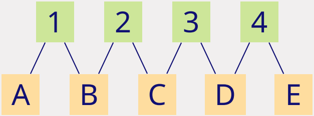
EINDHOVEN UNIVERSITY OF TECHNOLOGY

Performance Evaluation of Stochastic Non-Bipartite Matching Models

Céline Comte — c.m.comte@tue.nl — Eindhoven University of Technology
Jan-Pieter Dorsman — j.l.dorsman@uva.nl — University of Amsterdam

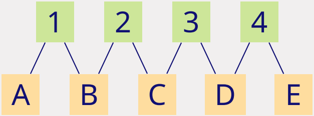
ASMTA 2021 — December 14, 2021

Stochastic bipartite matching model



Bipartite graph $G = (\mathcal{I}, \mathcal{K}, \mathcal{E})$

Stochastic bipartite matching model

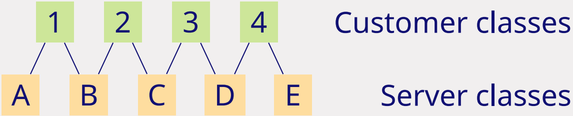


Customer classes

Bipartite graph $G = (\mathcal{I}, \mathcal{K}, \mathcal{E})$ with

- $\mathcal{I} \rightsquigarrow$ "customer" or "demand" classes

Stochastic bipartite matching model

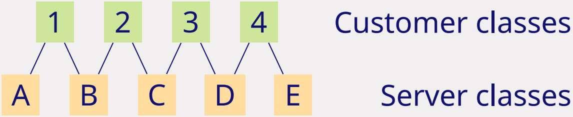


Bipartite graph $G = (\mathcal{I}, \mathcal{K}, \mathcal{E})$ with

- $\mathcal{I} \rightsquigarrow$ "customer" or "demand" classes
- $\mathcal{K} \rightsquigarrow$ "server" or "supply" classes

TU/e

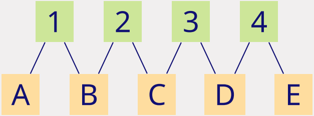
Stochastic bipartite matching model



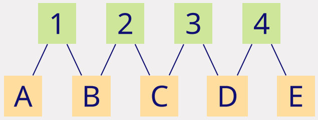
Bipartite graph $G = (\mathcal{I}, \mathcal{K}, \mathcal{E})$ with

- $\mathcal{I} \rightsquigarrow$ “customer” or “demand” classes
- $\mathcal{K} \rightsquigarrow$ “server” or “supply” classes
- $\mathcal{E} \rightsquigarrow$ authorized matchings

Infinite bipartite matching model



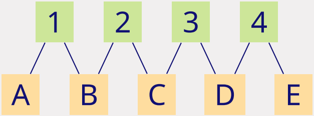
Infinite bipartite matching model



1 1 2 3 2 4 1 $\cdots$

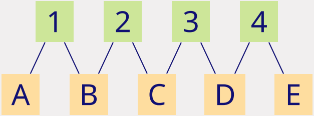
- Sequence of i.i.d. customer classes: class *i* with probability $\lambda_i, i \in \mathcal{I}$

Infinite bipartite matching model



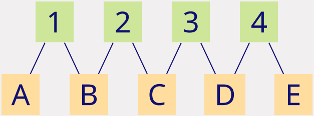
- Sequence of i.i.d. customer classes: class i with probability $\lambda_i, i \in \mathcal{I}$
- Sequence of i.i.d. server classes: class k with probability $\mu_k, k \in \mathcal{K}$

Infinite bipartite matching model



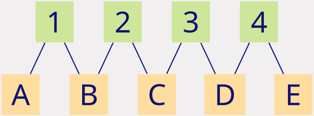
- Sequence of i.i.d. customer classes: class i with probability $\lambda_i, i \in \mathcal{I}$
- Sequence of i.i.d. server classes: class k with probability $\mu_k, k \in \mathcal{K}$
- First-come-first-matched policy

