Causality-driven RL-based Scheduling Policies for Diverse Delay Constraints

Dibbendu Roy
Department of Electrical Engineering
Indian Institute of Technology
Indore, India
droy@iiti.ac.in

James Gross
ISE Division
KTH Royal Institute of Technology
Stockholm, Sweden
jamesgr@kth.se

***Abstract*—We investigate the role of causal models in the context of obtaining scheduling policies that minimize delay violations. We consider multi-user queuing systems with random delay constraints, packet sizes and arrivals in Gilbert-Elliot wireless channels. Owing to Judea Pearl's landmark work on causality to achieve a higher level of cognitive ability, we demonstrate the role of counterfactual reasoning, leading to the well-investigated optimal EDF policy for wired channels. Due to the randomness associated with the wireless channels, finding an optimal policy is not straightforward, leading to RL-based approaches. We present CPPO (counterfactual-PPO) and CA2C (counterfactual-A2C) algorithms that use counterfactual examples generated using causal models during the training process. We argue how stochastic gradient based policy gradient RL algorithms benefit during training due to incorporation of counterfactuals. We show that these algorithms provably lead to lower variance indicating a robust learning performance. Our results demonstrate a $\sim 60\%$ increase in the number of cases where CA2C and CPPO outperform their non-counterfactual counterparts with reduced variance and negligible computation overhead.**

***Index Terms*—Counterfactuals, Causality, Reinforcement Learning, Diverse delay constraints, Scheduling policies**

I. Introduction

The rapid evolution of networks heralds a new era in telecommunications, marked by unprecedented diversity in application requirements and network conditions [1]. These networks are expected to support a wide array of services, from high-definition video streaming and augmented reality to critical applications such as autonomous driving and telemedicine. Consequently, network schedulers must manage packets with varying sizes and deadlines, ensuring efficient and timely data delivery in an automated fashion. This requires development of advanced scheduling policies capable of meeting stringent and diverse delay constraints that adapt to fluctuating network conditions [2].

Scheduling policies with delay deadlines have been extensively studied in the context of wired networks. Earliest Deadline First (EDF) is a well-established optimal scheduling policy for such systems under certain conditions, as it minimizes maximum lateness and is theoretically proven to be optimal for preemptive tasks with dynamic arrival times [3]. However, extending these concepts to general wireless settings introduces significant challenges due to the inherent randomness in arrivals and variability of wireless channels [2], [4]. Evidently, finding an optimal scheduling policy with delay deadlines involves solving a Markov Decision Process (MDP) or employing Reinforcement Learning (RL) techniques, both of which are computationally complex [4], [5].

Recent advances in artificial intelligence (AI), particularly in the realm of causal reasoning, offer promising avenues for addressing these challenges. Pearl's causal hierarchy [6] formalizes reasoning about causality into three levels—association (observing correlations as performed by current ML/DL models), intervention (understanding effects of actions), and counterfactuals (imagining alternative realities)—providing a structured framework to infer causation from data and guide decision-making. Leveraging these insights, we explore the potential of causal models in developing effective scheduling policies for wireless queueing systems.

Our investigation focuses on the application of causal inference to obtain scheduling policies to minimize expected delay violations in wireless channels. We show that arguments from causal inference that rely on counterfactuals lead us to the optimal EDF policy in wired networks, providing a solid foundation for extending this approach to wireless environments. However, the stochastic nature of wireless channels necessitates the adoption of reinforcement learning (RL) techniques to discover optimal policies. To use causal settings, we resort to the widely popular Gilbert-Elliot [7] channel modeling for wireless channels. We briefly present the relevant works in the areas of delay deadline-based scheduling and use of the causal techniques in reinforcement learning.

A. Related Works

Although the problem of scheduling with delay constraints is a widely investigated topic, recent approaches resort to formulating MDPs that suits the RL framework for obtaining automated policies [2], [4], [8], [9]. However, these works are not devoid of assumptions which defeats the goal of agnostic automated decision making. Ref. [8] assumes the cumulative distribution function (CDF) of channels for users are known constructs a CDF based scheduling policy for hard-deadline and soft-deadline cases. [4] considers a multi-hop network and formulates an MDP with the objective of maximizing the throughput of packets that do not violate their deadlines (termed timely throughput) to find routing, scheduling and power allocation policy. They show that under peak average power constraints, several dependencies in the model are relaxed leading to an LP based optimal policy while such guarantees are not available for other cases. [9] considers a WirelessHart based network control system and formulates an MDP to optimize delay violation probability using throughput as an upper bound. The authors in [2] propose a new Deep RL architecture to optimize the throughput subject to average resource constraints. The model considers each user to have homogeneous delay.

