Decision Algorithm Based on the Modified Bellman Equation to Deal with EMI-induced Errors in Hamming-based Communications

Miriam Gonzalez-Atienza*, Dries Vanoost*, Mathias Verbeke**, and Davy Pissoort*

* ESAT-WaveCoRe, M-Group, KU Leuven Bruges Campus, Belgium
**Declarative Languages and Artificial Intelligence (DTAI), M-Group, KU Leuven Bruges Campus, Belgium

***Abstract*—This paper presents a decision-making algorithm based on a modified version of the Bellman Equation to deal with Electromagnetic Interference errors in a communication channel inside a harsh electromagnetic reverberant environment.**

The Bellman equation is a fundamental concept in decision-making problems such as Markov Decision Processes. Such processes model decision-making in situations where the outcomes are partly random and partly controlled by an agent (decision-maker).

Recent studies have implemented Markov Decision Processes as a tool for risk assessment in areas such as robotics or aviation. However, so far, no research has been reported that uses the Bellman Equation or Markov Decision Processes to deal with risks related to electromagnetic disturbances.

In our study, a wired communication channel that uses Non-Return-to-Zero-Level data encoding and Hamming code for error detection and correction is disturbed. First, the packet error rate is calculated and compared with and without the proposed algorithm for different electromagnetic disturbance frequencies and bit rates. The gain is compared at different packet error rates when the algorithm's parameters (called rewards) are optimized and when these are set as random. Last, the influence of the rewards and the maximum number of resends on the algorithm's performance is also studied.

***Index Terms*—Decision-making, Communication, Electromagnetic Interference (EMI), reverberant environment, PER, Bellman Equation.**

I. Introduction

Nowadays, Electrical, Electronic, and Programmable Electronic (E/E/PE) devices are omnipresent in our daily lives. Moreover, they are increasingly performing safety-critical tasks in which a device failure might severely harm people or the environment. For an increasing number of these devices, reliable and robust communication is crucial. However, the communication medium will occasionally be disturbed by Electromagnetic Interference (EMI), resulting in the corruption of the transmitted data. Whether the disruptions occur by accident (unintentional EMI) or maliciously (intentional EMI), limiting the influence of these disturbances is essential.

Several Error Detection Codes (EDCs) and Error Correction Codes (ECCs) have been suggested to improve communication reliability [1] [2]. However, non of these codes are perfect. For example, there are still reported cases in which corrupted data is considered valid at the receiver (false negatives). In these cases, the system does not know the data was corrupted, meaning that EDCs and ECCs could not take any countermeasures. Therefore, these kinds of failures are the most dangerous ones in terms of safety.

One of the areas that is gaining attention is the use of Artificial intelligence (AI) techniques to predict/classify EMI [3], [4], or to address problems in electromagnetic compatibility (EMC) [5]. These techniques include using neural networks to model EMI and predict its impact on electronic systems [6], or using genetic algorithms to optimize electronic device design to improve EMC performance [7].

One of the recent focus areas in EMC is the implementation of reinforcement learning to develop control algorithms that can adapt to changing electromagnetic environments in real-time [8], [9]. Reinforcement learning algorithms can learn to make decisions based on feedback from the environment and can adjust the control parameters of electronic devices to improve their EMC performance.

Markov decision processes (MDPs) [10] are fundamental in reinforcement learning. They provide a mathematical framework for sequential decision-making problems for a fully observable, stochastic environment with a Markovian transition model and additive rewards. MDPs have been used in the context of safety assessment to find an optimal policy that minimizes a cost function that penalizes the risk of some specific conditions on the system [11], [12]. Several studies have applied MDP for decision-making in industrial applications [13]–[15], communications [16], [17], cyber security [18] or robotics [19], [20].

However, to the author's knowledge, no literature studies exist that employ Markov Decision Processes (MDPs) to deal with EMI-induced errors in a communication channel.

This paper aims to develop a decision-making algorithm based on the Bellman Equation (a key element in MDPs) that increases the certainty that the received data is correct and ensures the system's dependability when it is affected by EMI. Furthermore, this algorithm predicts the most optimal action using the available information from the electromagnetic environment, continuously leading the system to a minimal risk condition.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 955816. This publication reflects only the author's view exempting the European Union from any liability. Project website: https://eternity-project.eu/

The remainder of this paper is organized as follows. Section II presents the generic simulation setup and the system under study. Section III presents the error detection and correction technique (Hamming code) and the EM condition assessment. Section IV explains the theoretical foundations of the Bellman Equation and instantiates the modified model for the presented experimental setup. Section V provides experimental evaluations. Last, Section VI concludes the work.

II. Simulation Setup

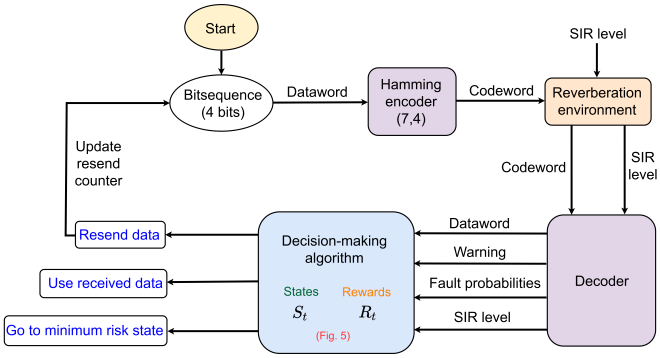


Fig. 1: Simulation Framework Outline

This paper relies on an in-house developed simulation framework that combines the paper of Hill [21], Magdowski et al. [22], and Vanhee et al. [23]. Different studies, as performed by [24], [25], use and extensively describe this framework. This section will shortly recapitulate this framework, which performs a Monte-Carlo simulation of single or multi-frequency continuous wave disturbances, which are incident to a device-under-test (DUT), in this case a Printed Circuit Board (PCB). The simulator extensively considers different randomized angles for the incident disturbance:

- Polar angle θ: $\arccos\left(U(-1,1)\right)$
- Azimuth angle φ: $U(0,\ 2\pi)$
- Polarization angle ψ: $U(0,\ \pi)$
- Phase shift angle α: $U(0,\ 2\pi)$

Where $U()$ denotes the uniform distribution. Randomization leads to the ability for a global assessment approach. The incident disturbance is superimposed on the sent voltages. We ran 16000 simulations for a specific Signal-to-Interference-Ratio (SIR), which represents the changing nature of a reverberation environment. The superimposed induced voltages result in the bit errors introduced within the PCB traces.

Fig. 1 depicts the communication between a sender and a receiver, simulated using a single PCB trace with matched source and load impedances. The sender encodes the data (4 bits) into codewords of 7 bits using a Hamming code $(7, 4)$. Those codewords are sent to the receiver using a particular voltage encoding. The received voltages are converted back into bits/symbols at the receiver side. The entire communication subsystem is subjected to an EMI disturbance, represented schematically in Fig. 2 by E_{inc}. Possible errors are detected and corrected using the Hamming code, which can detect up to two bit-flips and correct one bit-flip. A warning signal is raised if an error has been detected. A conceptual overview of the considered signal over time can be seen in Fig. 3. The decision algorithm then determines the most optimal action/decision using the following inputs:

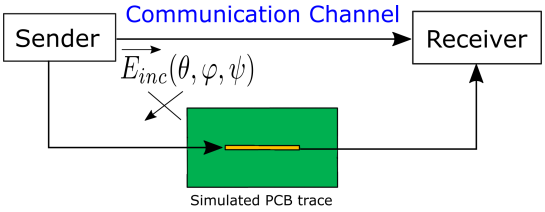


Fig. 2: Simulated Trace as Communication Channel

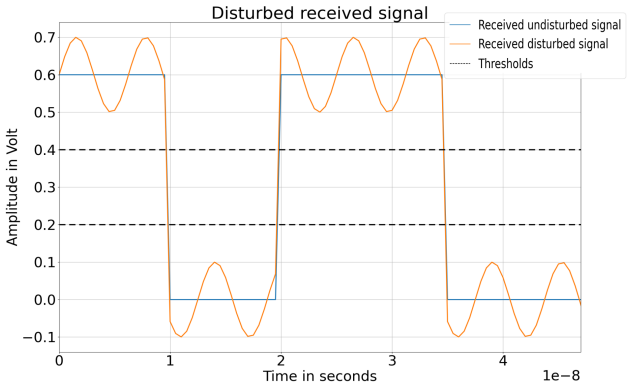


Fig. 3: The received disturbed signal in time domain. The blue line represents the undisturbed received signal. The orange line is the received signal plus the disturbance.

- Warning signal from the detector
- Fault probabilities of the received data

With this information, plus the comparison between the data received for the first time and the data received after a resend, the possible decisions are:

- Use the data received at a specific time
- Ask to resend the data
- Go to a minimum risk state

The full-wave simulations carried out in this article were performed using the Finite Difference Time Domain (FDTD) solver inside PathWave from Keysight Technologies [26]. All post-processing analysis is done using the in-house built simulation framework coded in Python in which the geometry, encoding and decoding scheme, bit pattern, bit frequency, EM environment, and error correction mechanism can be freely selected.

A. Geometry Under Study

The PCB includes an FR4 substrate of dimensions 10 cm by 16 cm and a thickness of 1.6 mm. The trace is 3 mm wide, corresponding to a characteristic impedance of 50 Ω, and has a length of 5 cm. A full ground plane covers the bottom of the PCB [24].

Since the output port consists of matched load, the maximum output voltage is half of the applied input signal.

B. Encoding Scheme

A random stream of $N_b = 112$ is first encoded into voltages (see Fig. 1) using a Non-Return-to-Zero-Level (NRZ-L) encoding scheme. A "0" bit is encoded as 0 V while a "1" is encoded as 1 V.

C. Calculation of Induced Voltages

This paper focuses on a reverberation environment. First, the geometry is exposed to reverberant disturbances while transmitting the encoded bit stream over the trace. Then, based on the superposition and the reciprocity theorem, the EMI-induced voltages are added to the encoded bit stream [23]. Similar to the real world, a reverberation environment consists of multiple reflections. These conditions are created using the plane wave integral representation for reverberation chambers as presented in [21], which states that a reverberation environment is represented by a superposition of randomly chosen plane waves (according to the correct statistical distributions). The final disturbances are a set of $N = 200$ plane waves, each one with random properties for the polar angle θ, azimuth angle φ, polarization angle ψ, and phase angle α. To represent the continuously changing nature of a reverberating environment, $M = 1000$ different sets are considered in this paper.

III. Hamming code

The Hamming code is a linear, single-error correction code capable of detecting up to two-bit errors and correcting single-bit errors [27]. This section considers the encoding and decoding scheme Hamming (7,4).

A. Hamming Encoding and Decoding

Hamming $(7, 4)$ encodes four bits into seven bits by adding three parity bits, ensuring that the minimum Hamming distance between any two correct codewords is three. If the distance between the received word and the transmitted codeword is at least one, the Hamming code decodes it to the most likely data word. Hamming code is formed in the following way, as explained in [28].

