

A 3-player game theoretic model of a choice
between two queueing systems with strategic
managerial decision making

Michalis Panayides

EURO 2021 Athens

About me



About me

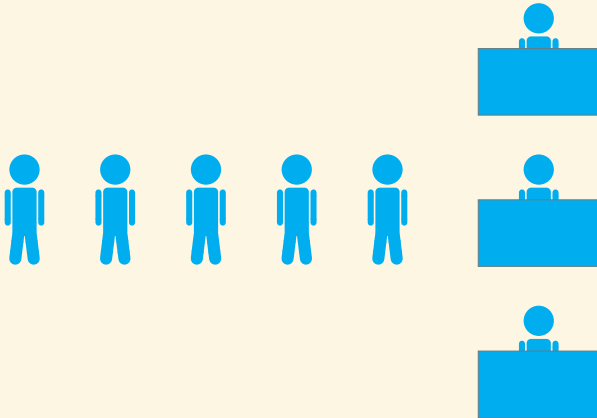


THIS.

Queues - Examples



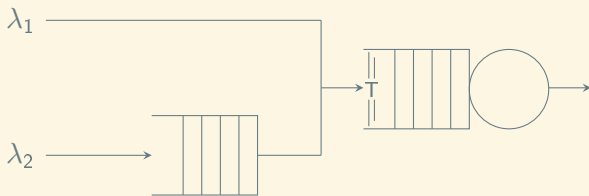
Queues - Examples



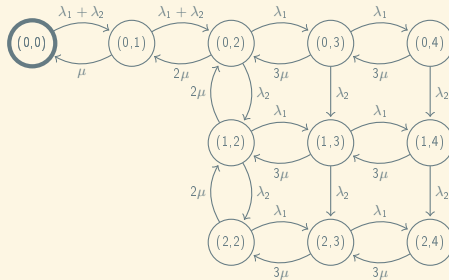
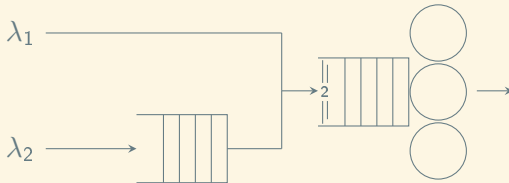
Queues - Examples



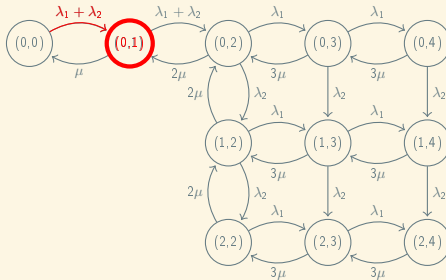
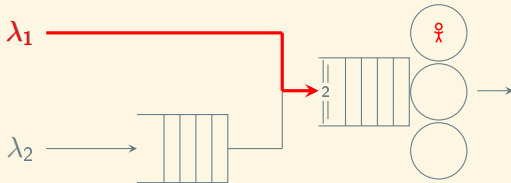
Queueing network structure



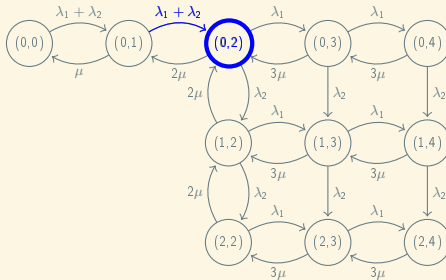
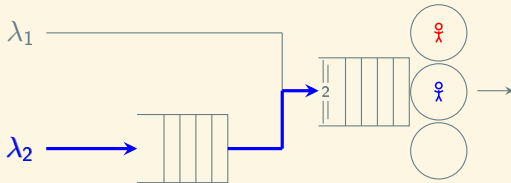
Markov Chain - Custom network



