2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Reinforcement Learning for Stochastic Max-Plus Linear Systems

Vignesh Subramanian, Farzaneh Farhadi, and Sadegh Soudjani

Abstract— **This paper studies the design of control policies for Discrete Event Systems under uncertainties. We capture the timing of the events using the framework of max-plus-linear systems in which the time between consecutive events depends on random delays with unknown distributions. Our policy synthesis approach is with respect to a cost function, and it can be extended directly to satisfy safety specifications on the timing of events. The main novelty of our approach is to translate the system evolution to a Markov decision process (MDP) that has an uncountable state space and develop a stochastic optimisation problem under the evolution of the MDP. To tackle the unknown distribution of uncertainties (thus unknown transition probabilities in the MDP), we employ model-free reinforcement learning to perform optimisations and find control policies for the system. Our implementation results on the 9-dimensional model of a railway network show superiority of our learning approach in comparison with the stochastic model predictive control approach.**

I. INTRODUCTION

A Discrete Event System (DES) is a dynamical system where the evolution of the states are triggered by some events. The state variables change only at definite points in time through instantaneous state transitions which we call *events*. These events could take place asynchronously over time. The states of the system remain unchanged in between events and could change only during the events. DES provides a general framework for many systems whose dynamics can be controlled by man-made constraints as well as physical laws. Examples of a DES include logistic systems, traffic networks, manufacturing systems and scheduling systems [1]. In all these application domains, we have finite resources that are shared over several jobs that tend to contribute towards the culmination of a final goal such as arrival and departure of trains through multiple stations over a period of time in a railway network, or manufacturing and assembling of products in a factory [2]. A DES also tends to be complex by having large state spaces depending on the number of resources and jobs the DES relies on to achieve its final goal.

Max-plus-linear (MPL) systems are a class of DES in which there is synchronisation (new operation starts once all the preceding operations have been finished) but no concurrency or choice is included in the model description. Analysis and control of MPL systems are difficult to be performed using mathematical concepts in conventional algebra due to the nonlinear nature of the dynamical equations and their high dimensionality. To reduce the complexity of the model, the state equations are interpreted in the max-plus algebra [2]. MPL systems become linear with respect to max-plus algebra, thus it is possible to use the features of the max-plus algebra to analyse MPL systems in a highly efficient manner and evaluate the characteristics and performance of the system.

Every MPL system has inherent uncertainties in its operation. These uncertainties can be caused by uncontrollable factors such as human errors, system input delays, weather conditions, or system utility malfunctions. These factors can cause the state variables of the DES to change. For example, an equipment malfunction in a production facility can cause uncertain delays in the manufacturing process. Therefore, it is desirable to incorporate uncertainties in the MPL model by including stochastic variables and use the framework of *stochastic MPL systems* [3].

A promising approach to tackle uncertainties with unknown distributions is to use model-free reinforcement learning for the control and optimisation of the system [4]. In recent years, reinforcement learning has grown in popularity due to its capability to deal with dynamic, large complex problem spaces and to encode safety constraints directly in its objective function. Reinforcement learning is able to train systems to respond to unforeseen environments because it learns through a continuous process of receiving rewards and punishments for each action taken.

The main contribution of this paper is to provide an optimisation strategy for *switching stochastic MPL systems* (SS-MPL), where the switching in the system models the control actions. Uncertainties in the system are assumed to have unknown distributions. To find the switching actions, we employ *Deep Q-learning* algorithm [5], which is a model-free reinforcement learning approach suitable for handling continuous state spaces and encoding safety constraints in its reward structure. This is achieved by translating the state evolution of the SS-MPL system to a *Markov decision process* that evolves probabilistically on the event index. We implement our approach on a 9-dimensional railway network model adapted from [6] by adding stochastic delays. We show the superiority of our technique in comparison with the stochastic model predictive control approach.

Related Work. One of the earliest studies on designing controllers for DES is performed by Ramadge and Wonham [7]. Their approach interprets the traces of the controlled system as a formal language and finds the controller such that this language realises the desired specifications. In the past years,

Vignesh Subramanian is with Georgia Institute of Technology, USA. Farzaneh Farhadi is with the School of Engineering, Newcastle University, UK. Sadegh Soudjani is with the School of Computing, Newcastle University, UK. The research of S.Soudjani is supported by the following grants: EPSRC EP/V043676/1, EIC 101070802, and ERC 101089047. Email: vsubramanian64@gatech.edu, F.Farhadi2@ncl.ac.uk, Sadegh.Soudjani@ncl.ac.uk.

many control strategies for DES have been formalised. The paper [8] proposes a hierarchical control approach for DES based on the concepts of control structures and observers. The paper [9] presents a learning approach to synthesise the largest nonblocking supervisor for DES by extending the L* algorithm [10]. The paper [11] proposes an iterative control approach for nondeterministic DES that computes a nonblocking supervisor from a possibly blocking one by remoning certain states.

Model-based control approaches designed for performing optimisations on MPL systems have been vastly improved in the recent years. The paper [12] proposes an internal model control approach for optimisation on MPL systems using a model that predicts the future behaviours. The paper [6] has used model predictive control (MPC) to optimise a switching MPL system which leads to mixed integer optimisations. The authors of [13] propose an adaptive MPC approach by incorporating a parameter identification method to adapt the MPL to changes of parameters. The paper [14] addresses the issue of partial synchronisation in MPL systems using MPC and feedforward control. The authors of [15] have considered infinite-horizon optimal control of MPL systems with additive discounted costs. Their approach uses full knowledge of the model and is based on space discretisations.

MPL systems have also been studied for satisfying logical requirements. The works [16], [17] study verification approaches to check properties of stochastic MPL systems using finite probabilistic abstractions. The work [18] studies reachability of MPL systems using finite abstract models, i.e., finding event timings that can be reached from an initial set of event timings. The paper [19] proposes an approach to perform reachability analysis of MPL systems using satisfiability modulo theories (SMT) solvers. A similar technique is used in [20] to study the transient of an MPL system, i.e., the number of steps leading to its periodic regime.

