_network.add_metabolites([Asp,OAC,AcCoA,Cit,CO2,AKG,Glu,S])
25 tca_metabolic_network.add_reactions([v1,v2,v3,v4,v5,v6,v7,v8,b1,b2,b3])
26
27 tca_metabolic_network.reactions.v1.build_reaction_from_string('OAC + A
28 tca_metabolic_network.reactions.v2.build_reaction_from_string('Cit -->
29 tca_metabolic_network.reactions.v3.build_reaction_from_string('AKG -->
30 tca_metabolic_network.reactions.v4.build_reaction_from_string('AKG -->
31 tca_metabolic_network.reactions.v5.build_reaction_from_string('Suc -->
32 tca_metabolic_network.reactions.v6.build_reaction_from_string('Fum -->
33 tca_metabolic_network.reactions.v7.build_reaction_from_string('OAC -->
34 tca_metabolic_network.reactions.v8.build_reaction_from_string('Asp -->
35
36 tca_metabolic_network.reactions.b1.build_reaction_from_string(' --> Asp
37 tca_metabolic_network.reactions.b2.build_reaction_from_string(' --> AcC
38 tca_metabolic_network.reactions.b3.build_reaction_from_string('Glu -->
39 tca_metabolic_network.reactions.b4.build_reaction_from_string('CO2 -->
40
41 tca_metabolic_network.reactions.v1.notes['CARBON TRANSITIONS'] = 'OAC
42 tca_metabolic_network.reactions.v2.notes['CARBON TRANSITIONS'] = 'CIT
43 tca_metabolic_network.reactions.v3.notes['CARBON TRANSITIONS'] = 'AKG
44 tca_metabolic_network.reactions.v4.notes['CARBON TRANSITIONS'] = 'AKG
45 tca_metabolic_network.reactions.v5.notes['CARBON TRANSITIONS'] = 'Suc
46 tca_metabolic_network.reactions.v6.notes['CARBON TRANSITIONS'] = 'Fum
47 tca_metabolic_network.reactions.v7.notes['CARBON TRANSITIONS'] = 'OAC
48 tca_metabolic_network.reactions.v8.notes['CARBON TRANSITIONS'] = 'Asp
49
50 S = get_stoichiometric_matrix( tca_metabolic_network )
51 rxns, mets = S.columns, S.index
52
53
54 #display(Markdown('### Nullspace of Extended Metabolic Reaction Network'))
55 #display(get_nullspace( S ).T)
56 display(Markdown('### Elementary modes of Extended Metabolic Reaction

```

```

57 elmo = calculate_elementary_modes( tca_metabolic_network, verbose=False)
58 display(elmo)
59 display(flux_map(tca_metabolic_network, display_name_format=lambda x:
60     flux_dict={rxn.id: elmo.sum(axis=0)[rxn.id]} for rxn in tca_me
61
62 for em in elmo.index:
63     with tca_metabolic_network:
64         tca_metabolic_network.remove_reactions([r for r in elmo.columns])
65         display(Markdown('### Elementary mode {} '.format(em)))
66         display(flux_map(tca_metabolic_network,
67                         display_name_format=lambda x: str(x.id),
68                         flux_dict={rxn.id: elmo.loc[em,rxn.id]
69                                     for rxn in tca_metabolic_network.reactions})
70

```

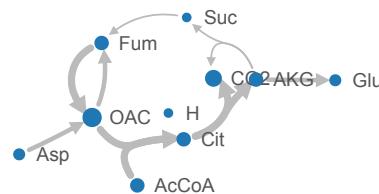
Elementary modes of Extended Metabolic Reaction Network

```

/Users/zuck016/.pyenv/versions/anaconda3-2020.11/lib/python3.8/subprocess
s.py:849: RuntimeWarning: line buffering (buffering=1) isn't supported in
binary mode, the default buffer size will be used
    self.stderr = io.open(errread, 'rb', bufsize)