Due to the linearity of Hamming codes, they can be computed in linear algebra terms through matrices. In the correction and decoding process, these matrices are needed: the parity check matrix $H(r \times n)$, the code generator matrix $G(k \times n)$, as shown in (1) and (2), and the syndrome vector s.

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{1}$$

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The transmitted codeword r (7 bits) is determined by taking the product of G and the dataword (sequence of 4 bits) p: $r = G^T p$. Multiplying H with the received codeword yields the syndrome s: $s = Hr$.

When a correct codeword is received, s is all zero. If there is an incorrect codeword, the syndrome indicates the bit position of that error. However, it is also possible for an incorrect codeword to yield an all-zero s, for example, when a codeword is transformed into another valid codeword. In the case of two or more errors in the codeword, a non-zero syndrome is obtained, but only a one-bit error is corrected. These cases will be explained in Section III-B. The correction is done by performing a bit-inversion in the position determined by the syndrome. After the correction, the codeword is decoded by multiplying it with G.

TABLE I: Overview of considered categories

Warning	Final codeword same as transmitted?	Category
0	Yes	True Positive (TP)
0	No	Channel False Negative (CFN)
1	Yes	False Positive (FP)
1	No	Channel True Negative (CTN)
2	-	Data True Negative (DTN)

B. EM Condition Assessment

The condition assessment (i.e., the assessment of the detector on the measured channel condition) used so far for EM risk analysis and proposed by [29] is shown in Table I. These categories are determined based on the syndrome calculated for the received data word. Thus, if the syndrome equals 0, we consider the warning equal to 0, and the corrected codeword is the same as the received. On the other hand, if the syndrome is non-zero, we consider the warning equal to 1 and try to correct the data. For example, if Hamming detects exactly two bit-errors, it cannot correct it, and the warning is equal to 2.

- True Positive (TP): The received output is correct, and the detector (the syndrome determined by the Hamming code) is also correct. In this case, the syndrome is zero, which indicates that the codeword has not been affected by any disturbance. This is the best scenario.
- Channel False Negative (CFN): The received output is incorrect. The channel is getting saturated, and EMI is deciding the outcome. This is the worst case since EMI enforces a wrong output, but no warnings are raised because the syndrome equals to zero. These cases are the most detrimental to overall performance and safety.
- False Positive (FP): The syndrome is non-zero, which indicates that the codeword has been affected by EMI. After the correction, the corrected codeword is the same as the transmitted. Thus the corrected codeword is valid.
- Channel True Negative (CTN): The channel is in control, and EMI decides the outcome. The system receives a warning (the syndrome is non-zero), and a correction is attempted. However, the corrected codeword is not the same as the transmitted.
- Data True Negative (DTN): The syndrome is non-zero, and the Hamming code detects two-bit errors in the received codeword, meaning it cannot correct the data.

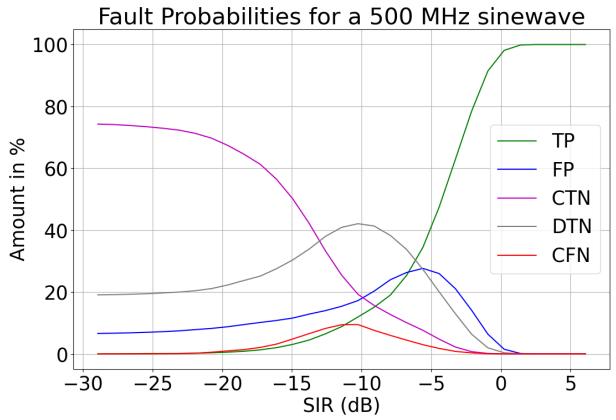


Fig. 4: Fault probabilities with $f_{\text{EMI}} = 500$ MHz and $f_{\text{BIT}} = 350$ MHz.

Therefore, the warning raised in this scenario differs from the one raised in the case of FP or CTN.

Figure 4 shows the response of the number of faults as a function of the SIR for a reverberation wave of $f_{\text{EMI}} = 500$ MHz and $f_{\text{BIT}} = 350$ MHz. The algorithm for computation of SIR is explained in [25]. To calculate these probabilities, a large number of codewords are sent through the simulation framework, and the Hamming code is applied to the received codewords. Depending on the final output, the codeword falls into one of the categories explained in Section III-B, obtaining the results above.

IV. The Bellman Equation

The following section describes the mathematical foundation of the Bellman Equation and how it is used in Markov Decision Processes.

Before explaining the Bellman Equation, we will introduce the key elements of Markov Decision Processes.

MDP is a discrete-time stochastic control process that models decision-making in situations where outcomes are partially random and partially under the control of a decision-maker. A MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where

- $\mathcal{S}$ is a finite set of states.
- $\mathcal{A}$ is a finite set of actions.
- $\mathcal{P}$ is a state transition probability matrix, with $\mathcal{P}^a_{ss'} = P[S_{t+1} = s' | S_t = s, A_t = a]$.
- $\mathcal{R}(s, a)$ is the reward for being in the state s and performing action a.
- γ is a discount factor $\gamma \in [0, 1]$.

It thus consists of a set of states and actions, a transition model, and a reward function. Like a Markov chain, the action outcomes only depend on the current state. At each step during this iterative process, the decision-maker (agent) may choose to take an available action in the current state, resulting in the model moving to the next step and offering the agent a reward.

A solution to an MDP is called a policy $\pi(s)$. It specifies an action a for each state s. In an MDP, we aim to find the optimal policy that yields the highest expected utility. Value iteration is an algorithm to find an optimal policy for an MDP. It calculates each state's utility $U(s)$, defined as the expected sum of discounted rewards from that state onward, using the Bellman equation [30].

$$U(s) = \max_a \left(R(s, a) + \gamma U(s') \right) \tag{3}$$

It expresses a recursive relationship between the values of the states. That is, the value of being in state s is equal to the expected immediate rewards from being in this state plus the value of being in the state we transition into. This relationship is helpful to approximate the state-action value function of the MDP.