Reinforcement learning is an emerging approach capable of learning optimal actions for systems that have temporal dynamical changes [4]. This approach has been used recently for DES modelled with finite state machines [21] and fuzzy automata [22]. To use reinforcement learning for the computation of control strategies on the MPL systems, the system needs to be presented as an MDP. The paper [17] formulates a stochastic MPL system as a Markov chain over a continuous space and approximates it with a finite one to verify properties of the system. We extend this idea to SS-MPL systems and translate them to MDPs (evolving over an event index) to be fitted into the reinforcement learning framework.

This paper is organised as follows. In Section II, we give the preliminaries on max-plus algebra, switching stochastic MPL systems, and the problem of designing control policies for such systems to optimise infinite-horizon discounted objectives. Section III formulates the MDP representation of the delay dynamics in the system and provides a solution approach using reinforcement learning. Section IV describes the details of a 9-dimensional case study for a railway network. Section V illustrates the performance of the proposed approach on the case study and compares it with the MPC approach. Finally, Section VI concludes the paper.

II. PRELIMINARIES AND PROBLEM STATEMENT

In this section, we introduce some basic definitions of the max-plus algebra and MPL systems.

A. Max-Plus Algebra

Define $\varepsilon := -\infty$ and $\mathbb{R}_{\mathsf{m}} := \mathbb{R} \cup \{\varepsilon\}$. Max-plus algebra has two basic operations, i.e., max-plus algebraic addition $\oplus$ and max-plus algebraic multiplication $\otimes$. The max-plus algebraic addition is denoted by $\oplus$ and is defined as $x \oplus y := \max(x, y)$ for any $x, y \in \mathbb{R}_{\mathsf{m}}$. Max-plus algebraic multiplication is denoted by $\otimes$ and is defined as $x \otimes y := x + y$ for any $x, y \in \mathbb{R}_{\mathsf{m}}$. These two operations are naturally extended to matrices. Denote the entries of any matrix X with x_{ij}. Then, we define

$$Z = X \oplus Y \text{ with } z_{ij} = x_{ij} \oplus y_{ij} = \max(x_{ij}, y_{ij}),$$

$$C = A \otimes B \text{ with } c_{ij} = \bigoplus_{k=1}^{n} a_{ik} \otimes b_{kj} = \max_{k=1,\ldots,n} (a_{ik} + b_{kj}),$$

for any matrices $X, Y \in \mathbb{R}_{\mathsf{m}}^{m \times n}$, $A \in \mathbb{R}_{\mathsf{m}}^{m \times n}$, and $B \in \mathbb{R}_{\mathsf{m}}^{n \times p}$

B. Stochastic MPL Systems

An MPL system is a DES in which there is synchronisation (a new operation starts as soon as all the preceding operations have been finished) but no concurrency or choice can be described by a state equation designed within the bounds of max-plus algebra. The state equation of an MPL system can be formulated as

$$x(k) = A(k) \otimes x(k-1) \oplus B(k) \otimes u(k), \tag{1}$$

with *event index* $k \in \mathbb{N} := \{1, 2, 3, \ldots\}$. The state $x(k)$ contains the time instants at which the internal events of the MPL system occur for the k^{th} time and $u(k)$ is the input which contains the time instants at which the input events occur for the k^{th} time. Matrices in (1) are $A \in \mathbb{R}_{\mathsf{m}}^{n \times n}$ and $B \in \mathbb{R}_{\mathsf{m}}^{n \times m}$, where n denotes the number of states and m denotes the number of inputs. Note that A, B could change with the event index k.

Stochastic MPL systems are a set of DES designed under the bounds of max-plus algebra similar to an MPL system. But unlike in an MPL system, the time intervals between successive event occurrences could be random quantities. For instance, in a model for a railway scheduling network, fixed scheduled delays such as railway track maintenance can be incorporated directly into the model but uncertain delays such as mechanical failures, weather conditions, boarding times of passengers, accidents, and the driver behaviour should be modelled using uncertain random variables.

A Stochastic MPL system is defined similar to the MPL system (1) but instead of introducing standard delays into the system via the inputs, the entry (i, j) of matrices $A(k)$ and $B(k)$ could depend on random delays $e_{ij}(k)$ that take

in each event index k a random value from a certain (known or unknown) probability distribution.

Example 1: Consider a two-dimensional stochastic MPL system in the form of (1) that models a railway network of two trains. The matrices of the system are

$$A(k) = \begin{bmatrix} 2+e_{11}(k) & \varepsilon \\ \varepsilon & 7+e_{22}(k) \end{bmatrix}, B(k) = \begin{bmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{bmatrix},$$

with initial state $x(0) = [\ \varepsilon \quad \varepsilon\ ]^T$. Let us consider a *timetable* $d(k)$ for the departure times of the two trains that requires the k^{th} departures of the trains to be at least $d(k)$ with

$$d(k) = \begin{bmatrix} 60k \\ 32+60k \end{bmatrix}, \quad k \in \mathbb{N}.$$

The above required departure time shows that the period of the timetable is 60 minutes. This timetable can be enforced by putting the input of the stochastic MPL system in (1) to be $u(k) = d(k)$. This results in the following state equations for the system in the usual algebra:

$$x_1(k) = \max(2 + e_{11}(k) + x_1(k-1), 60k),$$
$$x_2(k) = \max(7 + e_{22}(k) + x_2(k-1), 32 + 60k).$$

C. Switching Stochastic MPL Systems

If a stochastic MPL system is capable of switching between different modes of operation, it is known as *Switching Stochastic MPL* (SS-MPL) system. For an SS-MPL system with n_m modes of operation, the state equation at mode $l(k) \in \{1, 2, \ldots, n_m\}$ is given by

$$x(k) = A^{l(k)} \otimes x(k-1) \oplus B^{l(k)} \otimes u(k), \tag{2}$$

where $A^{l(k)}$ and $B^{l(k)}$ are the system matrices for mode $l(k)$. The switching between modes in general depends on previous state $x(k-1)$, previous mode $l(k-1)$, input $u(k)$ and possibly some additional control input $v(k)$. A switching variable $r(k) \in \mathbb{R}_{\mathsf{m}}^{n_r}$ that depends on these aforementioned variables can be written as

$$r(k) := \Phi(x(k-1), l(k-1), u(k), v(k)), \tag{3}$$

for some function $\Phi(\cdot)$. A switching mechanism is constructed using a partition of $\mathbb{R}_{\mathsf{m}}^{n_r}$ into n_m subsets each associated with a mode $l(k)$. For any $r(k)$ in each partition set, the associated mode is selected.

