Solving Nonstationary Non-Markovian Queueing Systems

A Transfer Learning Neural Network Approach

Yunan Liu

Principal Research Scientist, Supply Chain Optimization Technology

Joint work with Spyros Garyfallos, Amazon Connect, AWS

AMLC 2025

Queueing Theory

Long waiting line at the AMLC registration desk.

- A mathematical study of waiting lines or queues (a branch of operations research).
- Applications: customer call centers, healthcare, manufacturing systems, transportation, cloud computing, food services, etc.
- Entities in a queueing system:
  - Customers: consumers, patients, jobs, data packets, riders, etc.
  - Servers: agents, doctors, nurses, beds, machines, cloud, drivers, etc.

Real-World Service Systems

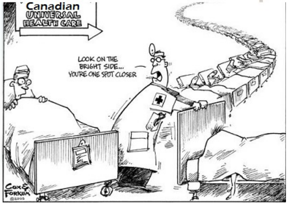
Complex dynamics:

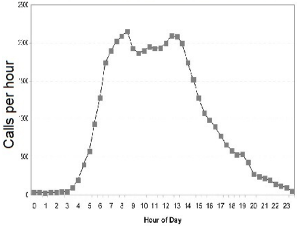
- **Time-varying demand:** Customer arrivals fluctuate throughout the day
- **Time-varying capacity:** Staffing levels adjust to match demand
- **Non-Markovian structure:** Service and abandonment times don't follow exponential distributions
- **Multiple performance objectives:** Need to satisfy concurrent service-level constraints
- **Others:** Complex network topologies, multiple customer classes, customer/server behavior.

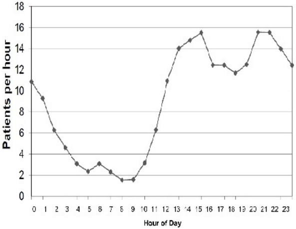
Key Question:

- **Performance prediction:** *How can we accurately predict system performance in such complex systems?*
- **Capacity planning:** *How can we optimally plan service capacity to meet service-level targets under nonstationary non-Markovian dynamics?*
- **Dynamic control:** *What real-time actions should be taken when service-level constraints are at risk of being violated?*

Time-Varying Demand

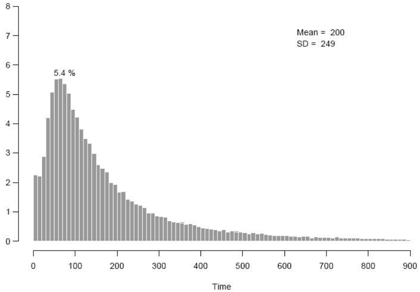


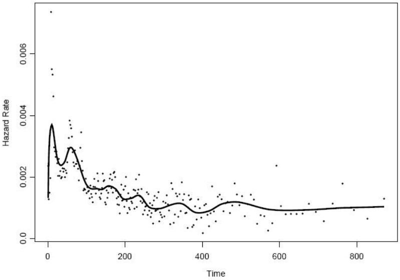




- **Call centers:** Peak hours during business days, lunch breaks
- **Healthcare:** Emergency department arrivals follow daily patterns
- **Cloud services:** Traffic varies by time of day, day of week

Non-Markovian Distributions

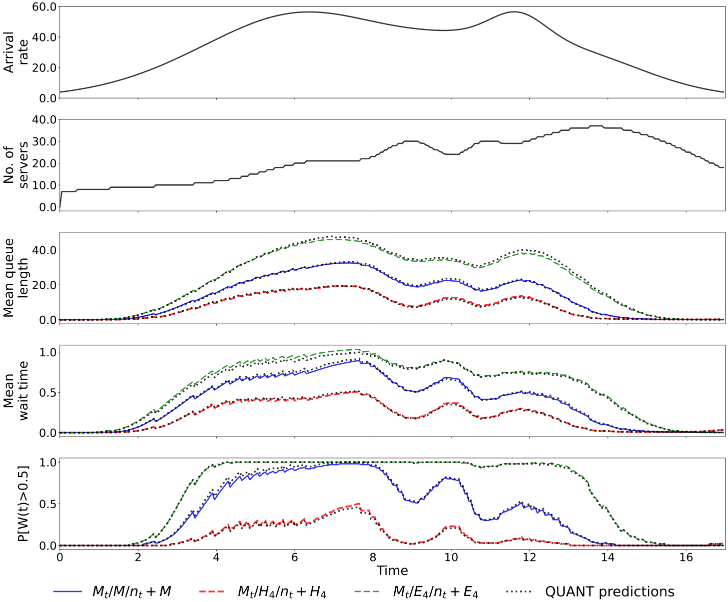




Real service and abandonment times are NOT exponential:

