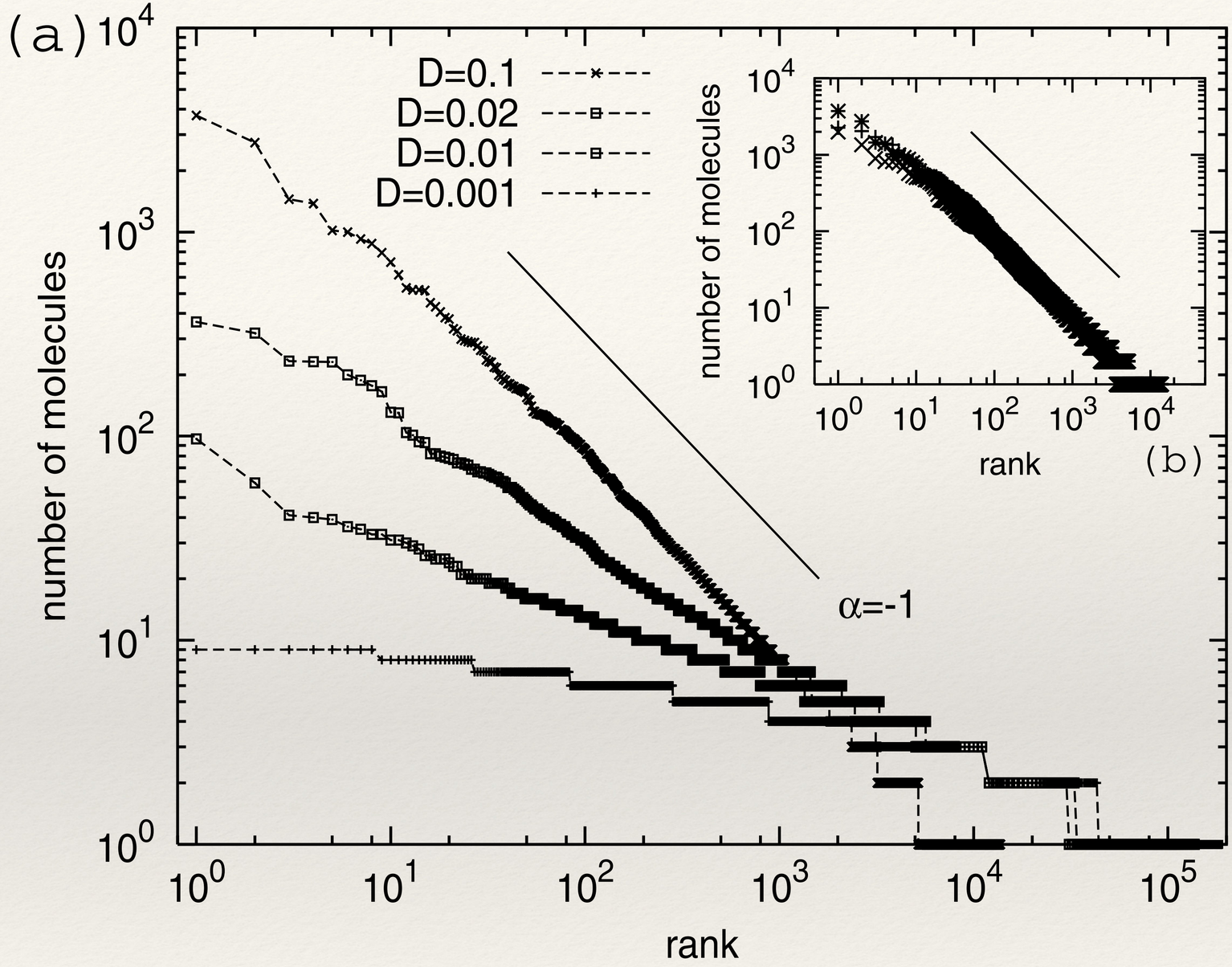
Discrete-time Stochastic Simulation of Chemical Reaction Networks

