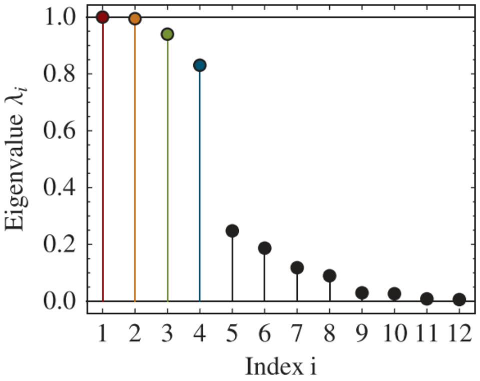
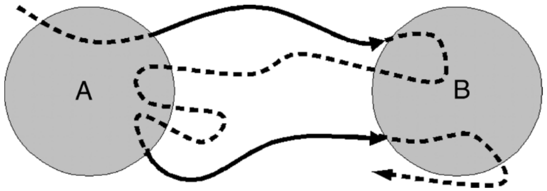
Coarse-graining MSMs with PCCA and Transition Path Theory analysis



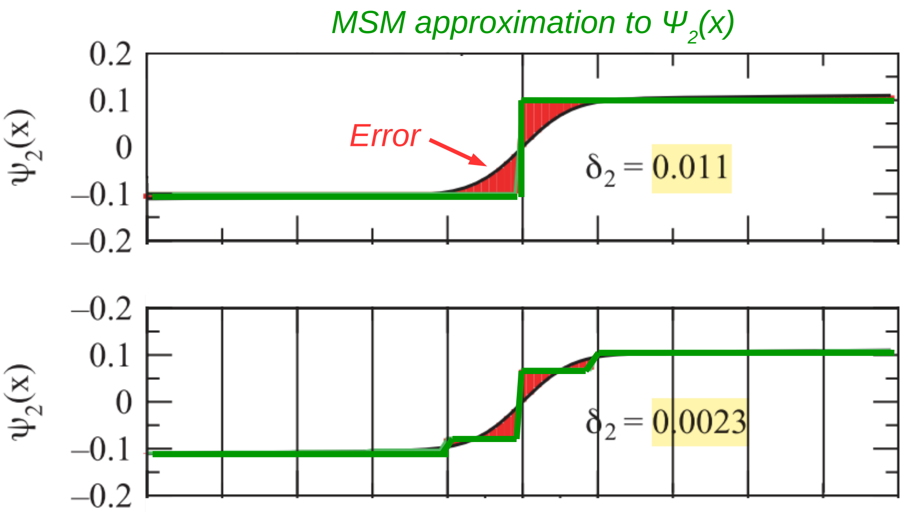


Computational Molecular Biology

Lluís Raich
2nd of March 2021

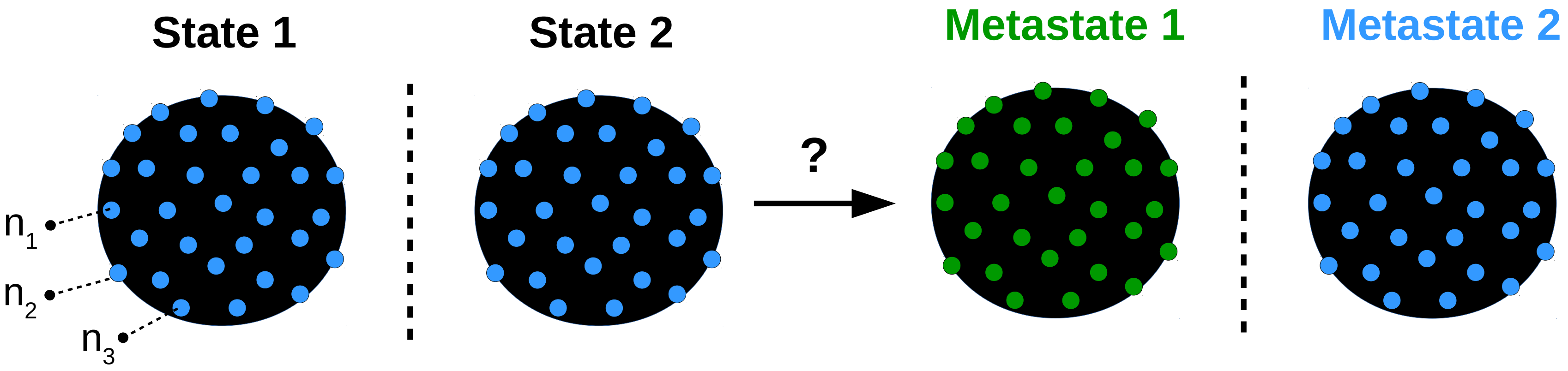
Freie Universität Berlin

Why do we want to coarse-grain MSMs?



Prinz et al. J. Chem. Phys. 134, 174105 (2011)

- MSMs approximate true eigenfunctions with (multi) step functions.
- The **better** the **discretization**, the **better** the **description** *$\Psi_2(x)$*
- The **more** discretization **clusters**, the more **difficult** is to **interpret** the MSM.



n Markov states (clusters)

n-1 non-stationary processes

$\mathbf{T}(\tau) \in \mathbb{R}^{n\times n}$

m metastable states

m-1 non-stationary processes

$\mathbf{T}_c(\tau) \in \mathbb{R}^{m\times m}$

Right eigenvectors of T(τ)

- Spectral decomposition of the transition matrix:

$$\mathbf{T}(\tau) = \mathbf{R}\,\mathbf{\Lambda}(\tau)\,\mathbf{L}$$

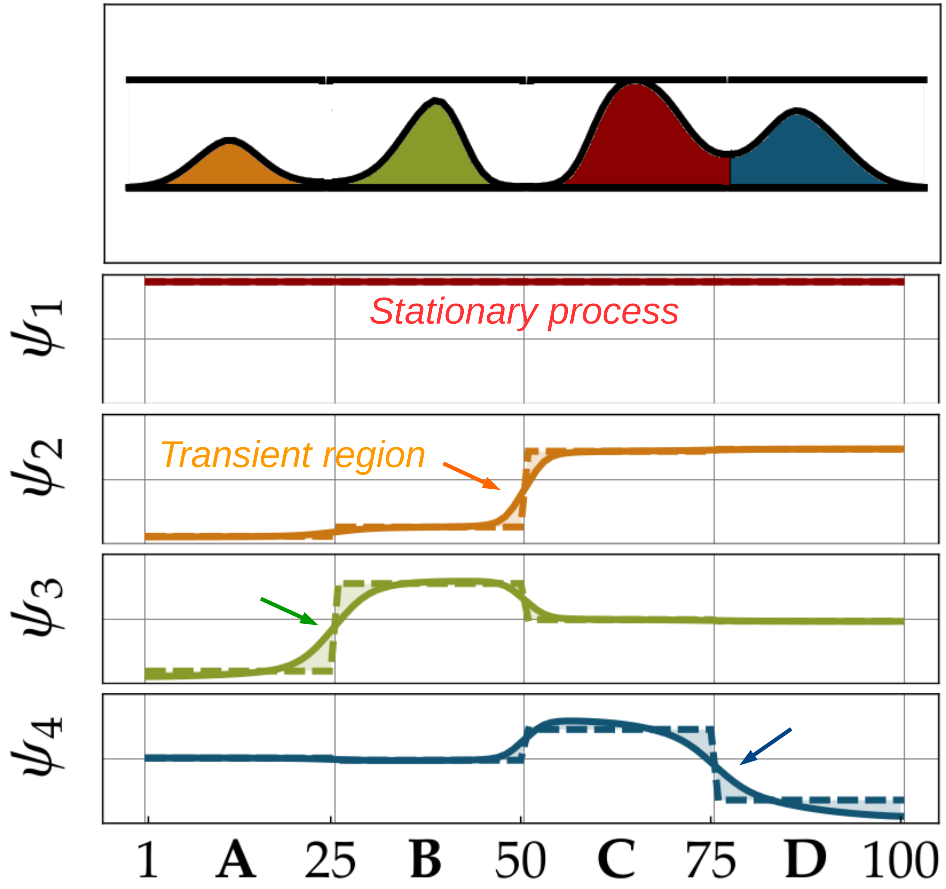


Figure courtesy of JH Prinz

- Right eigenvectors represent dynamical processes between the different states.
- They contain information about metastable and transient regions.
- Transient regions are characterized by (abrupt) changes, while metastable regions are flat.

We **aim** to **find** a cluster of **states** that is **maximally metastable**

How many **metastable states** do we **choose?**

How many metastable states do we choose?

- At long lag times **Λ**(τ) becomes sparse because of the exponential decay of eigenvalues:

$$\mathbf{\Lambda}(\tau) = \begin{pmatrix} \lambda_1 & & & & & \\ & \lambda_2 & & & & \\ & & \lambda_3 & & & \\ & & & \lambda_4 \approx 0 & & \\ & & & & \lambda_5 \approx 0 & \\ & & & & & \lambda_6 \approx 0 \end{pmatrix}$$

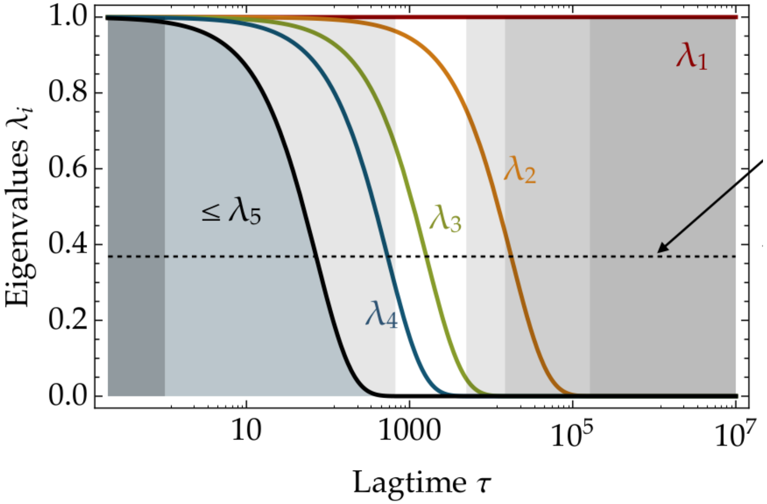


Figure courtesy of JH Prinz

- We can approximate the transition matrix using only the (Perron) cluster of m-dominant eigenvalues and eigenvectors:

$$\mathbf{T}(\tau) \approx \mathbf{R'}\,\mathbf{\Lambda'}(\tau)\,\mathbf{L'}$$

$$\mathbf{T}(\tau) \in \mathbb{R}^{n\times n} \xrightarrow{\checkmark} \mathbf{\Lambda'}(\tau) \in \mathbb{R}^{m\times m} \xrightarrow{?} \mathbf{T}_{\mathbf{c}}(\tau) \in \mathbb{R}^{m\times m}$$

Do a **cluster analysis** on the **Perron cluster** (PCCA)

The PCCA problem

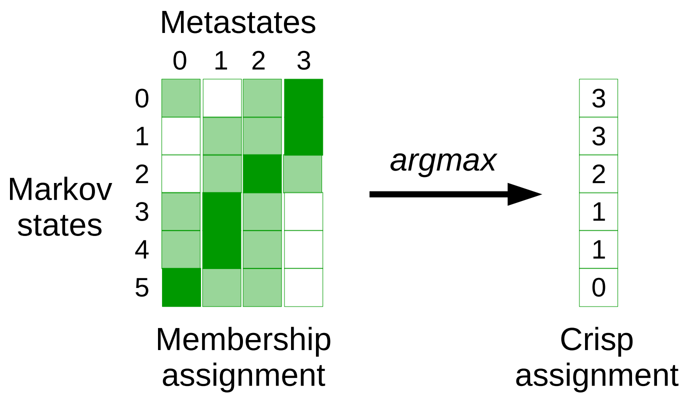
- **Goal:** find a non-singular (i.e. invertible) transformation matrix **A** such that:

$$\mathbf{M} = \mathbf{R'}\,\mathbf{A}$$

subject to the following constraints:

$$\mathbf{M}_j(i) \geq 0 \text{ (positivity)}$$
$$\textstyle\sum_j \mathbf{M}_j(i) = 1 \text{ (partition of unity)}$$

- The resulting matrix **M**, referred as the *membership matrix*, provides a fuzzy assignment over the metastable states (instead of a hard, crisp assignment to a single state).



- **Intuition:** for each cluster center we build a linear combination of (**A**-weighted) dominant eigenvectors that tell us how relevant they are in each metastable state.