Then, using the states' utilities, an optimal action is selected for each state.

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s') \tag{4}$$

A. Decision algorithm based on the Bellman Equation

In the proposed decision algorithm, we implemented a modified version of the Bellman Equation to calculate the utility value of each state. Hence, we do not perform value iteration as described in MDP. This is because our goal is not to learn a policy but to decide the best action in each state based on the calculated utility value.

The decision algorithm is visually represented in Fig. 5.

As described in Section III, different fault probabilities define each state in the process, depending on the warnings received and the comparison between the first received data and the data received after the resend. The following parameters characterize the decision algorithm:

- Central states. $\mathcal{S}$ = $\{s_1, s_2, s_3\}$.
  - **State 1:** Warning received at $t_{n-1} = 0$
  - **State 2:** Warning received at $t_{n-1} = 1$
  - **State 3:** Warning received at $t_{n-1} = 2$

  Where n indicates the number of resends that have been done. $n = 2, 3, ..., n_{\text{max}}$.
- Sub-states. $\mathcal{S}b$ = $\{s_{b11}, s_{b12}, s_{b13}, s_{b14}, s_{b21}, s_{b22}, s_{b23}, s_{b24}\}$.
  There are three central states in the model ($\mathcal{S}$). Those states are defined by the warning from the first received data. As seen from Fig.5, if the first received warning equals 0, the model is in state s_1. If the warning equals 1, it is in state s_2. And if the warning equals 2, it is in state s_3, which is not represented in Fig. 5 because the decision is always "resend data". Each of these central states is composed of two sub-states ($\mathcal{S}b$), from which the different transitions take place depending on the warning received after the resend and the comparison between the data received at different times (see Table II).
- $\mathcal{A}$ = {Resend data, Use data received at t_n, Go to minimum risk state}.
- Number of possible codewords $c = 16$.
- Discount factor $\gamma = 0.8$.
- Penalty resend $p_r = 0.4$.
- Initial reward function for each of the categories mentioned above $R = \{1, -1, 0.5, -0.25, -0.35\}$.

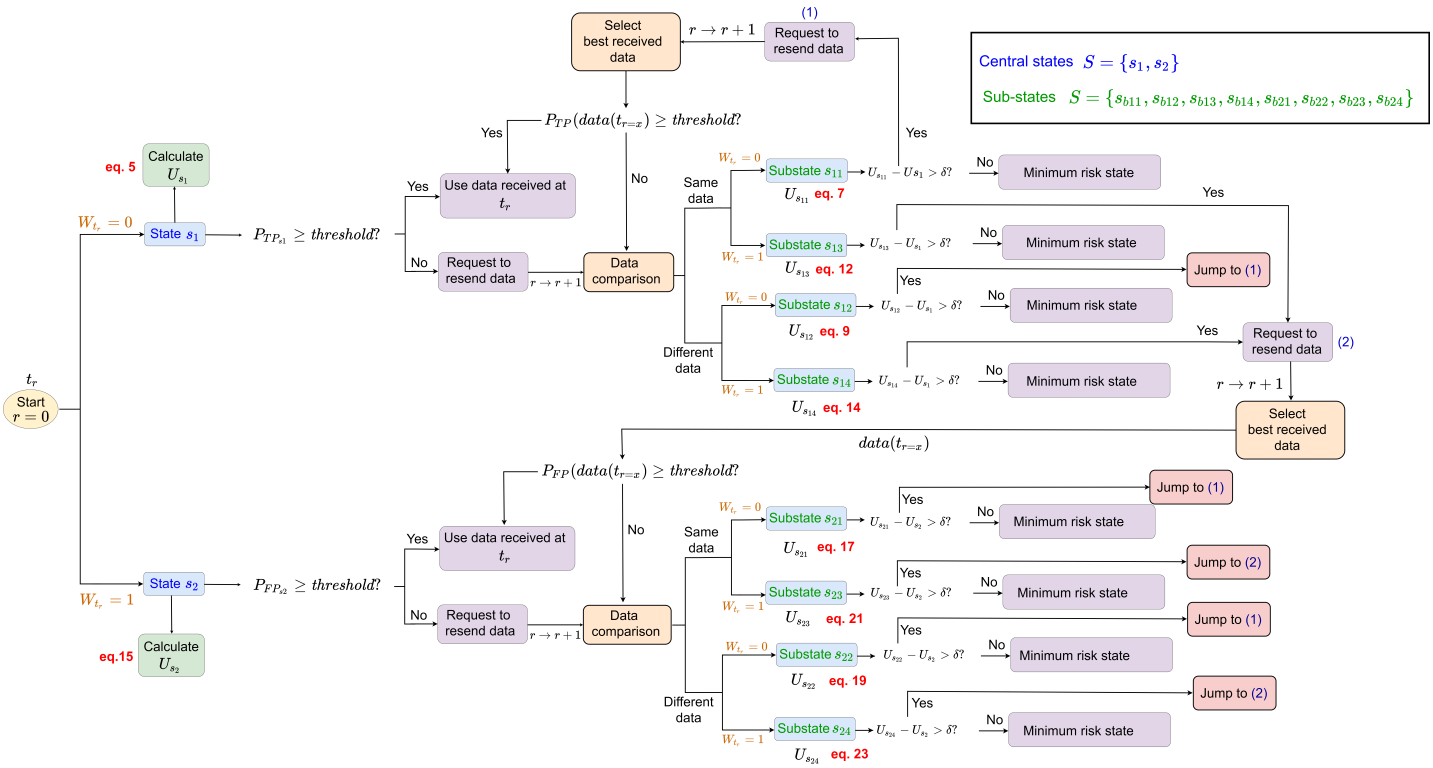


Fig. 5: Algorithm overview

A high value for the discount factor (close to 1) captures the effective long-term reward, while taking a discount factor closer to 0 makes our agent consider only immediate reward. Therefore, we have chosen a discount factor higher than 0.5 and closer to 1 to capture the effect of the utility in the following states instead of only considering the immediate reward (see Eq. 3).