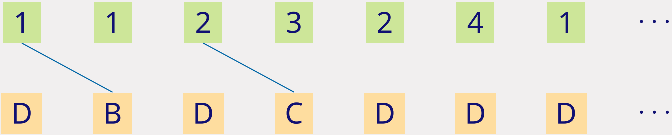
Infinite bipartite matching model



- Sequence of i.i.d. customer classes: class i with probability $\lambda_i, i \in \mathcal{I}$
- Sequence of i.i.d. server classes: class k with probability $\mu_k, k \in \mathcal{K}$
- First-come-first-matched policy

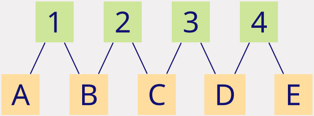
Infinite bipartite matching model

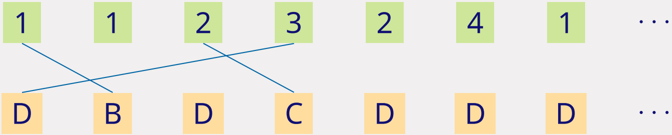




- Sequence of i.i.d. customer classes: class *i* with probability $\lambda_i, i \in \mathcal{I}$
- Sequence of i.i.d. server classes: class *k* with probability $\mu_k, k \in \mathcal{K}$
- First-come-first-matched policy

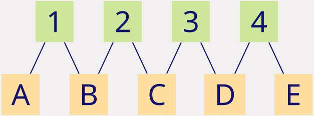
Infinite bipartite matching model

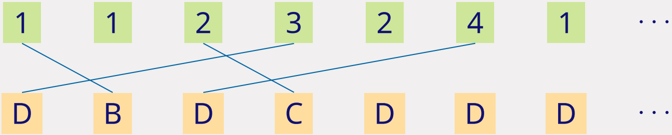




- Sequence of i.i.d. customer classes: class *i* with probability $\lambda_i, i \in \mathcal{I}$
- Sequence of i.i.d. server classes: class *k* with probability $\mu_k, k \in \mathcal{K}$
- First-come-first-matched policy

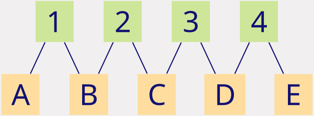
Infinite bipartite matching model

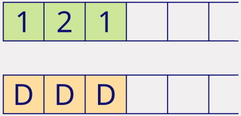




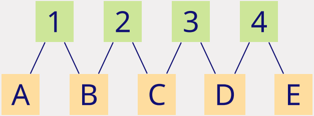
- Sequence of i.i.d. customer classes: class *i* with probability $\lambda_i, i \in \mathcal{I}$
- Sequence of i.i.d. server classes: class *k* with probability $\mu_k, k \in \mathcal{K}$
- First-come-first-matched policy

What a queueing theorist sees...

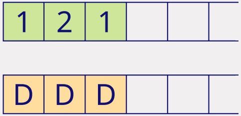




What a queueing theorist sees...

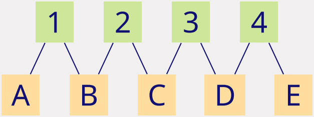


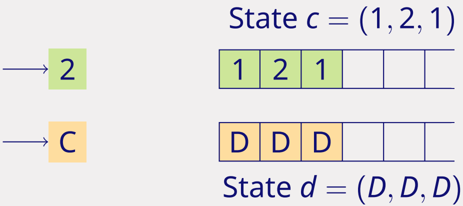
State $c = (1, 2, 1)$



State $d = (D, D, D)$

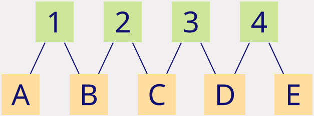
What a queueing theorist sees...



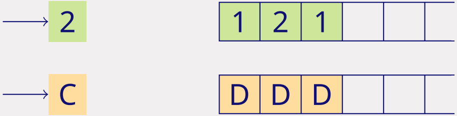


- At each time slot, reveal the next customer *and* the next server:
  - The customer belongs to class i with probability λ_i.
  - The server belongs to class k with probability μ_k.