Incorporating causality in RL paradigms has gained recent popularity, specifically in terms of theoretical possibilities. Recent approaches [10]–[12] extend multi-armed bandit algorithms to their causal counterparts. However, these works are limited to interventions (level 2 in causal hierarchy). [13] developed the concept of counterfactual (level 3 in causal hierarchy) reasoning in RL and proposes offline counterfactual policy evaluation strategies for improved performance. Although these works have promising theoretical underpinnings, they fall short on extending the causal framework to the online RL settings such as actor-critic (A2C) and proximal-policy optimization (PPO) [14] algorithms.

B. Contributions

In this paper, we present how a queuing system can be modeled as a causal graph without presumptions on arrival and service processes, with random delay deadlines. We formulate our scheduling problem with the objective of minimizing delay violations and construct a corresponding MDP for the same. Further, we show how counterfactual arguments are relevant in justifying the optimality of EDF policies in wired systems. We then extend our model to wireless settings with help of Gilbert-Elliot model and indicate why a simple policy such as EDF is not optimal. Motivated by this, we introduce novel causality-driven RL algorithms based on policy gradient techniques: Counterfactual Proximal Policy Optimization (CPPO) and (CA2C). We introduce an advantage term in the optimization update, computed using counterfactuals from the causal model. We show that the advantage term does not introduce bias in the gradient estimates. At the same time, under certain assumptions, it should lead to reduction in variance, thereby exploiting the utility of the causal model. The results demonstrate a significant gain in the percentage of successful packet transmissions without violating their delay constraints compared to the existing state of the art techniques, at the cost of minor computation overhead.

II. Model Description

We consider a simple single-server queuing system where packets arrive following a random arrival process. Each packet has a random delay deadline and the server must try to serve as many packets without violating the deadlines. Whenever the server is empty and the queue is non-empty, it faces a decision regarding which packet to serve from the queue at that moment in time. Such sequential decision making problems are typically solved in an RL framework.

A. System Model and MDP formulation

1) States: Let us denote the state of the queuing system at t_j by Q_{t_j}, that contains packets P_i. The number of packets in the queue at t_j is given by the cardinality $|Q_{t_j}|$. Each packet is characterized by its arrival time T_i^a, its size S_i, and its delay constraint D_i^c. These can be treated as random variables generated from random processes $A(t), S(t)$ and $D^c(t)$ denoting the arrival process, service process and a deadline generating process respectively. These processes are assumed to be independent and stationary. Q_{t_j} can then be described as a collection of tuples $Q_{t_j} = \{P_i = (T_i^a, S_i, D_i^c) \mid i = 1, 2, \ldots, |Q_{t_j}|\}$.

2) Decisions/Actions: Given a state, the server decides which packet to serve, denoted by the action variable $a_{t_j} = i$ if the packet $P_i \in Q_{t_j}$ is chosen to be served. We consider a non-preemptive server here i.e. once an action is taken, the next action is not taken until the completion of service of the chosen packet.

3) State Transition: Given a current state and action, the next state will not contain the packet chosen by the current action while new packets will be included that arrived by following $A(t)$ during the service time of the chosen action. Thus, we can compute the state $Q_{t_{j+1}}$ from Q_{t_j}, a_{t_j} with stochasticities due to $A(t), S(t)$ and $D^c(t)$. As computing $Q_{t_{j+1}}$ does not require the information of states or actions previous to Q_{t_j}, the environment is therefore completely described just by the single step transition probabilities $P(Q_{t_{j+1}}|Q_{t_j}, a_{t_j})$. Hence, the system follows the Markov property, justifying an MDP formulation.

4) Reward: After every action, the system evolves to a new state and a reward is computed for reaching this state. Since we want to maximize the number of packets that do not violate their delay bounds, we define the reward as 1 if delay of the leaving packet is within its deadline, else zero. To realize this mathematically we compute the delay for a packet $P_i \in Q_{t_{j+1}} \cap Q_{t_j}$, $D_i = t_j + S_i/C - T_i^a$, where C is the serving speed (fixed for wired scenarios). If $\mathbb{1}$ denotes an indicator function, the reward is:

$$R_{t_j}(a_{t_j}) = \mathbb{1}\Big(D^c_{a_{t_j}} \geq D_{a_{t_j}}\Big) \tag{1}$$

5) Policy: A policy $\pi = \{a_{t_j},\ j = 1, 2, \ldots K\}$ is a sequence of actions corresponding to its states where K denotes the horizon of an episode. Thus, it can be defined as a function of state that generates an action i.e. $\pi(Q_{t_j}) = a_{t_j}$. This is the case if the policy is deterministic. For a more general definition, it is defined as a probability distribution of possible actions given a state i.e. $\pi(\cdot \mid Q_{t_j})$.

