On Meeting a Maximum Delay Constraint Using Reinforcement Learning

HOSSEIN SHAFIEIRAD, (Member, IEEE), AND RAVIRAJ S. ADVE, (Fellow, IEEE)
Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5T 2S8, Canada
Corresponding author: Hossein Shafieirad (hossein.shafieirad@mail.utoronto.ca)

ABSTRACT Several emerging applications in wireless communications are required to achieve low latency, but also high traffic rates and reliabilities. From a latency point of view, most of the state-of-the-art techniques consider the average latency which may not directly apply to scenarios with stringent latency constraints. In this paper, we consider scheduling under a max-delay constraint; this is an NP-hard problem. We propose a novel approach to tackle the scheduling problem by directly addressing the constraint. We consider the downlink of a multi-cell wireless communication network with nodes communicating with users each facing their own delay constraint on randomly arrived packets. Packets must be scheduled to meet the users’ delay constraints. Our main contributions are first, proposing a new search approach, Super State Monte-Carlo Tree Search (SS-MCTS), as a version of regular MCTS modified for large-scale probabilistic environments; second, developing trained value and policy networks to reduce computational complexity, and finally, addressing the scheduling problem through a reinforcement learning framework. Our numerical results demonstrate that the proposed approach significantly improves the packet delivery rate over a baseline approach while meeting the max-delay constraint, and addressing the scalability as the main issues in large action-state spaces.

INDEX TERMS Monte-Carlo tree search, scheduling, reinforcement learning, max-delay constraints.

I. INTRODUCTION Latency plays a key role in a wide range of applications. Ultra-reliable low-latency communications is often a crucial requirement for many applications such as remote healthcare, virtual reality (VR), public safety, etc. [2], [17], [38]. Such applications have in common strict requirements in terms of capacity, hard end-to-end latency constraints and high reliabilities [31]. Considering the requirements of such applications, average delays are not of interest given that an instantaneous disruption in the transmitted data will lead to a poor performance of the overall system. Furthermore, in applications such as remote surgery, for instance, different tasks face different priority levels as well as tolerable deadlines to be served.

In this paper, our main goal is to consider scheduling a given number of flows (interchangeably called users) with random packet arrivals (with a known arrival rate) and a hard latency constraint (maximum tolerable delay) in the downlink of a wireless communication network. We wish to minimize the dropping of packets that comprise these flows; however, finding the optimal schedule for a set of flows is NP-hard [32]. This is the motivation for our work: we wish to find an efficient approach to tackle the scheduling problem in a realistic large-scale scenario. To the best of our knowledge, in considering the issue of latency, most state-of-the-art technologies focus on the average delay (which can, often, be translated into a throughput constraint [25]). Those addressing the hard delay constraint suffer from several limitations in large-scale probabilistic environments; we discuss these contributions and their limitations below.

Since our scheduling problem of interest is NP-hard, we propose a technique based on Monte-Carlo Tree Search (MCTS) to address the maximum delay constraints. We consider a multi-cell network with base-station (BS) serving multiple users simultaneously on multiple channels.
The packets meant for each user must meet the user’s hard delay constraints, i.e., it must be scheduled within a given time period or dropped. Our proposed approach can be applied to more general scenarios facing large-scale probabilistic environments with delay-sensitive tasks.

In this regard, our main contributions in this work are:

- First and foremost, proposing the Super State Monte-Carlo Tree Search (SS-MCTS) method as a modified version of regular MCTS to account for delay-sensitive large-scale probabilistic environments.
- We develop value and policy networks and model the problem through a reinforcement learning (RL) framework to address the issue of computational complexity.
- Finally, we apply the proposed SS-MCTS method to the scheduling problem in cellular networks considering the maximum delay constraint on flows and illustrate the gains possible using our deep learning framework.

It is worth mentioning that large-scale probabilistic scenarios with random packet arrivals have been studied with various low-complexity algorithms proposed in the literature [39]. However, our main focus in this work is to introduce SS-MCTS as an efficient tool which can be applied to such systems, especially for the max-delay constraint.

The MCTS approach is a best-first tree search algorithm which, unlike the full-tree search approaches, focuses on the more promising nodes by expanding the search tree based on random sampling of the search space. MCTS has gained significant attention after its breakthrough performance in the game of Go [27], [28]. In the MCTS method, a search tree is generated in which each node represents a possible state of the system and each branch represents an action which causes a transition to a new child node (new state) with an immediate reward. The application of MCTS is based on a large number of rollouts. In each rollout, the experiment (e.g., a move in the game) is played out until a terminal state is reached by selecting actions at random (or based on some algorithm). The final result of each rollout is then backed up to update the weights of the edges in the tree so that better nodes are more likely to be chosen in future action selections.

Since we consider a probabilistic environment, i.e., the flows to be scheduled are random, the idea of considering a super state, as the set of all possible next states resulting from different random flow arrivals, allows for the consideration of large scale probabilistic environments with lower complexity compared to the regular MCTS. As one of the secondary benefits of the proposed SS-MCTS search approach with fewer rollouts, and therefore less computational complexity, is that it can be integrated into asynchronous or synchronous distributed deep reinforcement learning (RL) methods resulting in an improvement in sample efficiency.

The SS-MCTS approach, combined with deep RL, addresses large-scale scenarios, as an extension to the work in [23], the main issue with tree search approaches. The main contribution and focus of this work compared to the work in [23] is considering a value and policy neural network combined with the SS-MCTS approach to improve the performance and reduce the computational complexity of the proposed approach. In this regard, we have proposed a distributed multi-agent RL framework which significantly improves the performance, in an online manner. Also, the value and policy neural networks, trained efficiently by SS-MCTS, can be used with much less computational complexity compared to the SS-MCTS. Furthermore, in this work, we consider a more realistic model for scheduling in the downlink of a multi-cell network and further reduce computational complexity by both modifying the action-selection in large-scale scenarios and addressing the scheduling problem through a distributed multi-agent RL framework. This efficiency can be traded off for the performance improvements of the proposed SS-MCTS approach for larger scale scenarios. We also analyse the complexity of the proposed approach. As well as the aforementioned contributions, the scheduling problem considered in this work is selection of multiple tuples of users and their assigned channels per each timeslot in a multi-cell scenario while in [23] a single user is selected per timeslot in a single-cell scenario.

The remainder of this paper is organized as follows. We first review the relevant literature on delay sensitive communication systems in Section II. Section III presents our system model. The proposed SS-MCTS technique is then described in Section IV. The framework on RL for the scheduling problem with max-delay constraints as well as the SS-MCTS approach combined with deep neural value and policy networks are introduced in Section V. In Section VI, we provide simulation results illustrating the performance of the proposed techniques. Finally, Section VII concludes the paper.

II. LITERATURE REVIEW

As our work covers such different areas as communications, tree search methods and reinforcement learning, here we discuss the most relevant works in these areas.

The related works accounting for a delay constraint mainly consider the throughput-delay trade-off, the delay-limited link capacity, and channel coding schemes for low-latency communications in 5G systems [26]. Importantly, these works generally consider the average delay in delivering data packets. While an average delay constraint can be converted to a throughput constraint [25] (and so is easier to address), applications such as remote surgeries may be better served by a max-delay constraint [14]. At the network layer, recent works include end-to-end delay bounds in wireless networks using large-deviations theory [30], the use of short transmission time intervals, and delay-limited throughput [6], [20], average delay of network coding in the downlink [37] and a trade-off between throughput and guaranteeable delay [17].

In [35], the problem of minimizing delay in edge information sharing for vehicle-to-vehicle links is investigated. In [2], a Terahertz (THz) cellular network is considered to provide high-rate VR services and the achievable reliability and latency of VR services over THz links are characterized. In remote surgery applications, the transmission delay of
signals as well as tasks’ operation (scheduling) delays need to be handled. Most of the works in this area are focused on signal delay and round trip signal latency [1], [19], [38]. However, efficient scheduling of arrived tasks with different priorities and task deadlines is also crucial. The authors in [9] consider a strict delay constraint for energy harvesting devices to find the best number of packets for transmission by modeling the problem as a Markov decision process (MDP). Solving the MDP through value iteration requires iterating over all states. Such an approach is, therefore, impractical in large-scale scenarios.