What a queueing theorist sees...



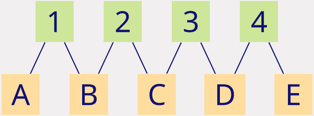
State $c = (1, 2, 1)$

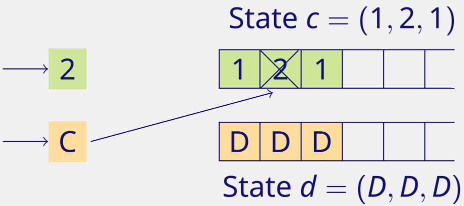


State $d = (D, D, D)$

- At each time slot, reveal the next customer *and* the next server:
  - The customer belongs to class i with probability λ_i.
  - The server belongs to class k with probability μ_k.
- First-come-first-matched service policy.

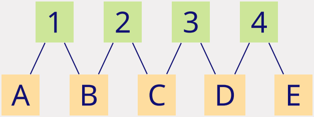
What a queueing theorist sees...

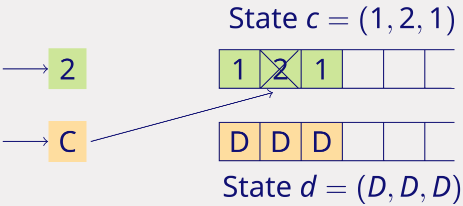




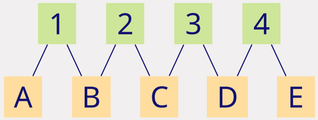
- At each time slot, reveal the next customer *and* the next server:
  - The customer belongs to class i with probability λ_i.
  - The server belongs to class k with probability μ_k.
- First-come-first-matched service policy.

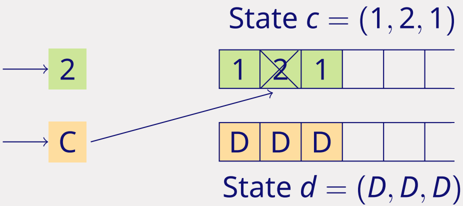
What a queueing theorist sees...





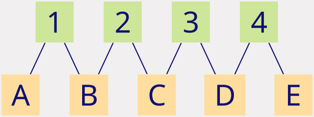
What a queueing theorist sees...

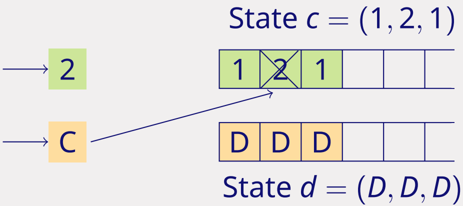




- There are always as many customers as servers in the queue.

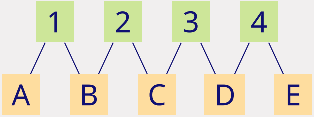
What a queueing theorist sees...



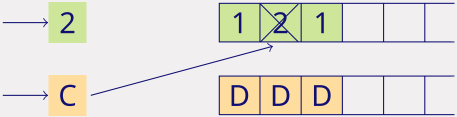


- There are always as many customers as servers in the queue.
- The set $\mathcal{A}$ of unmatched item classes satisfies:
    - $\mathcal{A}$ is an independent set of the graph G
    - $\mathcal{A} \cap \mathcal{I} \neq \emptyset$ if and only if $\mathcal{A} \cap \mathcal{K} \neq \emptyset$

What a queueing theorist sees...



