TRAINING FLUID QUEUES WITH DIRECT BACKPROPAGATION

Leon Li
Columbia University
New York
{al4263}@columbia.edu

ABSTRACT

In this paper, we propose a novel method for training fluid queues using direct backpropagation, capitalizing on their differentiable dynamics compared to traditional stochastic queueing networks. Fluid queues, which disregard the inherent stochasticity in arrival and service processes, offer a fully deterministic framework for modeling and optimization. By employing modern deep neural network architectures and leveraging the continuous nature of fluid queues, we can efficiently train a parametrized policy to minimize the cost of the system.

1 Introduction

Queueing systems are ubiquitous in various domains, ranging from traditional industries such as manufacturing and logistics, to modern tech industries such as cloud computing and data processing. They play a critical role in managing resources and ensuring efficient operations. The conventional approach to modeling and analyzing queueing systems is based on stochastic queueing networks, which capture the random nature of arrival and service processes. However, these stochastic models can be challenging to optimize due to their complex dynamics and computational requirements. Even under the context of modern reinforcement learning, these problems are notoriously hard due to infinite state space, unbounded costs, and long-run average cost objective when formulating an MDP[1].

On the other hand, fluid queues offer a promising alternative by representing the dynamics of the system in a fully deterministic manner, ignoring the inherent stochasticity in arrival and service processes. This differentiable framework enables the application of powerful optimization techniques, such as gradient-based methods, which are not readily applicable to stochastic models. The recent advancements in deep neural network architectures present a unique opportunity to leverage the differentiable properties of fluid queues for efficient policy optimization.

In the first part of the paper, we introduce a novel method for training fluid queues using direct backpropagation, taking advantage of the continuous nature of the fluid queue model. Our proposed approach employs a deep neural network architecture specifically designed for fluid problems, which directly optimizes the cost function of the system using backpropagation.

In the second part of the paper, we focus on the stability analysis of the direct backpropagation method. Specifically, we investigate the impact of varying arrival rates on the performance of the policy network. To address this challenge, we propose a Bayesian training method, which considers a distribution of arrival rates and trains the network by sampling from this distribution. We subsequently compare the performance outcomes between the direct backpropagation method and the Bayesian training method to highlight their respective strengths and limitations.

The main contributions of our work are as follows:

- We provide a detailed examination of the fluid queue model and its differentiable properties, establishing a solid foundation for our proposed method.
- We develop a deep neural network architecture for fluid control problems, capitalizing on the deterministic nature of fluid queues. We propose a direct backpropagation-based optimization technique that minimizes the cost function of the system.

- We propose a Bayesian training procedure that incorporates the uncertainty of varying arrival rates into the training process, enhancing the robustness and adaptability of the optimized policy network.
- We conduct extensive experiments to validate the effectiveness of our approach, demonstrating its potential in solving complex queue management problems in various applications.

The remainder of this paper is organized as follows: Section 2 presents an overview of fluid queues and their differentiable properties. Section 3 describes our proposed method of direct backpropagation and bayesian trianing method. Section 4 details the experimental setup and results. Finally, Section 5 concludes the paper and outlines future research directions.

2 Fluid Queues and Differentiable Properties

Without the loss of generality, we consider the case of single server multi queues(N) problem. For each of the queues, Class i job J_i arrive at server i with a Poisson arrival rate of λ_i. For each J_i, we let μ_i be the the exponential service rate of the job, which is the full capacity service rate. For our problem setup. We let $x_i(t)$ denote the number of class i jobs J_i at time t and let $u_i(t)$ denote the effort that the server spends on processing J_i at time t. The first constraint is that the total effort cannot exceed the capacity of the server.

$$\sum_{i=1}^{N} \frac{u_i(t)}{\mu_i} \leq 1$$

To maintain stability and avoid overflows in the system, we choose a T large enough time so that all the queues will be drained. To avoid overflow, we have to ensure that the traffic intensity is strictly less than 1, which is:

$$\sum_{i=1}^{N} \frac{\lambda_i(t)}{\mu_i} < 1$$

Let c_i be the holding cost per unit time for holding a class i job J_i. Then we can formulate our fluid control problem as to find $\mathbf{u}$ to minimize the total holding cost $\mathcal{C}$ over time $[0, T]$[2]:

$$\begin{aligned}
\min \quad & \mathcal{C} = \int_0^T \mathbf{c}^\top \mathbf{x}(t)\, dt \\
\text{s.t.} \quad & \dot{x}_i(t) = \lambda_i - u_i(t), \quad \forall i \in N, \forall t, \\
& \sum_{i=1}^{N} \frac{u_i(t)}{\mu_i} \leq 1, \quad \forall t, \\
& \mathbf{u}(t), \mathbf{x}(t) \geq 0, \quad \forall t.
\end{aligned}$$

We opt for a neural network-based policy that generates a random vector $\mathbf{A}$ with elements summing to 1. The relationship $a_i \mu_i = u_i$ represents the effort dedicated to serving queue i. Without loss of generality, we set $\mu_i = 1$ uniformly, resulting in the policy output $\mathbf{A} = \mathbf{u}$. Note that this is a deterministic problem with fixed arrival and service rates.