As mentioned earlier, scheduling to meet a maximum delay constraint in an NP-hard problem; as such, small-scale problems can be solved optimally using approaches such as Branch and Bound (B&B) [18]. However, for large-scale networks several heuristics have been proposed. One such set of heuristics is based on rules that decide which task to schedule according to the current state of the problem, known as dispatch rules [13]. Some other common “greedy” heuristics are based on choosing the task with the longest processing time remaining or the shortest processing time [22]. Another approach is to first schedule the packets with lowest time-to-drop (TTD) times and in case of finite buffer size, both TTD time and buffer capacity are considered. The Algebraic meta-model, is another proposed approach which makes collective decisions in successive process stages, not separately for individual objects [5]. Although heuristic approaches are simple and practical, especially in large-scale scenarios, as we will see, they suffer significantly in terms of their performance. A greedy heuristic will act as our baseline approach.

In this paper, we propose the use of RL and deep learning to address the max delay latency constraint through a sequential decision making model. The most common RL framework considered in the literature has been deep Q-learning (DQN), e.g., increasing the number of served users by considering the delay violation rate as in [40]. The delay constraint using DQN has been considered in some studies such as resource allocation for vehicle-to-vehicle communications, e.g., [41]. These approaches mainly consider the delay constraint as a penalty in the calculation of rewards, i.e., do not address max-delay as a constraint; also, the performance for large-scale scenarios is not discussed.

In [10], a network scheduler for deadline-driven data transfers is proposed to maximize the network utilization. In this problem, up to six users are considered and no flow is dropped if the deadline is not met and therefore, from our perspective is different in both scale and approach in addressing the max-delay constraint. In [36], DQN is used to design an uplink random access scheme for delay-constrained heterogeneous wireless networks in a two-user case.

The main issue with Q-learning based approaches is that the associated computational complexity grows exponentially as the problem parameters, such as the number of flows or users increase. Calculating the Q-value for all possible actions is not practical in large-scale scenarios and therefore an approximate method such as DQN is used in the literature which does not efficiently address the latency constraints in such scenarios.

Tree search-based methods form another category of value-based approaches to directly address the max-delay problem. Although through tree search methods more actions, and therefore more realistic models, can be considered, a full tree search in large search spaces is not practical. This limitation has been alleviated by Monte Carlo Tree Search approaches by reducing the depth of the search tree as well as using an efficient policy to decrease the effective breadth of the search tree [29]. Constrained MCTS for partially observable Markov decision processes (POMDP) [15] and MCTS with information layer [34] (which uses probabilistic information of the environment to estimate the expected value of future actions) are the works most relevant to our problem. In these schemes, the next state as a result of an action applied in the current state is not known deterministically. However, these schemes cannot be directly applied in scenarios with a large number of users since the calculation of the expected value of all future actions is not practical.

In [3], MCTS with deep neural networks is used for pilot-power allocation in the uplink of a massive MIMO system. In this work, a regular MCTS approach is considered unlike our work which focuses on modifying the MCTS approach to decrease computational complexities. Also, in [3] only a single decision is made while our maximization problem is defined over a finite time horizon and therefore, our proposed approach considers all decisions required within a finite time horizon which increases the complexity of search tree.

As mentioned, most state-of-the-art technologies focus on average delay. The approaches which consider hard delay do not address a max-delay constraint and have not discussed practical large-scale scenarios, the main focus of our work. As the problem of our interest in NP-hard, the proposed approaches in the literature to tackle this problem, such as tree search algorithms, must be modified in order to be an efficient tool to schedule a large number of users with individual max-delay constraints on their packets. In this work, our modifications of the MCTS approach also allows for the consideration of large scale probabilistic environments with lower complexity compared to the regular MCTS approach. The proposed SS-MCTS search approach with fewer rollouts, and therefore less computational complexity, is integrated into deep learning methods resulting in an improvement in sample efficiency and therefore performance.

III. SYSTEM MODEL

Considering the main purpose of this work is addressing a maximum delay constraint in probabilistic environments, in this paper we consider the problem of scheduling in the downlink of a multi-cell wireless network with multiple frequency bands available to each BS per cell.

We consider a network area partitioned into, for convenience, identical hexagonal cells, with one BS located at the
The system considered in this work is illustrated in Fig. 1 which depicts a 7-cell scenario with 2 frequency bands available per BS. Users with different priorities (i.e., different delay constraints and arrival rates) are illustrated in different colors. The frequency band assigned to each user is depicted with orange and green arrows (illustrating the 2 available frequency bands). The downlink channel from the \( m \)th BS to the user \( j \) associated with the base station \( n, j = n \), on the frequency band \( f_n \) is given by

\[
\beta_{m,n} = |g_{m,n}|^2 \sqrt{\frac{P_{m,n}}{d_0}} \sim \mathcal{CN}(0, 1),
\]

where \( \beta_{m,n} \) denotes the small-scale Rayleigh fading component and the large-scale path loss component is denoted by

\[
\beta_{m,n} = \left(1 + \frac{d_{m,n}}{d_0}\right)^{-\alpha},
\]

where \( d_{m,n} \) and \( d_0 \) denote the distance between the \( m \)th BS and the user \( j \) associated with the BS \( n \) and the reference distance for the path loss model, respectively. The path loss exponent is denoted by \( \alpha \). The Rayleigh fading components for the channel to a particular user on different frequency bands are assumed to be statistically independent (i.e., for \( f \neq f' \), \( g_{m,n} \) and \( g'_{m,n} \) are independent). Time division duplexing is employed and the small-scale fading components to any user in different timeslots are assumed to be statistically independent. Furthermore, we assume that the small scale fading components vary independently from one timeslot to another while the path loss components remain identical. The signal-to-interference-plus-noise (SINR) ratio for user \( j \) in BS \( b \) is given by

\[
\gamma_{j,b} = \frac{y_{j,b}}{\sum_{b'=1, b' \neq b} y_{j,b'} + \sigma^2_{j,b}},
\]

resulting in an achievable rate of

\[
R_{j,b} = W \log(1 + \gamma_{j,b})
\]

on the frequency band \( f \).

Therefore, the combined data rate achieved by the user \( j \) in timeslot \( t \) is given by summation of \( R_{j,b} \) for all \( F \) available frequency bands which is denoted by \( R_{j,b,\text{tot}} \) and given by

\[
R_{j,b,\text{tot}} = \sum_{f=1}^{F} R'_{j,b,f}
\]

A. PROBLEM FORMULATION

Our goal is to find a practical and efficient policy for each BS to select users such that the total number of dropped packets is minimized. Each time a user is selected, based on the quality of the channel, packets stored for that user in the BS can be transmitted. The scheduling of each packet can be interpreted as a task with a deadline. Specifically, we consider a finite

1 If the BS has multiple receive antennas, a simple beamforming scheme like matched filtering can be easily incorporated.
time horizon divided into $T$ timeslots $\mathcal{T} = \{1, 2, \ldots, T\}$. The total number of packets that arrive for the $j$th user associated with BS $b$ is denoted by $K_{j-b}$ (which is random). Specifically, packet $k_{j-b}$ arrives at time $t_{k_{j-b}} \in \mathcal{T}$.