Example 2: In a railway network, there are two types of constraints that determine a timetable: *connection constraints* and *follow constraints*. If a connection constraint is broken, the two involved trains no longer need to maintain their connection. If a follow constraint is broken, the two involved trains no longer need to follow each other as specified in the schedule and hence can change their order. The mode of operation for a railway network depends on these constraints. Consider a railway network with n_{ct} constraints. The control input $v(k)$ in (3) can be selected from the input set $\{0,1\}^{n_{ct}}$. The i^{th} entry $v_i(k) = 0$ corresponds to keeping the i^{th} constraint and $v_i(k) = 1$ means the i^{th} constraint is broken. The number of modes in the system is $l(k) \in \{1, 2, 3, \ldots, 2^{n_{ct}}\}$. The function Φ in (3) will map each control input $v(k)$ to a mode $l(k)$. These several modes enable us to disrupt railway connections and alter the sequence of trains providing us with a new timetable which alters the system matrix A.

Example 3: Consider the railway network of Example 1. Let us put a constraint on train 2 such that train 2 follows train 1 arrived in its station. The control input v is such that we can either allow train 2 leave independently by setting $v(k) = [1]$ or satisfy the constraint by setting $v(k) = [0]$. Then we have $n_{ct} = 1$ and $l(k) \in \{1, 2\}$. This will result in state matrices

$$A^{l(k)} = \begin{bmatrix} 2+e_{11}(k) & \varepsilon \\ c(l(k)) + e_{21}(k) & 7+e_{22}(k) \end{bmatrix}, \tag{4}$$

$$B^{l(k)} = \begin{bmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{bmatrix},$$

where $c(1) = \varepsilon$ for breaking the connection and $c(2) \neq \varepsilon$ takes a non-negative value computed based on the required follow constraint. The initial state is $x(0) = [\ \varepsilon \quad \varepsilon\ ]^T$ and the input is $u(k) = d(k)$.

D. Problem Statement

We raise the following assumption before giving the problem statement.

Assumption 1: Distributions of delays and other uncertainties in the SS-MPL system are possibly unknown but sample event timings $x(0), x(1), x(2), \ldots$ under different switching strategies are available.

This is a valid assumption since most realistic railway networks have simulators that can imitate the behaviour of the real system. Uncertain delays are generally added to such simulators to assess the robustness of the system to such delays. These models are not suitable for model-based probabilistic analysis due to the scale of the system having hundreds of states [2].

We define the objective at the event index k as

$$J(k) := \lambda_1 \sum_{j=1}^{n} D(k,j) + \lambda_2 \sum_{m=1}^{n_{ct}} W(k,m), \tag{5}$$

where $D(k,j)$ is the delay of the j^{th} internal event in the event index k, $W(k,m)$ is the cost involved in breaking an inherent constraint of the system, n_{ct} denotes the number of constraints, and n denotes the number of internal events in the system (i.e., the dimension of $x(k)$). Note that the first term in (5) gives the total delay and the second term gives the cost of switching to the mode $l(k)$. The second term is added since altering the nominal operation of the SS-MPL system can break the timetable and can lead to unwanted complications in the real world scenario. The coefficients λ_1, λ_2 are weights to create a tradeoff between the delays and the costs. Additional terms to encode safety constraints can also be added to the objective function. We consider the total discounted objective function

$$J := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k J(k)\right], \tag{6}$$

where $\mathbb{E}[\cdot]$ denotes expectation with respect to random uncertainties and $\gamma \in (0,1)$ is a discounting factor that puts more emphasis on delays and costs happening in the near event indices. This objective function creates a tradeoff between minimising delays and costs incurred by breaking the constraints. A *switching policy* in the form of (3) will specify the next mode in each time step (i.e., which constraints should be broken) to minimise the objective function.

Problem 1: Given an SS-MPL system with unknown delays under Assumption 1 with state equation (2), and objective function (6), find a switching policy in the form of (3) that minimises the objective function.

III. SOLUTION APPROACH

A. Delay Dynamics in SS-MPL Systems

When all the random delays are assumed to be zero ($e_{ij} = 0$), the resulting SS-MPL system induces a periodic nature if the input is set to be the timetable, $u(k) = d(k)$ for all k. Then, each event is repeated at regular intervals characterised by an event period p. Next theorem gives the delay dynamics of the system.

Theorem 1: Define the difference between the states of the original SS-MPL system and those of the periodic departure times in $d(k)$ as $z(k) = x(k) - d(k)$. The delay dynamics can be written as

$$z(k) = [A^{l(k)} + D_z] \otimes z(k-1) \oplus [B^{l(k)} + D_u], \qquad (7)$$

with matrices $D_z = [d^z_{ij}]_{i,j}$ and $D_u = [d^u_{ij}]_{i,j}$ such that $d^z_{ij} = d_j(0) - d_i(0) - p$ and $d^u_{ij} = d_j(0) - d_i(0)$.