The system's states, expressed as $\dot{x}_i(t) = \lambda_i - u_i(t), , \forall i \in N, \forall t$, and the actions, defined as $\mathbf{u} = \mu \mathbf{A}$, are fully differentiable. Consequently, the total cost $\mathcal{C}$, as a function of the policy network's parameters θ, is also fully differentiable. We can therefore optimize the policy network by directly backpropagating the gradient of $\mathcal{C}$ with respect to θ.

3 Training Methods

We introduce two distinct training methods, both of which rely on direct backpropagation. The first method involves training on constant arrival rates, while the second one focuses on training with a fixed distribution of arrival rates.

Let's define the total arrival rates as $tot = \sum \lambda_i$. With a given tot, we formulate a training distribution by scaling a uniform distribution $\mathcal{U}(0, 1)$ by tot:

$$\mathcal{P}_{tot} = \mathcal{U}(0,1) * tot$$

For the first algorithm, we just sample one $\boldsymbol{\lambda} \sim \mathcal{P}_{tot}$ and train the network on this arrival rates for every episode.

Algorithm 1: Direct BackProp

Training algorithm input: ;
A policy network $\mathcal{P}$ parametrized by θ;
A fixed arrival rate $\boldsymbol{\lambda}$;
Number of training episodes n;
Simulation time T;
for $i = 1$ *to* n **do**
  for $t = 1$ *to* T **do**
    Simulate the policy $\mathcal{P}$ on $\boldsymbol{\lambda}$ for time T;
  end for
  Compute the total cost $\mathcal{C}$;
  Backpropagate the gradient cost $\mathcal{C}$ to compute ∇_θ;
  $\theta \leftarrow \theta - \eta \nabla_\theta$;
end for
return θ

In the context of the second algorithm, we propose a Bayesian training approach that accommodates distribution shifts. Specifically, for each episode during the training phase, we draw a sample from the distribution $\boldsymbol{\lambda} \sim \mathcal{P}_{tot}$, which represents the arrival rate of events in the system. This sampled arrival rate is then utilized to generate training instances for the learning algorithm. The underlying rationale for this method is that, as the network is exposed to an increasing number of training examples with varying arrival rates, it would gradually become more robust to potential distribution shifts, enhancing its generalization capabilities in diverse and dynamic environments.

Algorithm 2: Bayesian Training

Training algorithm input: ;
A policy network $\mathcal{P}$ parametrized by θ;
A distribution of arrival rates $\mathcal{P}_{tot}$;
Number of training episodes n;
Simulation time T;
for $i = 1$ *to* n **do**
  draw $\boldsymbol{\lambda_i} \sim \mathcal{P}_{tot}$;
  for $t = 1$ *to* T **do**
    Simulate the policy $\mathcal{P}$ on $\boldsymbol{\lambda_i}$ for time T;
  end for
  Compute the total cost $\mathcal{C}$;
  Backpropagate the gradient cost $\mathcal{C}$ to compute ∇_θ;
  $\theta \leftarrow \theta - \eta \nabla_\theta$;
end for
return θ

4 Result and Analysis

In our experiment setting, we choose three different total arrival rates: $[0.3, 0.6, 0.9]$. For a certain tot, we let $\boldsymbol{\lambda_{tot}}$ denote the fixed training arrival rates for the fixed policy FixPolicy_$\{tot\}$, and let $\boldsymbol{\lambda'_{tot}}$ denote a random arrival rates sampled from $\mathcal{P}_{tot}$. We let BayesianPolicy_$\{tot\}$ denote the policy trained on the bayesian method we described above.

We focused on the single server 5 queue problems with each queues initialized with 10 jobs. We choose an ascending holding cost c of 5 queues: $[1.1, 1.5, 1.7, 2.1, 2.5]$. The non uniform holding cost helps us to study the behavior of the policy. We use a three layer neural network with 64 We train all networks for 1000 episodes and tests them on all 6 arrival rates.