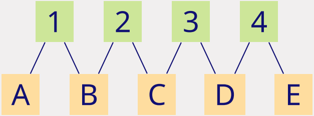
State $c = (1, 2, 1)$



State $d = (D, D, D)$

$\{1, C\}, \{1, D\}, \{1, E\}, \{1, C, D\}, \{1, C, E\},$
$\{1, D, E\}, \dots, \{1, 2, D\}, \{1, 2, E\}, \{1, 2, D, E\},$
$\{1, 3, E\}, \{1, 4, C\}, \dots$

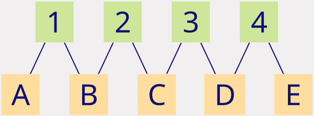
- There are always as many customers as servers in the queue.
- The set $\mathcal{A}$ of unmatched item classes satisfies:
  - $\mathcal{A}$ is an independent set of the graph G
  - $\mathcal{A} \cap \mathcal{I} \neq \emptyset$ if and only if $\mathcal{A} \cap \mathcal{K} \neq \emptyset$



Related works and contributions

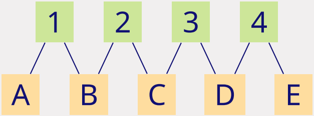
- Model introduction (Caldentey, Kaplan, and Weiss, 2009)

Related works and contributions



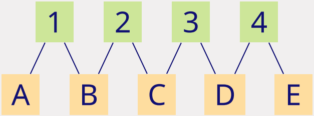
- Model introduction (Caldentey, Kaplan, and Weiss, 2009)
- Necessary and sufficient stability condition (Bušić, Gupta, and Mairesse, 2013)

Related works and contributions



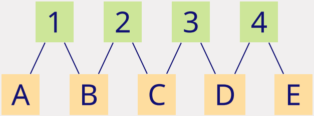
- Model introduction (Caldentey, Kaplan, and Weiss, 2009)
- Necessary and sufficient stability condition (Bušić, Gupta, and Mairesse, 2013)
- Performance evaluation
  - (Adan and Weiss, 2012)
  - (Adan, Bušić, Mairesse, and Weiss, 2017)

Related works and contributions



- Model introduction (Caldentey, Kaplan, and Weiss, 2009)
- Necessary and sufficient stability condition (Bušić, Gupta, and Mairesse, 2013)
- Performance evaluation
  - (Adan and Weiss, 2012)
  - (Adan, Bušić, Mairesse, and Weiss, 2017)
- Optimization and learning (Cadas, 2021)

Performance evaluation

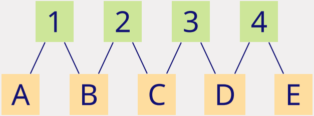


- **Stationary distribution** of the set of unmatched item classes

$$\Delta(\mathcal{A})\pi(\mathcal{A}) = \mu(\mathcal{A}\cap\mathcal{K}) \sum_{i\in\mathcal{A}\cap\mathcal{I}} \lambda_i \pi(\mathcal{A}\backslash\{i\}) + \lambda(\mathcal{A}\cap\mathcal{I}) \sum_{k\in\mathcal{A}\cap\mathcal{K}} \mu_k \pi(\mathcal{A}\backslash\{k\})$$
$$+ \sum_{i\in\mathcal{A}\cap\mathcal{I}} \sum_{k\in\mathcal{A}\cap\mathcal{K}} \lambda_i \mu_k \pi(\mathcal{A}\backslash\{i,k\}), \quad \text{if } \mathcal{A} \text{ is non-empty,}$$

where $\Delta(\mathcal{A}) = \mu(\mathcal{K}(\mathcal{A}\cap\mathcal{I}))\lambda(\mathcal{I}(\mathcal{A}\cap\mathcal{K})) - \lambda(\mathcal{A}\cap\mathcal{I})\mu(\mathcal{A}\cap\mathcal{K})$.

Performance evaluation

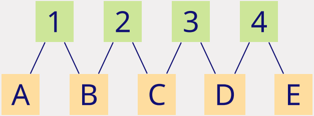


- **Stationary distribution** of the set of unmatched item classes

$$\Delta(\mathcal{A})\pi(\mathcal{A}) = \mu(\mathcal{A} \cap \mathcal{K}) \sum_{i \in \mathcal{A} \cap \mathcal{I}} \lambda_i \pi(\mathcal{A} \backslash \{i\}) + \lambda(\mathcal{A} \cap \mathcal{I}) \sum_{k \in \mathcal{A} \cap \mathcal{K}} \mu_k \pi(\mathcal{A} \backslash \{k\}) + \sum_{i \in \mathcal{A} \cap \mathcal{I}} \sum_{k \in \mathcal{A} \cap \mathcal{K}} \lambda_i \mu_k \pi(\mathcal{A} \backslash \{i, k\}), \quad \text{if } \mathcal{A} \text{ is non-empty,}$$