Proof: We use the fact that $u(k) = d(k)$ and the following two properties of the timetable $d(k)$:

$$d_i(k) - d_i(k-1) = p \text{ and}$$
$$d_i(k) - d_j(k) = d_i(0) - d_j(0), \quad \forall i,j,k.$$

Then we have for all i,

$$\begin{aligned}
z_i(k) &= x_i(k) - d_i(k) \\
&= \max(A^{l(k)}_{i1} + x_1(k-1) - d_i(k), \ldots, \\
&\qquad B^{l(k)}_{i1} + u_1(k) - d_i(k), \ldots) \\
&= \max(A^{l(k)}_{i1} + x_1(k-1) - d_i(k-1) - p, \ldots, \\
&\qquad B^{l(k)}_{i1} + u_1(k) - d_i(k), \ldots) \\
&= \max(A^{l(k)}_{i1} + x_1(k-1) - d_1(k-1) - d_i(0) + d_1(0) - p, \\
&\qquad \ldots, B^{l(k)}_{i1} + d_1(0) - d_i(0), \ldots) \\
&= \max(A^{l(k)}_{i1} + z_1(k-1) + d_i(0) - d_1(0) - p, \\
&\qquad \ldots, B^{l(k)}_{i1} + d_1(0) - d_i(0), \ldots),
\end{aligned}$$

which gives the expression in (7) for the delay dynamics. ■

B. SS-MPL System as a Markov Decision Process

An SS-MPL Σ can be equivalently represented as an MDP with a continuous state space denoted by the tuple $\Sigma = (Z, L, T_z)$, where Z is the state space of the system, L is the input space of the system and $T_z : \mathcal{B}(Z) \times Z \times L \to [0,1]$, is the conditional stochastic kernel that assigns to any $z \in Z$, and $l \in L$, a probability measure $T_z(\cdot \mid z, l)$ on the measurable space $(Z, \mathcal{B}(Z))$ so that for any set $\mathcal{A} \in \mathcal{B}(Z)$

$$\mathbb{P}(z(k+1) \in \mathcal{A} \mid z(k), l(k)) = \int_{\mathcal{A}} T_z\left(dz' \mid z(k), l(k)\right).$$

For given switching input $l(\cdot)$, the stochastic kernel T_z captures the evolution of the state of SS-MPL system Σ.

Theorem 2: The conditional stochastic kernel T_z of an SS-MPL system can be defined by the conditional density function $t_z(\overline{z} \mid z, l)$, where

$$t_z(\overline{z} \mid z, l) = \prod_{i=1}^{n} t_i\left(\overline{z}_i \mid z, l\right), \text{ with}$$

$$t_i(\overline{z}_i | z, l) = \boldsymbol{\delta}_{\alpha_i}(\overline{z}_i) \cdot \sum_{j=1}^{n} \left[t_{ij}(\overline{z}_i - d_{ij} - z_j, l) \prod_{\substack{k=1 \\ k \neq j}}^{n} T_{ik}(\overline{z}_i - d_{ik} - z_k, l) \right],$$

where $[\alpha_1, \ldots, \alpha_n]^T := \max(B^l + D_u)$ with max being taken in each row, and $\boldsymbol{\delta}_{\alpha_i}$ is the Dirac delta distribution located at α_i.

Proof: The independence property of $A^{l(k)}_{ij}(.) \otimes d_{ij}$, for all $i, j \in \{1, 2, \ldots, n\}$ leads to the multiplicative expression of $t_z(\overline{z}_i | z, l)$. In order to show the expression of the components $t_i(\overline{z}_i | z, l)$, first we compute the i^{th} conditional distribution function $T_i(\overline{z}_i | z, l)$, then we compute the i^{th} conditional density function $t_i(\overline{z}_i | z, l)$ by taking the derivative of $T_i(\overline{z}_i | z, l)$ w.r.t. $\overline{z}_i$.

$$\begin{aligned}
&T_i(\overline{z}_i | z, l) \\
&= \Pr\{\max\{A^l_{i1} + d_{i1} + z_1, \ldots, A^l_{in} + d_{in} + z_n, \alpha_i\} \leq \overline{z}_i | z\} \\
&= \Pr\{A^l_{i1} + d_{i1} + z_1 \leq \overline{z}_i, \ldots, A^l_{in} + d_{in} + z_n \leq \overline{z}_i, \alpha_i \leq \overline{z}_i | z\} \\
&= \mathbf{1}(\alpha_i \leq \overline{z}_i) \prod_{j=1}^{n} \Pr\{A^l_{ij} \leq \overline{z}_i - d_{ij} - z_j | z\} \\
&= \mathbf{1}(\alpha_i \leq \overline{z}_i) \prod_{j=1}^{n} T_{ij}(\overline{z}_i - d_{ij} - z_j | z).
\end{aligned}$$

Taking the derivative of T_i will give the intended result. ■

C. Reinforcement Learning

Q-learning. For an MDP $\Sigma = (Z, L, T_z)$, Q-learning finds a policy π for maximising the expected value of a total discounted reward function starting from the current state. In our case, the reward at each state $z(k)$ under input $l(k)$ will be $R(z(k), l(k)) = -J(k)$ with $J(k)$ defined in (5). At the heart of Q-learning there is a *Q-function* $Q^\pi(z, l)$ that gives the total discounted reward under a policy π starting from the state-input pair (z, l)

$$Q^\pi(z,l) = \mathbb{E}^\pi \left[\sum_{k=0}^{\infty} \gamma^k R(z(k), l(k)) | z(0) = z, l(0) = l \right], \qquad (8)$$

where $\mathbb{E}^\pi$ denotes the expectation over the solution of the system when the switching input $l(\cdot)$ is selected to be l in

the initial state $z(0) = z$ and follows the policy π. The goal is to find a policy that maximises $Q^{\pi}(z, l)$ in any state z.

When the state space Z is finite, Q-learning stores the values of $Q(z, l)$ in a table, called *Q-table*, and updates iteratively the values in the Q-table according to the following equation

$$Q(z,l) \leftarrow Q(z,l) + \alpha[R(z,l) + \gamma \max_{l'}(Q(z',l')) - Q(z,l)]. \tag{9}$$

In the right-hand side of (9), the term $Q(z, l)$ indicates the current values in the Q-table and $\max_{l'}(Q(z', l'))$ shows the maximum expected future rewards. During the learning, the agent will search through all the inputs l' for a particular state z' and chooses the state-input which the highest Q-value.