Our goal is to minimize the total number of dropped packets over a finite time horizon $\mathcal{T}$. In order to reach this goal, we aim for minimizing the total number of dropped packets in each cell. Mathematically, the problem of interest for all users in the network can be written as

$$\min_{x_{k_{j-b},f}} \left[ \sum_{b=1}^{B} \sum_{j=1}^{J} K_{j-b} \sum_{k_{j-b}=1}^{K_{j-b}} \mathbb{I}\{T_{k_{j-b}} - t_{k_{j-b}} > D_{j-b}\} \right]$$

subject to

$$\sum_{t=1}^{T} \sum_{f=1}^{F} x_{k_{j-b},f} \leq 1 \quad b \in \mathcal{B}, 1 \leq k_{j-b} \leq K_{j-b}$$

$$\sum_{k_{j-b}=1}^{K_{j-b}} x_{k_{j-b},f} \leq \frac{R_{b,f} \times T_{b}}{m} \quad t \in \mathcal{T}, f \in \mathcal{F}, b \in \mathcal{B}, 1 \leq j \leq J, 1 \leq k_{j-b} \leq K_{j-b}$$

$$x_{k_{j-b},f} \in \{0, 1\} \quad t \in \mathcal{T}, f \in \mathcal{F}, b \in \mathcal{B}, 1 \leq k_{j-b} \leq K_{j-b}$$

(1)

where $\mathbb{I}\{\cdot\}$ denotes the indicator function ($= 1$ if the statement is true, $= 0$ if false), $t_{k_{j-b}}$ denotes the arrival time of packet $k$ for user $j$ associated with BS $b$ (user $j - b$) and $T_{k_{j-b}}$ denotes the scheduled transmission time. In case the time duration that packet $k$ for user $j - b$ is stored, i.e., $T_{k_{j-b}} - t_{k_{j-b}}$, is less than its delay constraint, $D_{k_{j-b}}$, the packet can be transmitted. Otherwise, the packet will be dropped. $T_{b}$ denotes the length of each timeslot and $m$ represents the size of one packet, in bits (considering the rate is represented in bits per second).

The binary optimization variables are $x_{k_{j-b},f}$ such that $x_{k_{j-b},f} = 1$ if in timeslot $t$, the packet $k$ for user $j$, associated with BS $b$, is scheduled on frequency band $f$.

The first constraint in (1) ensures that each task is scheduled only once. The second constraint implies that in each timeslot $t$ and on each frequency band $f$, the total number of packets that can be scheduled for user $j - b$ is limited by the maximum data rate achievable on frequency band $f$. This constraint allows for simultaneous transmission of multiple stored packets for a user on each frequency band in each timeslot (if the achievable data rate allows).

The scheduled transmission time $T_{k_{j-b}}$ in Problem (1) is a function of the optimization variable $x_{k_{j-b},f}$ as $T_{k_{j-b}}$ denotes the time $x_{k_{j-b},f} = 1$. Mathematically, this can be written as

$$T_{k_{j-b}} = \inf(t \in \mathcal{T}|x_{k_{j-b},f} = 1),$$

though we note that (2) results in $T_{k_{j-b}}$ to be infinite in case the optimization variable $x_{k_{j-b},f}$ remains zero over the finite time horizon $\mathcal{T}$. The scheduling problem as stated in (1) is NP-hard, i.e., there is no polynomial-time algorithm to solve this optimization problem [16].

The optimization Problem (1) takes the whole finite time horizon into account for calculating the number of dropped packets within one cell. However, the future information about channels and arrivals is not available and therefore, it becomes non-causal resulting in Problem (1) to be infeasible. Therefore, one option is to re-visit the problem of interest by considering the optimization problem in each timeslot $t$ and minimizing the expected number of dropped messages in each timeslot. Thus, considering the same constraints in Problem (1), the objective function in this problem can be replaced by

$$\min_{x_{k_{j-b},f}} \mathbb{E} \left[ \sum_{b=1}^{B} \sum_{j=1}^{J} \sum_{k_{j-b}=1}^{K_{j-b}} \mathbb{I}\{T_{k_{j-b}} - t_{k_{j-b}} > D_{j-b}\} \right]$$

(3)

where $\mathbb{E}[\cdot]$ denotes expectation (over the future arrivals and channel states).

In deterministic scenarios, the B&B technique provides the optimal solution for the problem in (1) by implicitly enumerating all the possible solutions of the problem on a search tree. The complexity of the B&B approach grows exponentially with the size of the system. Therefore, even for deterministic arrivals, B&B is impractical for real-time execution in large-scale scenarios. With random arrivals, the B&B approach is infeasible because the objective function is impossible to evaluate in closed form and taking the expectation requires Monte-Carlo simulations.

To build toward an effective solution we recognize that the problem in (3) can be formulated as a constrained MDP and therefore, the optimal solution can be achieved through the well-known value-iteration method [21], [33]. In this MDP, the states at each time comprise the vector of arrived packets, the arrival time for each packet and the channel state information at that time; the actions are the scheduling decisions $\{x_{k_{j-b},f}\}$. However, solving the MDP directly through value-iteration is not practical in our problem as the complexity of solving an MDP is grows exponentially with the size of state and action spaces. Therefore, even for small number of users and channels, finding the optimal solution is impractical.

Heuristic techniques provide significantly worse performance than the optimal approaches such as the B&B technique, especially for large-scale problems.

These reasons constitute the main motivation for our use of the Monte-Carlo Tree Search technique to provide a balance between the optimal and heuristic approaches. The MCTS approach provides a near-optimal solution with lower computational complexity compared to the B&B approach [24]. Importantly, by fixing the number of rollouts, the computation load of MCTS approach can be bounded. In the following section, we first describe the regular MCTS technique and then provide our novel modified MCTS method, called Super State Monte-Carlo Tree Search (SS-MCTS).
IV. PROPOSED APPROACH
The MCTS approach is a best-first tree search algorithm which focuses the search on the more promising nodes [27], [28]. In the MCTS method, a search tree is generated in which each node at layer \( t \) of the tree represents a possible state of the system at time \( t \), denoted by \( S_t \) and each branch represents an action \( a_t \) (from the valid set of actions in each state), which causes a transition to a new child node, \( S_{t+1} \), with an immediate reward \( R_t \). In our problem, the immediate reward is proportional to the total number of dropped packets in a direct negative relationship as a function of maximum delay constraint on packets. Importantly, in our model, the number of actions are large, and, since packet arrivals are random, the next state is not known deterministically. This necessitates the modifications described in the next section.

A state at time \( t \) is defined as \( S_t = [s_m, s_s, h_t] \) in which \( s_m \) represents the vector of arrived packets, \( s_s \) the vector of arrival times for each packet and the channel state information is denoted by \( h_t \). The possible actions in timeslot \( t \), denoted by \( a_t \), are different choices of selection of users scheduled as well as the frequency bands assigned to them. Action selection is based on a policy function, denoted by \( \pi(a|s) \), defined as the probability distribution over possible actions, \( a \), from state \( s \). Our main goal is to find the best scheduling policy through interactions with the environment to minimize the rate of dropped packets. The environment includes cells, BSs, users and their randomly arriving delay-sensitive packets.

As illustrated in Fig. 2, MCTS comprises four steps: Selection, Expansion, Rollout and Back-propagation. Starting from a base node (root), in the selection step the search tree is traversed from the root node to an existing leaf by action selections using a tree policy \( \pi_t(.) \). The leaf node is then expanded in the expansion step and the newly expanded node, demonstrated by the green node in Fig. 2, is added to the tree. In the rollout step, Monte-Carlo rollouts are performed according to a default policy \( \pi_d(.) \) and the estimated value is then back up in the back-propagation step through parent nodes to update their state-action value estimates. MCTS uses random actions as the default rollout policy.

![FIGURE 2. MCTS Steps.](image)

One of the important parameters in implementation of MCTS is the best action selection policy (tree policy). By random exploration we can try out unexplored nodes; however, this might result in wasting computation resources in exploring bad actions. On the other hand, exploitation of only the most promising nodes might not give a chance to the other unexplored high-value nodes. The Upper Confidence Bounds for Trees (UCT) algorithm is the most widely used solution in this regard based on the principle of optimism in the face of uncertainty [29]. In the UCT algorithm, the next action is selected based on the following value

\[
Q(s, a) + c \times U(s, a)
\]

where \( Q(s, a) \) represents the total average reward received for taking action \( a \) in state \( s \) and \( U(s, a) \) is defined as

\[
U(s, a) \propto \frac{N}{1 + N(s, a)}
\]

where \( N \) denotes the total number of times state \( s \) has been visited and \( N(s, a) \) is the number of times the state-action pair \((s, a)\) has been selected. The constant \( c \) balances the two terms in (4); the first term is the state-action value function (exploitation term), while the second is the confidence term (exploration term). Each time an action \( a \) is selected at node \( s \), \( N(s, a) \) increments and therefore the uncertainty is presumably reduced, and, as it appears in the denominator in (4), the exploration term decreases. The goal is to run the simulations up to the level where this estimated value of \( Q(s, a) \) gets as close as possible to the true value, \( Q^*(s, a) \). Each time a new action is selected based on (4), an averaging backup is used to update the value estimates of all edges on the path from the root to the selected edge as follows.

\[
N(s, a) \leftarrow N(s, a) + 1
\]

\[
Q(s, a) \leftarrow Q(s, a) + \frac{R + Q(s, a)}{N(s, a)}
\]

where \( R \) represents the reward for taking action \( a \) in state \( s \). In the next section, we discuss how the estimated value is generated in the proposed SS-MCTS approach.