6) Value Function and Optimal Policy: The value function of a state in a MDP is defined as the cumulative reward that may be achieved in an episode by starting from the concerned state and following a policy π. Thus, the value function may be defined as:

$$V_\pi(Q_{t_j}) = \mathbb{E}_\pi \left[\sum_{t_j|j\in\mathbb{N}}^{t_{j+K}} R_{t_j} \middle| Q_{t_j} \right] = R_{t_j} + P^T V_\pi(.) \qquad (2)$$

where the term $P^T V_\pi(.)$ represents the sum $\sum_{j'} P\left(Q_{t_{j'}} \middle| Q_{t_j}, \pi(Q_{t_j})\right) V_\pi(Q_{t_{j'}})$. An optimal policy π^* maps every state to an optimal action $a^*_{t_j}$ that maximizes the value function for every state.

B. Causal Model for Queuing Systems

As described in [6], a structural causal model (SCM) is represented with help of a graph whose vertices depict random variables and edges represent the dependencies among them. If $X \rightarrow Y$, it implies that Y is a direct cause of X and hence is a parent of Y. The dependency between them can be modeled as a function $Y = f(X) + U_Y$. The variable U_Y can be conceived as source noise, also termed as *exogenous* (out of the system) variables while the observed variables (X, Y in this context) are *endogenous*. When SCM for a system is known, [6] shows that one can compute interventions and counterfactuals (alternate realities) by following abduction, action and prediction steps on the SCM. The SCM for a queuing system can be construed as follows:

1) Wired Systems: Fig.1 shows the causal model of a queuing system that works at constant rate (representing wired link). Since packets are generated as discussed in Section II-A1, the queue is always governed by the random processes $A(t), S(t)$ and $D^c(t)$ as shown in Fig.1(a). The dotted lines indicate that they are exogenous variables since the system is oblivious of the generating processes while the queues, actions and rewards are endogenous as they are observable. The next queue states $Q_{t_{j+1}}$ directly depends on the current action a_{t_j} and state Q_{t_j}, and the new arrivals which are generated by the random processes. Each action leads to the departure of a packet which generates a delay based on its size and serving rate, say C bits/sec. Once an action is decided, the reward can be computed using (1).

2) Wireless Systems: We extend the model for a wireless system with multiple users. The controller maintains separate buffers for each user having different channel conditions and can decide on which packet to serve given the channel condition and other QoS requirements. Fig.1(b) shows the resulting causal model with aggregated queue and action states for all users. Time is assumed to be slotted (t_j is replaced by slot j). We employ the classical 2-state Gilbert-Elliot Model, [7] where the channel is either in a good state or a bad one, based on which the service rates change for each user.

III. Counterfactual Policy Design

A. Policy design for Wired Systems

Given the system model, we present the following theorem using counterfactual arguments.

Theorem 1. The optimal policy for the formulated MDP is given by:
If $D^c_i - D_i \geq 0$ for some $P_i \in Q_{t_j}$, then

$$\pi^*(Q_{t_j}) = \underset{i|D^c_i - D_i \geq 0}{\arg\min}\ (D^c_i + T^a_i) \text{ (Earliest Deadline First)}$$

Otherwise,

$$\pi^*(Q_{t_j}) = \emptyset$$

Proof. Given an action a_{t_j}, for policy π, the first term of the value function is deterministic (since Q_{t_j} is also given). The rest of the terms depend on the random nature of new arrivals, their sizes/service times, and their delay constraints.

$$\begin{aligned} R_{t_{j+1}} &= \mathbb{1}\left(D^c_{a_{t_{j+1}}} \geq D_{a_{t_{j+1}}}\right) \\ \mathbb{E}[R_{t_{j+1}}] &= \mathbb{E}\left[\mathbb{1}\left(D^c_{a_{t_{j+1}}} \geq D_{a_{t_{j+1}}}\right)\right] \\ &= P(D^c_{a_{t_{j+1}}} \geq D_{a_{t_{j+1}}}) \end{aligned}$$

Thus, the terms of expected reward are probabilities that the packets at future queue states do not violate the delay bounds and it is our objective to maximize this. To prove that the mentioned policy is optimal, we consider the following cases.

Case 1: If $D^c_i - D_i \geq 0$ for some $P_i \in Q_{t_j}$. This implies there are some packets that can be served without violating the delay bound. Consider two policies, one of which follows EDF denoted by π^* and let $\pi^*(Q_{t_j}) = i$ (packet index). The other policy π also follows EDF except at t_j, where $\pi(Q_{t_j}) = k$. Since $R^{\pi^*}_{t_j} = 1$, $R^{\pi^*}_{t_j} \geq R^{\pi}_{t_j}$. If we followed π, then $P_i \in Q_{t_{j+1}}$. The intuition here is that *had we not scheduled P_i, it is always more likely that its delay will be violated at any later schedule* (**counterfactual argument**) and hence there is a higher probability of delay violation at $Q_{t_{j+l}}, l \in \mathbb{N}$ i.e., $P_\pi(D^c_i < D_i) \geq P_{\pi^*}(D^c_k < D_k) \Rightarrow \mathbb{E}_{\pi*}[R_{t_{j+l}}] \geq \mathbb{E}_\pi[R_{t_{j+l}}]$ where $j + l$ represents a future time index by which both i and k are served following either policy.