The PCCA problem

- Subset of m-dominant eigenvalues and eigenvectors from the spectral decomposition:

$$\mathbf{T}(\tau) \approx \mathbf{R'}\,\mathbf{\Lambda'}(\tau)\,\mathbf{L'}$$

- Introduce matrix **A** and its inverse such that:

$$\mathbf{T}(\tau) \approx \mathbf{R'}\,(\mathbf{A}\mathbf{A}^{-1})\,\mathbf{\Lambda'}(\tau)\,(\mathbf{A}\mathbf{A}^{-1})\,\mathbf{L'}$$

$$\mathbf{T}(\tau) \approx \underbrace{(\mathbf{R'}\,\mathbf{A})}_{\mathbf{M}}\,\underbrace{(\mathbf{A}^{-1}\,\mathbf{\Lambda'}(\tau)\,\mathbf{A})}_{\mathbf{T}_c(\tau)}\,\underbrace{(\mathbf{A}^{-1}\,\mathbf{L'})}_{\mathbf{D}}$$

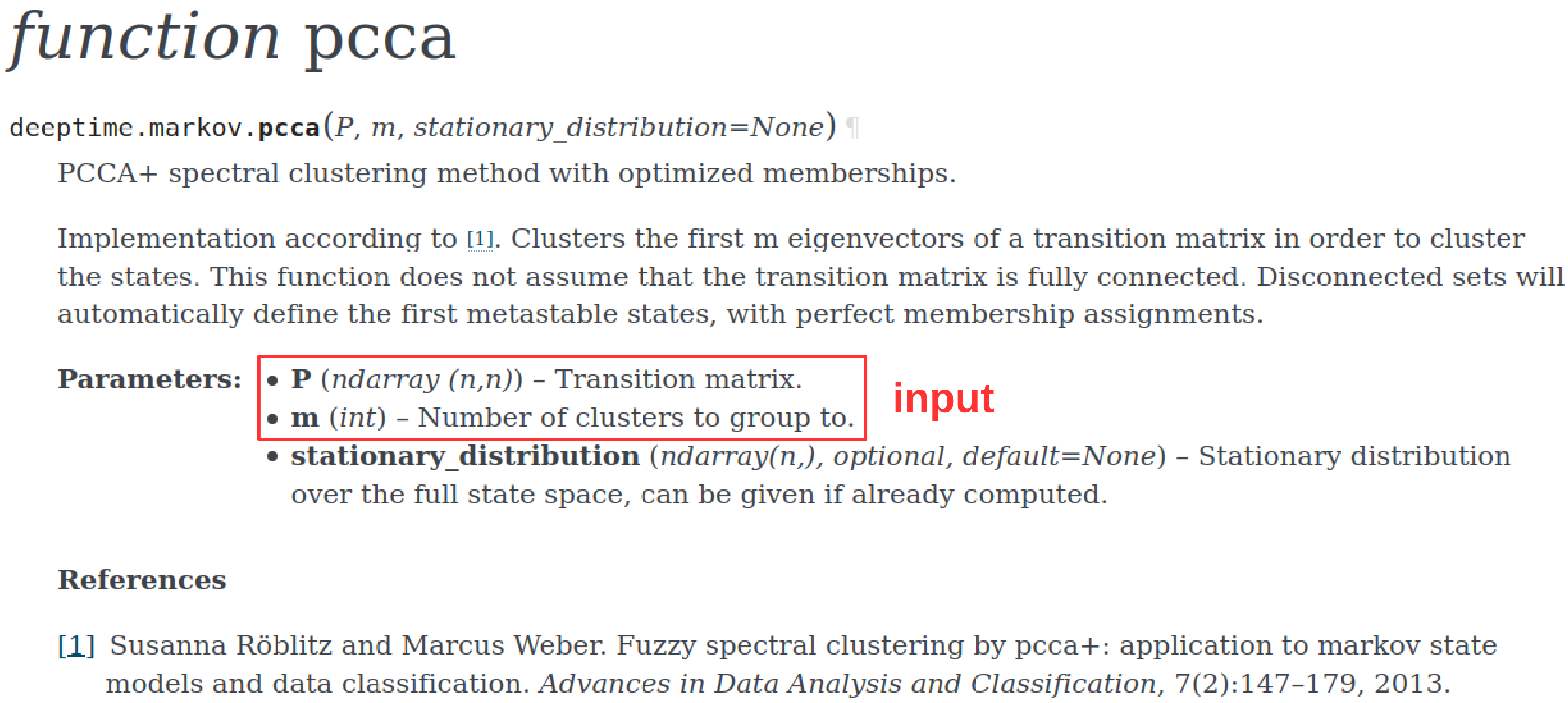
Matrix of memberships: $P(metastable\ state\ i \mid cluster\ j)$
Coarse-grained **T**(τ): $P(metastable\ state\ i \mid metastable\ state\ j)$
Matrix of metastable distributions: $P(cluster\ i \mid metastable\ state\ j)$

- The coarse-grained matrix $\mathbf{T}_c(\tau)$ is small ($\mathbb{R}^{m\times m}$), preserves eigenvalues of the full transition matrix, and represents an approximation of the dynamics between metastable states.

$$\mathbf{T}(\tau) \in \mathbb{R}^{n\times n} \xrightarrow{\checkmark} \mathbf{\Lambda'}(\tau) \in \mathbb{R}^{m\times m} \xrightarrow{\checkmark} \mathbf{T}_c(\tau) \in \mathbb{R}^{m\times m}$$

- Unfortunately $\mathbf{T}_c(\tau)$ is not anymore a stochastic matrix since it can contain negative values.

DeepTime API: PCCA



function pcca

`deeptime.markov.pcca`(*P*, *m*, *stationary_distribution=None*) ¶

PCCA+ spectral clustering method with optimized memberships.

Implementation according to [1]. Clusters the first m eigenvectors of a transition matrix in order to cluster the states. This function does not assume that the transition matrix is fully connected. Disconnected sets will automatically define the first metastable states, with perfect membership assignments.