where $\Delta(\mathcal{A}) = \mu(\mathcal{K}(\mathcal{A} \cap \mathcal{I}))\lambda(\mathcal{I}(\mathcal{A} \cap \mathcal{K})) - \lambda(\mathcal{A} \cap \mathcal{I})\mu(\mathcal{A} \cap \mathcal{K})$.
The value of the **normalization constant** $\pi(\emptyset)$ follows by normalization.

Performance evaluation

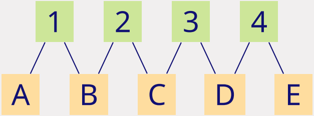


- **Stationary distribution** of the set of unmatched item classes

$$\Delta(\mathcal{A})\pi(\mathcal{A}) = \mu(\mathcal{A} \cap \mathcal{K}) \sum_{i \in \mathcal{A} \cap \mathcal{I}} \lambda_i \pi(\mathcal{A} \backslash \{i\}) + \lambda(\mathcal{A} \cap \mathcal{I}) \sum_{k \in \mathcal{A} \cap \mathcal{K}} \mu_k \pi(\mathcal{A} \backslash \{k\}) + \sum_{i \in \mathcal{A} \cap \mathcal{I}} \sum_{k \in \mathcal{A} \cap \mathcal{K}} \lambda_i \mu_k \pi(\mathcal{A} \backslash \{i, k\}), \quad \text{if } \mathcal{A} \text{ is non-empty,}$$

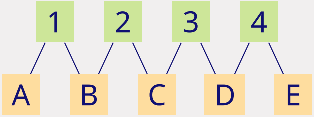
  where $\Delta(\mathcal{A}) = \mu(\mathcal{K}(\mathcal{A} \cap \mathcal{I}))\lambda(\mathcal{I}(\mathcal{A} \cap \mathcal{K})) - \lambda(\mathcal{A} \cap \mathcal{I})\mu(\mathcal{A} \cap \mathcal{K})$.
  The value of the **normalization constant** $\pi(\emptyset)$ follows by normalization.
- Similar expressions for waiting probability, mean waiting time...

Discussion



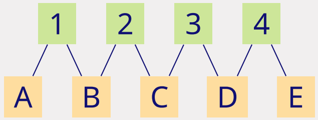
- **Time complexity.** $O(I \cdot K \cdot ((I + K) \cdot M) + N)$, where
  - I = number of customer classes,
  - K = number of server classes,
  - M = number of maximal independent sets,
  - N = number of independent sets.

Discussion



- **Time complexity.** $O(I \cdot K \cdot ((I + K) \cdot M) + N)$, where
  - $I =$ number of customer classes,
  - $K =$ number of server classes,
  - $M =$ number of maximal independent sets,
  - $N =$ number of independent sets.
- **Flexibility.** This approach can be easily adapted to derive other performance metrics (e.g., matching rates, mean length of a busy sequence).

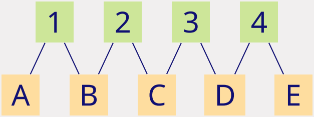
Numerical results



Numerical results

$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$

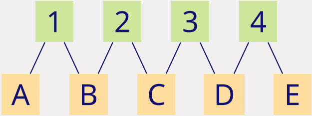
$\mu_A = \frac{\rho}{4} \qquad \mu_B = \mu_C = \mu_D = \frac{1}{4} \qquad \mu_E = \frac{1-\rho}{4}$