To show this, by our choice of π^*, we should have

$$\begin{aligned} & D^c_i + T^a_i \leq D^c_k + T^a_k \\ & \Rightarrow P_\pi\left(t_j + \frac{S_k + S_i}{C} > D^c_i + T^a_i\right) \\ & \quad \geq P_{\pi^*}\left(t_j + \frac{S_i + S_k}{C} > D^c_k + T^a_k\right) \\ & \Rightarrow P_\pi(D_i \geq D^c_i) \geq P_{\pi^*}(D_k \geq D^c_k) \quad \text{for } t > t_j \end{aligned} \qquad (3)$$

Since, for all other time indices, EDF is followed and the distributions $A(t), S(t), D^c(t)$ are identical, the expected rewards should not differ. As k was chosen arbitrarily, we have that $\mathbb{E}_{\pi^*}[R_{t_{j+l}}] \geq \mathbb{E}_\pi[R_{t_{j+l}}]$ and we can also run an induction on this for subsequent time steps whenever EDF is not followed. As this is true for any time step following t_j, adding the reward terms can only increase the value function and not reduce it. Thus, $V_{\pi^*} \leq V_\pi$.

Case 2: In this case, none of the packets can satisfy the delay bounds and the reward R_{t_j} cannot improve by selecting any packet. Therefore, serving any of the packets in the buffer

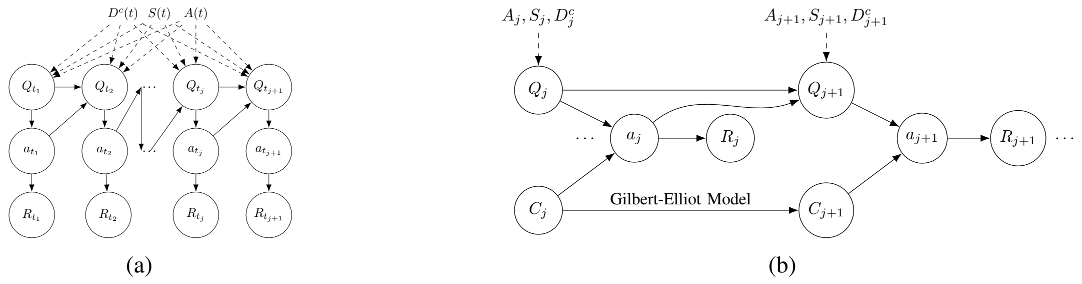

$D^c(t)$ $S(t)$ $A(t)$
Q_{t_1} Q_{t_2} Q_{t_j} $Q_{t_{j+1}}$
a_{t_1} a_{t_2} a_{t_j} $a_{t_{j+1}}$
R_{t_1} R_{t_2} R_{t_j} $R_{t_{j+1}}$
(a)
A_j, S_j, D^c_j
$A_{j+1}, S_{j+1}, D^c_{j+1}$
Q_j Q_{j+1} a_j R_j a_{j+1} R_{j+1} C_j C_{j+1}
Gilbert-Elliot Model
(b)

Fig. 1: (a) The figure on the left is the causal graph for a queuing system with fixed serving speed. (b) A possible causal graph for a wireless scenario. C denotes the channel state. The subscript j corresponds to a time slot. The scheduling decision a_j at time slot j depends on both the queue state and the channel state. The transition process of the channel between C_j and C_{j+1} is a 2-state Gilbert-Elliot Markov Model [7].

will increase the probability of violation of packets arriving in the future which leads to lower future rewards. ■

It is also important to note the importance of the constant service rate C. If the service rate C changes for the packets with time, a fixed term will not be added to t_j to account for the service times under the two policies and hence (3) cannot be guaranteed. Therefore, we try to employ RL techniques for scheduling in wireless channels.

B. RL Algorithms using Counterfactuals

Recent advancements in RL have demonstrated considerable performance improvements by employing policy gradient techniques [15]. The policy $\pi(a|s)$, is parameterized as $\pi(a|s,\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the parameter (maybe implemented as a neural network), a represents the action and s represents the state. If $\mathcal{Q}_{\pi_\theta}(s,a)$ represents the q-value and $V_{\pi_\theta}(s)$ represent the value of a state as defined in (2) (considering now that the policy is parameterized with the parameter $\boldsymbol{\theta}$, [16] (Chapter 13) presents the policy update equation as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \sum_t \hat{R}_t \left[\nabla \ln \pi(a|s_t,\boldsymbol{\theta}_t)\right] \tag{4}$$

where s_t is the state at time t and a_t is the corresponding action sampled at time t. The hat represents a learned approximation to the actual q-value. This method (known as REINFORCE) forms the basis of most modern RL algorithms such as proximal policy optimization (PPO) and advantage actor-critic (A2C) models [17].