based on Furusawa & Kaneko's model in PRL 90.088102

Tuan Pham

UNIVERSITY OF COPENHAGEN

Niels Bohr Institute
Biocomplexity Department



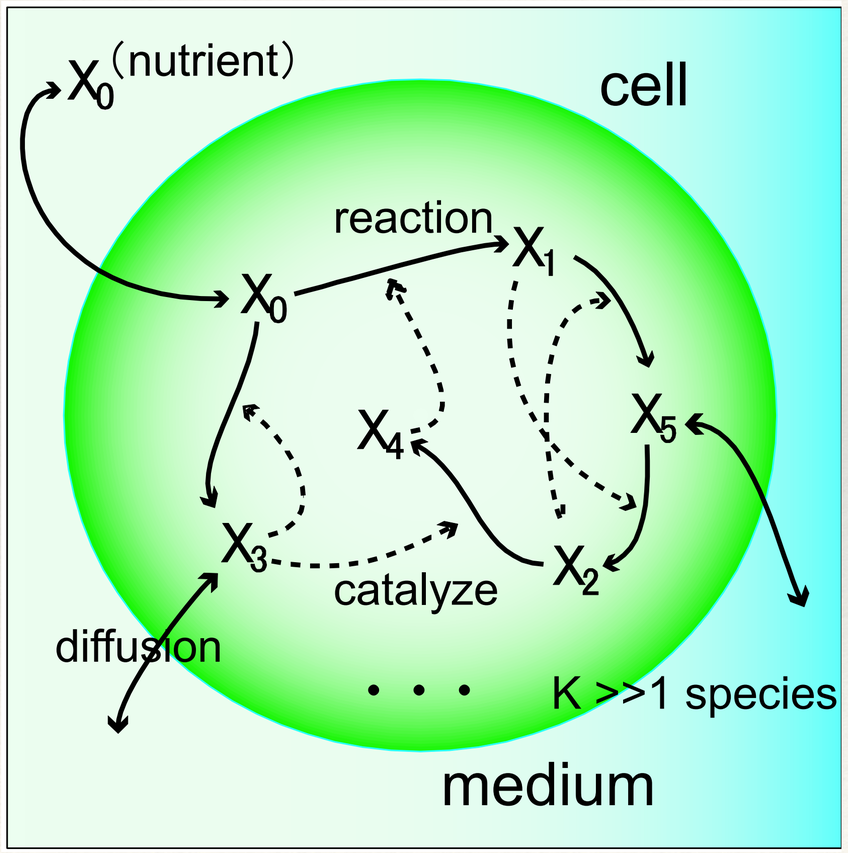
$$k = 5 \times 10^6, N = 5 \times 10^5, \rho = 0.022$$

Algorithm

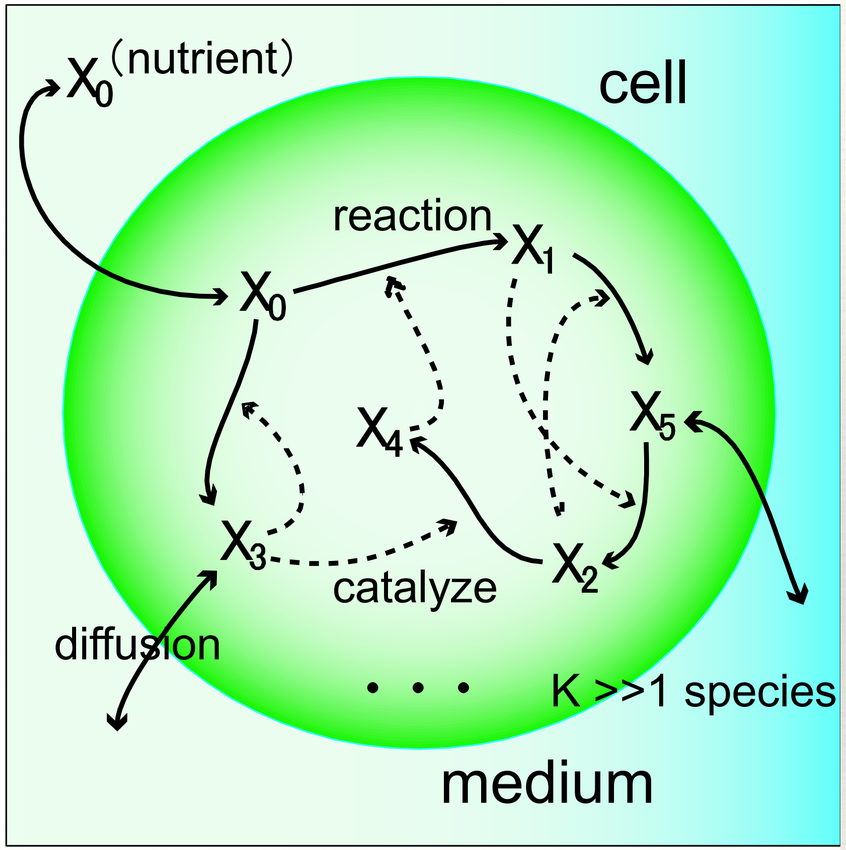
- 0) Creating a vector of k components, denoted by x and assign to each of its component a random integer $\in [1,N]$. Let's keep the total number of molecules less than equal 2N before anything happens, i.e. $\sum_{i=1}^{k} x_i(\text{intial time}) = \alpha N, \alpha \in (0,2)$, note that this is not important as a constraint as long as the initial condition, if set close to 2N, i.e α close to 2, will reach step 4 FASTER.
- 1) Generating the chemical reaction network G with k species (nodes) and a probability ρ for i and j being connected and producing l as specified by $i + j \to l + j$, note ρ is independent from the pair i and j
- 2) At every trial, choose a pair of species i and j at random with probability proportional to the product of their current abundances and if $i + j \to l + j$, $x_i \to x_i - 1$ and $x_l \to x_l + 1$
- 3) After every D of such trials, let a *fixed* nutrient come in with probability 1 and go out with probability proportional to its concentration inside the cell. For simplicity, incoming makes $x_{\text{nutrient}} \to x_{\text{nutrient}} + 1$, whereas outgoing makes $x_{\text{nutrient}} \to x_{\text{nutrient}} - 1$,
- 4) Cell division happens if $\sum_{i=1}^{k} x_i = 2N$, use $x = (1-p)x^{(1)} + px^{(2)}$, where $p = \text{rand}()$, keep one among the two $x^{(1)}$ and $x^{(2)}$
- Once step 4 is completed, go back to 2 and keep repeating this cycle 2-3-4-2 (for the same G that is generated at step 1) for a total number of cell_div