Penalty resend is the value subtracted from the original reward function to reduce future rewards when the algorithm resends the data. In those cases, resending the data increases the system's latency, which may lead to an unsafe scenario.

We assign a different reward in the range $[-1, 1]$ depending on the probability of the categories received at each state. If the category received is a TP, $R = 1$. If it is a CFN, $R = -1$, $R = 0.5$ for a FP, $R = -0.25$ for a CTN, and $R = -0.35$ if it is a DTN. However, in Section V-D, we show how to learn/optimize these rewards.

Table II illustrates the different transitions depending on the warnings received and the data comparison.

At t_0, we check if the probability of the first received data being correct (TP or FP) is higher than a predefined threshold (for this paper, set as 96%). If so, we decide to directly use the data received at t_0 (see Fig. 5). If not, we request to resend the data and move to the next time step (t_1). This threshold is related to the certainty that the received data is correct. Thus, increasing this threshold means that the algorithm must request more resends to increase that certainty. We chose a value between 95 - 99% to achieve a trade-off between latency and availability.

Then, depending on the warning received after the resend and the comparison between data, we transit to a different state, for which we calculate its utility, following these equations:

- **Sub-state 1** (s_{b1}) : Warning received at t_{n-1} equals 0 ($W_{t_{n-1}} = 0$). In this state, we can either receive a TP or a CFN.

The utility at s_1 is:

$$U_{s_1} = \frac{P(\text{TP}_{s_1})R_{\text{TP}} + P(\text{CFN}_{s_1})R_{\text{CFN}}}{P(\text{TP}_{s_1}) + P(\text{CFN}_{s_1})} \tag{5}$$

Where $P(\text{TP}_{s_1})$ and $P(\text{CFN}_{s_1})$ are the probabilities of receiving a TP or a CFN at state s_1, respectively. R_{TP} and R_{CFN} are the rewards for receiving a TP or a CFN at state s_1.

- **Sub-state 11** (s_{b11}) : $W_{t_n} = 0$ and $data(t_n) = data(t_{n-1})$. The data received after resending (at t_n) is the same as the previously received data (at t_{n-1}). The transition probabilities in state s_{11} are the following:

$$\begin{cases} P_1 = P_{\text{TP}_{s_1}\text{TP}_{s_{11}}} = P(\text{TP}_{s_{11}} \cap \text{TP}_{s_1}) = P(\text{TP}_{s_{11}})P(\text{TP}_{s_1}) \\ P_2 = P_{\text{CFN}_{s_1}\text{CFN}_{s_{11}}} = P(\text{CFN}_{s_{11}} \cap \text{CFN}_{s_1}) \\ \qquad = P(\text{CFN}_{s_{11}})P(\text{CFN}_{s_1})\dfrac{1}{c-1} \end{cases} \tag{6}$$

TABLE II: Possible transitions in each state of the decision algorithm

Warnings	Transitions	
	Same data	Different data
$W_{t_{n-1}}=0$	TP-TP	TP-CFN
$W_{t_n}=0$	$(P_{TP_{s1}TP_{sb11}})$ CFN-CFN	$(P_{TP_{s1}CFN_{sb12}})$ CFN-TP
	$(P_{CFN_{s1}CFN_{sb11}})$	$(P_{CFN_{s1}TP_{sb12}})$ CFN-CFN
		$(P_{CFN_{s1}CFN_{sb12}})$
Sub-state:	s_{b11}	s_{b12}
$W_{t_{n-1}}=0$	TP-FP	TP-CTN
$W_{t_n}=1$	$(P_{TP_{s1}FP_{sb13}})$ CFN-CTN	$(P_{TP_{s1}CTN_{sb14}})$ CFN-FP
	$(P_{CFN_{s1}CTN_{sb13}})$	$(P_{CFN_{s1}FP_{sb14}})$ CFN-CTN
		$(P_{CFN_{s1}CTN_{sb14}})$
Sub-state:	s_{b13}	s_{b14}
$W_{t_{n-1}}=1$	FP-TP	FP-CFN
$W_{t_n}=0$	$(P_{FP_{s2}TP_{sb21}})$ CTN-CFN	$(P_{FP_{s2}CFN_{sb22}})$ CTN-TP
	$(P_{CTN_{s2}CFN_{sb21}})$	$(P_{CTN_{s2}TP_{s2}})$
Sub-state:	s_{b21}	s_{b22}
$W_{t_{n-1}}=1$	FP-FP	FP-CTN
$W_{t_n}=1$	$(P_{FP_{s2}FP_{sb23}})$ CTN-CTN	$(P_{FP_{s2}CTN_{sb24}})$ CTN-FP
	$(P_{CTN_{s2}CTN_{sb23}})$	$(P_{CTN_{s2}FP_{sb24}})$
		CTN-CTN
		$(P_{CTN_{s2}CTN_{sb24}})$
Sub-state:	s_{b23}	s_{b24}