Even though we have an understanding of the causal model as described in Section II-B2, and we may compute counterfactuals given a policy's outcomes, it is not evident how to use the same for training purposes. This is more so because the counterfactuals do not give entire trajectories as following a policy would. Each counterfactual is a possible deviation branch for the observed policy. Let $\mathcal{A}_t = a_t$ denote the action implemented due to policy π_θ while a^c_t denote an action which was not implemented. We compute the counterfactual reward $R^c_t = R(a^c_t)$ using steps in [6]: Given that $R(a_t)$ is known, we should go back to compute the exogenous variables that affect the parents of the variable we wish to change (*abduction*). Then using the values of the exogenous variables, we perform an intervention on action $\mathcal{A}_t = a^c_t$ by disconnecting it from its parents (Q and C) (*action*) which yields the desired value (*prediction*). We have the following lemma due to the causal graph Fig.1(b).

Lemma 1. If the random variables $S_{j,j\geq 0}$, $D^c_{j,j\geq 0}$ and $C_{j,j\geq 0}$ are independent and identically distributed, then so are R_j and R^c_j.

Proof. Due to the channel condition at C_j, the service rate can be represented as a deterministic function of $C_j = L(C_j)$. $R_j = \mathbb{1}(D^c_{a_j} \geq D_{a_j}) = \mathbb{1}(D^c_{a_j} \geq t_j + S_{a_j}/L(C_{a_j}))$. When R_j is given, it implies that t_j time has already been spent and is known. Hence, $R^c_j = \mathbb{1}(D^c_{a^c_j} \geq D_{a^c_j}) = \mathbb{1}(D^c_{a^c_j} \geq t_j + S_{a^c_j}/L(C_{a^c_j})|t_j)$. Since each term in R^c_j is independent to the corresponding term of R_j, we have the lemma. ■

While the independence assumption can be understood (each user is independently generating its packets), they might not be identical as they are being generated from different users. However, across the packets of the same user, the assumption should hold.

Let us consider a modified version of the algorithm

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta}_t + \frac{\alpha_t}{2} \sum_t [\hat{R}_t - \frac{1}{|\mathcal{A}|} \sum_{a^c} \hat{R}^c_t] \underbrace{[\nabla \ln \pi(a|s_t,\boldsymbol{\theta})]}_{g_t} \tag{5}$$

We show that due to the lemma, we may be able to reduce the variance of the gradient estimate in (5). Usually, this is termed as *baseline* $b(s_t)$ which is subtracted from R_t to reduce variance (chapter 13 [16]). However, we consider the counterfactual term along with the existing baseline used for policy gradients. In PPO and A2C, the baselines are the estimated values of a state obtained by using a value network called the critic $V(s)$ while the policy network ($\pi(a|s,\theta)$) is called the actor. In these methods both the actor and critic are learnt during the training process.

Proposition 1. Introducing the counterfactual rewards in the gradient yields an unbiased gradient estimate with reduced variance.

Proof. We have, $\mathbb{E}_{\pi_\theta}\left[\frac{1}{2}\sum_t \left[\hat{R}_t - \frac{1}{|\mathcal{A}|}\sum_{a^c} \hat{R}^c_t\right] g_t\right]$

$= \mathbb{E}_{\pi_\theta}\left[\frac{1}{2}\sum_t \left[2\hat{R}_t - \frac{1}{|\mathcal{A}|}\sum_a \hat{R}_t^a\right] g_t\right]$

$= \mathbb{E}_{\pi_\theta}\left[\sum_t \left[\hat{R}_t - \frac{1}{2|\mathcal{A}|}\sum_a \hat{R}_t^a\right] g_t\right]$, where $a \in \mathcal{A}$ represents any action from the set of all actions. Note that we can invariably set $\hat{R}_t = 0$ for actions which are invalid. Since the sum is over all possible actions, the term is independent of action a. We need bother only about $\mathbb{E}_{\pi_\theta} \sum_a \hat{R}_t^a g_t$ as the rest of the terms are same as the original gradient in (4). Thus, $\mathbb{E}_{\pi_\theta}\left[\sum_a \hat{R}_t^a g_t\right] = \sum_a \hat{R}_t^a \sum_{a'} \pi(a'|s_t, \boldsymbol{\theta}) g_t$. Since, $\sum_{a'} \pi(a'|s_t, \boldsymbol{\theta}) g_t = 0$ ($\because$ g_t is a score function [16], [18]), have that $\mathbb{E}_{\pi_\theta}\left[\frac{1}{2}\sum_t \left[\hat{R}_t - \frac{1}{|\mathcal{A}|}\sum_{a^c} \hat{R}_t^c\right] g_t\right] = \mathbb{E}_{\pi_\theta}\left[\sum_t \hat{R}_t g_t\right]$.