In (9), γ is the discount factor and α is a *learning rate*. Rearranging the above equation, we get

$$Q(z,l) \leftarrow (1-\alpha)Q(z,l) + \alpha \underbrace{[R(z,l) + \gamma \max_{l'}(Q(z',l'))]}_{\textbf{TD}},$$

where **TD** is the temporal difference target which gives an updated policy to choose the optimal l' in each state z', and the learning rate α creates a weighted sum of current Q-values and the **TD** to update the Q-table.

The major disadvantage of Q-learning is that it is only applicable to finite state spaces and the Q-table does not scale very well when there is a large set of state-input pairs. Therefore, Q-learning is not directly applicable to the model of SS-MPL systems that have continuous uncountable state spaces of the form $\mathbb{R}^n$. Deep Q-learning solves this issue by replacing the Q-table with neural networks [5].

Reward Function. Suppose $J(k, i)$ is the objective in (5) at epoch number i. To encourage the learning to minimise the objective across epochs, we define the reward function

$$R(k,i) = J(k,i-1) - J(k,i).$$

This gives a positive reward to the learning agent if $J(k, i) < J(k, i-1)$ and a negative reward if $J(k, i) > J(k, i-1)$. We take $\lambda_1 = 0.01$ and $\lambda_2 = 0.001$ in (5).

Deep Q-learning. The implementation of the Q-table is a key distinction between Deep Q-learning and the traditional Q-learning. Deep Q-learning substitutes the Q-table with a neural network known as the Deep Q-Network (DQN). Instead of mapping a state-action pair to a Q-value as it is done in Q-tables, A DQN maps states to (action, Q-value) pairs. For an n-dimensional state space and an action space consisting of m actions, the neural network is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$ (the DQN will have m outputs and the value inside each output node shows the Q-value). This is also shown in Fig. 1.

In order to find the parameters θ of the DQN, the network is trained with an appropriate loss function, which is the squared error of the target Q-value and predicted Q-value. Deep Q-learning utilises a prediction network $Q(z, l; \theta^{\text{pred}})$ as well as a target network $\hat{Q}(z, l; \theta^{\text{target}})$ to estimate the prediction and the target Q-values for the stability of the algorithm.

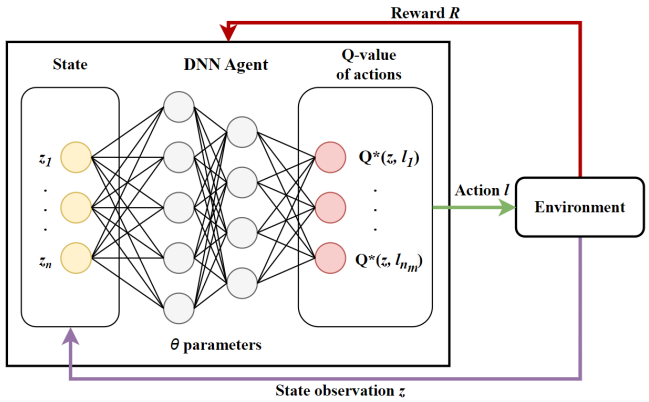

Fig. 1. Deep Q-learning framework where the Q-table is replaced by a deep neural network with appropriate number of inputs and outputs. The *Environment* block is the SS-MPL system that generates the state observation z and reward R under action l.

During learning, experience replay plays an important role: the learning agent builds up an experience dataset $\mathcal{D} = \{e_1, e_2, \ldots, e_t\}$ consisting of experiences $e_k = (z(k), l(k), z(k+1), R(k))$ from many iterations. When training the DQN, instead of merely using the present experience, as required by conventional **TD** learning, the Deep Q-network is trained by evenly sampling mini-batches of experiences from dataset $\mathcal{D}$. By allowing samples to be reused, experience replay increases sample efficiency. Additionally, experience replay in the context of neural networks permits mini-batch updates, which improves computing efficiency. Uniform sampling from the replay buffer reduces variance by minimising correlation between the samples used in the update. Using the above information, the loss function now can be formulated as

$$L_i(\theta_i) = \mathbb{E}_{(z,l,z',R)\sim\mathcal{U}(\mathcal{D})}\left[\left(y_i - Q(z,l;\theta_i^{\text{pred}})\right)^2\right], \tag{10}$$

where i is the epoch number, $\mathcal{U}(\mathcal{D})$ is the uniform distribution over the emulation dataset $\mathcal{D}$, and y_i is the update target values given by the target network $\hat{Q}$:

$$y_i = R(z,l) + \gamma \max_{l'} \hat{Q}(z',l';\theta_{i-1}^{\text{target}}).$$

When optimising the loss function $L_i(\theta_i)$ the parameters of θ^{target} from iteration $i-1$ are frozen for a fixed number of iterations while updating the parameters of the prediction network θ^{pred} to improve the stability of the network. We then get the following gradient by differentiating the loss function (10) with respect to the weights:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{(z,l,z',R)}\left[\left(y_i - Q(z,l;\theta_i^{\text{pred}})\right)\nabla_{\theta_i} Q(z,l;\theta_i^{\text{pred}})\right].$$

Remark 1: It is shown that Q-learning converges to the optimal policy and the optimal Q-values with probability 1 whenever all the actions are repeatedly sampled in all states under appropriate selection of the learning rate [23]. In contrast, Deep Q-learning will only converge to a sub-optimal solution due to a fixed structure for the DQN that may not be sufficient for representing the optimal Q-function on a continuous state space.

TABLE I

PERIODIC TIMETABLE OF THE RAILWAY NETWORK USED AS A CASE STUDY SECTION IV. ALL THE TIMES ARE MENTIONED IN MINUTES.