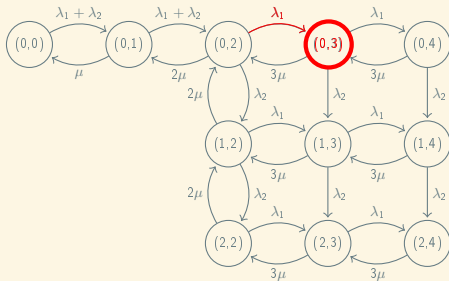
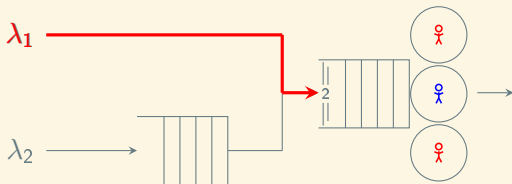
Markov Chain - Custom network



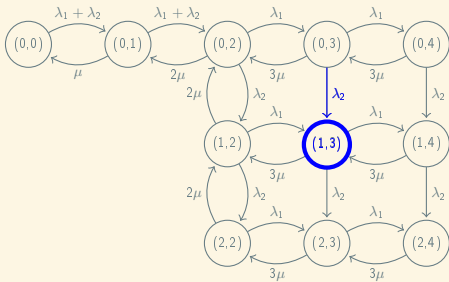
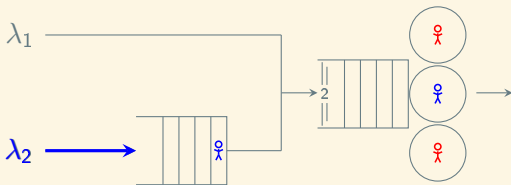
Markov Chain - Custom network



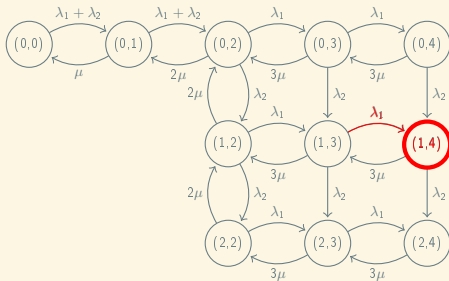
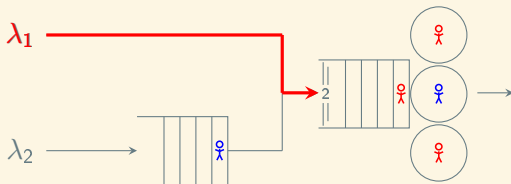
Markov Chain - Custom network



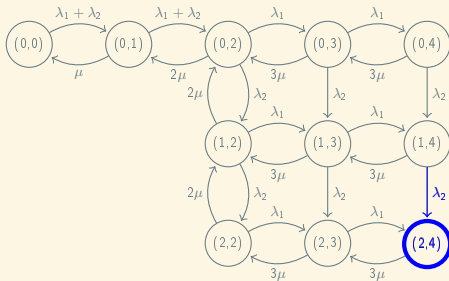
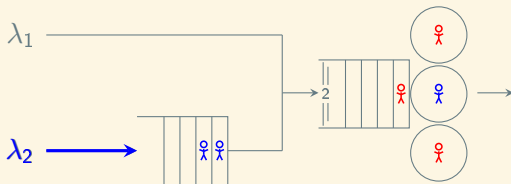
Markov Chain - Custom network



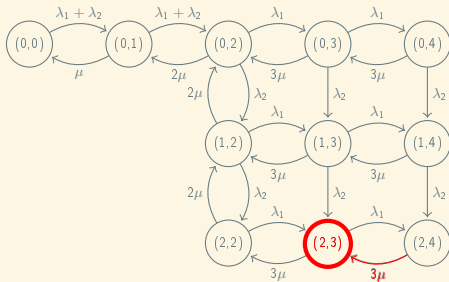
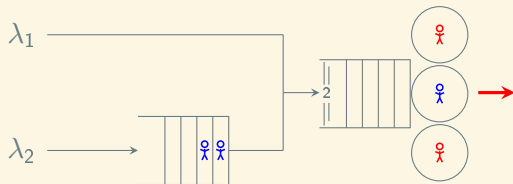
Markov Chain - Custom network