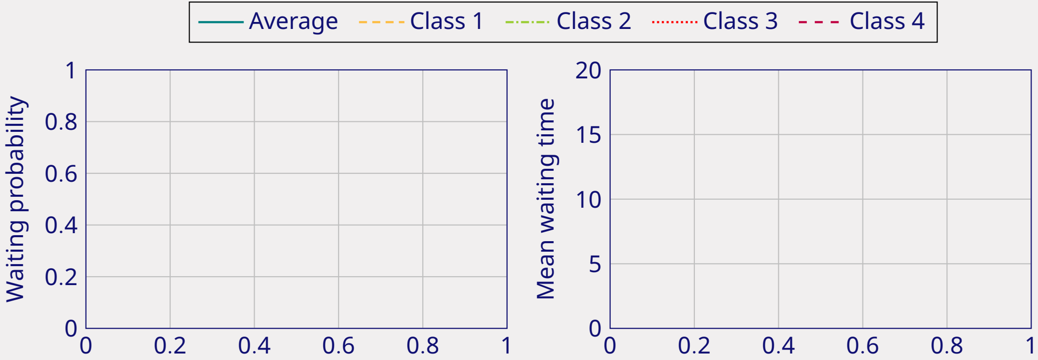


Numerical results

$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$

$\mu_A = \frac{\rho}{4} \quad \mu_B = \mu_C = \mu_D = \frac{1}{4} \quad \mu_E = \frac{1-\rho}{4}$



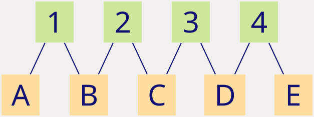


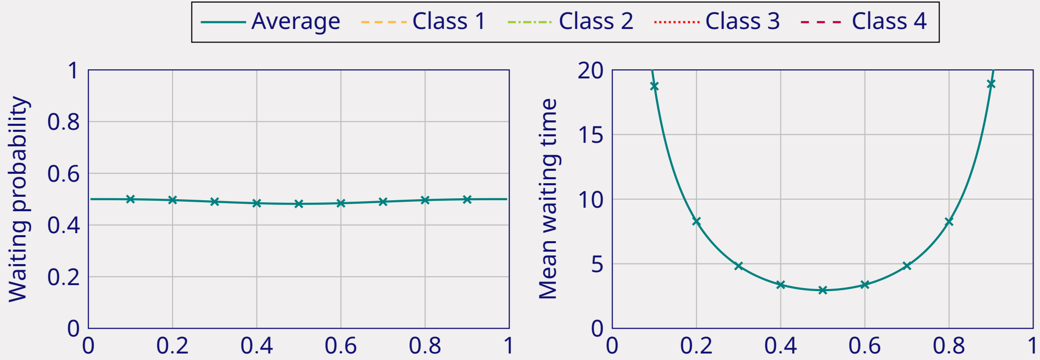
TU/e

Numerical results

$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$

$\mu_A = \frac{\rho}{4} \qquad \mu_B = \mu_C = \mu_D = \frac{1}{4} \qquad \mu_E = \frac{1-\rho}{4}$

