Emergent Behaviour in Stochastic Queueing Systems

Vincent Knight, *Geraint Palmer*, & Thomas Watson

28th November 2024

Overview

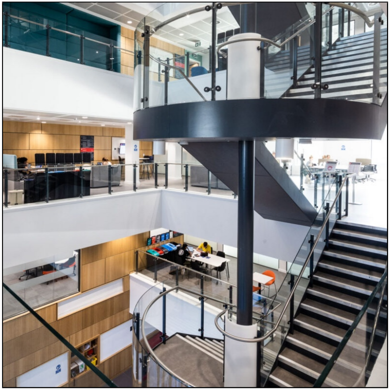
i. Motivation - Abacws
ii. Background to game theory & emergent behaviour
iii. Discrete Event Simulation
iv. Discrete Event Population Updates
v. Recovering known results
vi. Abacws jockeying scenario

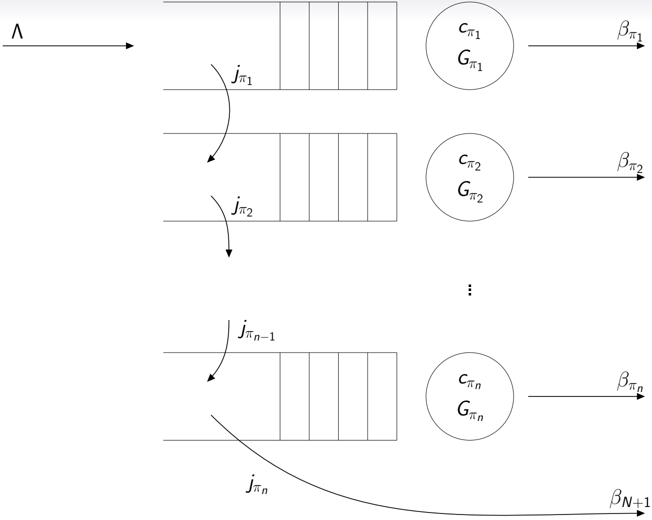
Motivation & Inspiration

Using Replicator Dynamics to Model Seating Strategies in Abacws

Anthony Holman, Katherine Zverovich, Megan Allen and Rachel Banks

April 2024





Λ
c_{π_1}
G_{π_1}
β_{π_1}
j_{π_1}
c_{π_2}
G_{π_2}
β_{π_2}
j_{π_2}
$\vdots$
$j_{\pi_{n-1}}$
c_{π_n}
G_{π_n}
β_{π_n}
j_{π_n}
β_{N+1}

$$s = (\pi, j)$$

$$s = (\pi = (3, 1, 2, 1),\ j = (0.5, 1.2, 0.4, 3.3))$$

Corresponds to:

- First queue at node 3, and wait a maximum of 0.5 time units;
- If still waiting after for service 0.5 time units, jockey to node 1, and wait a maximum of 1.2 time units;
- If still waiting after for service 1.2 time units, jockey to node 2, and wait a maximum of 0.4 time units;
- If still waiting after for service 0.4 time units, jockey to node 1, and wait a maximum of 3.3 time units;
- If still waiting after for service 3.3 time units, renege from the system.

Fitness of a Strategy

$$f_s = e^{-\kappa \mathbb{E}(C)} = e^{-\kappa\left(\mathbb{E}(L)\beta + \sum_{k=0}^{K} (\mathbb{E}(W_{\pi_k}) + \mathbb{E}(T_{\pi_k}))\right)}$$

Where:

- κ is the selection intensity,
- L is a binary variable indicating if the customer was lost,
- β_{N+1} is the cost of being lost,
- β_{π_k} is the cost of being served at π_k,
- W_{π_k} is the waiting time at node π_k,
- $T_{\pi_k} = \epsilon g_{\pi_k} + \beta_{\pi_k}$ service time plus cost of service at node π_k, if they were served there, 0 otherwise.