Train	From-To	Departure-Arrival	Constraints
1	D-A	00-12	same train as 3^-, connects to 9^-, and follows 7^-
2	A-B	15-27	same train as 1, connects to 6^-, and follows 4^-
3	B-D	30-50	same train as 2
4	A-B	19-31	same train as 6^-, follows 2, and connects to 7
5	B-C	34-44	same train as 4
6	C-A	47-12	same train as 5
7	D-A	04-16	same train as 9^-, follows 1
8	A-C	19-44	same train as 7
9	C-D	47-57	same train as 8, connects to 5

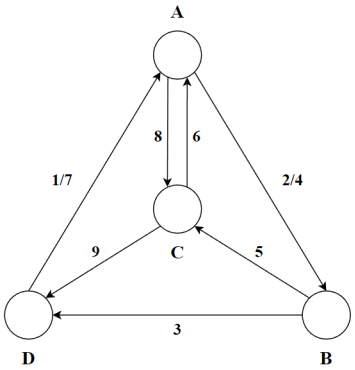

Fig. 2. The railway network used in Section IV.

IV. CASE STUDY

We consider a railway network and model its departure times with a 9-dimensional SS-MPL system. This case study is adapted from [6] by including stochastic delays in the model. The goal is to minimise delays using only data gathered from the railway network and find a policy for breaking the constraints in the network with the least disruption to the expected normal operation.

A. Railway Network

The railway network is shown in Fig. 2 and has four stations A, B, C, and D. There are three physical trains that follow respectively the routes (D,A,B,D), (A,B,C,A), and (D,A,C,D). The network has five single tracks (1/7, 2/4, 3, 5, and 9) and one double-track (6 and 8). The periodic timetable for scheduling the trains is presented in Table I. The nominal departure and arrival stations and times of the trains are included in the table with having period of 60 minutes. Although the network has three physical trains, we want to show the departure time of a specific train from a particular station. Therefore, we define 9 virtual trains to specify physical trains on specific tracks and show the information of these virtual trains in the timetable I. There are also three types of constraints that must be respected in the network (e.g., to dictate the arrival and departure times):

- *Continuity constraints* relate virtual trains to physical trains. For example, trains on tracks 1, 2, and 3 are physically the same train.
- *Connection constraints* allow the passengers to change trains. For example, train 1 has to wait for train 9 in the previous cycle with minimum connection time of 3 minutes.
- *Follow constraints* guarantee sufficient separation time between two trains on the same track. For example, train 4 follows train 2 with a minimum separation time of 4 minutes.

All the constraints are included in Table I, where the minus superscript indicates train from the previous cycle. The minimum stopping time of train j at station j is also fixed to 1 minute.

B. SS-MPL Model of the Railway Network

The state equations of the SS-MPL model of the train network when all the constraints are respected are as follows:

$$
\begin{aligned}
x_1(k) &= \max(x_3(k-1)+21+e_{13}(k), x_7(k-1)+4 \\
&\quad + e_{17}(k), x_9(k-1)+13+e_{19}(k), 60k) \\
x_2(k) &= \max(x_1(k)+13, x_4(k-1)+4+e_{24}(k), \\
&\quad x_6(k-1)+28+e_{26}(k), 15+60k) \\
x_3(k) &= \max(x_2(k)+13+e_{32}(k), 30+60k) \\
x_4(k) &= \max(x_2(k)+4+e_{42}(k), x_6(k-1)+26 \\
&\quad +e_{46}(k-1), x_7(k)+15+e_{47}(k), 19+60k) \\
x_5(k) &= \max(x_4(k)+13+e_{54}(k), 34+60k) \\
x_6(k) &= \max(x_5(k)+11+e_{65}(k), 47+60k) \\
x_7(k) &= \max(x_9(k-1)+11+e_{79}(k-1), x_1(k)+4 \\
&\quad + e_{71}(k), 4+60k) \\
x_8(k) &= \max(x_7(k)+13+e_{87}(k), 19+60k) \\
x_9(k) &= \max(x_8(k)+26+e_{98}(k), x_5(k)+13+e_{95}(k), \\
&\quad 47+60k).
\end{aligned}
$$

The timetable is $u(k) = d(k) = d(0) + 60k$ with $d(0) = [0, 15, 30, 19, 34, 47, 4, 19, 47]^T$. The state equations are modified appropriately for each constraint being broken. Note that our learning algorithm does not need to know the state equations nor the mathematical distributions of the uncertainties. It just requires knowing the state and actions spaces, initial state, and sampled trajectories under different control policies.

State Space. Let $x_j(k)$, $j \in \{1, 2, \ldots, 9\}$, be the time instant at which train j departs from its station for the k^{th} time. We also denote by $d_j(k)$ the nominal departure time for this train according to the timetable and use $a_j(k)$ for the travel time of this train. The state space of this 9-dimensional SS-MPL system is $\mathbb{R}_\varepsilon^9$.

Initial State. The first period starts at time $t = 0$. At the beginning of the first period the physical trains are respectively in stations D, A, and D. The initial state of the SS-MPL system is set to $x(0) = [\varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon]^T$, and the timetable is encoded in the input $u(k)$.

Action Space. The number of possible switching inputs of the SS-MPL system is determined by the number of constraints. The railway network has $n_{ct} = 8$ constraints

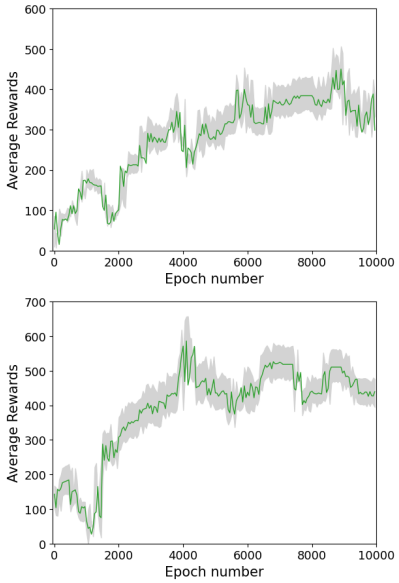

Fig. 3. Reward of the Deep Q-learning as a function of epoch number for the SS-MPL railway network with uncertainties having Exponential (top plot) and Gamma (bottom plot) distributions.