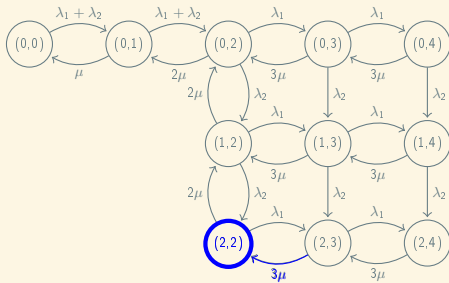
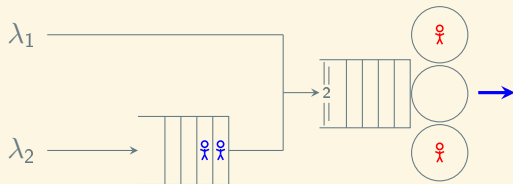
Markov Chain - Custom network



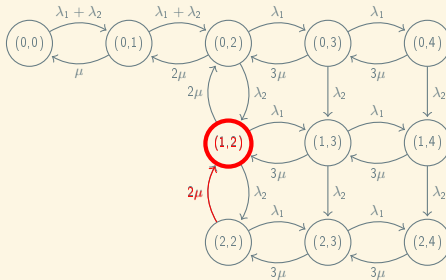
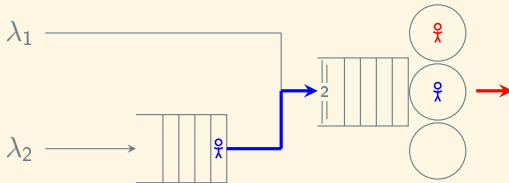
Markov Chain - Custom network



Markov Chain - Custom network

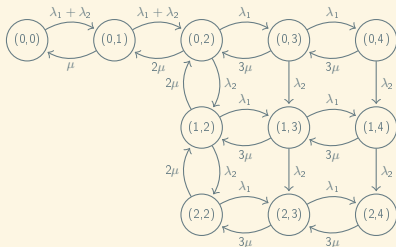


Markov Chain - Custom network



Steady state probabilities - Custom network

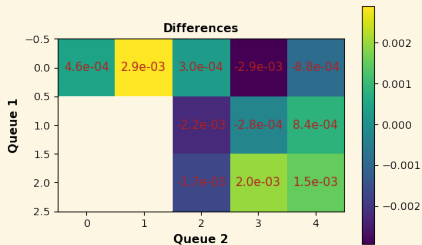
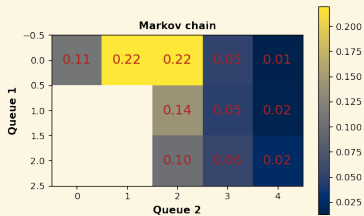
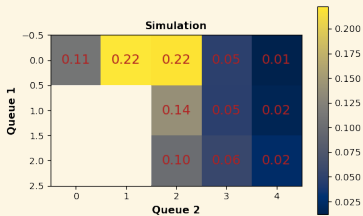
$$Q = \begin{matrix} & \begin{matrix} (0,0) & (0,1) & (0,2) & \dots & (2,3) & (2,4) \end{matrix} \\ \begin{matrix} (0,0) \\ (0,1) \\ (0,2) \\ \vdots \\ (2,3) \\ (2,4) \end{matrix} & \begin{pmatrix} -\lambda_1 - \lambda_2 & \lambda_1 + \lambda_2 & 0 & \dots & 0 & 0 \\ \mu & -\mu - \lambda_1 - \lambda_2 & \lambda_1 + \lambda_2 & \dots & 0 & 0 \\ 0 & 2\mu & -2\mu - \lambda_1 - \lambda_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\lambda_1 - 3\mu & \lambda_1 \\ 0 & 0 & 0 & \dots & 3\mu & -3\mu \end{pmatrix} \end{matrix}$$