Where $P(\text{TP}_{s_{11}} \cap \text{TP}_{s_1})$ and $P(\text{CFN}_{s_{11}} \cap \text{CFN}_{s_1})$ are the joint probabilities of receiving twice a TP or twice a CFN at states s_1 and s_{11}. Then, we calculate the utility in state s_{11}

$$U_{s_{b11}} = U_{s_1} + \gamma \frac{P_1(R_{\text{TP}} - p_r) + P_2(R_{\text{CFN}} - p_r)}{P_1 + P_2} \tag{7}$$

Once the utility of states s_1 and s_{b11} is calculated, we use the threshold to determine the decision at t_n :

$$\begin{cases} \text{if} \quad U_{s_{b11}} - U_{s_1} > 0.01, & \text{Resend data} \\ \text{else}, & \text{Go to minimum risk state} \end{cases}$$

If the algorithm decides to resend at t_n, the decision at t_{n+1} is based on the transition probabilities P_{TPTP} and P_{CFNCFN}. That is, if the algorithm decides not to resend and the probability of receiving a TP at t_{n-1} is greater than the predefined threshold, then the decision at t_{n+1} is to use the data received at t_{n-1}. If not, the decision is "Go to a minimum risk state". On the other hand, if the algorithm decides to resend and the probability of receiving a TP at t_{n-1} and another TP at t_n is greater than the predefined threshold, then the decision at t_{n+1} is to use the data received after the resend (at t_n). If not, the decision is to resend again and we use the algorithm to determine the decision at t_{n+2}.

- **Sub-state 12** (s_{b12}) : $W_{t_n} = 0$ and data$(t_n) \neq$ data(t_{n-1}). That is, the data received after resending (at t_n) is not the same as the previously received data (at t_{n-1}). The transition probabilities in state s_{12} are the following:

$$\begin{cases} P_1 = P_{\text{TP}_{s1}\text{CFN}_{sb12}} = P(\text{TP}_{s_{b12}} \cap \text{CFN}_{s_1}) \\ P_2 = P_{\text{CFN}_{s1}\text{TP}_{sb12}} = P(\text{CFN}_{s_{b12}} \cap \text{TP}_{s_1}) \\ P_3 = P_{\text{CFN}_{s1}\text{CFN}_{sb12}} = P(\text{CFN}_{s_{b12}} \cap \text{CFN}_{s_1}) \end{cases} \tag{8}$$

Then, we calculate the utility in state s_{b11}

$$U_{s_{b12}} = U_{s_1} + \gamma \frac{P_1(R_{\text{CFN}} - p_r) + P_2(R_{\text{TP}} - p_r) + P_3(R_{\text{CFN}} - p_r)}{P_1 + P_2 + P_3} \tag{9}$$

$$\begin{cases} \text{if} \quad U_{s_{b12}} - U_{s_1} > 0.01, & \text{Resend data} \\ \text{else}, & \text{Go to minimum risk state} \end{cases} \tag{10}$$

As in the previous case, if the algorithm decides not to resend and the probability of receiving a TP at t_{n-1} is greater than the predefined threshold, the decision at t_{n+1} is to use the data received at t_{n-1}. If not, the decision is "Go to minimum risk state". If the algorithm decides to resend and the probability of receiving a CFN at t_{n-1} and a TP at t_n is greater than the predefined threshold. In that case, the decision at t_{n+1} is to use the data received after the resend (at t_n). If not, the decision is to resend it again.

- **Sub-state 13** (s_{b13}) : $W_{t_n} = 1$ and data$(t_n) =$ data(t_{n-1}). In this state, we can receive a FP or a CTN. The transition probabilities in state s_{b13} are the following:

$$\begin{cases} P_1 = P_{\text{TP}_{s1}\text{FP}_{sb13}} = P(\text{TP}_{s_1} \cap \text{FP}_{s_{b13}}) \\ P_2 = P_{\text{CFN}_{s1}\text{CTN}_{sb13}} = P(\text{CFN}_{s_1} \cap \text{CTN}_{s_{b13}}) \end{cases} \tag{11}$$

We calculate the utility in state s_{b13}

$$U_{s_{b13}} = U_{s_1} + \gamma \frac{P_1(R_{\text{FP}} - p_r) + P_2(R_{\text{CTN}} - p_r)}{P_1 + P_2} \tag{12}$$

In this case, if the algorithm decides not to resend and the probability of receiving a FP at t_{n-1} is greater than the predefined threshold, the decision at t_{n+1} is to use the data received at t_{n-1} because in that case, the data received is assumed to be correct. If the decision is to resend and the probability of receiving a TP at t_{n-1} and a FP at t_n is greater than the predefined threshold, the decision at t_{n+1} is to use the data received after the resend (at t_n). If not, the decision is to resend it again.

- **Sub-state 14** (s_{b14}) : $W_{t_n} = 1$ and data$(t_n) \neq$ data(t_{n-1}).

$$\begin{cases} P_1 = P_{\text{TP}_{s1}\text{CTN}_{sb14}} = P(\text{TP}_{s_1} \cap \text{CTN}_{s_{b14}}) \\ P_2 = P_{\text{CFN}_{s1}\text{FP}_{sb14}} = P(\text{CFN}_{s_1} \cap \text{FP}_{s_{b14}}) \\ P_3 = P_{\text{CFN}_{s1}\text{CTN}_{sb14}} = P(\text{CFN}_{s_1} \cap \text{CTN}_{s_{b14}}) \end{cases} \tag{13}$$

We calculate the utility in state s_{b14}

$$U_{s_{b14}} = U_{s_1} + \gamma \frac{P_1(R_{\text{CTN}} - p_r) + P_2(R_{\text{FP}} - p_r) + P_3(R_{\text{CTN}} - p_r)}{P_1 + P_2 + P_3} \tag{14}$$

In any case, the warning received equals 2, meaning the received data cannot be trusted, and should be sent again.