As shown in [18], computing the variance reduces to computing $\mathbb{E}\left[\left(\hat{R}_t - \frac{1}{|\mathcal{A}|}\sum_{a^c} \hat{R}_t^c\right)^2\right] = \mathbb{E}[\hat{R}_t^2] + \frac{1}{|\mathcal{A}|^2}\mathbb{E}[(\sum_{a^c} \hat{R}_t^c)^2] = \frac{1}{|\mathcal{A}|}\mathbb{E}\left[(\hat{R}_t)^2\right]$. The cross terms vanish due to independence from Lemma 1. The second equality is due to independence (cross terms of the sum for counterfactuals vanish) and imposing Lemma 1. ■

Algorithm 1: Collect Rollouts for CA2C and CPPO

```
Input: Environment $env$, Policy $\pi_\theta$
Output: Rollout buffer $\mathcal{R}$

Initialize rollout buffer $\mathcal{R}$;
Observe state $s_t$;
Sample action $a_t \sim \pi_\theta(a|s_t)$;
Execute action $a_t$ in the environment;
Observe reward $R_t$ and next state $s_{t+1}$;
Store $(s_t, a_t, R_t, s_{t+1})$ in $\mathcal{R}$;
for each counterfactual action $a_t^c \in \mathcal{A} \setminus \{a_t\}$ do
    Compute $a_t^c$ to obtain counterfactual reward $R_t^c$;
    Store $(s_t, a_t^c, R_t^c)$ in $\mathcal{R}$;
end
return $\mathcal{R}$;
```

Based on the developed gradient update, we present the algorithm for training in case of policy gradient based algorithms. Two popular algorithms are A2C and PPO. We state the changes to be performed on the existing implementation available using stable_baselines3 [14] to obtain their counterfactual versions (CA2C and CPPO). As shown in 1, we compute and collect the counterfactual states and rewards in the rollout buffer by the *abduction*, *action* and *predicton* steps as entailed in [6]. As shown in Algorithm 2, the adjustment in the temporal difference error is presented in the advantage computation step which is then used in the training process for the stochastic gradient update. The advantage has the intuitive interpretation that it computes the importance of the current action with respect to the average counterfactual return. If the average counterfactual reward is greater than the reward from the current action, the direction of the gradient switches from positive to negative. The following section presents our results on employing the developed algorithm.

Algorithm 2: Compute Advantages for CA2C and CPPO

```
Input: Rollout buffer $\mathcal{R}$, Value function $V_\phi$
Output: Updated rollout buffer with advantages

for each step $t$ in $\mathcal{R}$ do
    Compute TD error:
        $\delta_t = R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
    Compute averaged counterfactual reward:
        $\bar{R}_t^c = \frac{1}{|\mathcal{A}|} \sum_{a_t^c} R_t^c$
    Compute adjusted advantage:
        $A_t = \delta_t - \bar{R}_t^c$
    Store $A_t$ in $\mathcal{R}$;
end
```

IV. Results and Discussions

We illustrate the performance of the proposed algorithms CPPO and CA2C and compare them with their non-counterfactual versions.

A. Setup

We use simpy [19] which is a python-based discrete event simulator to simulate the queuing system. As discussed in Section III-B, we customize our environment to use stable_baselines3 implementations of PPO and A2C. The parameters for the setup are provided in Table I. The parameters ensure that the queue sizes do not blow up, thereby ensuring stability. Both the action and state spaces are bounded by the buffer sizes. For each packet in the buffer, we define the state as a 4-tuple containing 1) the remaining time to bound 2) number of slots required to serve the packet at current time 3) current time slot 4) channel condition. It is important to note that the channel condition for each user is different but the packets of the same user will have identical conditions.

Since the action space is discrete, only few of the stable baseline algorithms support out of the box namely DQN, A2C and PPO, wherein, DQN is an off-policy algorithm which learns the optimal policy (target) from a different policy (behavorial policy) used for collecting the experiences. On the contrary, A2C and PPO are on-policy algorithms that scouts the optimal policy using stochastic gradient ascent while collecting experiences.[1]

B. Learning Performance of CPPO and CA2C

Fig.2 shows the first 100 and 400 epochs or episodes of the different RL algorithms, where each episode consists of 1000 action steps (packet departures). We define a successful packet transmission, if it departs without violating its delay constraint. The mean % of successful packets is plotted in