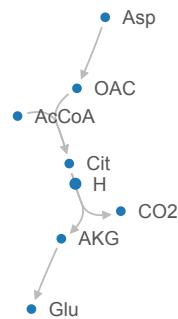
```

	v1	v2	v3	v4	v5	v6	v7	v8	b1	b2	b3	b4
EM1	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0
EM2	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
EM3	0.5	0.5	0.0	0.5	0.5	0.5	0.0	0.0	0.0	0.5	0.0	1.0



[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

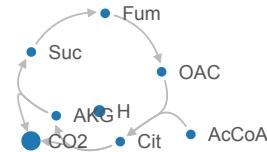
Elementary mode EM1

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

Elementary mode EM2

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

Elementary mode EM3

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

4.2 Combined EMU network for TCA Cycle

In [24]:

```

1
2 display(Markdown('### Figure 12\n'))
3 display(Image('Figure12.png'))
4

```

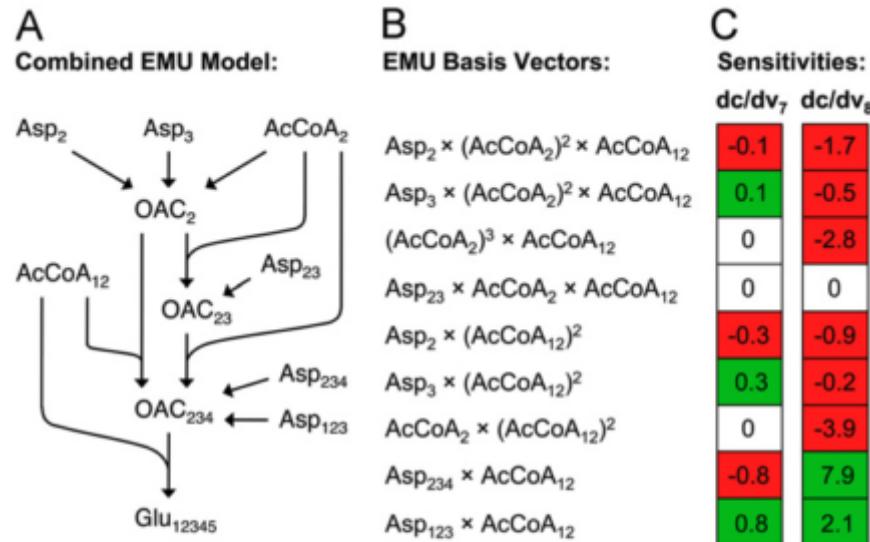
Figure 12

Fig. 12. (A) Combined EMU reaction network for the TCA cycle showing the flow of isotopic labeling from aspartate and acetyl-coenzyme A to glutamate. (B) Nine EMU basis vectors corresponding to the production of glutamate in the TCA cycle. (C) Sensitivity of contribution coefficients ($\times 10^3$) with respect to the free fluxes v_7 and v_8 . In the case of dc/dv_7 , there were three equal but opposite paired sensitivities. As a result, if an aspartate tracer was chosen such that $\text{Asp}_2=\text{Asp}_3$ and $\text{Asp}_{123}=\text{Asp}_{234}$, each pair of sensitivities canceled, and functionality with respect to v_7 was lost.

In [25]:

```

1
2
3 emu_network = Model('figure12_combined_EMU_network')
4 Asp_2 = Metabolite('Asp_2')
5 Asp_3 = Metabolite('Asp_3')
6 Asp_2_3 = Metabolite('Asp_2_3')
7 Asp_2_3_4 = Metabolite('Asp_2_3_4')
8 Asp_1_2_3 = Metabolite('Asp_1_2_3')
9 AcCoA_2 = Metabolite('AcCoA_2')
10 AcCoA_1_2 = Metabolite('AcCoA_1_2')
11 OAC_2 = Metabolite('OAC_2')
12 OAC_2_3_4 = Metabolite('OAC_2_3_4')
13 OAC_2_3 = Metabolite('OAC_2_3')
14 Glu_1_2_3_4_5 = Metabolite('Glu_1_2_3_4_5')
15
16 R1 = Reaction('R1')
17 R2 = Reaction('R2')
18 R3 = Reaction('R3')
19 R4 = Reaction('R4')
20 R5 = Reaction('R5')
21 R6 = Reaction('R6')
22 R7 = Reaction('R7')
23 R8 = Reaction('R8')
24 R9 = Reaction('R9')
25 R10 = Reaction('R10')
26 R11 = Reaction('R11')
27 R12 = Reaction('R12')
28 R13 = Reaction('R13')
29 R14 = Reaction('R14')
30 R15 = Reaction('R15')
31 R16 = Reaction('R16')
32 R17 = Reaction('R17')
33 R18 = Reaction('R18')
34
35 emu_network.add_metabolites(
36     [Asp_2, Asp_3, Asp_2_3, Asp_2_3_4, Asp_1_2_3,
37      AcCoA_2, AcCoA_1_2,
38      OAC_2, OAC_2_3, OAC_2_3_4,
39      Glu_1_2_3_4_5])
40
41 emu_network.add_reactions([R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11,
42
43 emu_network.reactions.R1.build_reaction_from_string('--> Asp_2')
44 emu_network.reactions.R2.build_reaction_from_string('--> Asp_3')
45 emu_network.reactions.R3.build_reaction_from_string('--> Asp_2_3')
46 emu_network.reactions.R4.build_reaction_from_string('--> Asp_2_3_4')
47 emu_network.reactions.R5.build_reaction_from_string('--> Asp_1_2_3')
48 emu_network.reactions.R6.build_reaction_from_string('--> AcCoA_2')
49 emu_network.reactions.R7.build_reaction_from_string('--> AcCoA_1_2')
50 emu_network.reactions.R8.build_reaction_from_string('Asp_2 --> OAC_2')
51 emu_network.reactions.R9.build_reaction_from_string('Asp_3 --> OAC_2')
52 emu_network.reactions.R10.build_reaction_from_string('AcCoA_2 --> OAC_2')
53 emu_network.reactions.R11.build_reaction_from_string('AcCoA_2 + OAC_2')
54 emu_network.reactions.R12.build_reaction_from_string('AcCoA_2 + OAC_2')
55 emu_network.reactions.R13.build_reaction_from_string('Asp_2_3 --> OAC_2')
56 emu_network.reactions.R14.build_reaction_from_string('Asp_2_3_4 --> OAC_2')

```

```

57 emu_network.reactions.R15.build_reaction_from_string('Asp_1_2_3 --> C')
58 emu_network.reactions.R16.build_reaction_from_string('OAC_2 + AcCoA_1')
59 emu_network.reactions.R17.build_reaction_from_string('OAC_2_3_4 + AcC')
60 emu_network.reactions.R18.build_reaction_from_string('Glu_1_2_3_4_5 -')
61
62 S = get_stoichiometric_matrix(emu_network)
63 rxns, mets = S.columns, S.index
64 external_mets = ['Asp_2', 'Asp_3', 'AcCoA_2', 'AcCoA_1_2', 'Asp_2_3',
65 external_rxns = ['R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R18']
66 output_met = 'Glu_1_2_3_4_5'
67 output_rxn = 'R18'
68
69 bv_order = ['$Asp_{2} \times \left(AcCoA_{2}\right)^{2} \times AcCoA_{1,2}$',
70 '$Asp_{3} \times \left(AcCoA_{2}\right)^{2} \times AcCoA_{1,2}$',
71 '$\left(AcCoA_{2}\right)^{3} \times AcCoA_{1,2}$',
72 '$AcCoA_{2} \times AcCoA_{1,2} \times Asp_{2,3}$',
73 '$Asp_{2} \times \left(AcCoA_{1,2}\right)^{2}$',
74 '$Asp_{3} \times \left(AcCoA_{1,2}\right)^{2}$',
75 '$AcCoA_{2} \times \left(AcCoA_{1,2}\right)^{2}$',
76 '$AcCoA_{1,2} \times Asp_{2,3,4}$',
77 '$AcCoA_{1,2} \times Asp_{1,2,3}$']
78
79 ebv, emu_elmo = get_EMU_basis_vectors_and_pathways(emu_network, exte
80 #cobra.io.save_json_model(emu_network, "figure12_combined_EMU_netw
81
82 display(Markdown('### D3Flux Combined EMU transition network\n'))
83 display(flux_map(cobra.io.load_json_model("figure12_combined_EMU_netw
84 flux_dict={rxn.id: emu_elmo.sum(axis=0)[rxn.id] for rxn in emu_n
85
86 display(Markdown('### EMU basis vectors \n'))
87 display(HTML(ebv.reset_index().set_index('EMU Basis Vector').loc[bv_o
88
89 #display(ebv['EMU Basis Vector'].values)
90
91
92 display(Markdown('### EMU Pathways \n'))
93 emu_elmo['EMU Basis Vector'] = ebv['EMU Basis Vector']
94 display(HTML(emu_elmo.reset_index().set_index('EMU Basis Vector').loc[em
95
96 for em in emu_elmo.index:
97     with emu_network:
98         emu_network.remove_reactions([r for r in emu_elmo.columns if
99 display(Markdown('#### EMU Pathway {} {}'.format(em) + ebv.loc[em, 'P
100 display(flux_map(emu_network,
101 display_name_format=lambda x: str(x.id),
102 flux_dict={rxn.id: emu_elmo.loc[em, rxn.id]
103 for rxn in emu_network.reactions}))
104