- **State 2** (s_2) : Warning received at t_{n-1} equals 1 ($W_{t_{n-1}} = 1$). In this state, we can either receive a FP or a CTN. The utility at s_2 is:

$$U_{s_2} = \frac{P(\text{FP}_{s_2})R_{\text{FP}} + P(\text{CTN}_{s_2})R_{\text{CTN}}}{P(\text{FP}_{s_2}) + P(\text{CTN}_{s_2})} \tag{15}$$

- **Sub-state 21** (s_{b21}) : $W_{t_n} = 0$ and $data(t_n) =$ data(t_{n-1}).

$$\begin{cases} P_1 = P_{\text{FP}_{s2}\text{TP}_{sb21}} = P(\text{FP}_{s_2} \cap \text{TP}_{s_{b21}}) \\ P_2 = P_{\text{CTN}_{s2}\text{CFN}_{sb21}} = P(\text{CTN}_{s_2} \cap \text{CFN}_{s_{b21}}) \end{cases} \tag{16}$$

$$U_{s_{b21}} = U_{s_2} + \gamma \frac{P_1(R_{\text{TP}} - p_r) + P_2(R_{\text{CFN}} - p_r)}{P_1 + P_2} \tag{17}$$

- **Sub-state 22** (s_{b22}) : $W_{t_n} = 0$ and data$(t_n) \neq$ data(t_{n-1}).

$$\begin{cases} P_1 = P_{\text{FP}_{s2}\text{CFN}_{sb22}} = P(\text{FP}_{s_2} \cap \text{CFN}_{s_{b22}}) \\ P_2 = P_{\text{CTN}_{s2}\text{TP}_{sb22}} = P(\text{CTN}_{s_2} \cap \text{TP}_{s_{b22}}) \end{cases} \tag{18}$$

$$U_{s_{b22}} = U_{s_2} + \gamma \frac{P_1(R_{\text{CFN}} - p_r) + P_2(R_{\text{TP}} - p_r)}{P_1 + P_2} \tag{19}$$

- **Sub-state 23** (s_{b23}) : $W_{t_n} = 1$ and data$(t_n) =$ data(t_{n-1}).

$$\begin{cases} P_1 = P_{\text{FP}_{s2}\text{FP}_{sb23}} = P(\text{FP}_{s_2} \cap \text{FP}_{s_{b23}}) \\ P_2 = P_{\text{CTN}_{s2}\text{CTN}_{sb23}} = P(\text{CTN}_{s_2} \cap \text{CTN}_{s_{b23}}) \end{cases} \tag{20}$$

$$U_{s_{b23}} = U_{s_2} + \gamma \frac{P_1(R_{\text{FP}} - p_r) + P_2(R_{\text{CTN}} - p_r)}{P_1 + P_2} \tag{21}$$

- **Sub-state 24** (s_{b24}) : $W_{t_n} = 1$ and data$(t_n) \neq$ data(t_{n-1}).

$$\begin{cases} P_1 = P_{\text{FP}_{s2}\text{CTN}_{sb24}} = P(\text{FP}_{s_2} \cap \text{CTN}_{s_{b25}}) \\ P_2 = P_{\text{CTN}_{s2}\text{FP}_{sb24}} = P(\text{CTN}_{s_2} \cap \text{FP}_{s_{b24}}) \\ P_3 = P_{\text{CTN}_{s2}\text{CTN}_{sb24}} = P(\text{CTN}_{s_2} \cap \text{CTN}_{s_{b24}}) \end{cases} \tag{22}$$

$$U_{s_{b24}} = U_{s_2} + \gamma \frac{P_1(R_{\text{CTN}} - p_r) + P_2(R_{\text{FP}} - p_r) + P_3(R_{\text{CTN}} - p_r)}{P_1 + P_2 + P_3} \tag{23}$$

V. Simulation Results

A. Systematic Fault Injection Model

Within this fault injection model, all possible bit flips within the 7-bit Hamming code are injected systematically: first, all single-bit bit-flips, then all double-bit bit-flips, all triple-bit bit-flips until the last injection will flip all seven bits within the codeword. Then, the category of each codeword is determined as explained in Section III-B. Injecting a bit flip is done by performing the XOR value of the codeword with a logical "1". The XOR function transforms a "0" into a "1" and vice versa. The results from the systematic fault injection are shown in Fig. 6.

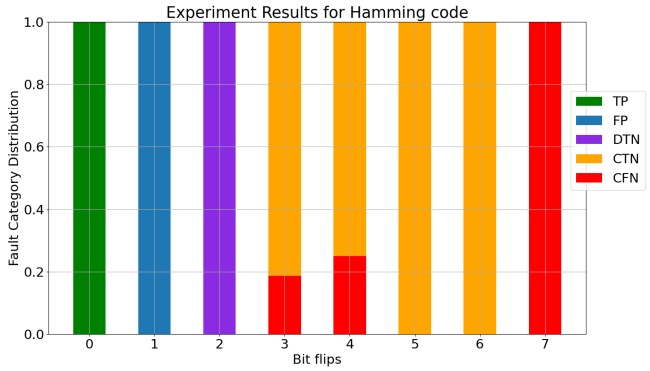


Fig. 6: Results of the Systematic Fault Injection Model on Hamming code (7,4).