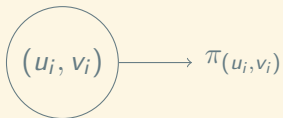
$$\frac{d\pi}{dt} = \pi Q = 0, \quad \sum \pi_{(u,v)} = 1$$

$$\pi = \begin{bmatrix} \pi(0,0) \\ \pi(0,1) \\ \pi(0,2) \\ \vdots \\ \pi(2,3) \\ \pi(2,4) \end{bmatrix}$$

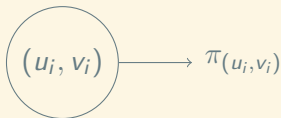
Steady state probabilities - Comparison



Performance Measures - Number of individuals



Performance Measures - Number of individuals



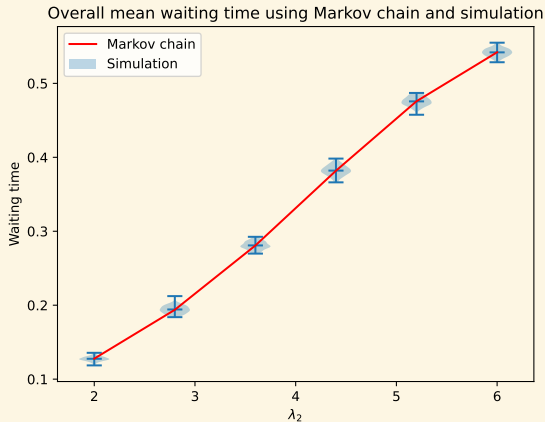
$$L = \sum_{i=1}^{|\pi|} \pi_i (u_i + v_i)$$

$$L_1 = \sum_{i=1}^{|\pi|} \pi_i u_i$$

$$L_2 = \sum_{i=1}^{|\pi|} \pi_i v_i$$

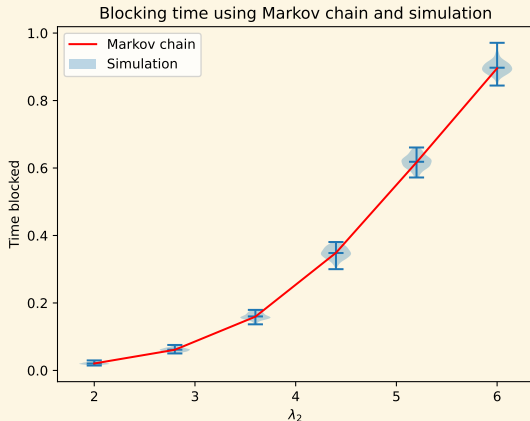
Performance Measures - Waiting time

$$W = \frac{\lambda_1 P_{L'_1}}{\lambda_2 P_{L'_2} + \lambda_1 P_{L'_1}} W^{(1)} + \frac{\lambda_2 P_{L'_2}}{\lambda_2 P_{L'_2} + \lambda_1 P_{L'_1}} W^{(2)} \quad (1)$$



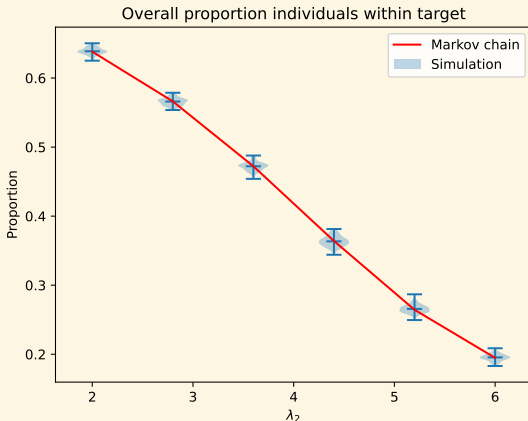
Performance Measures - Blocking time

$$B = \frac{\sum_{(u,v) \in S_A^{(2)}} \pi(u,v) b(u,v)}{\sum_{(u,v) \in S_A^{(2)}} \pi(u,v)} \quad (2)$$