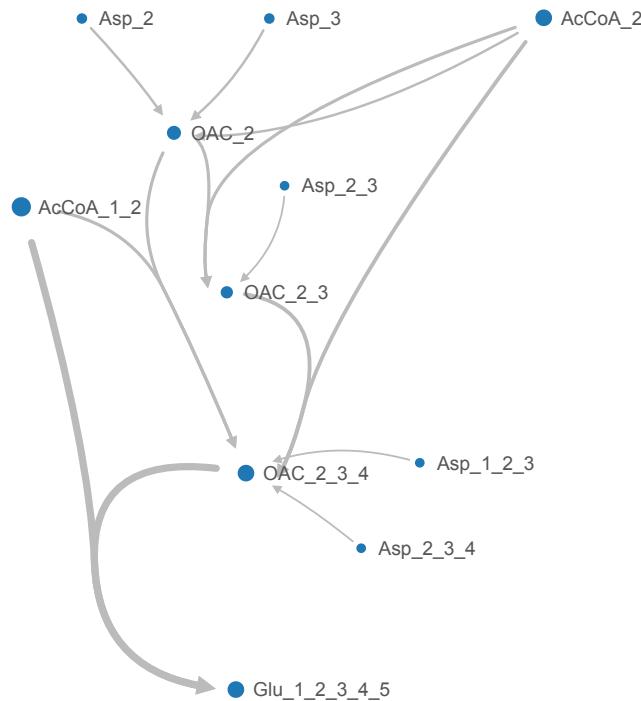
```

```

/Users/zuck016/.pyenv/versions/anaconda3-2020.11/lib/python3.8/subprocess.py:849: RuntimeWarning: line buffering (buffering=1) isn't supported in binary mode, the default buffer size will be used
    self.stderr = io.open(errread, 'rb', bufsize)

```

D3Flux Combined EMU transition network

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

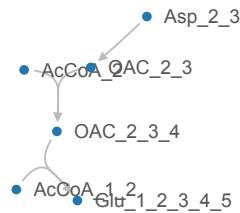
EMU basis vectors

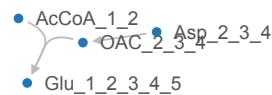
		index	Asp_2	Asp_3	AcCoA_2	AcCoA_1,2	Asp_2_3	Asp_1_2_3	Asp_2_3_4	Glu_1_2_3_4_5	
		EMU Basis Vector									
Asp ₂	× (AcCoA ₂) ²	EM5	-1.0	-0.0	-2.0	-1.0	-0.0	-0.0	-0.0	-0.0	
Asp ₃	× (AcCoA ₂) ²	EM6	-0.0	-1.0	-2.0	-1.0	-0.0	-0.0	-0.0	-0.0	
(AcCoA ₂) ³	× AcCoA _{1,2}	EM9	-0.0	-0.0	-3.0	-1.0	-0.0	-0.0	-0.0	-0.0	
AcCoA ₂	× AcCoA _{1,2}	EM1	-0.0	-0.0	-1.0	-1.0	-1.0	-0.0	-0.0	-0.0	
	× Asp _{2,3}										

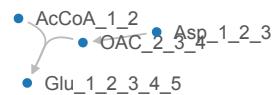
	index	Asp_2	Asp_3	AcCoA_2	AcCoA_1_2	Asp_2_3	Asp_1_2_3	Asp_2_3_4	Glu_1_2
EMU Basis Vector									
$Asp_2 \times (AcCoA_{1,2})^2$	EM7	-1.0	-0.0	-0.0	-2.0	-0.0	-0.0	-0.0	-0.0
$Asp_3 \times (AcCoA_{1,2})^2$	EM4	-0.0	-1.0	-0.0	-2.0	-0.0	-0.0	-0.0	-0.0
$AcCoA_2 \times (AcCoA_{1,2})^2$	EM8	-0.0	-0.0	-1.0	-2.0	-0.0	-0.0	-0.0	-0.0
$AcCoA_{1,2} \times Asp_{2,3,4}$	EM2	-0.0	-0.0	-0.0	-1.0	-0.0	-0.0	-1.0	-1.0
$AcCoA_{1,2} \times Asp_{1,2,3}$	EM3	-0.0	-0.0	-0.0	-1.0	-0.0	-1.0	-0.0	-0.0

EMU Pathways

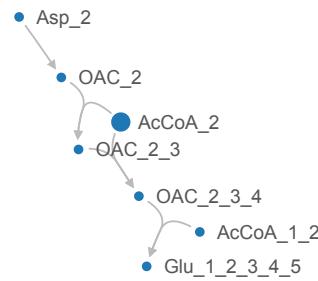
	index	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	I
EMU Basis Vector																		
$Asp_2 \times (AcCoA_2)^2 \times AcCoA_{1,2}$	EM5	1.0	0.0	0.0	0.0	0.0	2.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
$Asp_3 \times (AcCoA_2)^2 \times AcCoA_{1,2}$	EM6	0.0	1.0	0.0	0.0	0.0	2.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