Travel Times between Service Centres

$$c_1 = 2, \quad c_2 = 5$$

- $s_1 = ((1, 2), (5, 5))$
- $s_2 = ((2, 1), (6, 8))$
- $s_3 = ((2, 1, 2), (4, 3, 3))$

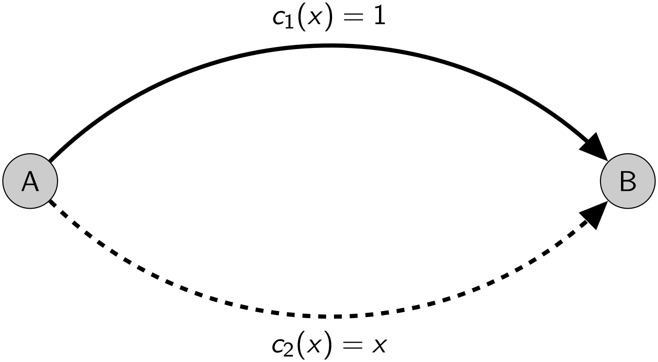
$$T = \begin{pmatrix} 0 & 10 \\ 7 & 0 \end{pmatrix}$$

$$c_1 = 2, \quad c_2 = 5, \quad c_3 = 0$$

- $s_1 = ((1, 3, 2), (5, 10, 5))$
- $s_2 = ((2, 3, 1), (6, 7, 8))$
- $s_3 = ((2, 3, 1, 3, 2), (4, 7, 3, 10, 3))$

i. Motivation - Abacws
ii. Background to game theory & emergent behaviour
iii. Discrete Event Simulation
iv. Discrete Event Population Updates
v. Recovering known results
vi. Abacws jockeying scenario

Example of a Game - Pigou's Routing Game



x = proportion taking the shortcut

Social (overall) Optimum

Strategy $\hat{x}$ that minimises the sum of everyone's travel times:

$$\begin{aligned}\hat{x} &= \arg\min_x \; (1-x)c_1(x) + xc_2(x) \\ &= \arg\min_x \; x^2 - x + 1 \\ &= \frac{1}{2}\end{aligned}$$

Social (overall) Optimum

Strategy $\hat{x}$ that minimises the sum of everyone's travel times:

$$\begin{aligned}\hat{x} &= \arg\min_x \ (1-x)c_1(x) + xc_2(x) \\ &= \arg\min_x \ x^2 - x + 1 \\ &= \frac{1}{2}\end{aligned}$$

Selfish Equilibrium

Strategy $x^\star$ that causes no reason to move, when travel times are equal:

$$\begin{aligned}c_1(x^\star) &= c_2(x^\star) \\ 1 &= x^\star\end{aligned}$$

Example of a Game - Stag Hunt

		Column Player	
		Stag	Hare
Row Player	Stag	(10, 10)	(1, 8)
	Hare	(8, 1)	(5, 5)

Replicator Dynamics

If there is a population of players, with proportion x_s playing strategy s, all playing against each other, then:

$$\frac{dx_s}{dt} = x_s \left(f_s - \phi \right) \quad \text{for all } s \in S$$

where $\phi = \sum_{s \in S} x_s f_s$ is the population's average fitness.

A stable population is when $\frac{dx_s}{dt} = 0$ for all $s \in S$.

Replicator Dynamics - Stag Hunt

Proportion of stag hunters $= y$, Proportion of hare hungers $= 1 - y$

$$f_{\text{stag}} = (10)(y) + (1)(1-y) = 9y + 1$$
$$f_{\text{hare}} = (1)(y) + (5)(1-y) = -4y + 5$$

$$\begin{aligned} \phi &= y f_{\text{stag}} + (1-y) f_{\text{hare}} \\ &= y(9y+1) + (1-y)(-4y+5) \\ &= 13y^2 - 8y + 5 \end{aligned}$$