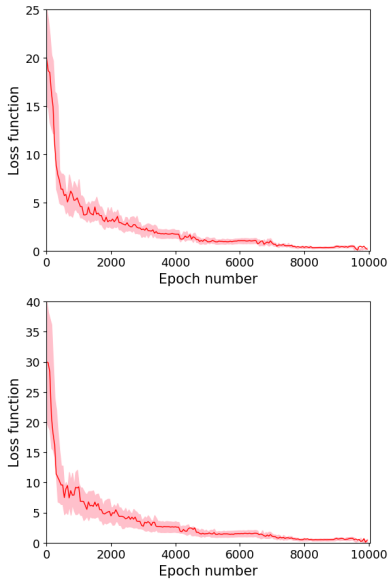

Fig. 4. Loss function of the Deep Q-learning as a function of epoch number for the SS-MPL railway network with uncertainties having Exponential (top plot) and Gamma (bottom plot) distributions.

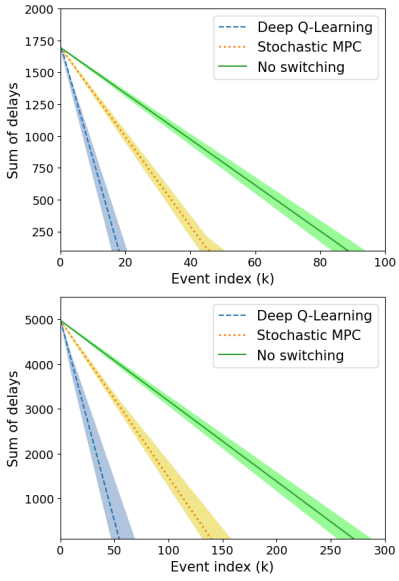

Fig. 5. Sum of delays for the SS-MPL railway network with uncertainties having Exponential (top plot) and Gamma (bottom plot) distributions under the policies computed with different approaches.

that can be broken (4 follow constraints and 4 connection constraints) giving an input space of size 2^8. The control input is a binary decision vector $v(k) \in \{0,1\}^{n_{ct}}$ that specifies whether any constraint is followed or broken.

Objective Function. We consider the objective in (5) for minimising delays while assigning the following cost to breaking the constraints $\sum_{m=1}^{n_{ct}} W(k,m) = Cv(k)$, where $v(k)$ is the binary vector indicating which constraints are broken, and C is a row vector containing the costs of breaking any particular constraint: $C = [w_{c_1}, w_{c_2}, w_{c_3}, w_{c_4}, w_{f_1}, w_{f_2}, w_{f_3}, w_{f_4}]$.

V. IMPLEMENTATION RESULTS

To model the stochastic delays in the railway system, we consider Exponential and Gamma distributions. These two types of distributions are widely used in real-time systems and can provide an accurate representation of the uncertainties [3]. We consider the random delays e_{ij} in the train travelling times following either Exponential distributions with parameter $\beta = 30$ or Gamma distributions with parameters $\theta = 40$ and $k = 3$. We select the DQN as a neural network with 6 layers and 10 nodes in each hidden layer. The input layer has 9 nodes corresponding to states of the system and the output layer has 8 nodes corresponding to the actions. Activation functions of the hidden layers are selected to be Rectified Linear Unit (ReLU). This neural network architecture has been effective in our experiments since the dynamics of the SS-MPL case study can also be considered as a neural network with ReLU activation functions.

We run our experiment for 10000 epochs each with the maximum event index k of 100 and 300 respectively for Exponential and Gamma distributions. We set the discount factor $\gamma = 0.99$ and learning rate $\alpha = 10^{-5}$. We use *Adam optimiser* which is an extension of stochastic gradient decent algorithm [24] to minimise the loss function $L(\theta)$ in (10). The learning also has another hyper-parameter ϵ that creates a trade-off between utilising newly acquired information and environment exploration. It is the probability of choosing to explore: $\epsilon = 1$ means exploring any policy randomly and $\epsilon = 0$ means selecting the input with the highest Q-value. In our experiment, ϵ is reduced linearly from 1 to 0.05.

Fig. 3 shows the average reward per epoch in the Deep Q-learning on the SS-MPL railway network with uncertainties having Exponential distribution (top plot) or Gamma distribution (bottom plot). The average reward curve is noisy due to the combination of exploration and exploitation employed by the learner to find the policy. Since the system is stochastic, we perform our experiment on the system 50 times and provide the average of quantities with a cloud around the average showing the min and the max. The plots in Fig. 4 show the loss function of the Deep Q-learning, which demonstrate the convergence of the learning to a (sub-optimal) policy for both types of distributions.

Performance Comparison. To assess the performance of the computed sub-optimal policy, we show the sum of delays as a function of event index k in Fig. 5 under three different policies: (a) the policy computed via our Deep Q-learning approach; (b) the policy computed via stochastic MPC adapted from [6] by using empirical mean in the optimisation objective; and (c) no-switching policy (the uncontrolled case when all the constraints are respected).

The comparison is also reported in Table V for both Exponential and Gamma distributions by running the algorithms 50 times and reporting the average and standard deviation of quantities. In the SS-MPL system with Exponential uncertainties, our Deep Q-learning approach reduces the delays in the network to zero and brings the network to its nominal timetable in average after 19 events (departures), while

TABLE II

COMPARING THE PERFORMANCE OF OUR DEEP Q-LEARNING APPROACH WITH STOCHASTIC MPC AND NO SWITCHING BY REPORTING AVERAGE AND STANDARD DEVIATION OF RUNNING THE ALGORITHMS 50 TIMES.