[1]Implementation is available at https://github.com/dibbend8/CausalRL.git

TABLE I: Values of chosen parameters for the simulation setup

Parameter	**Value**	**Parameter**	**Value**
Queuing Parameters			
# Users	2^*	Buffer Size/User	10
Mean Packet Size μ	1500 bytes	Min Packet Size	0.25μ
Max packet size	2μ	Minimum Delay bound	1 ms
Maximum Delay Bound	5 ms	Slot duration	1 ms
Minimum arrival rate	30 packets/sec	Maximum arrival rate	120 packets/sec
Service rate (Good)	100 Mbps	Service rate (Bad)	50 Mbps
p_{01}	0.5^*	p_{10}	0.5^*
Learning Parameters			
Learning rate α	0.001	Discount factor γ	0.99
Number of epochs	10	Rollout buffer size	2048
Minibatch size	64	Policy network architecture	64×64
Value network architecture	64×64	Optimizer	Adam

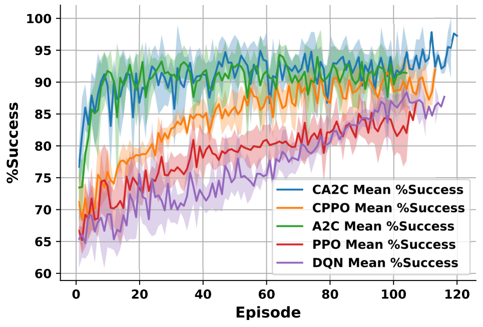
* Unless otherwise specified

bold while the shaded areas represent the deviations around the mean generated by repeating the experiments with five different seeds. Fig.2 (a) shows that CA2C and A2C have a similar convergence rate of almost 20 episodes while CPPO converges faster than PPO with higher % success. Both perform significantly better than an off-policy like DQN. Further, 2 (b) exhibits a long-run improvement of CA2C over A2C as the blue lines peak over the green ones.

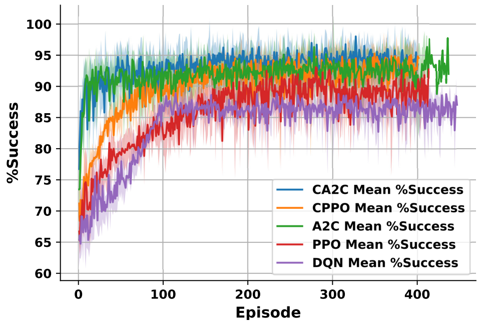
TABLE II: Variance of Rewards and Computation Times for Different RL Techniques

Method	**Variance of %Success**	**Time/Episode (s)**	**Success ($\%X > Y$)**
CA2C	**5.279**	**0.94**	**57.45%**
A2C	10.416	0.79	
CPPO	**1.174**	**1.61**	**62.95%**
PPO	6.299	0.65	
DQN	3.937	0.49	

To investigate this further, Table II shows the variance for the employed methods, the average training time per episode and the percentage of episodes where the success of an algorithm is higher than other. As expected, the variance reduces which shows the robustness of the developed counterfactual algorithms. The variance reduction is significantly higher in case of CPPO possibly due to the fact that PPO uses a minibatch training process compared to A2C and the same might be the reason for an increased training time per episode required for counterfactual computations in CPPO compared to CA2C. The last column shows that there is $\sim 60\%$ gain in terms of number of episodes where CA2C and CPPO outperforms A2C and PPO respectively. Although the 1s additional computation time for CPPO might appear as a significant overhead, considering the number of episodes for it to converge (~ 80), the computation time for convergence (128s) is similar to that of PPO (130s for ~ 180 episodes) with significantly reduced variance.



(a)



(b)

Fig. 2: Performance of training with various RL methods using Stable Baselines3 for (a) 100 episodes and (b) 400 episodes

C. *Performance comparison of policies*

Table III shows the comparison w.r.t percentage of packets served without delay violations. Evidently, the counterfactual versions marginally outperform their counterparts while EDF fails drastically as it does not adapt to channel conditions.

TABLE III: Average Success Rates on testing different policies

	CA2C	**A2C**	**CPPO**	**PPO**	**DQN**	**EDF**
%Success	97	95	93	90	85	51

D. *Performance with varying number of users*

Fig.3 shows the training performance when the number of users is increased in multiples of two while the service rates are increased proportionately (else, queues will be unstable).