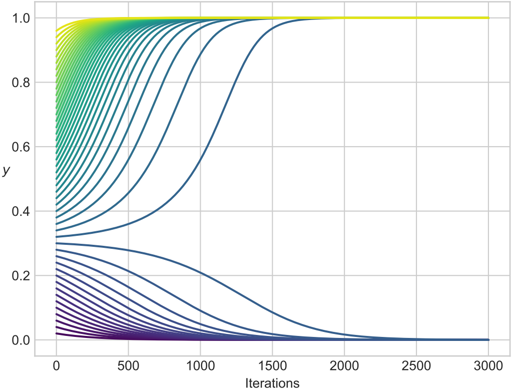
$$\begin{aligned} \frac{dy}{dt} &= y(f_{stag} - \phi) \\ &= y\left((9y+1) - (13y^2 - 8y + 5)\right) \\ &= -13y^3 + 17y^2 - 4y \end{aligned}$$

$$y = 0, \quad y = \frac{4}{13}, \quad y = 1$$

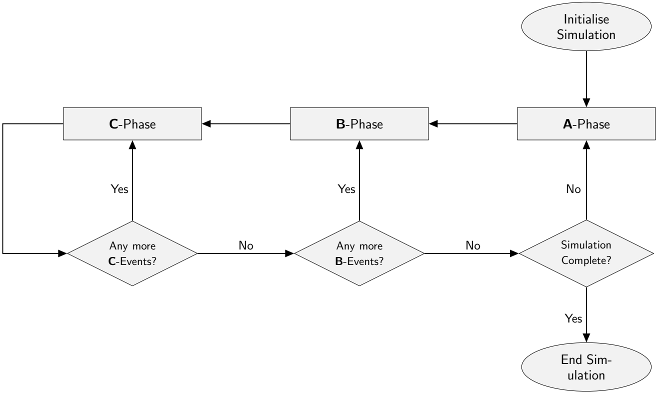
Numerical Methods

e.g. Euler method:

$$y_{t+1} = y_t + \frac{dy}{dt}\Delta t$$

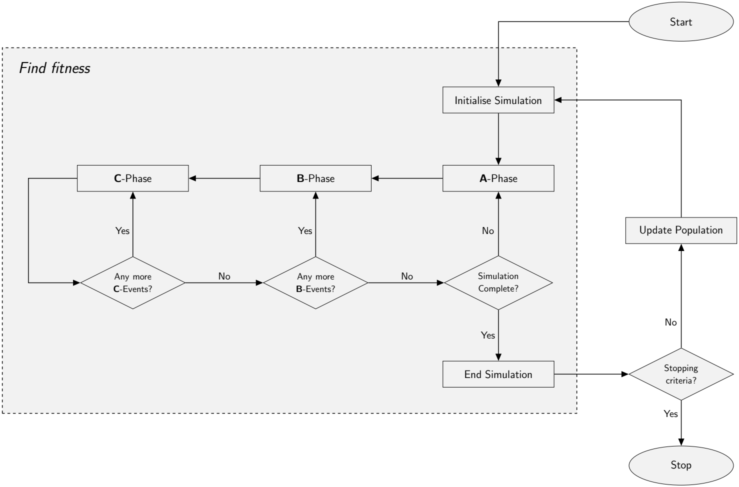


Discrete Event Simulation

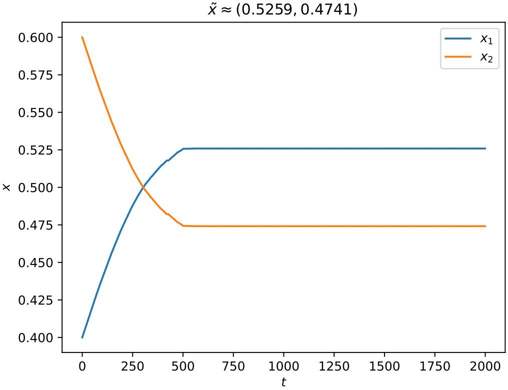


```
>>> import ciw
>>> N = ciw.create_network(
...     arrival_distributions=[ciw.dists.Exponential(5)],
...     service_distributions=[ciw.dists.Exponential(2)],
...     number_of_servers=[4]
... )
>>> ciw.seed(0)
>>> Q = ciw.Simulation(N)
>>> Q.simulate_until_max_time(1000)
>>> recs = Q.get_all_records()
>>> waits = [r.waiting_time for r in recs if r.arrival_date > 100]
>>> sum(waits) / len(waits)
0.0957353996505342
```