$(AcCoA_2)^3 \times AcCoA_{1,2}$	EM9	0.0	0.0	0.0	0.0	0.0	3.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
$AcCoA_2 \times AcCoA_{1,2} \times Asp_{2,3}$	EM1	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
$Asp_2 \times (AcCoA_{1,2})^2$	EM7	1.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
$Asp_3 \times (AcCoA_{1,2})^2$	EM4	0.0	1.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
$AcCoA_2 \times (AcCoA_{1,2})^2$	EM8	0.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
$AcCoA_{1,2} \times Asp_{2,3,4}$	EM2	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
$AcCoA_{1,2} \times Asp_{1,2,3}$	EM3	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

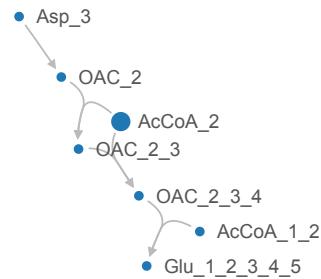
EMU Pathway EM1 ($AcCoA_2 \times AcCoA_{1,2} \times Asp_{2,3}$)[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)**EMU Pathway EM2 ($AcCoA_{1,2} \times Asp_{2,3,4}$)**

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)**EMU Pathway EM3 ($AcCoA_{1,2} \times Asp_{1,2,3}$)**

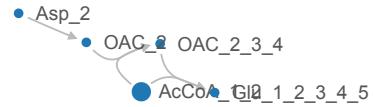
[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

EMU Pathway EM4 ($Asp_3 \times (AcCoA_{1,2})^2$)

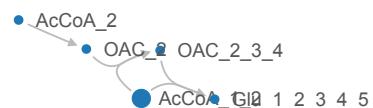
EMU Pathway EM5 ($Asp_2 \times (AcCoA_2)^2 \times AcCoA_{1,2}$)[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)**EMU Pathway EM6 ($Asp_3 \times (AcCoA_2)^2 \times AcCoA_{1,2}$)**

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

EMU Pathway EM7 ($Asp_2 \times (AcCoA_{1,2})^2$)



EMU Pathway EM8 ($AcCoA_2 \times (AcCoA_{1,2})^2$)

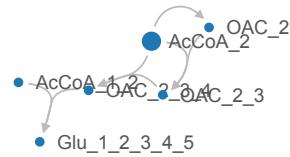


[Save JSON](#)

[Show Reaction Nodes](#)

[Download SVG](#)

EMU Pathway EM9 ($(AcCoA_2)^3 \times AcCoA_{1,2}$)

[Save JSON](#)[Show Reaction Nodes](#)[Download SVG](#)

4.3 Tracer Compositions for TCA EMU Network

In [26]:

```

1 import itertools
2
3 display(Image("Figure11.png"))
4 Asp_1_2_3_4 = EMU("Asp_1_2_3_4")
5 AcCoA_1_2 = EMU("AcCoA_1_2")
6
7 Asp_isotopic_labeling_states = []
8 AcCoA_isotopic_labeling_states = []
9 for i in range(2 ** Asp_1_2_3_4.get_num_atoms()):
10     Asp_isotopic_labeling_states.append(
11         pd.DataFrame(
12             [{"%": 100, 1: (i) & 1, 2: (i >> 1) & 1, 3: (i >> 2) & 1,
13              index=[r"$100\%" + r"\text{Asp}_{\text{d}\text{d}\text{d}}$"
14                  % ((i) & 1, (i >> 1) & 1, (i >> 2) & 1, (i >> 3) & 1),
15                  ],
16             )
17         )
18     )
19
20 for j in range(2 ** AcCoA_1_2.get_num_atoms()):
21     AcCoA_isotopic_labeling_states.append(
22         pd.DataFrame(
23             [{"%": 100, 1: (j) & 1, 2: (j >> 1) & 1}],
24             index=[r"$100\%" + r"\text{AcCoA}_{\text{d}\text{d}}$" % ((j) & 1),
25             ]
26         )
27     )
28
29 tracer_mixtures = [
30     dict(
31         Asp=Tracer("Asp", Asp_1_2_3_4.get_num_atoms(), Asp_il),
32         AcCoA=Tracer("AcCoA", AcCoA_1_2.get_num_atoms(), AcCoA_il),
33     )
34     for Asp_il, AcCoA_il in itertools.product(
35         Asp_isotopic_labeling_states, AcCoA_isotopic_labeling_states
36     )
37 ]
38 emu_basis_vectors = [EMU_Basis_Vector(ebv.iloc[e, :-1]) for e in range(len(emu_basis_vectors))]
39 mid_comparison = {}
40 isl_comparison = {}
41 for tracers in tracer_mixtures:
42     for emu_basis_vector in emu_basis_vectors:
43         for emu in emu_basis_vector.get_input_emus():
44             emu_isl = emu.get_isotopic_labeling_state(tracers)
45             emu_isl.index = pd.MultiIndex.from_tuples([
46                 (" and ".join([tracers[t].get_name() for t in sorted(emu_isl.columns)]), a)
47                 for atom in emu_isl.columns
48             ])
49             for atom in emu_isl.columns:
50                 key = (emu_basis_vector.get_name(), emu.get_name())
51                 if key in isl_comparison:
52                     isl_comparison[key] = isl_comparison[key].append(emu_isl[atom])
53                 else:
54                     isl_comparison[key] = emu_isl[atom]
55
56 mid_comparison[

```