- 1) Generating the chemical reaction network G with k species (nodes) and a probability ρ for i and j being connected and producing l that is chosen as a random integer to specify the reaction $i + j \rightarrow l + j$.

- Note ρ is independent of the pair i and j



- 1) Generating the chemical reaction network G with k species (nodes) and a probability ρ for i and j being connected and producing l as specified by $i + j \rightarrow l + j$. A_{ij} = 0, 1 D_{ij} = product , random integer. G_{ij}=A_{ij}D_{ij

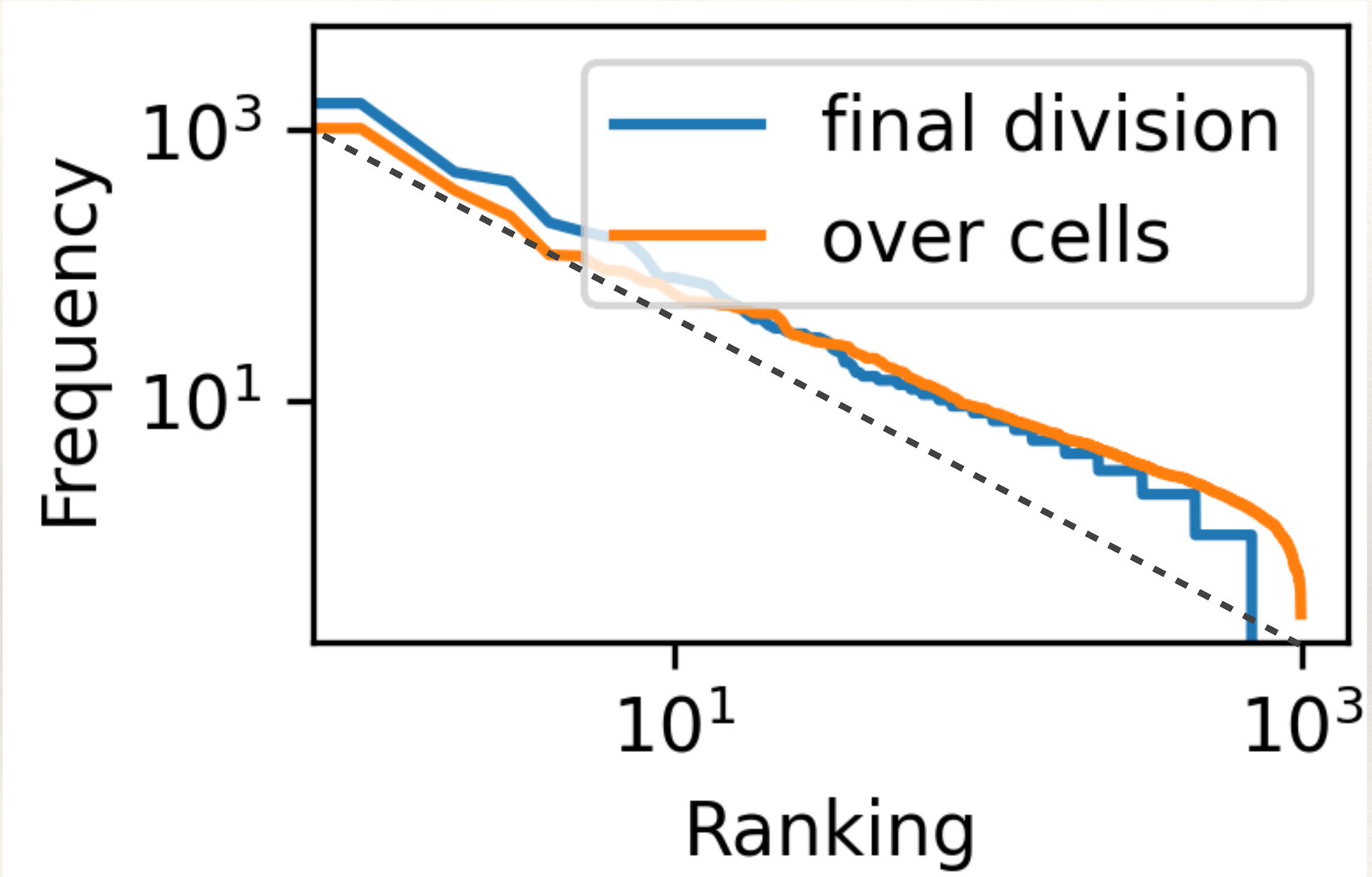



```python
import numpy as np
import matplotlib.pyplot as plt
import random
from tqdm.auto import tqdm

def random_reaction_network(rho, k):
    """Generates a random reaction network with k nodes and density rho."""
    # Create an array of nodes indices
    nodes = np.arange(k)

    # Keep generating edge degrees until the sum is even
    while True:
        edge_degrees = np.random.poisson(rho * k, size=k)
        if sum(edge_degrees) % 2 == 0:
            break

    # Repeat each node index by its edge degree to get the list of edges
    edges = np.repeat(nodes, edge_degrees)

    # Shuffle the edges and reshape into a 2D array
    np.random.shuffle(edges)
    Is, Js = edges.reshape((2, -1))

    # Assign a random reaction product to each edge and store in a dictionary
    G = {(i, j): np.random.randint(k) for i, j in zip(Is, Js)}
    return G
```

- Reaction
  - 2) At every trial, choose a pair of species i and j at random with probability proportional to the product of their current abundances and if $i + j \to l + j$, $x_i \to x_i - 1$ and $x_l \to x_l + 1$