The state-action value function \( Q(\cdot, \cdot) \) plays a key role in action selection. In regular MCTS, in order to estimate the value of a newly explored node, Monte-Carlo rollouts are performed using a random or some heuristic guided policies (default policy) from the expanded node to a terminal node. This value is then backed up through the tree in the back-propagation step.

For large action and state spaces, in order to get an appropriate estimate of the nodes’ values, a large number of rollouts are required which results in having high computational complexities; this issue is exacerbated in probabilistic environments where the calculation of averages over possible actions is required. In the next section, we discuss our proposed approach to alleviate this limitation of regular MCTS.

A. SUPER STATE MCTS
As mentioned earlier, the MCTS approach combined with neural networks has gained significant attention after its performance in the game of Go [27], [28] and has been used in several works [11]. Our main contribution in this work is to modify the MCTS approach for large-scale probabilistic scenarios and also, apply this technique to a cellular network
scenario with the max-delay constraint as well as the other constraints listed in Problem (1).

In probabilistic environments MCTS requires a large number of rollouts to achieve an appropriate estimate of node values. There are works in the literature on modifying MCTS approach for probabilistic environments [7]. In our case, the next state is generated according to the current state, selected action, channel state information and randomly arrived tasks (which, clearly, are a priori unknown). In large-scale environments, even more rollouts are required. Another challenge is how to define the reward function in order to directly and efficiently address the max-delay problem. This motivates the development of Super State MCTS.

The reward function in our model is depends on the number of dropped tasks. The selection and expansion steps in “Super State MCTS” method are done using the UCT algorithm similar to regular MCTS, as stated in Section IV, such that the next action is selected as follows

\[
a_t = \arg\max_a \{ Q(S_t, a) + c \times U(S_t, a) \} \tag{8}
\]

However, in SS-MCTS, we modify the rollout step in MCTS in order to efficiently address the unknown random arrivals and calculate the newly explored node’s value. In the rollout step of SS-MCTS approach, instead of requiring a large number of random Monte-Carlo rollouts to find an estimated value, a next “super state” is considered as the set of all possible next states. Specifically, for every state-action pair \((S_t, a_t)\) the super state \(SS_{S_t, a_t}\) is the set of all states that can be generated as a result of action \(a_t\) taken at the current state \(S_t\) and also random task arrivals.

In order to calculate \(Q(S_t, a_t)\) in the super state \(SS_{S_t, a_t}\), we calculate the average sum of dropped tasks within a super state by taking all the possible random arrivals into account. This requires knowledge of the arrival rates. However, as we will discuss in Section VI, it is important that the proposed reward function provides a very accurate ranking of the actions’ values instead of providing a correct estimate of values (i.e., an appropriate choice of reward function which is higher for better actions).

The expected number of dropped tasks for the super state \(SS_{S_t, a_t}\) is calculated as follows

\[
\sum_n \Pr((S_t, a_t) \rightarrow S_n) \times d((S_t, a_t) \rightarrow S_n) \tag{9}
\]

where \(\Pr((S_t, a_t) \rightarrow S_n)\) is the probability of reaching state \(S_n\) by taking action \(a_t\) in state \(S_t\), and \(d((S_t, a_t) \rightarrow S_n)\) is the number of dropped packets as a result of transition from state-action \((S_t, a_t)\) to \(S_n\). The summation in (9) is over all possible arrivals which results in a set of possible states in the super state \(SS_{S_t, a_t}\).

After calculation of the expected number of dropped tasks for the super state, the calculated value is backed up through the path taken to this node in the tree (similar to the back-up step in regular MCTS) and therefore the values of the edges along this path are updated.

The use of SS-MCTS results in a significant decrease in the required number of rollouts compared to the regular MCTS approach as the knowledge of arrival rates is used instead of considering a large number of rollouts to reach a good value estimate. Effectively, we bypass the need for Monte-Carlo trials to account for the random arrivals. Therefore, the runtime and computational complexity decreases.

The SS-MCTS steps as proposed are demonstrated in Fig. 3. In the rollout step of Fig. 3, as an example, two future super states are considered (green boxes). The first super state is the set of states resulting from the state-action pair connected to that state. The second super state is the set of all states as a result of states in the first super state. In this way, the size of super states grows exponentially by considering more look-ahead steps. As we discuss in Section VI, the number of required look-ahead super states depends on the problem parameters. By taking more future steps into account, better value estimation is achieved but this also increases the computational complexity of the algorithm.

In order to calculate the expected number of dropped packets using the super state, we analyze the probability transition model of states. In this regard, we consider a Markov model such that states in every timeslot depends on their value in the last timeslot. Denote as \(b_t\) the number of stored packets at time \(t\), the number of remaining packets to be scheduled at time \(t + 1\) can be derived as

\[
b_{t+1} = \min(S, b_t - c_t + r_t) \tag{10}
\]

where \(r_t\) is the number of randomly arrived packets and \(c_t\) is the number of scheduled packets at time \(t\). Thus, the probability transition model from the state with \(b_t\) stored packets to the next state with \(b_{t+1}\) stored packets in case \(r_t\) packets arrive in timeslot \(t\) is calculated as follows

\[
\Pr(b_{t+1} = b' | b_t = b, c_t = c, r_t = r) = \Pr(b' = b - c + r) = \Pr(r = b' - b + c) \tag{11}
\]

The probabilities are calculated using the Poisson distribution and known arrival rates. The probability transition model in (11) can be used in the rollout step to calculate the average number of dropped packets in the super state rollout.

**B. MODIFICATION IN MCTS**

As mentioned earlier, the actions in the scheduling problem of our interest are in form of tuples of selected users for
scheduling and their assigned frequency bands. For a scenario with 2 users in one cell and 3 frequency bands, this has been demonstrated in Fig. 4.b. The total number of children for cell $b$ with $J_b$ users and $F_b$ frequency bands is $J_b^{F_b}$, which is a main concern for large-scale scenarios (i.e., large number of users and/or frequency bands). One way to alleviate this limitation is to consider frequency bands one by one in each timeslot and assign users to each frequency band (see Fig. 4.a). Although the depth of the original tree in this way will be multiplied by the number of frequency bands, however, by decreasing the branching factor, as we discuss in the next section, the complexity of the decision making approach as in Fig. 4.a is much lower than that of the tree in Fig. 4.b. Therefore, in this work, we consider the tree represented in Fig. 4.a for the proposed SS-MCTS approach.

### C. TIME COMPLEXITY ANALYSIS

Here we consider the simplest version of MCTS for the complexity analysis since the analysis for the SS-MCTS case is similar. We assume a total number of $I$ iterations for the MCTS approach at each root node. The expansion phase in MCTS can happen in constant time. For a tree with branching factor of $W$ and depth of $D$, the expansion step has the time complexity of $O(WD)$. The complexity of the back-propagation step is also proportional to the depth of tree. In what follows we discuss the complexity of simulation step.

There are several ways to consider the simulation step in MCTS. In general, a total number of $L$ random rollouts are considered which in every rollout a random child is selected at each level of tree until a terminal node is reached.