Parameters:

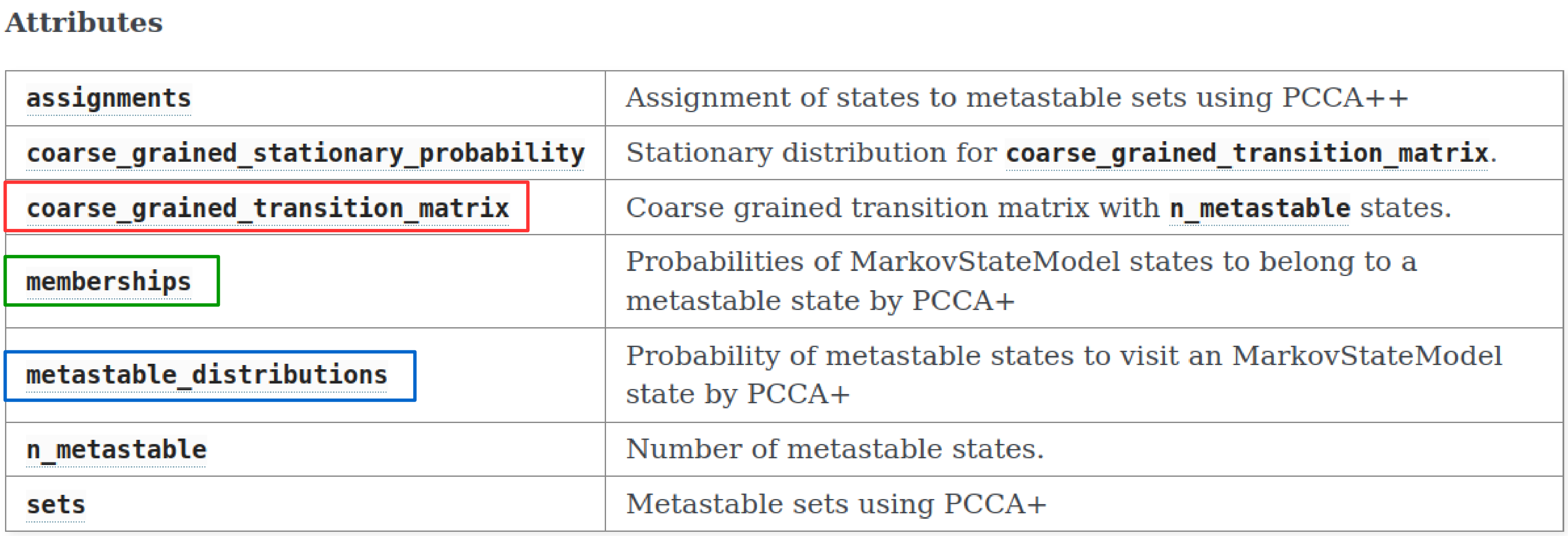
- **P** (*ndarray (n,n)*) – Transition matrix.
- **m** (*int*) – Number of clusters to group to.
- **stationary_distribution** (*ndarray(n,), optional, default=None*) – Stationary distribution over the full state space, can be given if already computed.

input

References

[1] Susanna Röblitz and Marcus Weber. Fuzzy spectral clustering by pcca+: application to markov state models and data classification. *Advances in Data Analysis and Classification*, 7(2):147–179, 2013.

DeepTime API: PCCA



Attributes

`assignments`	Assignment of states to metastable sets using PCCA++
`coarse_grained_stationary_probability`	Stationary distribution for `coarse_grained_transition_matrix`.
`coarse_grained_transition_matrix`	Coarse grained transition matrix with `n_metastable` states.
`memberships`	Probabilities of MarkovStateModel states to belong to a metastable state by PCCA+
`metastable_distributions`	Probability of metastable states to visit an MarkovStateModel state by PCCA+
`n_metastable`	Number of metastable states.
`sets`	Metastable sets using PCCA+

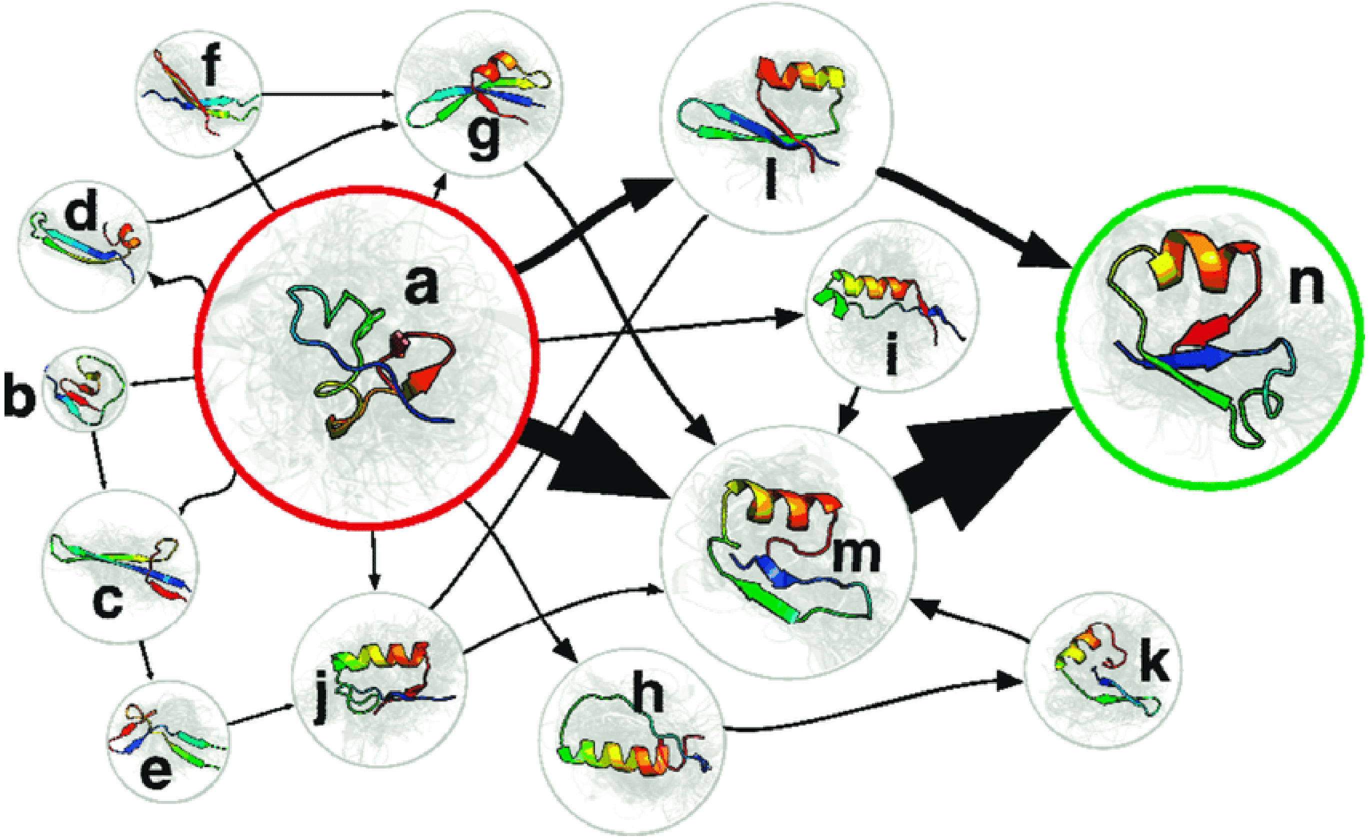
$$\mathbf{T}(\tau) \approx \underbrace{(\mathbf{R}'\,\mathbf{A})}_{\mathbf{M}}\ \underbrace{(\mathbf{A}^{-1}\,\mathbf{\Lambda}'(\tau)\,\mathbf{A})}_{\mathbf{T}_{\mathbf{c}}(\tau)}\ \underbrace{(\mathbf{A}^{-1}\,\mathbf{L}')}_{\mathbf{D}}$$