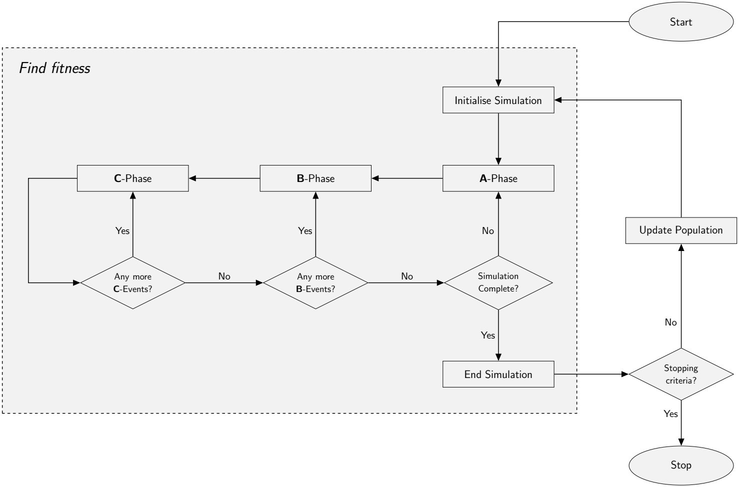
Numerical Method with Discrete Event Simulation



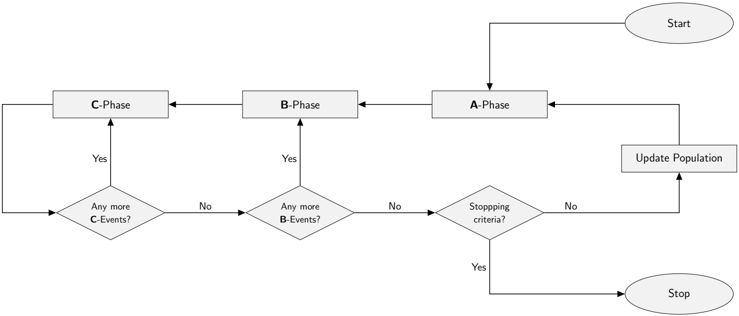
Jockeying example:



Numerical Method with Discrete Event Simulation

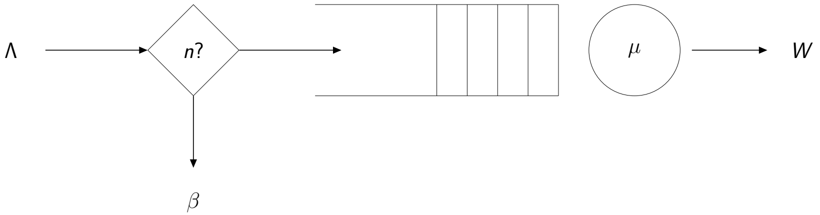


DEPU - Discrete Event Population Updates



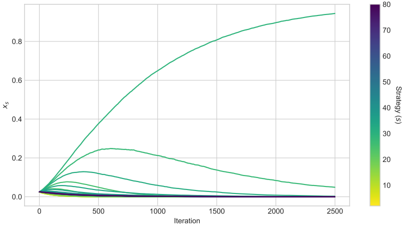
$$f_{s,i+1} \leftarrow (1-\alpha) f_{s,i} + \alpha f^{\star}$$

Naor (1969)



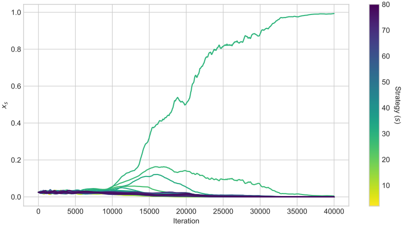
Emergent strategy: $n = \beta\mu$

Numerical Method + Simulation



(≈ 5 hours)