Most implementations consider only one rollout (i.e., $L = 1$), and therefore in applications similar to what we consider, as the terminal node is the last layer of tree (i.e., last timeslot), the simulation step takes time on $O(D)$. Therefore, for $L = 1$ rollout only, the total time complexity of MCTS approach is $O(I(WD + D) + D) = O(IWD)$. However, in random environments a large number of rollouts are required (i.e., $L \gg W$) for an accurate value estimation of the newly expanded node resulting in the simulation step having time complexity of $O(LD)$. This results in time complexity of $O(I(W + L + 1)D)$ or complexity of $O(ILD)$ for MCTS.

Therefore, the total number of rollouts linearly increases the complexity of the MCTS approach. In the proposed SS-MCTS approach, by considering the super states and one look-ahead step, the complexity of the tree through policy improvement techniques such as the UCT approach.

### V. MCTS WITH DEEP LEARNING

As discussed, the MCTS approach reduces the depth of the search tree by truncating the tree at newly explored states and replacing the truncated sub-tree by an approximate evaluation of the nodes’ value; MCTS also reduces the breadth of the tree through policy improvement techniques such as the UCT approach.

The proposed SS-MCTS approach decreases the computation required in the rollout step of MCTS. As we discuss in Section VI, the more number of super states (i.e., more number of look-ahead steps) results in better estimation of the newly explored node in SS-MCTS. However, the calculation of estimated values through super state rollouts for large number of look-ahead steps increases the computational complexity of this technique since the size of the super states grows exponentially when considering multiple steps. In this regard, in order to decrease the computational complexity of the SS-MCTS, as well as to improve the performance, we benefit from deep multi-agent reinforcement learning (MARL) by incorporating policy and value networks coupled with the SS-MCTS approach.

The structure of the RL framework for the scheduling problem comprises agents (in our problem, the BS in each cell) and a probabilistic environment in which packets arrive randomly for users. The environment is everything within a multi-cell network including BSs, users and their randomly arriving delay-sensitive packets. The agents interact with the environment by taking actions, denoted by $a$, indicating the scheduling of users based on the observed state of the system, denoted by $s$, and according to the policy $\pi(s)$. The state of the system comprises the stored packets of each user, the arrival time of each packet and the channel state information. The policy $\pi(s)$ is a function that selects the next action that can be taken at each state $s$ and is determined based on the values of action $a$, that can be taken at each state $s$, denoted by the function $Q(s,a)$.

We consider a multi-agent RL scenario and consider each BS in a cell as an agent which interacts with the users associated with that BS. Our goal here is to find
a policy for BSs which can independently schedule users within their cell. In order to get the best policy we benefit from the MCTS methods as explained in Section IV. The proposed SS-MCTS approach with deep multi-agent RL for the scheduling problem with Max-delay constraint is discussed in the next section.

\section{SS-MCTS WITH NEURAL NETWORKS}
In order to consider SS-MCTS approach with deep MARL for the scheduling problem of interest, one approach is to use a policy network which is trained to output the next action probabilities at each state and a value network trained to output state evaluations coupled with the SS-MCTS approach. Our goal is to have an effective value function, which accurately estimates the number of dropped packets from every single state $s$ and an accurate policy function which outputs the probabilities of dropped packets of choosing an action (user and their assigned channels) in every state. The policy network results in guiding the search toward the actions that are optimal with high probability, while the value network helps the SS-MCTS with evaluating actions of the tree.

In order to train the value and policy neural networks (VP-NN), the ideal approach is to generate labeled data listing the optimal solution through, for instance, B&B or full tree search methods and couple the trained networks with the SS-MCTS. However, as discussed in Section III, an optimal search is not practical in large state-action spaces. In this regard, one alternative approach is to first generate data from SS-MCTS and then consider supervised learning of achieved data to train neural networks. The trained value and policy networks are then combined with the SS-MCTS approach.

On the other hand, RL has been shown to be robust to disturbances in the dynamics \cite{8}. Also, unlike the supervised learning which requires a large amount of pre-sampled data and also training requires a large amount of memory and computation resources, by using (Multi-agent) RL framework, we can train the network more robust and with less computational costs. Our goal is to find a policy through which BSs can schedule users. Therefore, in this work, we consider an online training of neural networks using RL such that value and policy neural networks are coupled with SS-MCTS approach.

In the training of VP-NN using the SS-MCTS approach, both the value and policy networks learn features of the inputs (states in our case) which are useful for their training purposes. Although different features are learned for the different tasks, the extracted features might be useful for other tasks as well. In the defined deep learning framework for the scheduling problem, since the inputs to both value and policy networks are the same set of states, combining both networks into a single network can avoid multiple networks computing the same features; we therefore consider a single neural network $f_{\theta}(s)$ for both value and policy networks instead of having two separate networks. Hence, $\theta$ denotes the network parameters. The network takes states as its input and outputs both value $V_{\theta}$ and policy vector $\pi_{\theta}$. Multi-task learning can result in extracting richer and more efficient features which improves the training performance \cite{4}, and also avoids overfitting.

The input to the neural networks are the states $s$ passed through the value network and policy network. The value network’s output is the nodes’ estimated value $V_{\theta}$ and the policy network outputs the probability vector $\pi_{\theta}$ as the probability distribution over actions. The training goal is, therefore, to update the neural networks to minimize the error between the value network’s output and the terminal value, $Z$. 

\begin{figure}[ht]
\centering
\includegraphics[width=\textwidth]{figure4.pdf}
\caption{Different consideration of actions in one cell with 2 users (U’s) and 3 available frequency bands (or channels, C’s): (a) considering the set of users as actions, (b) considering the set of tuples of (frequency band, user) as actions.}
\end{figure}
(total number of dropped messages in one realization) and to maximize the similarity between the policy network’s output \( \rho_i \) and the vector of normalized visit counts \( \pi_i \).

For training the neural network \((V_\theta, \pi_\theta) = f_\theta(s)\), we randomly initialize the parameters of network \( \theta \). For each step of training, we run multiple episodes of the SS-MCTS approach using the current value and policy of VP-NN and generate new set of data used for training of the VP-NN. Let the total number of dropped packets at each run of SS-MCTS be \( Z \). \( Z \) is used as the target value for all states that have led to this value on the path of the tree that the terminal node has been reached. Then, the network parameters \( \theta \) are adjusted by minimizing the overall loss as follows

\[
I(\theta) = (Z - V_\theta)^2 - \pi^T \log \pi_\theta + \lambda ||\theta||^2
\]

where \( \pi \) is the vector of visit counts of each state’s children. The first term of \( I(\theta) \) is the mean-squared error, the second term is the cross-entropy loss, and the third term is used for \( L_2 \) weight regularization, controlled with a parameter \( \lambda \).

In the multi-cell scenario of our interest, since BSs in each cell are considered as the agents responsible for making the scheduling decisions, we consider a separate VP-NN per each cell (which includes one BS) separately. In this way, there is no need to consider all cells together and store the data from all cells which reduces the data transmission between BSs, and making the process of online training faster. This approach can be also useful for scenarios in which BSs in each cell serve different number of users, arrivals rates, etc.

Fig. 5 demonstrates the SS-MCTS approach and the process of data generation for each step of VP-NN training. As depicted in this figure, in each state \( S_i \), the SS-MCTS approach is executed and \( \Pi_i \), which is the normalized visit counts of all possible upcoming actions from \( S_i \) is obtained. The more simulations at state \( S_i \), the closer the \( \Pi(\cdot) \) gets to the true probability distribution over actions. The next action at each state \( S_i \) is selected according to the UCT approach as stated until the terminal state \( S_T \) is reached with terminal value \( Z \). The terminal value is the (negative of) total number of dropped packets in each realization. This describes one realization of SS-MCTS resulting in states \( S_1 \) to \( S_T \) as input data to the neural networks with target values \( Z \) and normalized probability vector \( \pi = \{\Pi_1, \ldots, \Pi_T\} \) accordingly to train the VP-NN.

We execute the SS-MCTS approach combined with VP-NN for a large enough number of realizations until the required training accuracy is achieved.