Matrix of memberships: $P(metastable\ state\ i \mid cluster\ j)$

Coarse-grained **T**(τ): $P(metastable\ state\ i \mid metastable\ state\ j)$

Matrix of metastable distributions: $P(cluster\ i \mid metastable\ state\ j)$

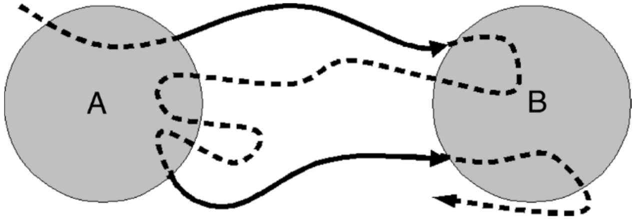
Which of the **pathways** between **A** and **N** is **most likely**?



Voelz et al. *J. Am. Chem. Soc.*, 132, 4702 (2010)

Transition path theory

- Transition path theory is a mathematical framework to **study** the **ensemble** of **transition paths** between **metastable states**

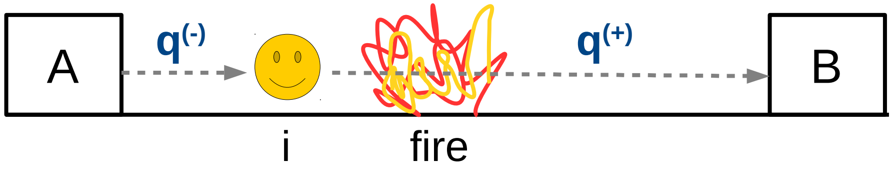


- **A-B reactive trajectory**: parts where the system goes from A to B without coming back to A (or A-B transition path ensemble for a set of such trajectories).

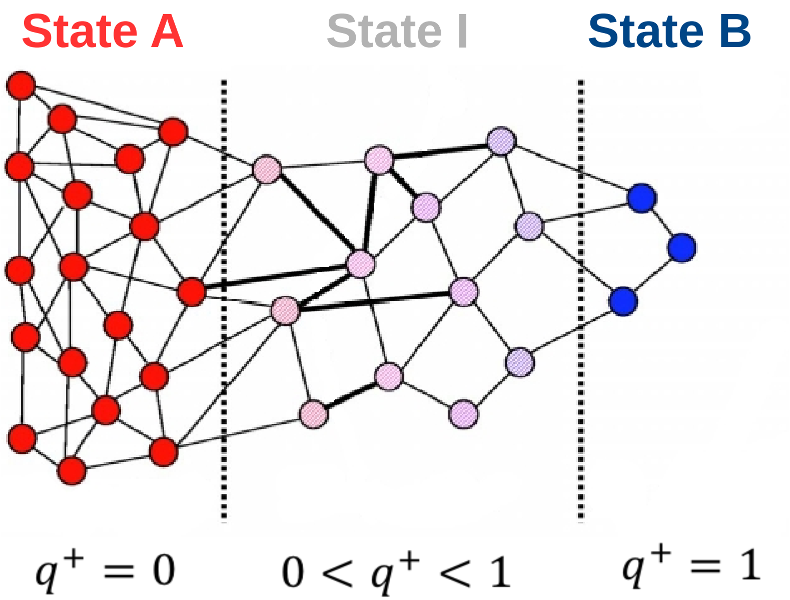
- The essential ingredient required to compute the statistics of transition pathways is the **committor probability function** from A to B:

  - **Forward** committor $\mathbf{q^{(+)}}$: how likely is that a trajectory starting at state "i" reaches state B before A.

  - **Backward** committor $\mathbf{q^{(-)}}$: how likely is that a trajectory arriving at "i" was previously in state A.



Transition path theory



- We can compute the **forward committor function** from A to B (q^+) taking into account the following set of equations:

$$
\begin{aligned}
q_i^+ &= 0 \ \text{ for } i \in A \\
q_i^+ &= 1 \ \text{ for } i \in B \\
q_i^+ &= \sum_{j \in S} \underbrace{T_{ij} q_j^+}_{} \ \text{ for } i \notin \{A, B\}.
\end{aligned}
$$

Probability to jump from “i” to any state ∈ S (Tij) and subsequently reach B (q_j^+).

- Boundary value problem that we can solve numerically:

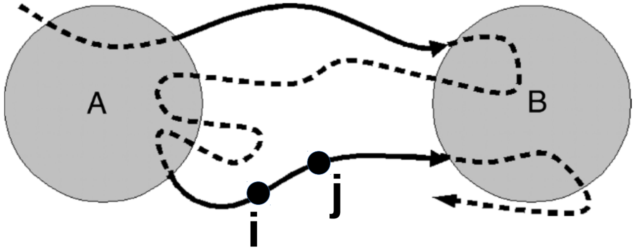
$$-q_i^+ + \sum_{k \in I} T_{ik} q_k^+ = -\sum_{k \in B} T_{ik} \ \text{ for } i \in I.$$

- For a system in equilibrium, we can define the **backward committor** simply as:

$$q^- = 1 - q^+$$

Transition path theory

- We can use the committor function together with **T**(τ) and **π** to obtain information about **fluxes** of trajectories (number of observed A→B transitions per unit time).