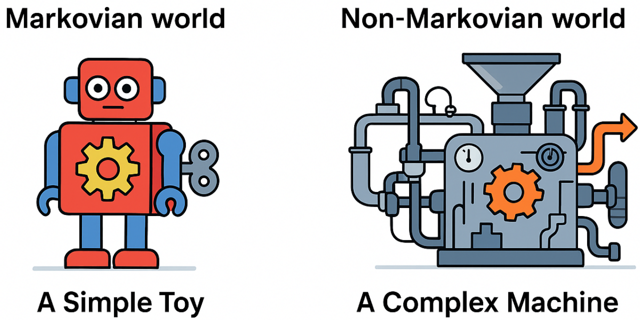
- Call durations often have high variability (lognormal)
- Customer abandonment times show complex patterns
- Distribution shape significantly impacts performance

Markovian vs. Non-Markovian: Performance Divergence



Markovian Model vs. Non-Markovian Model

Methodological complexity:



Markovian Model vs. Non-Markovian Model

Markovian queueing systems:

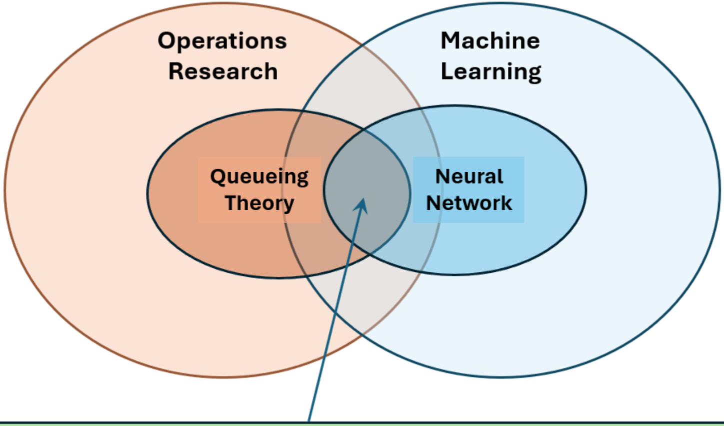
- Memoryless property: Future depends only on current state
- State = simple queue length
- Exact solutions via uniformization and matrix methods
- Computationally tractable

Non-Markovian queueing systems:

- Must track customer ages, residual service times
- State space becomes infinite-dimensional
- No closed-form solutions
- Heavy-traffic approximations are complex and often inaccurate
- Simulation is slow and computationally expensive

QUANT:
QUeueing Analysis via Neural-Network Transfer Learning

Where Are We?



QUANT – **QU**eueing **A**nalysis via **N**eural-network **T**ransfer learning

The $G_t/GI/n_t + GI$ Queueing Model

System Components:

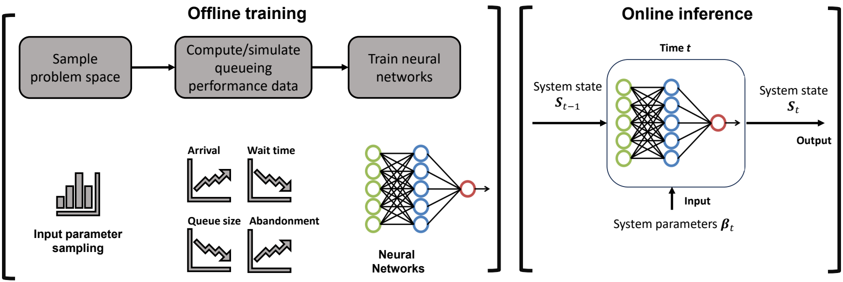
- G_t: General arrival process with nonstationary rate $\lambda(t)$
- GI: General independent service times (distribution G, mean $1/\mu$)
- n_t: Time-varying number of servers $n(t)$
- $+GI$: Customer abandonment (distribution F, mean $1/\theta$)