Distribution	**Approach**	**Sum of delays over event periods**		**First event index k_0 with zero delay**		$\max_j x_j(k_0)$	
		Average	**Standard Deviation**	**Average**	**Standard Deviation**	**Average**	**Standard Deviation**
Exponential ($\beta = 30$)	Deep Q-learning	17129.08	1096.43 (6.4%)	18.42	1.73 (9.39%)	1264.24	95.63 (7.56%)
	Stochastic MPC	44385.28	4211.71 (9.48%)	52.36	8.28 (15.81%)	3115.28	329.95 (10.59%)
	No switching	81593.84	6100.51 (7.47%)	95.06	10.24 (10.77%)	5912.18	580.16 (9.81%)
Gamma ($\theta = 40$, $k = 3$)	Deep Q-learning	138572.96	3712.08 (2.67%)	60.24	3.04 (5.05%)	6699.08	436.71 (6.51%)
	Stochastic MPC	353018.04	22891.23 (6.48%)	149.28	12.52 (8.38%)	17149.41	1620.43 (9.44%)
	No switching	687263.33	32486.32 (4.72%)	278.06	17.24 (6.20%)	32936.94	2669.67 (8.10%)

stochastic MPC and no-switching policies take respectively 53 and 96 events in average. The sum of all the delays and the time when delays become zero in the whole network are also reported in the table. When compared with no switching, our approach reduces the total delays in the network by 79% and the stochastic MPC approach reduces the total delays by 46%. Our approach is 2.84 times faster than stochastic MPC in clearing the delays. Similarly in the SS-MPL system with Gamma uncertainties, our approach brings the network to its nominal schedule within 61 events in average, while stochastic MPC and no-switching policies take respectively 150 and 279 events. When compared with no switching, our approach reduces the total delays in the network by 80%, and the stochastic MPC approach reduces the total delays by 49%. Our approach is 2.47 times faster than stochastic MPC in clearing the delays. Over 50 experiments on a laptop with Intel Core i5-8300H processor, 2.30GHz, and 4GB GTX GPU, the computation time for our RL policy is in average 5 min and maximum 13 min, while MPC policy takes 6 min in average and maximum 9 min.

VI. CONCLUSION

We studied the problem of learning policies for optimising objective functions on switching stochastic max-plus-linear systems. Our approach translates the system to a Markov decision process and utilises the model-free Deep Q-learning using sampled trajectories of the system. Our implementations on a 9-dimensional model of a railway network showed superior performance in comparison with the stochastic model predictive control approach for two types of uncertainties. Future work includes enforcing temporal logic properties on the system that requires keeping track of probabilistic dependencies in different event indices.

REFERENCES

[1] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.

[2] B. Heidergott, G. J. Olsder, J. Van Der Woude, and J. van der Woude, *Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications*. Princeton University Press, 2006, vol. 13.

[3] B. Heidergott, *Max-plus linear stochastic systems and perturbation analysis*. Springer, 2007.

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

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[6] T. J. van den Boom and B. De Schutter, "Modelling and control of discrete event systems using switching max-plus-linear systems," *Control engineering practice*, vol. 14, no. 10, pp. 1199–1211, 2006.

[7] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[8] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dynamic Systems*, vol. 6, no. 3, pp. 241–273, 1996.

[9] H. Zhang, L. Feng, and Z. Li, "A learning-based synthesis approach to the supremal nonblocking supervisor of discrete-event systems," *IEEE Trans. on Automatic Control*, vol. 63, no. 10, pp. 3345–3360, 2018.

[10] C. De la Higuera, *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

[11] J. Li and S. Takai, "Maximally permissive supervisors for nonblocking similarity control of nondeterministic discrete event systems," *IEEE Transactions on Automatic Control*, pp. 1–16, 2022.

[12] J.-L. Boimond and J.-L. Ferrier, "Internal model control and max-algebra: Controller design," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 457–461, 1996.

[13] Y. Shinada, S. Masuda, and H. Goto, "Adaptive model predictive control for the max-plus linear system," *IFAC Proceedings Volumes*, vol. 37, no. 12, pp. 481–486, 2004.

[14] X. David-Henriet, J. Raisch, L. Hardouin, and B. Cottenceau, "Modeling and control for max-plus systems with partial synchronization," *IFAC Proceedings Volumes*, vol. 47, no. 2, pp. 105–110, 2014.

[15] J. Xu, L. Buşoniu, T. van den Boom, and B. De Schutter, "Receding-horizon control for max-plus linear systems with discrete actions using optimistic planning," in *2016 13th International Workshop on Discrete Event Systems (WODES)*. IEEE, 2016, pp. 398–403.

[16] D. Adzkiya, S. Soudjani, and A. Abate, "Finite abstractions of stochastic max-plus-linear systems," in *Proceedings of the International Conference on Quantitative Evaluation of Systems*, ser. LNCS. Springer Verlag, 2014, vol. 8657, pp. 74–89.

[17] S. Soudjani, D. Adzkiya, and A. Abate, "Formal verification of stochastic max-plus-linear systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2861–2876, Oct 2016.

[18] D. Adzkiya, B. De Schutter, and A. Abate, "Computational techniques for reachability analysis of max-plus-linear systems," *Automatica*, vol. 53, pp. 293–302, 2015.

[19] M. S. Mufid, D. Adzkiya, and A. Abate, "Symbolic reachability analysis of high dimensional max-plus linear systems," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 459–465, 2020.

[20] A. Abate, A. Cimatti, A. Micheli, and M. S. Mufid, "Computation of the transient in max-plus linear systems via SMT-solving," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2020, pp. 161–177.

[21] K. M. Zielinski, L. V. Hendges, J. B. Florindo, Y. K. Lopes, R. Ribeiro, M. Teixeira, and D. Casanova, "Flexible control of discrete event systems using environment simulation and reinforcement learning," *Applied Soft Computing*, vol. 111, p. 107714, 2021.

[22] H. Ying, F. Lin, and R. Sherwin, "Fuzzy discrete event systems with gradient-based online learning," in *2019 IEEE international conference on fuzzy systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6.

[23] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2015.