The proposed framework allows for online learning of values, unlike the approaches in the literature in which the value model is trained in an offline manner and used to pick actions. In the proposed SS-MCTS with VP-NN approach, a (local) model good enough to take an action in the current state is trained and used for action selection while the training continues.

In order to consider the VP-NN in the SS-MCTS approach, we use the policy vector \( \pi_\theta \) such that the constant term \( c \) in the UCT decision rule, as defined in (4), is multiplied by the probability of selection of an action. The action selection as stated in (8) also, needs to be updated accordingly. In this way, the policy vector guides the UCT algorithm toward more promising actions which, in return, results in improving the action-selection policy. The value \( V_\theta \) is also used in order to estimate nodes’ values. In this regard, we consider a linear combination of the super states’ estimated value, calculated as stated in (9), and the neural network’s output \( V_\theta \) to update nodes’ values. In this way, as shown in Section VI, using VP-NN improves computation and enhances performance in two ways: first, improving the policy and estimated values in the tree; second, reducing both the depth and breadth of the search tree.

VI. SIMULATION RESULTS
The previous section described the proposed SS-MCTS approach. In this section, we present the results of simulations illustrating the efficacy and performance of the proposed technique. To evaluate the performance of the proposed approach, we first consider the simplest non-trivial case of a single-cell in which channels having only two states of “good” and “bad”. Then, we simulate the performance of the SS-MCTS combined with VP-NN in downlink of a 7-cell network with wraparound and multiple frequency bands available to each BS. The proposed approach can be applied to a wide range of large-scale sequential decision making problems dealing with a max-delay constraint in probabilistic environments.

The evaluation metrics are the average rate of dropped tasks and the total run-time.² Since the optimal approach is not practical for the large-scale case, as the baseline we use a full tree search approach for a small number of users and for large-scale scenarios we use one good greedy choice, selecting the user with the largest number of tasks having the least remaining time assigned to the best channel. This takes the number of stored messages, channels and the delay constraint into account.

²The simulation results as well as execution times are obtained using a desktop computer with a 3.5GHz Intel® Core™ i9 — 9900X CPU, 65536 MB of RAM and NVIDIA GeForce 2080 Ti GPU.
A. SINGLE-CELL SCENARIO

1) DROPPED PACKETS

We first consider the performance of the SS-MCTS approach in terms of the number of dropped packets.

Fig. 6 plots the average number of dropped packets of the SS-MCTS technique in relation to that of the greedy approach in the small-scale scenario with 5 and 7 users distributed within the cell. In this figure, we demonstrate the impact of number of users, delay constraint on users and total number of look-ahead super states on the performance of SS-MCTS approach compared to a baseline greedy approach. The greedy choice for the single cell scenario selects users with highest number of stored messages and highest time for packets stored in the buffer.

Fig. 6 also represents two different cases of all tasks facing the same delay constraint of 4 timeslots (dashed curves) and users with different delay constraints of 3 to 7 timeslots chosen randomly (solid curves). In both scenarios, users experience different arrival rates and a large enough buffer size of $B = 20$ is considered for each user’s packets over a time horizon of $T = 20$ timeslots.

As shown in Fig. 6, the performance of the SS-MCTS, as expected, improves by taking more look-ahead super states into account for the state-action value estimation. It is worth mentioning that although by going deeper into the tree to calculate the value of an action, a better estimate of node values is achieved, this increases the complexity of the SS-MCTS approach exponentially as the size of super states grows exponentially with the number of future steps considered. Our results show that the more features considered in system parameters, the better performance of SS-MCTS achieved compared to the greedy choice. In this regard, for the case of varying delay constraints, as illustrated in Fig. 6 with solid curves, the tree search approach results in a better understanding of the problem parameters compared to the greedy choice which only takes the current state into account. The number of future steps to be considered for the rollout and estimating the state-action values depends on the desired level of improvement which based on the scale of the problem.

2) COMPARISON WITH THE OPTIMAL SOLUTION

Table 1 demonstrates the performance of SS-MCTS with the optimal solution (full tree search) for scenarios with up to 5 users. The results are for SS-MCTS[3] (the SS-MCTS with three look-ahead super states), normalized to the greedy choice which demonstrate a significant performance of the SS-MCTS compared to the optimal solution for computationally practical small-scale scenarios. In order to calculate the average number of dropped packets in this table, we run 10000 realizations of packet arrivals and calculate the averaged number of dropped packets in each simulation run for both SS-MCTS and full-tree search, and normalize to the value to the same value for the greedy choice to compare the performance of SS-MCTS approach with the optimal solution.

Table 1. Average dropped packets of SS-MCTS vs full tree search (single-cell scenario).

|                | 2 Users | 3 Users | 4 Users | 5 Users |
|----------------|---------|---------|---------|---------|
| SS-MCTS[3]     | 0.974   | 0.858   | 0.819   | 0.785   |
| Full tree search| 0.960   | 0.842   | 0.800   | 0.767   |

The results demonstrate that the SS-MCTS approach provides performance similar to the optimal solution for small-scale scenarios where a full tree search is computationally practical.

3) EXECUTION TIME

Table 2 compares the average execution time for each run of the SS-MCTS normalized to the greedy approach and regular MCTS approach. In order to calculate the average time per simulation, we run 10000 realizations of the SS-MCTS approach for different set of random arrivals and calculate the time that each full run of SS-MCTS approach takes to generate all the scheduling decisions compared to that of the regular MCTS approach, and normalize the runtime to the greedy value simulation time.

As is clear, the SS-MCTS based approaches are faster than the traditional MCTS approach. It is worth mentioning that the results for regular MCTS approach are with 500 rollouts for each node’s expansion which is significantly higher than that of SS-MCTS which takes only one up to a few look-ahead super states into account. In some applications as in [12], MCTS is performed with an extremely large number of rollouts (on order of $10^4$). However, in case the value estimation can be addressed through limited-depth rollouts as in our problem of interest, using super states can significantly decrease the computational complexity and therefore the time to select actions.
B. MULTI-CELL SCENARIO WITH VP-NN

In this section, we evaluate the performance of SS-MCTS with VP-NN in the downlink of a 7-cell hexagonal network with wraparound and multiple frequency bands available to each BS and adjacent BSs spaced 1000 m apart. We consider the same number of users in each cell. However, the proposed multi-agent RL scenario can be applied to more general cases. The numerical values of the simulation parameters utilized in this section are listed in Table 3.

Regarding the VP-NN, as discussed in Section V-A, by using a policy and value network we can reduce the depth and breadth of the SS-MCTS tree search resulting in a more scalable and efficient approach. In this section, we demonstrate how combining the SS-MCTS approach with neural networks affects the performance of the proposed approach. As mentioned, we use a single combined value and policy network, denoted by VP-NN, which comprises 4 layers with states as input and one single output for the value of each given state and a softmax vector as the output for the policy network generating the probability distribution of the next action at each state. We use the ReLU activation function for all layers. In order to train the VP-NN, 20000 realizations of the SS-MCTS combined with VP-NN are considered. The data used for training after each realization consists of all states in one realization, total number of dropped messages and the vector of normalized visit counts for all possible actions of each state. The training process repeats until the required accuracy level is reached. Therefore, we use online training and our approach improves over time; this is a crucial advantage of the proposed approach compared to offline learning scenarios.

1) DROPPED PACKETS

Fig. 7 demonstrates the advantage in the performance of the SS-MCTS approach with VP-NN compared to the greedy approach as well as the regular SS-MCTS technique. The total number of frequency bands per cell is one fourth of the number of users. Results are demonstrated for $T = 20$ timeslots with up to 40 users per cell each having different strict delay constraints ranging from 1 to 5 timeslots as well as different random arrival rates. This accounts for different priority and load of tasks for users. It is worth emphasizing that it is only the use of the SS-MCTS approach that allows us to address such a large number of users.

We have also considered a large-enough buffer size of 20, which for the considered system model parameters results in close to zero buffer overflow. This is important for evaluation purposes as the main goal here is to monitor the number of dropped packets due to expiration of delay only, and not due to buffer overflow.