- The **gross flux** (or effective flux) is a matrix whose elements describe the probability flux along the edge i,j contributing to the transition from A to B:

$$f_{ij} = \pi_i q_i^- T_{ij} q_j^+$$

- We can also define the **total flux** as the sum over all the f_{ij} elements starting in $i \in A$ and ending in $j \notin A$:

$$F = \sum_{i \in A} \sum_{j \notin A} \pi_i T_{ij} q_j^+ .$$

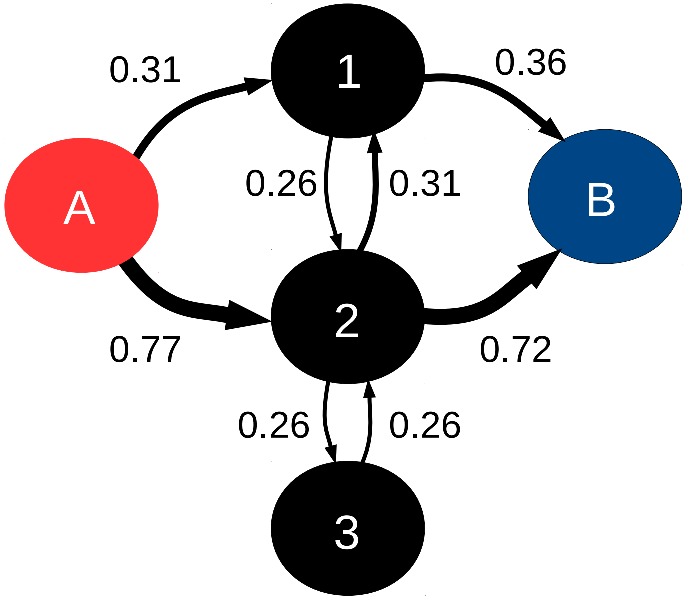
- From the total flux we can compute the **rate** from A to B:

$$k_{AB} = F \Big/ \left(\tau \sum_{i=1}^{m} \pi_i q_i^- \right) .$$

TPT: gross flux of trajectories from A to B

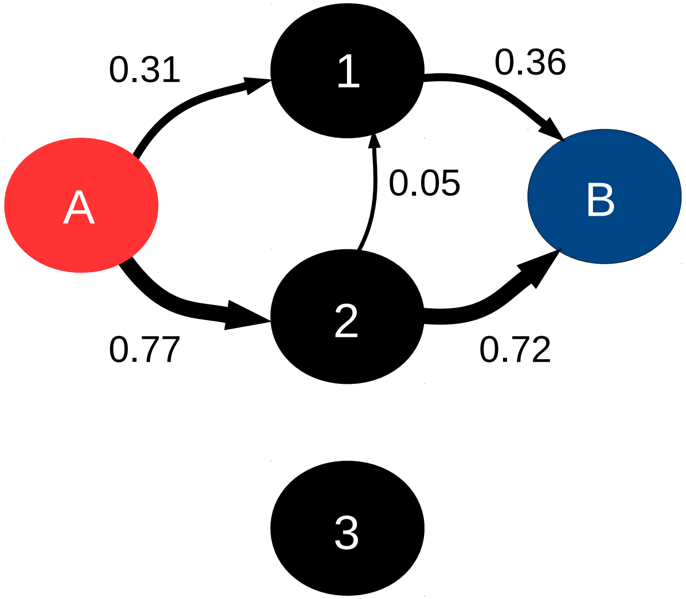
Gross flux

$$f_{ij} = \pi_i q_i^- T_{ij} q_j^+$$



Net flux

$$f_{ij}^+ = \max\{f_{ij}^{AB} - f_{ji}^{AB}, 0\}$$

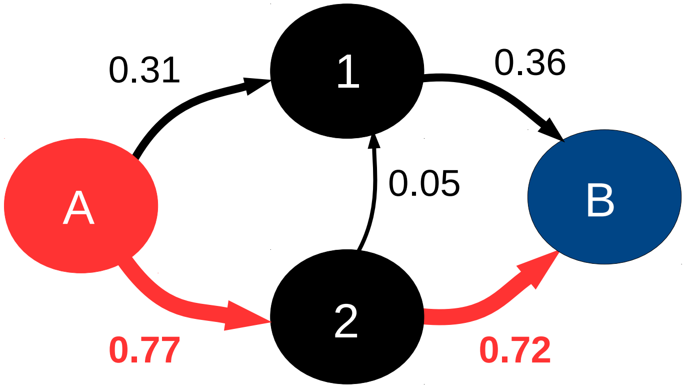


(net flux = 0)

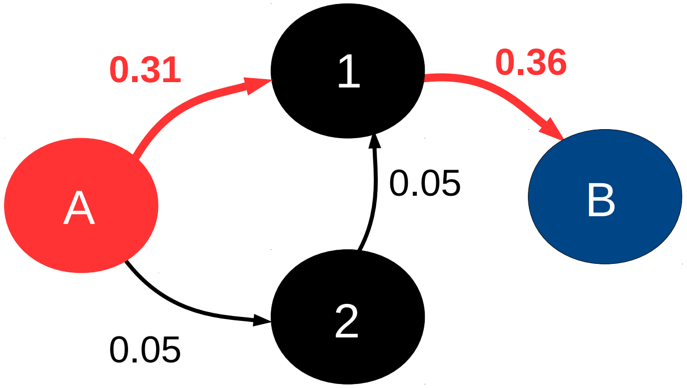
Pathway decomposition

- We can find all pathways from A to B by “subtracting” them iteratively:

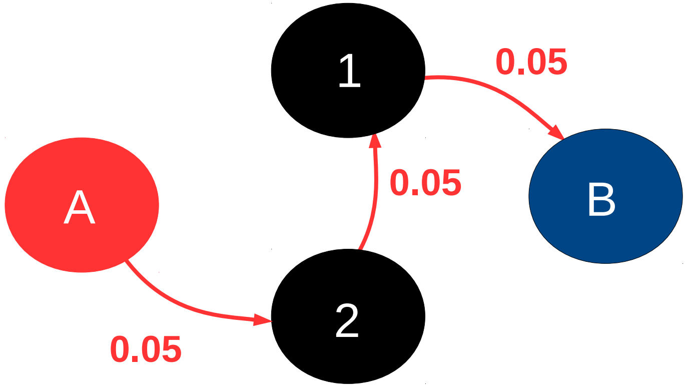
Pathway 1
f=1.49 (65%)