Policy	Arrival Rates					
	$\boldsymbol{\lambda_{0.3}}$	$\boldsymbol{\lambda'_{0.3}}$	$\boldsymbol{\lambda_{0.6}}$	$\boldsymbol{\lambda'_{0.6}}$	$\boldsymbol{\lambda_{0.9}}$	$\boldsymbol{\lambda'_{0.9}}$
FixPolicy_0.3	**2605.75**	2748.47	22054.92	5018.62	52502.58	45997.90
BayesianPolicy_0.3	2633.80	**2638.42**	5110.40	**4417.71**	13319.25	**14213.36**
FixPolicy_0.6	4262.89	4346.97	**4658.72**	16849.66	27717.10	26790.05
BayesianPolicy_0.6	2679.70	2686.58	5023.37	4530.68	13900.46	14745.38
FixPolicy_0.9	4221.82	4023.87	24646.26	9497.71	**12719.02**	19002.86
BayesianPolicy_0.9	2986.72	3053.41	5707.38	5011.98	13311.79	14276.96

Table 1: Costs for different policies and arrival rates (largest value in each column bolded)

This study investigates the cost implications of various policies under different arrival rates. We initially found that with a fixed training arrival rate, denoted as $\boldsymbol{\lambda_{tot}}$, the FixPolicy_$tot$, which was exclusively trained on these specific arrival rates for all episodes, exhibits superior performance. Nonetheless, it was observed that these Fixed Policies demonstrated proficiency only at the arrival rates they were trained on. Even when a different arrival rate was sampled from the same distribution, $\mathcal{P}_{tot}$, these policies failed to approach optimal performance.

In contrast, Bayesian Policies not only showcased comparable performance on the $\boldsymbol{\lambda_{tot}}$ drawn from their training distribution $\mathcal{P}_{tot}$, but also performed well on unseen distributions. The cost was found to be slightly higher than FixPolicy_tot when tested on $\boldsymbol{\lambda_{tot}}$ for all tot. This finding was unexpected particularly for BayesianPolicy_0.3, which had never been exposed to congested situations prior to the study. Despite this lack of exposure, the policy exhibited the lowest cost among all the tested policies. This outcome highlights the potentially superior adaptability of Bayesian Policies under varying conditions, suggesting a fruitful direction for further research.

In order to elucidate the behavior of the policies, we visually represent the actions within a single rollout. The graph reveals interesting dynamics for a Bayesian policy train on $\mathcal{P}_{0.3}$ and tested on $\boldsymbol{\lambda} \sim \mathcal{P}_{0.9}$. (Figure 1) The policy initially aims to deplete the queues with the highest cost, allocating all available resources to that specific queue. This approach is followed by a rapid convergence to the arrival rate of the highest cost queue, ensuring that the queue always remains at zero. Subsequently, the policy proceeds to target the queue with the second highest cost. All remaining effort is allocated to serve this queue, effectively draining it. A rapid convergence to the arrival rate is also observed following the depletion of the first queue. This process illustrates an efficient and adaptive strategy employed by the Bayesian policy, where it prioritizes queue management based on cost and adjusts resource allocation dynamically to maintain optimal performance. The effect can be further visualized by the cost and queues over time (Figure 2). And because the strategy the Bayesian policy learned is almost independent from $\boldsymbol{\lambda} \sim \mathcal{P}_{tot}$, the policy is robust to distribution shift in arrival rate. This finding further underscores the efficacy of Bayesian policies in managing complex and fluctuating conditions.