- Diffusion
  - 3) After every D of such trials, let a *fixed* nutrient come in with probability 1 and go out with probability proportional to its concentration inside the cell. For simplicity, incoming makes $x_{\text{nutrient}} \to x_{\text{nutrient}} + 1$, whereas outgoing makes $x_{\text{nutrient}} \to x_{\text{nutrient}} - 1$,
- Cell division
  - 4) Cell division happens if $\sum_{i=1}^{k} x_i = 2N$, use $x = (1 - p)x^{(1)} + px^{(2)}$, where $p = \text{rand}()$, keep one among the two $x^{(1)}$ and $x^{(2)}$

- Reaction
- Diffusion
- Cell division

```
time = 0
r = np.array(list(G.keys()))
P = x[r[:,0]] * x[r[:,1]]
P = P / P.sum()

for division in tqdm(range(celldivision)):

    # If the total number of molecules exceeds 2N, divi
    print('Division count: ', division)
    while np.sum(x) < 2 * N:

        # Choose two random nodes
        index = np.random.choice(len(r), p=P)
        i, j = tuple(r[index])

        # If they are connected by a reaction and both
        if ((i, j) in G) and (x[i] > 0) and (x[j] > 0):
            time += 1
            l = G[(i, j)]
            x[i] -= 1
            x[l] += 1

            # Every D iterations, add a nutrient molecu
            if time % D == 0:
                x[0] += 1
                leaking_out = x[0] / np.sum(x)
                if np.random.random() < leaking_out:
                    x[0] -= 1

            P = x[r[:,0]] * x[r[:,1]]
            P = P / P.sum()
    x = x // np.random.randint(2, 10)
```



$$k = 10^3, N = 10^4, \rho = 0.11$$