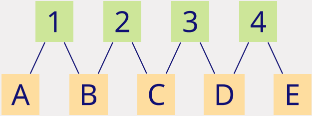


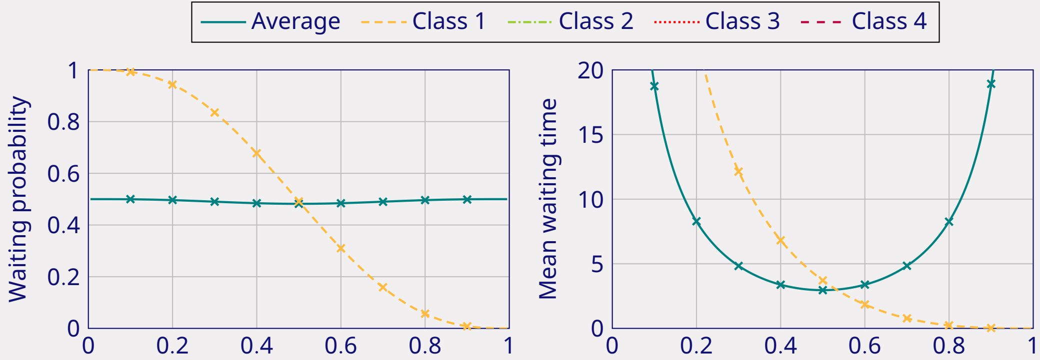


Numerical results

$\lambda_1=\lambda_2=\lambda_3=\lambda_4=\frac{1}{4}$

$\mu_A=\frac{\rho}{4}$ $\mu_B=\mu_C=\mu_D=\frac{1}{4}$ $\mu_E=\frac{1-\rho}{4}$

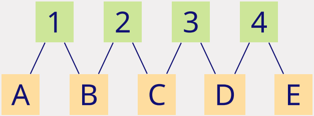


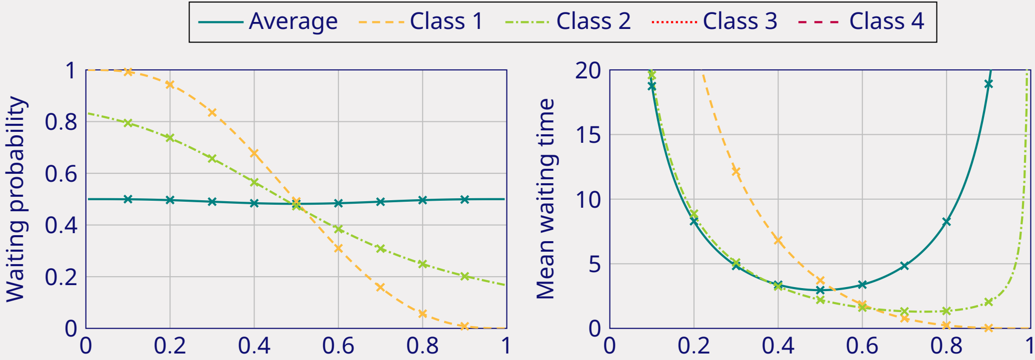


Numerical results

$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$

$\mu_A = \frac{\rho}{4} \quad \mu_B = \mu_C = \mu_D = \frac{1}{4} \quad \mu_E = \frac{1-\rho}{4}$

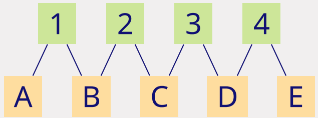


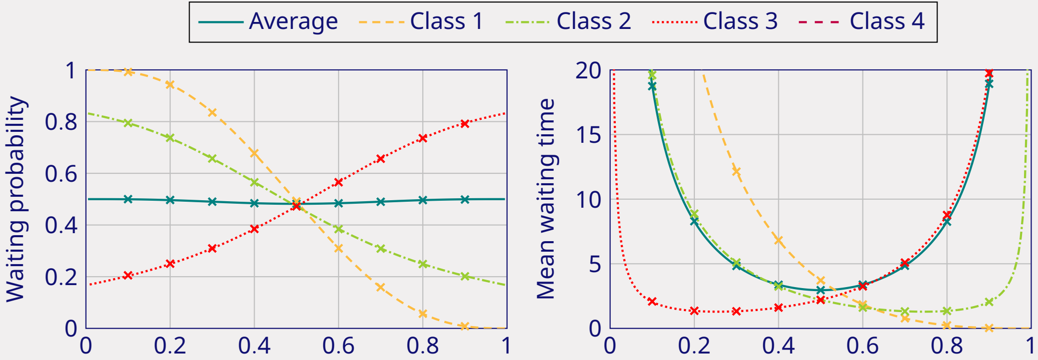


Numerical results

$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$

$\mu_A = \frac{\rho}{4} \qquad \mu_B = \mu_C = \mu_D = \frac{1}{4} \qquad \mu_E = \frac{1-\rho}{4}$