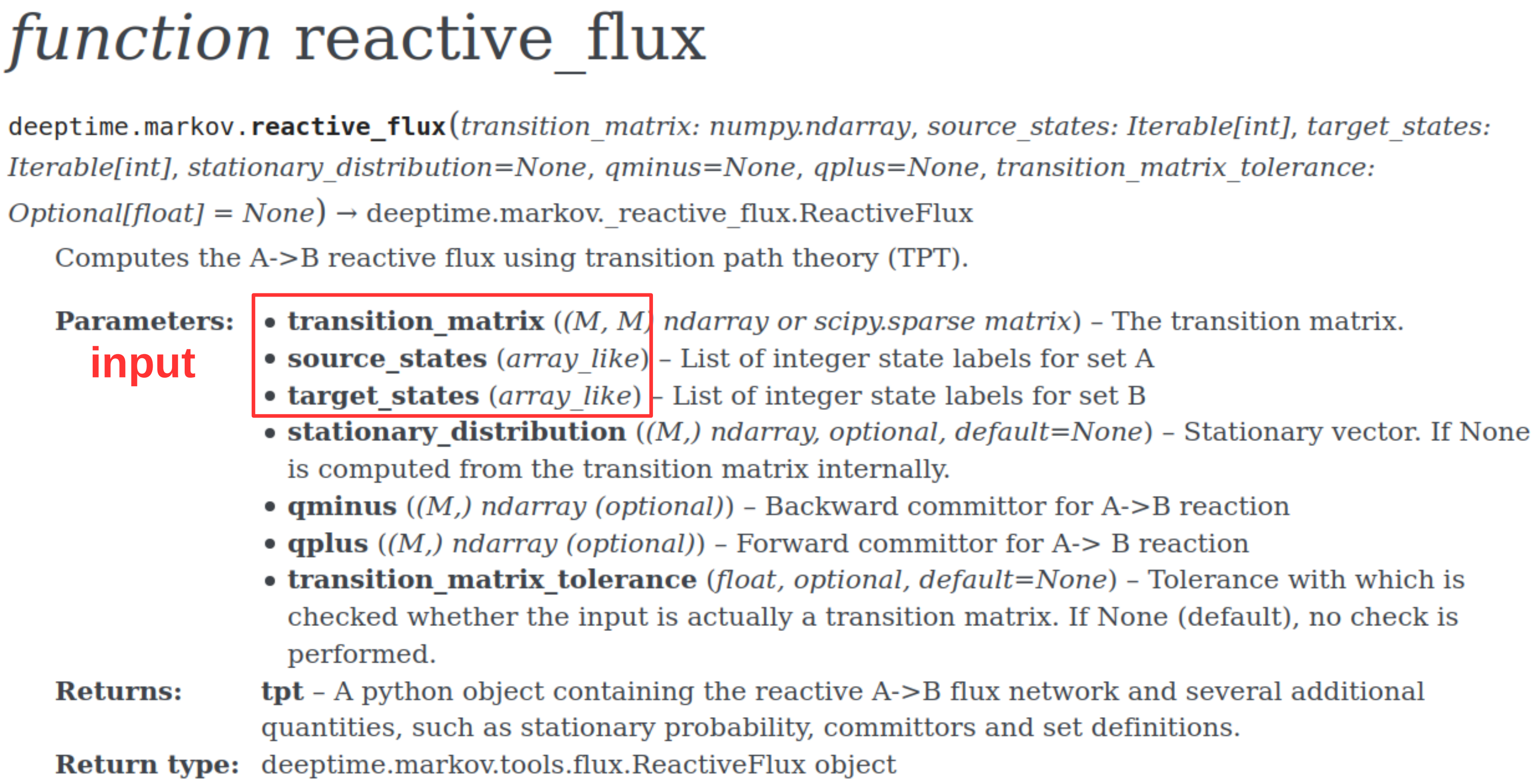
Pathway 2
f=0.67 (29%)



Pathway 3
f=0.15 (6%)



DeepTime API: TPT



function reactive_flux

`deeptime.markov.`**`reactive_flux`**(*transition_matrix: numpy.ndarray, source_states: Iterable[int], target_states: Iterable[int], stationary_distribution=None, qminus=None, qplus=None, transition_matrix_tolerance: Optional[float] = None*) → deeptime.markov._reactive_flux.ReactiveFlux

Computes the A->B reactive flux using transition path theory (TPT).

Parameters:

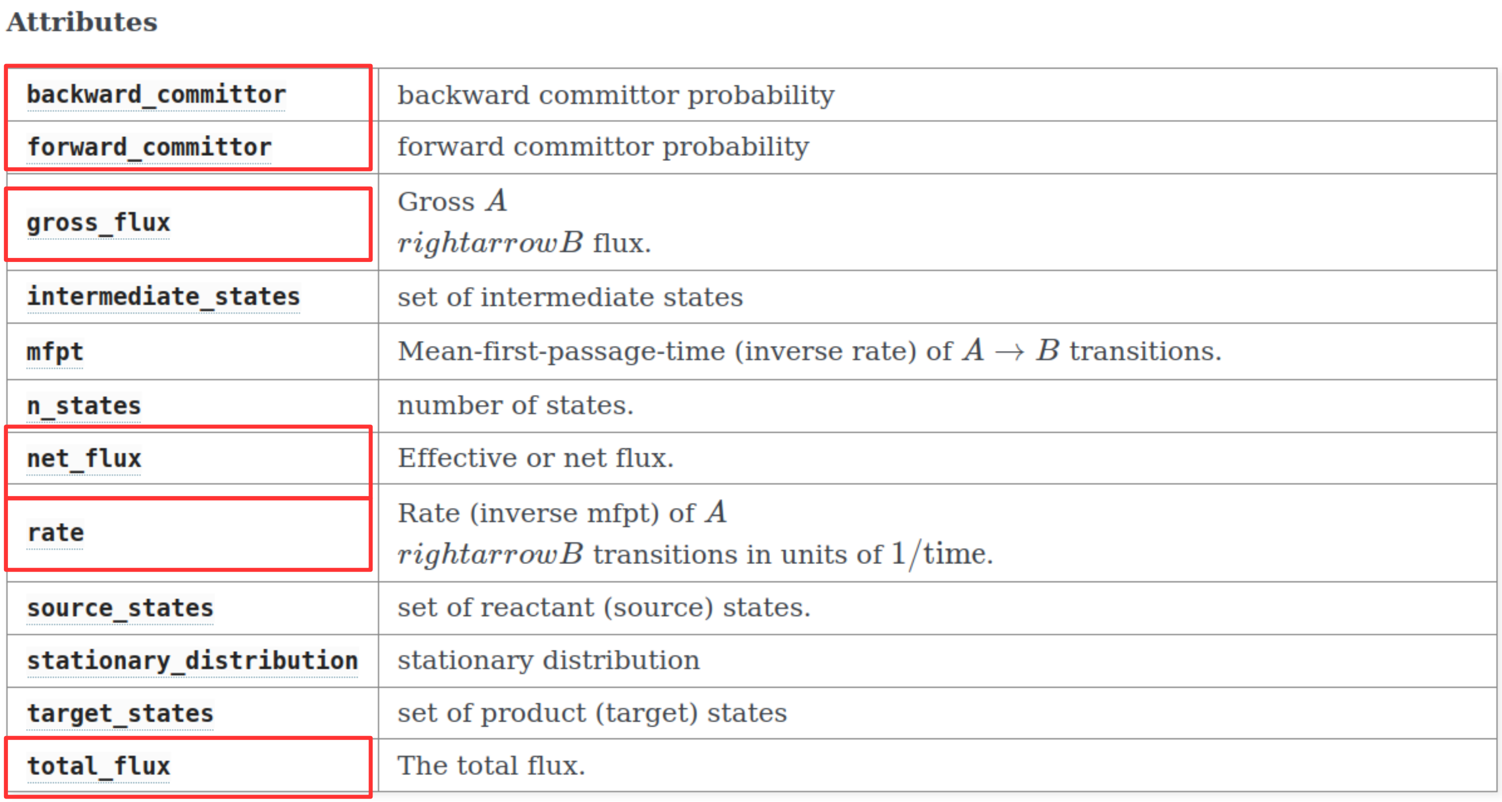
input

- **transition_matrix** (*(M, M) ndarray or scipy.sparse matrix*) – The transition matrix.
- **source_states** (*array_like*) – List of integer state labels for set A
- **target_states** (*array_like*) – List of integer state labels for set B
- **stationary_distribution** (*(M,) ndarray, optional, default=None*) – Stationary vector. If None is computed from the transition matrix internally.
- **qminus** (*(M,) ndarray (optional)*) – Backward committor for A->B reaction
- **qplus** (*(M,) ndarray (optional)*) – Forward committor for A-> B reaction
- **transition_matrix_tolerance** (*float, optional, default=None*) – Tolerance with which is checked whether the input is actually a transition matrix. If None (default), no check is performed.

Returns: **tpt** – A python object containing the reactive A->B flux network and several additional quantities, such as stationary probability, committors and set definitions.

Return type: deeptime.markov.tools.flux.ReactiveFlux object

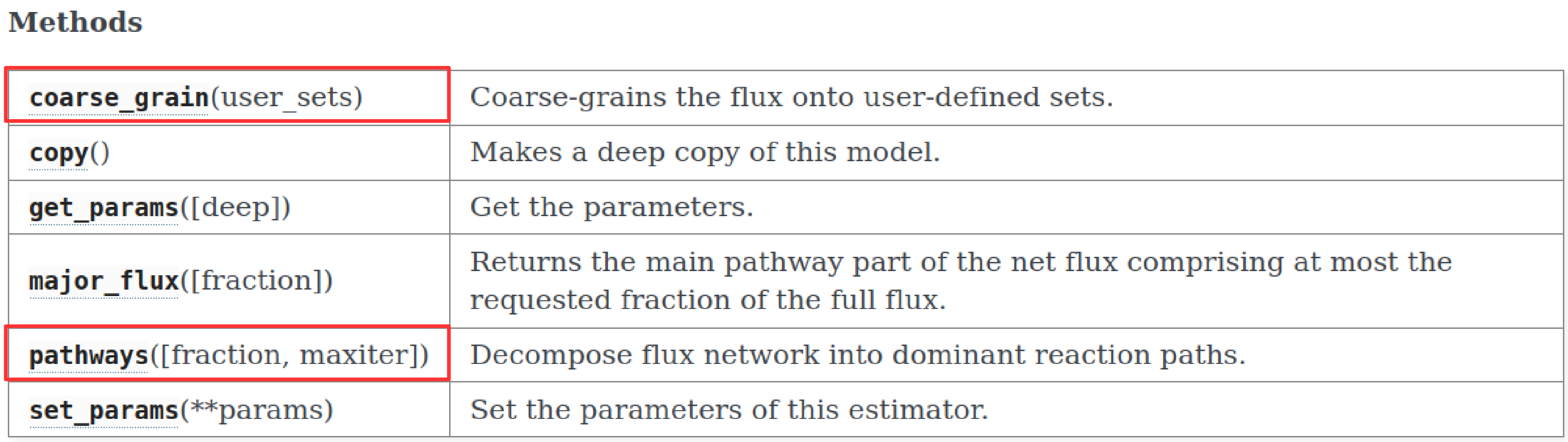
DeepTime API: TPT



Attributes

$q^{(-)}$	`backward_committor`	backward committor probability
$q^{(+)}$	`forward_committor`	forward committor probability
f_{ij}	`gross_flux`	Gross A $rightarrow B$ flux.
	`intermediate_states`	set of intermediate states
	`mfpt`	Mean-first-passage-time (inverse rate) of $A \rightarrow B$ transitions.
	`n_states`	number of states.
f_{ij}^{+}	`net_flux`	Effective or net flux.
k_{AB}	`rate`	Rate (inverse mfpt) of A $rightarrow B$ transitions in units of $1/$time.
	`source_states`	set of reactant (source) states.
	`stationary_distribution`	stationary distribution
	`target_states`	set of product (target) states
F	`total_flux`	The total flux.

DeepTime API: TPT



Methods

coarse_grain(user_sets)	Coarse-grains the flux onto user-defined sets.
copy()	Makes a deep copy of this model.
get_params([deep])	Get the parameters.
major_flux([fraction])	Returns the main pathway part of the net flux comprising at most the requested fraction of the full flux.
pathways([fraction, maxiter])	Decompose flux network into dominant reaction paths.
set_params(**params)	Set the parameters of this estimator.

Further reading

MSM coarse graining

1. “Fuzzy spectral clustering by PCCA+: application to Markov state models and data classification” Röblitz & Weber *Adv. Data Anal. Classif.* (**2013**) 7, 147–179.
2. “Improved coarse-graining of Markov state models via explicit consideration of statistical uncertainty” Bowman *J. Chem. Phys.* (**2012**), 137, 134111.
3. “A Minimum Variance Clustering Approach Produces Robust and Interpretable Coarse-Grained Models” Husic et al. *J. Chem. Theory Comput.* (**2018**), 14, 1071−1082.

Transition path theory

1. “Towards a theory of transition paths” Weinan & Vanden-Eijnden. *J. Stat. Phys.* (**2006**) 123:503–523.
2. “Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations” Noé et al. *PNAS.* (**2009**) 106:19011-19016.
3. https://deeptime-ml.github.io/notebooks/tpt.html