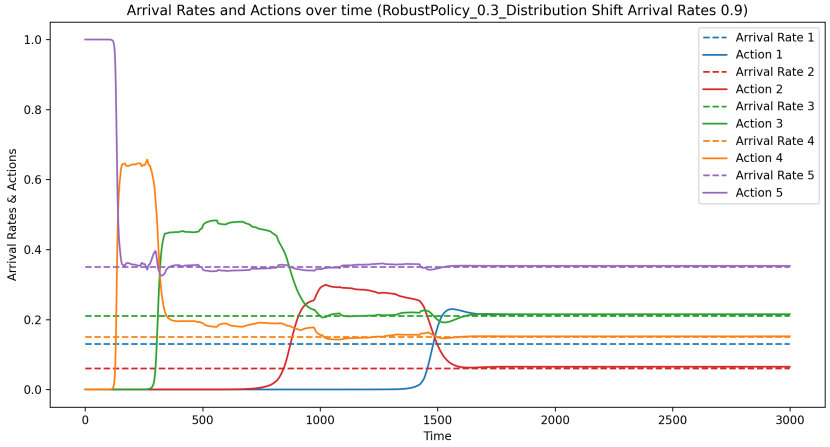

Figure 1: Actions over time

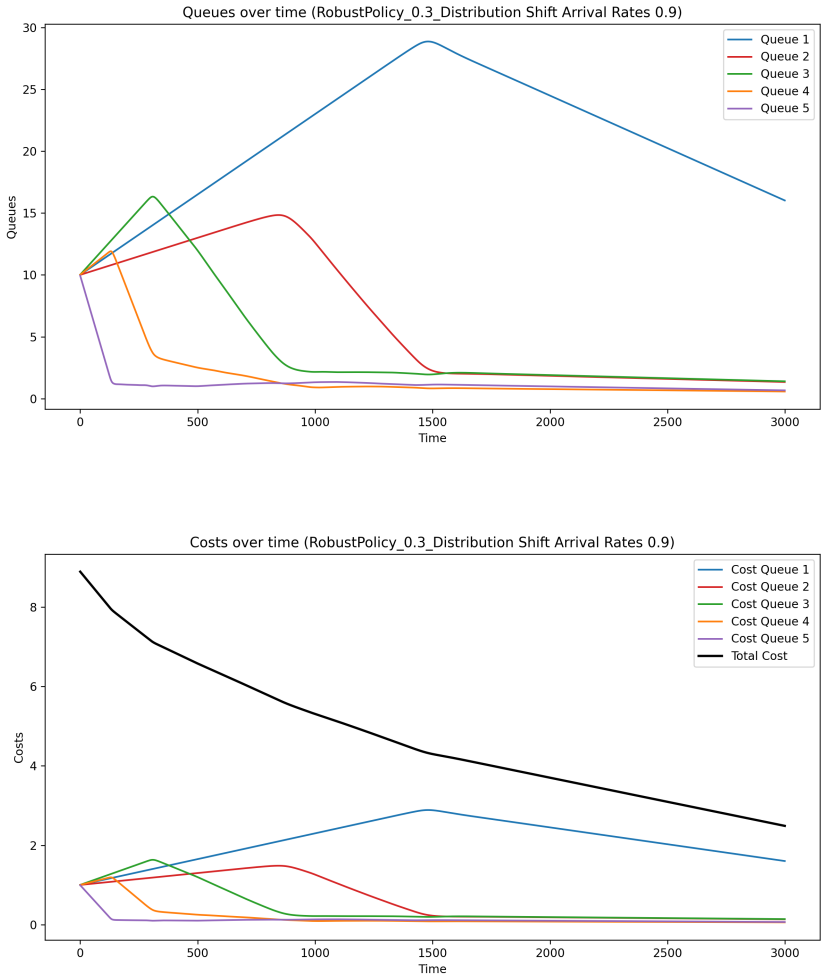

Figure 2: The queues and cost over time

5 Conclusion and Future directions

We proposed a novel approach for training fluid queues using direct backpropagation, leveraging the unique advantage of fluid queues' differentiable dynamics. By creating a deterministic framework, the fluid queue model enables the application of powerful optimization techniques such as gradient-based methods, which are often not applicable to stochastic models. Through the employment of deep neural network architectures, we demonstrated that our proposed method can efficiently optimize a parametrized policy to minimize the cost of the system.

Our main contributions include the detailed examination of the fluid queue model, the development of a deep neural network architecture for fluid control problems, and the proposal of a Bayesian training procedure. The Bayesian approach, in particular, has demonstrated robustness and adaptability in managing varying arrival rates during the training process. The experiments conducted validate the effectiveness of our approach, highlighting its potential in solving complex queue management problems in various applications.

Looking towards the future, our work raises several interesting questions and possible directions for further research. One critical aspect that warrants further exploration is the application of our policy in a stochastic queueing network. While we have demonstrated the effectiveness of our approach within deterministic fluid queues, it remains to be seen how the policy would perform when directly put into a stochastic setting. Future work should therefore explore the performance and potential adaptations required for the policy to operate effectively within stochastic queueing networks.

In our study, we assumed that the service distribution is stationary, an assumption that may not always hold in real-world systems. Therefore, another important area of future research is to find a systematic way to deal with the distribution shift in service rate. This could involve developing adaptive policies that can respond to dynamic changes in the service rate or incorporating anticipated changes in service rate into the training process.

In conclusion, this paper has presented a promising new approach for optimizing queueing systems using fluid queues and deep neural networks. The proposed method and its extensions could have far-reaching implications for a wide range of industries, including manufacturing, logistics, cloud computing, and data processing, where efficient queue management is crucial.

References

[1] J. G. Dai and Mark Gluzman. Queueing network controls via deep reinforcement learning.

[2] Dimitris Bertsimas and Ioannis Ch. Paschalidis. Robust fluid processing networks.