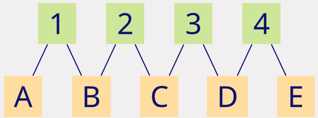


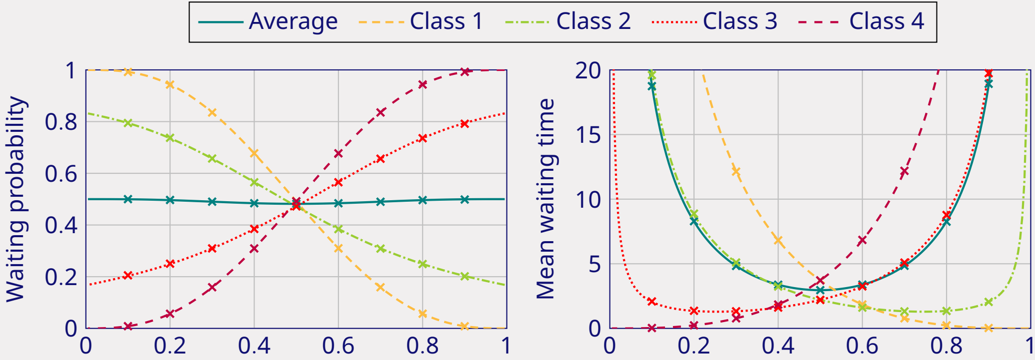
TU/e

Numerical results

$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$

$\mu_A = \frac{\rho}{4} \quad \mu_B = \mu_C = \mu_D = \frac{1}{4} \quad \mu_E = \frac{1-\rho}{4}$

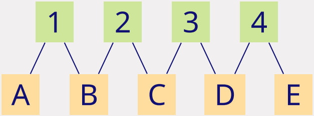


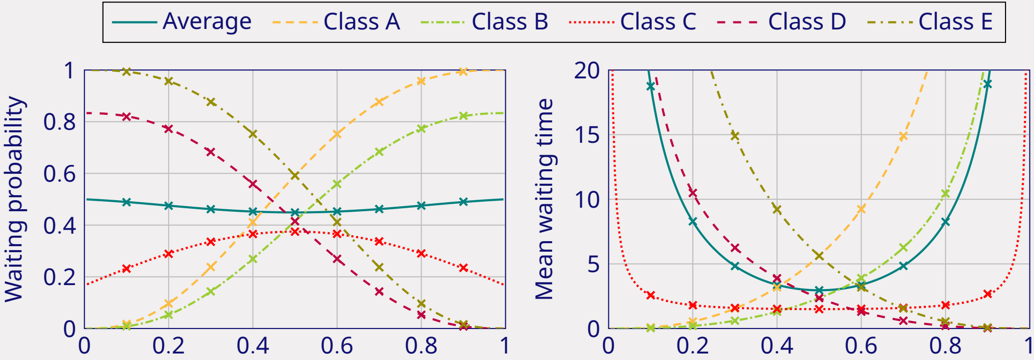


Numerical results

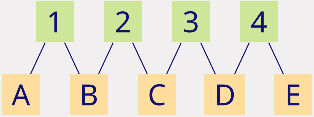
$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$

$\mu_A = \frac{\rho}{4} \quad \mu_B = \mu_C = \mu_D = \frac{1}{4} \quad \mu_E = \frac{1-\rho}{4}$



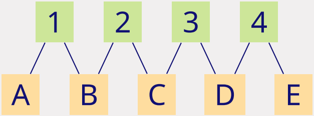


Conclusion



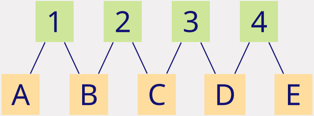
- New closed-form expressions for performance metrics in the stochastic bipartite matching model.

Conclusion



- New closed-form expressions for performance metrics in the stochastic bipartite matching model.
- Numerical evaluations on toy examples.

Conclusion



- New closed-form expressions for performance metrics in the stochastic bipartite matching model.
- Numerical evaluations on toy examples.
- Self-advertising ☺ $\rightsquigarrow$ (Comte, Stochastic Models, 2021)
  Similar expressions for the stochastic non-bipartite matching model
  (with additional comments on order-independent queues!)