The blue curve is the SS-MCTS approach in which the algorithm looks only one super state ahead in the rollout step. The orange dashed curve (SS-MCTS + VP-NN) represents the SS-MCTS approach combined with VP-NN such that, in the rollout step, a linear combination of the value network’s output and one super step ahead is considered and the policy is guided using the policy network’s output.

As demonstrated in Fig. 7, the combination of super state rollouts with value networks results in a better performance compared to the regular MCTS approach. As the number of users per cell increases, the gap between the two curves also increases showing the better performance of SS-MCTS + VP-NN in larger-scale environments compared to the regular SS-MCTS approach. Furthermore, the higher number of users per cell results in higher interference to surrounding cells causing the concavity of curves for larger number of users.

### TABLE 2. Average time per simulation (10 users) in seconds (single-cell scenario).

| Method     | SS-MCTS[1] | SS-MCTS[2] | SS-MCTS[3] | MCTS |
|------------|------------|------------|------------|------|
| Runtime    | 2.34       | 4.86       | 9.02       | 11.8 |

### TABLE 3. Numerical values of parameters used for evaluations (multi-cell scenario).

| Parameter                  | Value   |
|----------------------------|---------|
| BS transmit power          | $P_t = 43$ dBm |
| Total BW                   | $BW = 20$ MHz |
| Noise figure               | $N_f = 9$ dBm |
| Path-loss exponent         | $\alpha = 4$ |
| Reference distance in the path-loss model | $d_0 = 0.392$ m |

### TABLE 4. Average number of dropped packets for 10 timeslots, 3 users and 2 frequency bands per cell.

| Method                  | Greedy | SS-MCTS[2] | SS-MCTS + VP-NN | Full-tree Search |
|-------------------------|--------|------------|-----------------|-----------------|
|                         | 9.99   | 5.25       | 5.06            | 4.87            |
2) COMPARISON WITH THE OPTIMAL SOLUTION

The advantage in the performance of SS-MCTS and SS-MCTS+VP-NN approaches considering larger number of super state steps compared with full-tree search is demonstrated in Fig. 8. The results here are averaged over 2 to 9 users per cell, with a fixed number of 3 frequency bands and a total of $T = 10$ timeslots. As shown in Fig. 8, as expected, the performance of both the SS-MCTS and SS-MCTS+VP-NN improves by taking more look-ahead super states into account in the rollout step. As mentioned before, although by going deeper into the tree a better value estimation is achieved, the complexity increases as the size of super state grows exponentially by considering deeper layers of tree.

This is worth noting that since only a total of $T = 10$ timeslots are considered here, considering 2 or 3 super state steps gives an adequate estimate of the node values. However, for larger number of timeslots (for which the full-tree search is very time consuming and infeasible), this cannot be true. In this case, the trained neural networks combined with the tree can play a key role in improving the performance even by considering one or two super state steps ahead. Even in this scenario, the performance of SS-MCTS+VP-NN with only one or two super state steps is comparable with SS-MCTS approach with larger number of super states. Therefore, the combined neural networks with even one super state step in rollout can result is reasonable performance; a choice depending on the problem parameters can be made.

The exact number of dropped packets for a scenario with 3 users and 2 frequency bands available per cell for $T = 10$ timeslots is represented in Table 4. The small gap between both SS-MCTS and SS-MCTS with neural networks and the optimal approach makes it an efficient approach with much less computational complexity compared to the full-tree search.

3) EXECUTION TIME

In Table 5, the impact of number of timeslots (i.e., depth of tree) on execution time for a scenario with 3 users and 2 frequency bands per cell is demonstrated. As shown here, the runtime grows exponentially in the full-tree search as the number of timeslots increases while in SS-MCTS, the runtime grows almost linearly (as discussed in Section IV-C).

C. MODIFYING TRAINING BASED ON SS-MCTS

In order to train the neural networks, considering the actions as only selection of users, as demonstrated in Fig. 4.a, results in nodes in different levels of tree having different distributions. The reason is that for a scenario with $F$ number of frequency bands per cell, the packets arrive at the beginning of each timeslot (i.e., orange nodes in Fig. 4.a, where packet arrivals are applied to states and channels are updated). Therefore, we have used two separate VP-NNs which are trained separately based on levels of the tree which represent the beginning of a timeslot and based on the rest of levels. This resulted in a better performance compared to feeding all levels’ data to the same value-policy neural network.

As different states in one realization are correlated, considering all states of one realization of SS-MCTS labeled with the same terminal value $Z$ (the total number of dropped messages in one realization) results in an overfitting of neural networks since increased training of value networks results in memorizing the output labels. To avoid this issue, for the training of the neural networks we randomly select a few states from each realization to reduce the correlation between data samples.

The simulation results demonstrated that considering the more features in the problem, such as delay constraints on tasks, arrival rates and number of users, results in a better performance of SS-MCTS compared to the greedy approach. The main reason is that the greedy choice is fixed and cannot adapt itself with the new parameters. However, the SS-MCTS, by interacting with the environment gets a better understanding of how different features are affecting nodes’ values and therefore by considering a good action selection, searches for the most promising nodes.

| $T$ (timeslots) | SS-MCTS | Full-tree Search |
|----------------|---------|------------------|
| 4              | 0.026   | 0.016            |
| 6              | 0.034   | 0.12             |
| 8              | 0.047   | 1.13             |
| 10             | 0.066   | 10.25            |
| 12             | 0.076   | 94.59            |

![Figure 8. Effect of number of look-ahead steps in rollout step on the average dropped packets (multi-cell scenario).](image-url)
VII. CONCLUSION AND FUTURE WORKS
In this paper, we propose an efficient method which directly addresses the task scheduling problem in the downlink of wireless networks with delay-sensitive tasks. The key difference in our approach is that we consider a max-delay constraint on packets. This is an NP-hard problem which cannot be addressed efficiently in polynomial-time. The “Super State Monte-Carlo Tree Search” (SS-MCTS) algorithm combined with Value and Policy networks significantly decreases the computational complexity of the regular MCTS approach in large-scale scenarios as well as allowing for easier implementation in probabilistic environments using tree search based approaches.

Our main contributions are first, proposing the SS-MCTS method and considering it with value and policy neural networks as a modified version of regular MCTS approach to consider the delay-sensitive tasks in probabilistic environments. Further, modeling the problem through a value-based reinforcement learning framework and finally, modifying the proposed method for large-scale scenarios and applying it to the scheduling problem in multi-cell wireless networks considering the maximum delay constraint on tasks.

The significance of this work is due to the NP-hardness of the scheduling problem with delay-sensitive tasks. The SS-MCTS method, as a modified version of regular MCTS approach, can be applied to a wide range of probabilistic sequential decision making problems including communication systems scenarios. Also, combining SS-MCTS with value and policy networks reduces the computational complexity of the tree search approach, resulting in a more scalable and efficient algorithm for large-scale scheduling problems in probabilistic environments.

Our numerical results demonstrate that the proposed approach can significantly improve the delivery rate of packets because we directly address the maximum delay constraint in large-scale probabilistic scenarios unlike the state-of-the-art technologies.

Finally, since the complexity of tree search algorithms increases exponentially with the number of actions (users and their assigned channels in our case), those RL techniques focusing more on the policy (e.g., policy gradient methods) than value can be considered as an efficient tool compared to value-based techniques in case the scale of the system is larger than what can be considered even through the SS-MCTS approach combined with RL. Also, the impact of different parameters considered in different cells on the used multi-agent RL approach can be investigated.