```

57         " and ".join([tracers[t].get_name() for t in sorted(tracers)])
58         emu_basis_vector.get_name(),
59     ] = emu_basis_vector.get_mid(tracers)
60
61 display(Markdown("### Isotopomer labeling states"))
62 isl_df = pd.DataFrame(isl_comparison)
63 isl_df.columns = isl_df.columns.set_names(["EMU Basis Vectors", "EMUs"])
64 isl_df.index = isl_df.index.set_names(["Tracer Compositions", "Tracers"])
65 # display(HTML(isl_df.to_html()))
66 display(Markdown("### Mass Isotopomer Distribution"))
67 mid_df = pd.DataFrame(mid_comparison)
68 mid_df.columns = mid_df.columns.set_names(["Tracer Compositions", "EMUs"])
69
70
71 # display(HTML(mid_df.T.to_html()))
72 rank4 = mid_df.xs(
73     r"$100\%" + r"\text{AcCoA}_{00}" and $100\%" + r"\text{Asp}_{11}" +
74     axis=1,
75     level="Tracer Compositions",
76 )
77 display(rank4)
78 np.linalg.matrix_rank(rank4)
79 table_template = "| {} |\n" * 6
80
81 accoa_isl = [accoa.index[0] for accoa in AcCoA_isotopic_labeling_states]
82 asp_isl = [asp.index[0] for asp in Asp_isotopic_labeling_states]
83
84 rank_df = pd.DataFrame(0, index=accoa_isl, columns=asp_isl)
85 for accoa in accoa_isl:
86     for asp in asp_isl:
87         rank_df.loc[accoa, asp] = np.linalg.matrix_rank(
88             mid_df.xs(accoa + " and " + asp, axis=1, level="Tracer Compositions")
89         )
90 display(rank_df)
91 # table = table_template.format(' | '.join([' | '] + asp_isl),
92 #                                 '/'.join(['-----'] + ['-----' for _ in asp_isl]),
93 #                                 ' | '.join([accoa_isl[0]] + [str(np.linalg.matrix_rank(
94 #                                     mid_df.xs(accoa_isl[0], axis=1, level="Tracer Compositions")))]),
95 #                                 ' | '.join([accoa_isl[1]] + [str(np.linalg.matrix_rank(
96 #                                     mid_df.xs(accoa_isl[1], axis=1, level="Tracer Compositions")))]),
97 #                                 ' | '.join([accoa_isl[2]] + [str(np.linalg.matrix_rank(
#                                     mid_df.xs(accoa_isl[2], axis=1, level="Tracer Compositions")))]),
98 #                                 ' | '.join([accoa_isl[3]] + [str(np.linalg.matrix_rank(
#                                     mid_df.xs(accoa_isl[3], axis=1, level="Tracer Compositions")))])
99 # display(Markdown(table))

```

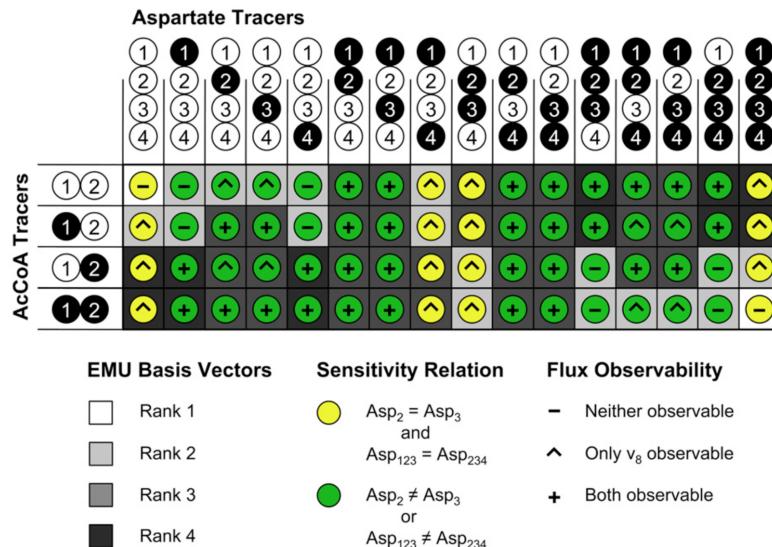


Fig. 11. Analysis of flux observability and EMU basis vector classification for the TCA cycle. Flux observability was evaluated for all combinations of aspartate tracers and acetyl-CoA tracers. The square colors denote the EMU basis vector rank, the circle colors correspond to whether v_7 functionality was lost due to tracer selection, and the symbol within the circle denotes the flux observability. Observability of both free fluxes in the model required an EMU basis vector rank of 3 or 4, and required that Asp₂ ≠ Asp₃ and/or Asp₁₂₃ ≠ Asp₂₃₄.

Isotopomer labeling states

Mass Isotopomer Distribution

EMU Basis Vectors	Asp _{2,3} × AcCoA _{1,2} × AcCoA ₂	Asp _{2,3,4} × AcCoA _{1,2}	Asp _{1,2,3} × AcCoA _{1,2}	Asp ₃ × (AcCoA _{1,2}) ²	Asp ₂ × (AcCoA ₂) ² × AcCoA _{1,2}	Asp ₃ × (AcCoA ₂) ² ×
M+0	0.0	0.0	0.0	0.0	0.0	0.0
M+1	0.0	0.0	0.0	100.0	100.0	
M+2	100.0	100.0	0.0	0.0	0.0	0.0
M+3	0.0	0.0	100.0	0.0	0.0	0.0
M+4	0.0	0.0	0.0	0.0	0.0	0.0
M+5	0.0	0.0	0.0	0.0	0.0	0.0

	100% Asp ₀₀₀₀	100% Asp ₁₀₀₀	100% Asp ₀₁₀₀	100% Asp ₁₁₀₀	100% Asp ₀₀₁₀	100% Asp
100% AcCoA ₀₀	1	2	2	3	2	
100% AcCoA ₁₀	2	2	3	3	3	
100% AcCoA ₀₁	4	4	3	3	3	
100% AcCoA ₁₁	4	4	3	3	3	