Performance Metrics to Predict:

- Mean waiting time: $\mathbb{E}[W(t)]$
- Mean queue length: $\mathbb{E}[Q(t)]$
- Mean busy servers: $\mathbb{E}[B(t)]$
- Tail probability of delay: $\mathbb{P}(W(t) \leq \tau)$ (e.g., $\tau = 60$ sec)

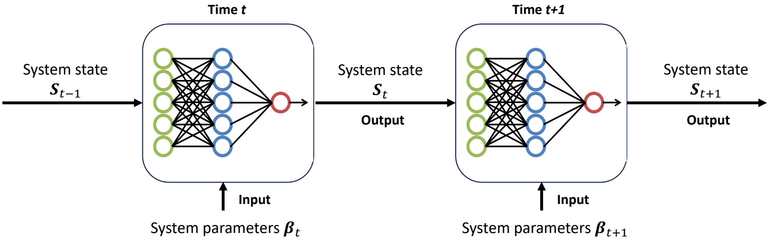
Goal: Predict these metrics over entire horizon $t \in [0, T]$ for any $\lambda(t)$ and $n(t)$

The QUANT Framework



- Goal: Predicts queue performance under time-varying demand and capacity
- Method: Neural network models trained on exact and simulated data
  - Offline phase: generate data and train networks
  - Online phase: fast prediction in real time
  - Inputs: demand rate, staffing level, current system state
  - Outputs: waiting time, queue length, busy servers, service level.

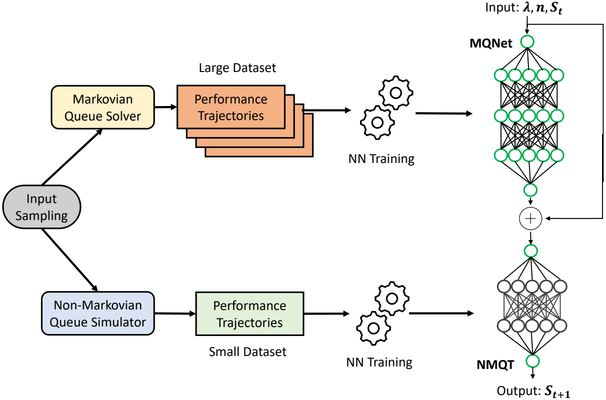
QUANT Architecture: Macroscopic Level



Recurrent Structure:

- One-step prediction: $\mathbf{S}_{t+1} = \mathcal{F}(\mathbf{S}_t, \boldsymbol{\beta}_{t+1})$
- State: $\mathbf{S}_t = (\mathbb{E}[W(t)], \mathbb{E}[Q(t)], \mathbb{E}[B(t)])$
- Input: $\boldsymbol{\beta}_{t+1} = (\lambda(t+1), n(t+1))$
- Roll out predictions recursively over time horizon
- Each step takes $\sim$1 millisecond on GPU

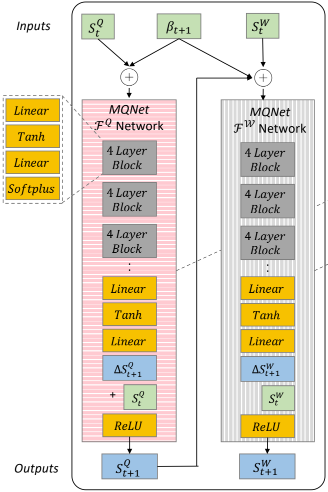
QUANT Architecture: Mesoscopic Level

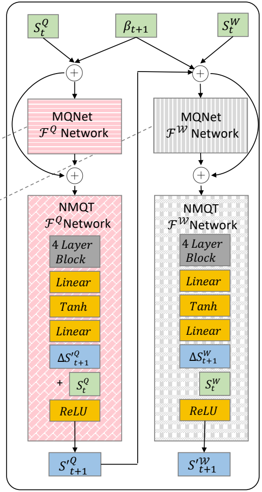


Two-Module Design:

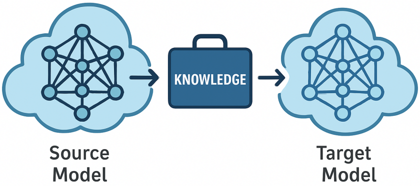
- Markovian Queue Network (MQNet): Pre-trained on $M_t/M/n_t + M$ systems (exact results)
- Non-Markovian Queue Transformer (NMQT): Learns corrections for non-Markovian $G_t/G/n_t + GI$ (simulated results)

QUANT Architecture: Microscopic Level





Why Transfer Learning?



Features Characterizing A Queueing System

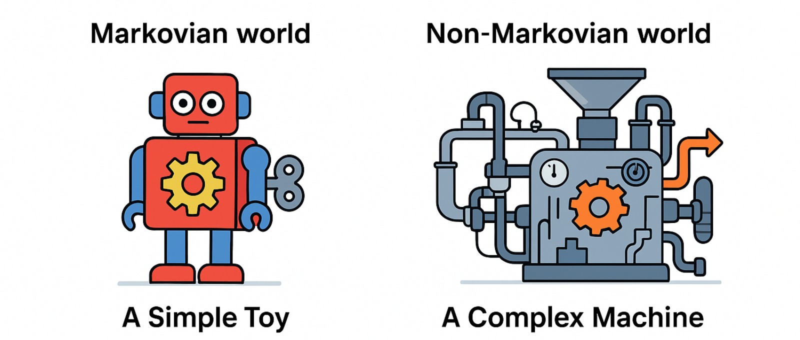
1. **System Structure**
   - Topology: single, pooled, parallel, tandem, or networked queues
   - Capacity: number of servers; time-varying or fixed
2. **Operational Rules**
   - Service discipline: FCFS, LCFS, Priority, Processor Sharing
   - Routing: fastest-server-first, join-shortest-queue
   - Customer behavior: balking, abandonment, retrial
3. **Random Elements: Averages**
   - Arrival rate: constant or time-varying
   - Average service time, average abandonment time
4. **Random Elements: Distribution beyond Means**
   - Arrival process: Poisson vs. non-Poisson
   - Service- and abandonment-time distribution: exponential vs. general

Features Characterizing A Queueing System

1. **System Structure**
   - Topology: single, pooled, parallel, tandem, or networked queues
   - Capacity: number of servers; time-varying or fixed
2. **Operational Rules**
   - Service discipline: FCFS, LCFS, Priority, Processor Sharing
   - Routing: fastest-server-first, join-shortest-queue
   - Customer behavior: balking, abandonment, retrial
3. **Random Elements: Averages**
   - Arrival rate: constant or time-varying
   - Average service time, average abandonment time
4. **Random Elements: Distribution beyond Means**
   - Arrival process: Poisson vs. non-Poisson
   - Service- and abandonment-time distribution: exponential vs. general

Markovian Model vs. Non-Markovian Model

Methodological complexity:



Transfer Learning: The Key Innovation

Why Transfer Learning?

- Markovian systems: Exact solutions available via MQSolver (cheap!)
- Non-Markovian systems: Only simulation available (expensive!)
- Generate 10,000 trajectories for Markovian training: $\sim$7 CPU-days
- Generate 10,000 trajectories for non-Markovian: $\sim$100 CPU-days

Our Solution:

1. Pre-train MQNet on 10,000 Markovian trajectories
2. Fine-tune with only 100 non-Markovian trajectories (1% of data!)
3. Achieves near-optimal accuracy with 99% less simulation
4. Total training: $\sim$8 CPU-days vs. $\sim$100 CPU-days

$\Rightarrow$ **Orders of magnitude reduction in training cost**

Data Generation Strategy

Synthetic Trajectory Generation:

- Time-series composition of Gaussian processes
- Creates realistic nonstationary patterns: single-peak, multi-peak, high-volatility
- Covers operational bounds: $\lambda \in [0, 100]$, $n \in [1, 100]$