Performance Measures - Proportion within time

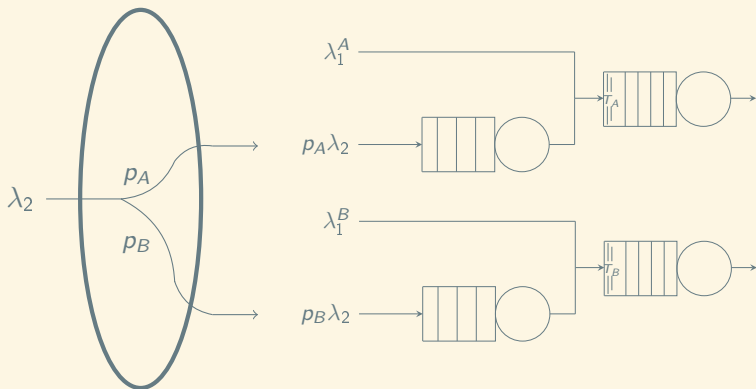
$$P(X < t) = \frac{\lambda_1 P_{L'_1}}{\lambda_2 P_{L'_2} + \lambda_1 P_{L'_1}} P(X^{(1)} < t) + \frac{\lambda_2 P_{L'_2}}{\lambda_2 P_{L'_2} + \lambda_1 P_{L'_1}} P(X^{(2)} < t) \quad (3)$$



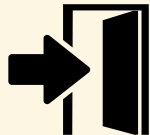
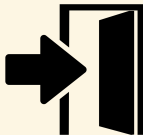
Game - Definition



Game - Players



Game - Strategies



$$p_A, p_B \in [0, 1]$$

$$T_A \in [1, N_A]$$

$$T_B \in [1, N_B]$$

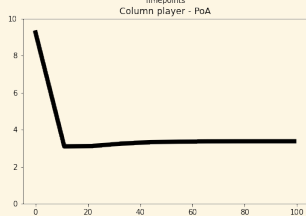
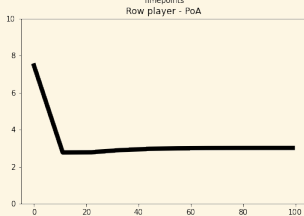
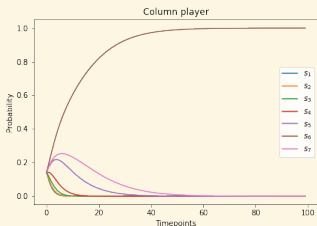
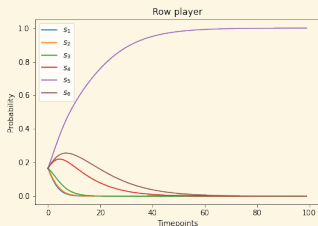
$$p_A + p_B = 1$$

Game - Formulation

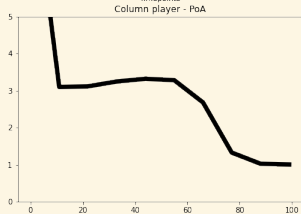
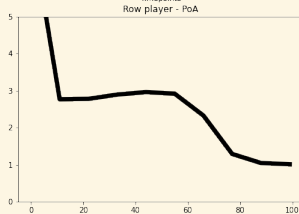
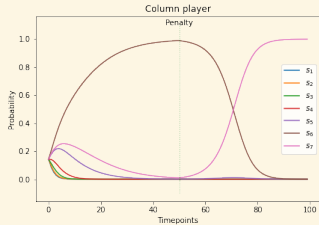
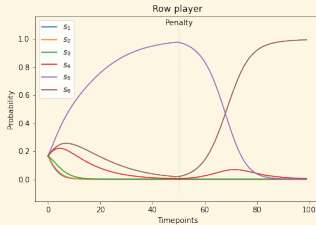
$$A = \begin{pmatrix} U_{1,1}^A & U_{1,2}^A & \cdots & U_{1,N_B}^A \\ U_{2,1}^A & U_{2,2}^A & \cdots & U_{2,N_B}^A \\ \vdots & \vdots & \ddots & \vdots \\ U_{N_A,1}^A & U_{N_A,2}^A & \cdots & U_{N_A,N_B}^A \end{pmatrix}, \quad B = \begin{pmatrix} U_{1,1}^B & U_{1,2}^B & \cdots & U_{1,N_B}^B \\ U_{2,1}^B & U_{2,2}^B & \cdots & U_{2,N_B}^B \\ \vdots & \vdots & \ddots & \vdots \\ U_{N_A,1}^B & U_{N_A,2}^B & \cdots & U_{N_A,N_B}^B \end{pmatrix}$$