REFERENCES
[1] M. Anvari, T. Broderick, H. Stein, T. Chapman, M. Ghodoussi, D. Birch, C. McKinley, P. Trudeau, S. Dutta, and C. Goldsmith, “The impact of latency on surgical precision and task completion during robotic-assisted remote telepresence surgery,” Comput. Aided Surg., vol. 10, no. 2, pp. 93–99, 2005.
[2] C. Chaccour, R. Anmer, B. Zhou, and W. Saad, “On the reliability of wireless virtual reality at terahertz (THz) frequencies,” in Proc. 10th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS), 2019, pp. 1–5.
[3] J. Chen, S. Luo, L. Zhang, C. Zhang, and B. Cao, “IPAS: A deep Monte Carlo tree search-based intelligent pilot-power allocation scheme for massive MIMO system,” Digit. Commun. Netw., vol. 7, no. 3, pp. 362–372, 2021.
[4] M. Crawshaw, “Multi-task learning with deep neural networks: A survey,” 2020, arXiv:2009.09796.
[5] E. Dudek-Dyduch, “Algebraic logical meta-model of decision processes—New metaphysics,” in Proc. Int. Conf. Artif. Intell. Soft Comput., 2015, pp. 541–554.
[6] A. Eryilmaz, A. Ozdaglar, and M. Medard, “On delay performance gains from network coding,” in Proc. 40th Annu. Conf. Inf. Sci. Syst., 2006, pp. 864–870.
[7] J. Eyck, J. Ramon, F. Guiza, G. Meyfroidt, M. Bruynooghe, and G. Van Den Berge, “Guided Monte Carlo tree search for planning in learned environments,” J. Mach. Learn. Res., vol. 29, pp. 33–47, Jan. 2013.
[8] B. Eysenbach and S. Levine, “Maximum entropy RL (provably) solves some robust RL problems,” 2021, arXiv:2103.06257.
[9] I. Fawaz, M. Sarkiss, and P. Ciblat, “Delay-optimal resource scheduling of energy harvesting-based devices,” IEEE Trans. Green Commun. Netw., vol. 3, no. 4, pp. 1023–1034, Dec. 2019.
[10] D. Ghosal, S. Shukla, A. Sin, A. V. Thakur, and K. Wu, “A reinforcement learning based network scheduler for deadline-driven data transfers,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2019, pp. 1–6.
[11] A. Guez, T. Weber, I. Antonoglou, K. Simonyan, O. Vinyals, D. Wierstra, R. Munos, and D. Silver, “Learning to search with MCTS Nets,” in Proc. 35th Int. Conf. Mach. Learn., vol. 80, 2018, pp. 1822–1831.
[12] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, “Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning,” in Proc. Adv. Neural Inf. Process. Syst., vol. 27, Red Hook, NY, USA: Curran Associates, 2014, pp. 1–9.
[13] H. Ingimundardottir and T. R. Philip, “Supervised learning linear priority dispatch rules for job-shop scheduling,” in Learning and Intelligent Optimization, Berlin, Germany: Springer, 2011, pp. 263–277.
[14] J. S. Ladoiye, “Control of surgical robots with time delay using model predictive control,” M.S. thesis, Dept. Mech. Eng., University of Ottawa, Ottawa, Canada, 2018.
[15] J. Lee, G.-H. Kim, P. Poupart, and K.-E. Kim, “Monte-Carlo tree search for constrained POMDPs,” in Proc. Adv. Neural Inf. Process. Syst., vol. 31, Red Hook, NY, USA: Curran Associates, 2018, pp. 1–10.
[16] J. Leung, L. Kelly, and J. H. Anderson, Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Boca Raton, FL, USA: CRC Press, 2004.
[17] D. Malak, M. Médard, and E. M. Yeh, “Tiny codes for guaranteeable delay,” IEEE J. Sel. Areas Commun., vol. 37, no. 4, pp. 809–825, Apr. 2019.
[18] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, “Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning,” Discrete Optim., vol. 19, pp. 79–102, Feb. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572528616000062
[19] E. M. Navarro, A. Nahome, R. Álvarez, F. Irene, and S. Anguiano, “A new teletherapy generation supported by 5G technology: Benefits and future trends,” Proc. Comput. Sci., vol. 200, pp. 31–38, Jan. 2022.
[20] G. Pocovi, K. I. Pedersen, B. Soret, M. Lauridsen, and P. Mogensen, “On the impact of multi-user traffic dynamics on low latency communications,” in Proc. Int. Symp. Wireless Commun. Syst. (ISWCS), 2016, pp. 204–208.
[21] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st ed. Hoboken, NJ, USA: Wiley, 1994.
[22] G. Ramya and M. Chandrasekaran, “Solving job shop scheduling problem based on employee availability constraint,” in Materials and Diverse Technologies in Industry and Manufacture (Applied Mechanics and Materials), vol. 376, Wolfera, Switzerland: Trans Tech Publications, Nov. 2013, pp. 197–206.
[23] H. Shafieirad and R. S. Adve, “On meeting a maximum delay constraint using RL,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), 2021, pp. 1–6.
[24] M. Shaghaghi, R. Adve, and J. Ding, “Multifunction cognitive radar task scheduling using Monte Carlo tree search and policy networks,” IET Radar, Sonar and Navigation, vol. 12, pp. 1437–1447, Sep. 2018.
[25] A. Sharifian and R. S. Adve, “Queue-aware joint dynamic interference coordination and heterogeneous QoS provisioning in OFDMA networks,” IEEE Trans. Wireless Commun., vol. 17, no. 4, pp. 2571–2587, Apr. 2018.
[26] M. Shirvanimoghaddam, M. S. Mohammadi, R. Abbas, A. Minja, C. Yue, B. Matuz, G. Han, Z. Lin, W. Liu, Y. Li, S. Johnson, and B. Vucetic, “Short block-length codes for ultra-reliable low latency communications,” *IEEE Commun. Mag.*, vol. 57, no. 2, pp. 130–137, Feb. 2019.

[27] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–503, 2016.

[28] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis, “Mastering the game of go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[29] M. Świerchowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, “Monte Carlo tree search: A review of recent modifications and applications,” *Artif. Intell. Rev.*, pp. 1573–7462, 2022.

[30] V. Y. F. Tan, C.-H. Liu, and I.-H. Wang, “Delay scaling laws of random wireless networks: Impact of blocklength,” in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 409–413.

[31] H. Tatari, M. Shafii, A. F. Molisch, M. Dohler, H. Sjöland, and F. Tufvesson, “6G wireless systems: Vision, requirements, challenges, insights, and opportunities,” *Proc. IEEE*, vol. 109, no. 7, pp. 1166–1199, Jul. 2021.

[32] J. D. Ullman, “NP-complete scheduling problems,” *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000075800080

[33] V. Varagapriya, V. V. Singh, and A. Lisser, “Constrained Markov decision processes with uncertain costs,” *Oper. Res. Lett.*, vol. 50, no. 2, pp. 218–223, 2022.

[34] J. Veness, K. S. Ng, M. Hutter, W. Uther, and D. Silver, “A Monte-Carlo AI* approximation,” *J. Artif. Intell. Res. (JAIR)*, vol. 40, pp. 95–142, Oct. 2011.

[35] R. Wang, X. Jiang, Y. Zhou, Z. Li, D. Wu, T. Tang, A. Fedotov, and V. Badenko, “Multi-agent reinforcement learning for edge information sharing in vehicular networks,” *Digit. Commun. Netw.*, vol. 8, no. 3, pp. 267–277, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864821000584

[36] D. Wu, L. Deng, Z. Liu, Y. Zhang, and Y. S. Han, “Reinforcement learning random access for delay-constrained heterogeneous wireless networks: A two-user case,” 2021, arXiv:2103.14917.

[37] H. Xing, Y. Liu, A. Nallanathan, Z. Ding, and H. V. Poor, “Optimal throughput fairness tradeoffs for downlink non-orthogonal multiple access over fading channels,” *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 3586–3571, Jun. 2018.

[38] S. Xu, M. Perez, K. Yang, C. Perrenot, J. Felblinger, and J. Hubert, “Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the d-v-trainer simulator,” *Surgical Endoscopy*, vol. 28, no. 9, pp. 2569–2576, 2014.

[39] H. H. Yang, A. Arafa, T. Q. S. Quek, and H. V. Poor, “Age of information in random access networks: A spatiotemporal study,” 2020, arXiv:2008.07717.

[40] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar, “Deep reinforcement learning based resource allocation in low latency edge computing networks,” in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2018, pp. 1–5.

[41] H. Ye, G. Y. Li, and B.-H. F. Juang, “Deep reinforcement learning based resource allocation for V2V communications,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.