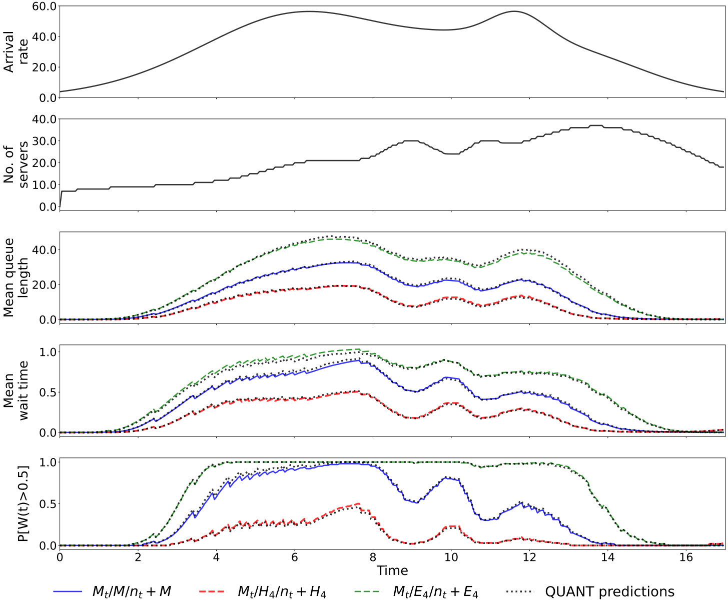
MQSolver: Exact Markovian Solution

- Exploits birth-death process structure
- Uniformization for time-inhomogeneous systems
- Computes exact $\mathbb{P}(W(t) \leq \tau)$ for all t, τ
- Runtime: 60 seconds per trajectory (vs. 900s for simulation)

Non-Markovian Simulation:

- 10,000 replications for smooth estimates
- Only 100 trajectories needed for transfer learning

QUANT Prediction: An Example



Runtime Performance

Method	Runtime (seconds)
QUANT (GPU)	**0.200**
QUANT (CPU)	0.350
MQSolver	60
Simulation	900

Speedup Analysis:

- **300× faster** than exact Markovian solver
- **4,500× faster** than simulation
- Enables real-time optimization with thousands of evaluations
- Critical for production capacity planning systems

Prediction Accuracy and Robustness

Configuration	TAMSE[1]
Base Markovian System	
Queue/Busy metrics	0.09999
Waiting time	0.00131
End-to-end	**0.07032**
By System Scale	
Small scale (0-41 servers)	0.08621
Medium scale (41-72 servers)	0.07559
Large scale (72-100 servers)	0.06377
By Distribution (Non-Markovian)	
$M_t/H_2/n_t + H_2$ (SCV=4)	0.07144
$M_t/E_4/n_t + E_4$ (SCV=1/4)	0.07092

Key Findings:

- Consistent accuracy (error < 0.1) across all configurations
- Minimal degradation from Markovian to non-Markovian
- Scale-invariant performance

[1] TAMSE: time-averaged mean square error

Transfer Learning Efficiency

How much non-Markovian data is actually needed?

Fine-tuning Data	TAMSE[2]	Relative Error
Baseline (10,000 trajectories, no transfer)	0.0703	–
50% (5,000)	0.0709	0.8%
10% (1,000)	0.0714	1.6%
1% (100)	**0.0734**	**4.4%**
0.5% (50)	0.0872	24.0%

Key Insight:

- Robust performance down to **1% of data (100 trajectories)**
- Accuracy within 5% of full dataset
- Sharp degradation below this threshold
- **Practical guideline: Use 100-500 trajectories for new systems**

[2] TAMSE: time-averaged mean square error

QUANT-based Optimal Staffing

Optimal Staffing under Service-Level Constraints

Problem: Given $\lambda(t)$ and constraints, find optimal $n^*(t)$

Example Constraints:

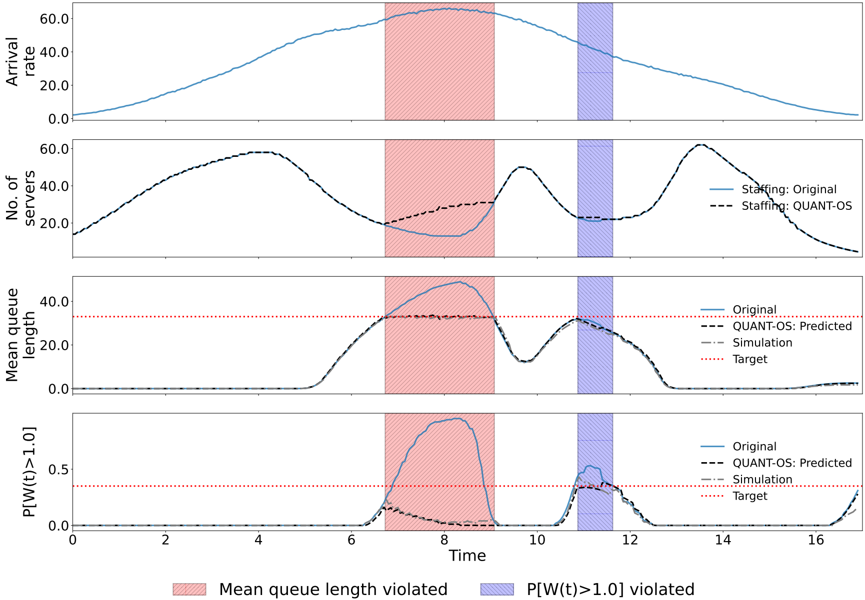
- Mean waiting time: $\mathbb{E}[W(t)] \leq w$ (e.g., $w = 30$ seconds)
- Mean queue length: $\mathbb{E}[Q(t)] \leq q$ (e.g., $q = 10$)
- Tail probability: $\mathbb{P}(W(t) > \tau) \leq \alpha$ (e.g., $\tau = 60$ seconds, $\alpha = 20\%$)
- Meeting multiple constraints simultaneously!

Three Optimization Methods:

1. **Binary Search:** Exploits monotonicity, ∼8 iterations
2. **Batch Evaluation:** Parallel GPU evaluation, 1 pass
3. **Gradient-Based:** Auto-differentiation through QUANT, ∼35 iterations

All methods complete in seconds (vs. hours for simulation-based optimization)

QUANT-OS: A Multi-Constraint Example



Optimization Performance

Method	Iterations	Time (s)	Success
Binary Search	8.2	1.2	100%
Batch Evaluation	1.0	0.8	100%
Gradient-Based	34.5	10.1	97.2%

Validation Against Simulation:

Constraint Type	Pred. Error	Violation Rate
$\mathbb{E}[W] \leq w$	0.082-0.107	2.5-3.5%
$\mathbb{E}[Q] \leq q$	0.080-0.096	2.0-3.0%
$\mathbb{P}(W > \tau) \leq \alpha$	0.131	5.0%
Multi-constraint	0.140	6.0%

$\Rightarrow$ **Over 94% constraint satisfaction in all cases**

Robustness Across Configurations

Configuration	Convergence	Satisfaction	Pred. Error
$M_t/M/n_t + M$	99.2%	98.5%	0.121
$M_t/E_4/n_t + E_4$	98.5%	97.0%	0.224
$M_t/H_2/n_t + H_2$	97.8%	95.5%	0.360
Small scale	96.7%	96.5%	0.156
Medium scale	97.9%	97.0%	0.140
Large scale	98.4%	98.5%	0.132

Key Findings:

- Consistent performance across all distributions tested
- Works for both high and low variability systems
- Scale-independent (works for small and large systems)
- Maintains $> 95\%$ satisfaction even for high-variability systems

Conclusions

Summary:

- Recurrent neural network for nonstationary non-Markovian queues with time-varying capacity
- Transfer learning achieves 99% data reduction
- Real-time multi-constraint optimization in seconds

Future directions:

- Multiple customer classes with priority scheduling
- Network topologies using Graph Neural Networks
- Reinforcement learning with QUANT as environment model
- Multi-objective optimization beyond constraint satisfaction

Thank You!

Spyros Garyfallos: spyrosg@amazon.com
Yunan Liu: yunanliu@amazon.com

Paper: Transfer Learning Neural Networks for Nonstationary Non-Markovian Queueing Systems. *Submitted to INFORMS Journal on Computing.*

Website: `yliu48.github.io`