$$R = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,N_B} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,N_B} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N_A,1} & p_{N_A,2} & \cdots & p_{N_A,N_B} \end{pmatrix}$$

Learning algorithms - Asymmetric replicator dynamics



Learning algorithms - Asymmetric replicator dynamics



Code / Test

```
149 > def get_probability_of_accepting(...
155 ):
156     """
157     Generates the probability of acceptance for both class types of a given
158     Markov model.
159
160     Parameters
161     -----
162     all_states : list
163     pi : numpy.array
164     threshold : int
165     system_capacity : int
166     buffer_capacity : int
167
168     Returns
169     -----
170     list
171         The probability of accepting an individual upon its arrival for class 0
172         and class 1
173     """
174     prob_accept = [
175         np.sum(
176             [
177                 pi[state]
178                 for state in all_states
179                 if is_accepting_state(
180                     state=state,
181                     class_type=class_type,
182                     threshold=threshold,
183                     system_capacity=system_capacity,
184                     buffer_capacity=buffer_capacity,
185                 )
186             ]
187         )
188         for class_type in range(2)
189     ]
190     return prob_accept
```

```
239 > def test_get_probability_of_accepting_example():
240     """
241     Test that the probability of accepting an individual is as expected
242     """
243     all_states = [
244         (0, 0),
245         (0, 1),
246         (0, 2),
247         (0, 3),
248         (1, 3),
249         (2, 3),
250         (0, 4),
251         (1, 4),
252         (2, 4),
253         (0, 5),
254         (1, 5),
255         (2, 5),
256     ]
257     pi = np.array(
258         [
259             [0.1, 0.1, 0.1, 0.1, 0.1, 0.1],
260             [np.nan, np.nan, np.nan, 0.1, 0.1, 0.05],
261             [np.nan, np.nan, np.nan, 0.05, 0.05, 0.05],
262         ]
263     )
264
265     assert np.allclose(
266         get_probability_of_accepting(
267             all_states=all_states,
268             pi=pi,
269             threshold=2,
270             system_capacity=5,
271             buffer_capacity=2,
272         ),
273         [0.8, 0.85],
274     )
```

Michalis Panayides, 3 weeks ago via PR

GitHub repository

master 19 branches 0 tags

Go to file Add file + Code +

tlmichalis11 and drvinceknight Build game theoretic model (#41)

✓ 7 checks 22 days ago 56 commits

github/workflows	Build game theoretic model (#41)	22 days ago
data	Build game theoretic model (#41)	22 days ago
exp	Restructure modules and notations in repo (#40)	8 months ago
nbs	Build game theoretic model (#41)	22 days ago
src/ambulance_game	Build game theoretic model (#41)	22 days ago
tests	Build game theoretic model (#41)	22 days ago
tex	Build game theoretic model (#41)	22 days ago
.gitignore	Build game theoretic model (#41)	22 days ago
README.md	README file edit	16 months ago
environment.yml	Build game theoretic model (#41)	22 days ago
requirements.txt	Build game theoretic model (#41)	22 days ago
setup.py	Build game theoretic model (#41)	22 days ago

README.md

Ambulance Decision Game: A python library that attempts to explore a game theoretic approach to the EMS - ED interface

Installing and running tests

Install a development version of this library with the command:

```
$ python setup.py develop
```

Run all tests developed by first installing pytest (pip install pytest) and then executing the command:

```
$ pytest -
```