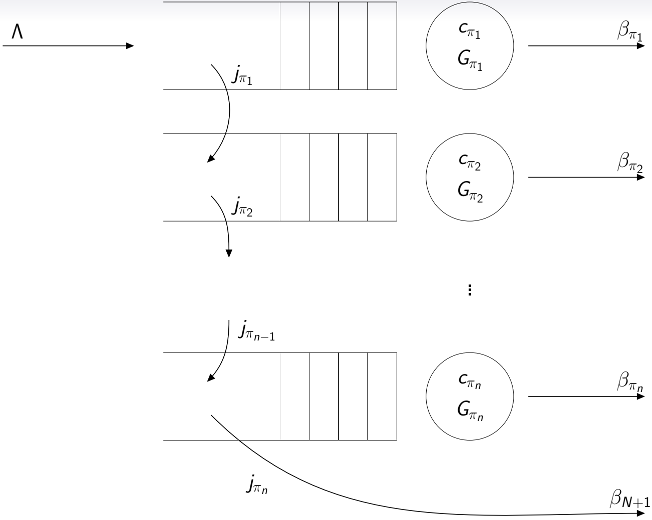
DEPU



(≈ 5 minutes)

$$\Lambda = 30,\ \mu = 20,\ \beta = 1.5$$

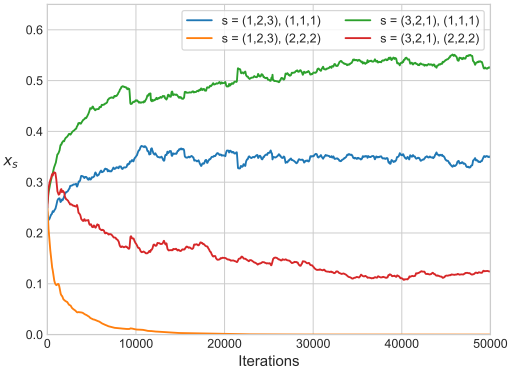
i. Motivation - Abacws
ii. Background to game theory & emergent behaviour
iii. Discrete Event Simulation
iv. Discrete Event Population Updates
v. Recovering known results
vi. Abacws jockeying scenario


Λ
c_{π_1}
G_{π_1}
β_{π_1}
j_{π_1}
c_{π_2}
G_{π_2}
β_{π_2}
j_{π_2}
$j_{\pi_{n-1}}$
c_{π_n}
G_{π_n}
β_{π_n}
j_{π_n}
β_{N+1}

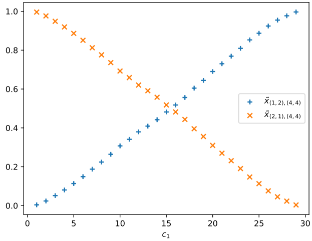
$$s = (\pi, j)$$

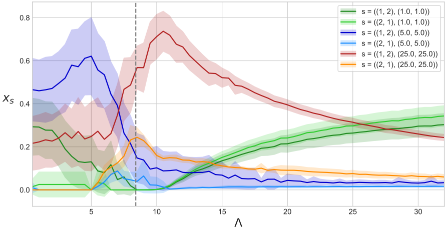
Applying DEPU

- $\Lambda = 32$,
- $c_1 = 1$, $c_2 = 3$, $c_3 = 1$,
- $\mu_1 = 4$, $\mu_2 = 8$, $\mu_3 = 12$,
- $\beta_1 = \beta_2 = \beta_3 = 0$,
- $\beta_4 = 5$,
- $\epsilon = 0$,
- $\kappa = 0.2$,
- $s_1 = ((1, 2, 3), (1, 1, 1)$,
- $s_2 = ((1, 2, 3), (2, 2, 2)$,
- $s_3 = ((3, 2, 1), (1, 1, 1)$,
- $s_4 = ((3, 2, 1), (2, 2, 2)$,
- $\Delta t = 0.01$,
- $\alpha = 0.1$,



Effect of Parameters





Next steps...

- Investigate other update rules
- Incorporate variability in fitness function