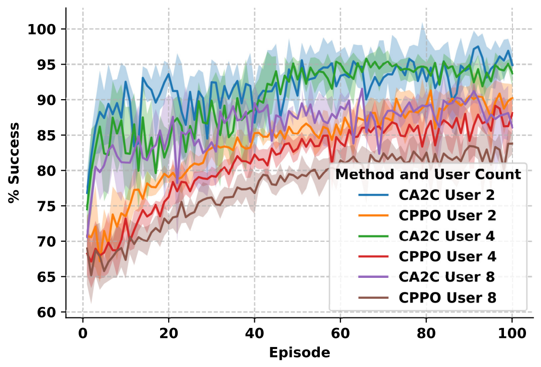


Fig. 3: Training performance with varying number of users

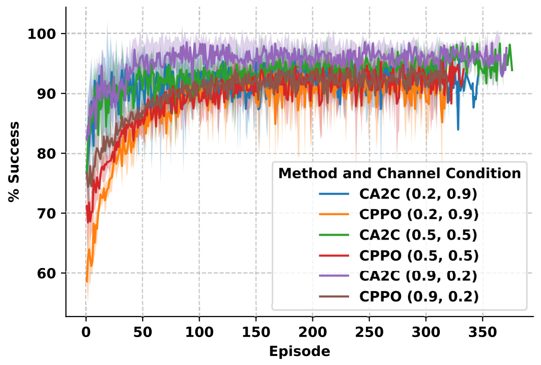


Fig. 4: Training performance with varying channel conditions

Clearly, CA2C outperforms CPPO in all scenarios. However, a decrement in the success percentage is observed with increase in the number of users. This shows an inability of the policy to scale with number of users. It is important to note that each user brings in more heterogeneity in the traffic since the arrival rates for the users are randomly allocated in the model which is pragmatic if all users have different service requirements. Since the traffic pattern is not a part of our state, this might cause the model to fail with scale in users.

E. Performance with varying channel conditions

Fig.4 shows the training performance under different channel conditions. The tuple represents (p_{01}, p_{10}) values. We take high low combinations to understand if the model adapts to varying channel conditions. As expected, when the transition probability from bad to good channel is high compared to good to bad transitions, the success probability is initially high since the service rate is high and vice-versa. However, with episodes, they converge appropriately.

V. Conclusions

This paper explores the importance of causal models in scheduling for wireless networks. We demonstrate how these models can assist in deriving scheduling policies aimed at minimizing delay violations in wired networks. Using counterfactuals, we establish that EDF is the optimal policy, which leads us to explore the potential of causal models for wireless channels. We develop a causal model for wireless channels and propose counterfactual-based policy gradient algorithms with proven performance improvement guarantees. Our findings indicate a substantial reduction in training time and variance, offering a robust learning framework. However, we also note that the learned policies do not scale well with an increasing number of users, necessitating further investigation.

References

[1] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjöland, and F. Tufvesson, "6g wireless systems: Vision, requirements, challenges, insights, and opportunities," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1166–1199, 2021.

[2] P. Hu, Y. Chen, L. Pan, Z. Fang, F. Xiao, and L. Huang, "Multi-user delay-constrained scheduling with deep recurrent reinforcement learning," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 2344–2359, 2024.

[3] S. Shakkottai and R. Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," in *Proceedings of the 2nd ACM international workshop on Wireless mobile multimedia*, pp. 35–42, 1999.

[4] R. Singh and P. Kumar, "Throughput optimal decentralized scheduling of multihop networks with end-to-end deadline constraints: Unreliable links," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 127–142, 2018.

[5] P. Nuggehalli, V. Srinivasan, and R. Rao, "Energy efficient transmission scheduling for delay constrained wireless networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 3, pp. 531–539, 2006.

[6] J. Pearl, "Causal inference in statistics: An overview," *Statistics Surveys*, vol. 3, no. none, pp. 96 – 146, 2009.

[7] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell system technical journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[8] P. C. Nguyen and B. D. Rao, "Delay control for cdf scheduling with deadlines," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3103–3107, IEEE, 2015.

[9] S. Zoppi, J. P. Champati, J. Gross, and W. Kellerer, "Dynamic scheduling for delay-critical packets in a networked control system using wirelesshart," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2020.

[10] E. Bareinboim, A. Forney, and J. Pearl, "Bandits with unobserved confounders: A causal approach," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[11] F. Lattimore, T. Lattimore, and M. D. Reid, "Causal bandits: Learning good interventions via causal inference," *Advances in neural information processing systems*, vol. 29, 2016.

[12] Y. Lu, A. Meisami, and A. Tewari, "Causal markov decision processes: Learning good interventions efficiently," *arXiv preprint arXiv:2102.07663*, 2021.

[13] L. Buesing, T. Weber, Y. Zwols, S. Racaniere, A. Guez, J.-B. Lespiau, and N. Heess, "Woulda, coulda, shoulda: Counterfactually-guided policy search," *arXiv preprint arXiv:1811.06272*, 2018.

[14] "Welcome to stable baselines docs! - rl baselines made easy — stable baselines 2.10.3a0 documentation." https://stable-baselines.readthedocs.io/en/master/. (Accessed on 07/25/2024).

[15] "Vanilla policy gradient — spinning up documentation." https://spinningup.openai.com/en/latest/algorithms/vpg.html. (Accessed on 07/25/2024).

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] "Going deeper into reinforcement learning: Fundamentals of policy gradients." https://danieltakeshi.github.io/2017/03/28/going-deeper-into-reinforcement-learning-fundamentals-of-policy-gradients/. (Accessed on 07/18/2024).

[19] "Overview — simpy 4.1.1 documentation." https://simpy.readthedocs.io/en/latest/. (Accessed on 07/26/2024).