As the Hamming code is linear, if a codeword undergoes complete corruption (i.e., 7 bit-flips), it will result in another valid codeword. This scenario can occur in the case of a Channel False Negative (CFN), and the error-checking algorithm will fail to identify the error. The results indicate that all double-bit bit-flips are detectable and categorized as Data True Negatives (DTN). The 5-bit and 6-bit bit-flips are also detectable, as well as the 3-bit and 4-bit bit-flips. Those are the cases in which a correction was attempted, but the final codeword is incorrect. A correction attempt yields the incorrect result for all the detectable multi-bit errors. In the case of a single bit-blip, the correction leads to the correct codeword (False Positives).

B. Packet Error Rate

Previous studies reported a correlation between the ratio of f_{EMI} and bit rate and the susceptibility to EMI [31], [32]. Therefore, seven frequencies were selected in our study and the bit rate was swept from 200 MHz to 450 MHz, while also sweeping the EMI frequency f_{EMI} from 100 MHz to 500 MHz in steps of 66.67 MHz.

The results for these combinations are shown in Table III. For each simulation, a total of 16.000 packets were transmitted.

Firstly, the Packet Error Rate (PER) is calculated for the first received data (without the decision algorithm) and the data selected by the algorithm. The PER is calculated as the number of incorrectly received packets divided by the total

number of transmitted packets (16000). Figure 7 shows the PER obtained with and without the decision algorithm, and the PER when the algorithm rewards are random or learned for different combinations of EMI frequency and bit rates. The process of learning these rewards is explained in Section V-D.

Table III summarizes the gain obtained with the decision algorithm when the rewards are random or learned. For the calculations, the reference PER is the case without the algorithm. Some of these results are shown in Fig. 7.

TABLE III: Simulation results summary

f_{EMI} (MHz)	Bit rate (MHz)	Gain (dB) Random rewards | Learned rewards			
		200	**250**	**300**	**450**
100	**Ratio**	**0.50**	**0.40**	**0.33**	**0.22**
	PER= 5%	1.05|2.08	0.82|1.20	1.76|2.02	0.80|1.22
	PER= 10%	1.49|2.29	1.06|1.38	1.80|1.97	1.08|1.44
	PER= 20%	2.19|2.60	1.22|1.64	1.66|1.98	1.36|1.67
166.67	**Ratio**	**0.83**	**0.67**	**0.56**	**0.37**
	PER= 5%	1.22|1.53	1.27|1.61	0.70|1.22	0.81|1.21
	PER= 10%	1.01|1.39	0.83|1.39	0.59|1.25	1.06|1.27
	PER= 20%	1.66|1.74	0.69|1.13	0.88|1.43	1.23|1.43
233.33	**Ratio**	**1.17**	**0.93**	**0.78**	**0.52**
	PER= 5%	0.90|1.08	0.94|1.11	0.67|0.83	1.31|1.44
	PER= 10%	0.85|1.39	0.98|1.38	0.92|1.37	1.13|1.37
	PER= 20%	1.20|1.60	1.14|1.43	0.82|1.68	1.49|1.63
300	**Ratio**	**1.50**	**1.20**	**1.00**	**0.67**
	PER= 5%	0.74|1.42	0.90|0.96	1.46|1.54	0.68|0.86
	PER= 10%	1.01|1.63	0.93|0.99	1.61|1.63	0.92|1.25
	PER= 20%	1.26|1.55	1.15|1.33	1.46|1.49	0.68|1.28
366.67	**Ratio**	**1.83**	**1.47**	**1.22**	**0.81**
	PER= 5%	1.03|1.10	0.48|0.77	0.76|0.86	1.06|1.30
	PER= 10%	0.75|1.01	0.29|0.70	1.13|1.24	0.80|1.07
	PER= 20%	1.12|1.27	0.86|0.98	1.23|1.24	1.12|1.36
433.33	**Ratio**	**2.17**	**1.73**	**1.44**	**0.96**
	PER= 5%	0.56|0.58	0.63|0.83	0.34|0.97	0.47|0.87
	PER= 10%	1.04|1.11	0.91|1.21	0.63|1.22	0.30|0.76
	PER= 20%	0.95|0.97	0.76|1.04	0.34|0.85	0.49|1.11
500	**Ratio**	**2.50**	**2.00**	**1.67**	**1.11**
	PER= 5%	0.56|1.01	0.90|1.33	0.88|1.26	1.09|1.12
	PER= 10%	0.83|1.09	0.63|1.21	0.73|0.92	1.17|1.20
	PER= 20%	0.64|0.80	0.90|1.47	1.10|1.32	0.92|0.94

Where:

- $\Delta_{\text{learned}} = \text{PER}_{\text{baseline}} - \text{PER}_{\text{with learned rewards}}$
- $\Delta_{\text{random}} = \text{PER}_{\text{baseline}} - \text{PER}_{\text{with random rewards}}$

and "Baseline" is the PER obtained without the proposed decision algorithm. That is, we calculated the PER between the first received data and the sent data. These results are obtained with a maximum of 8 resends (at each SIR level, the algorithm chooses between 0 and 8 resends).

As stated above, we swept the EMI frequency from 100 MHz to 500 MHz in steps of 66.67 MHz while changing the bit rate from 200 MHz to 450 MHz. The results for these combinations are shown in Table III while Fig. 7 shows some of these results. Table III shows that the gain with learned rewards is higher than in the case where the rewards are set as random, and this is for all the chosen frequency combinations. The maximum gain (2.60 dB) is obtained with $f_{\text{EMI}} = 100$ MHz and $f_{\text{BIT}} = 200$ MHz, at a PER =20% and with learned rewards. It can also be seen from Fig. 7b that the maximum gain (16.94 %) is obtained at SIR = -13.58 dB, with the MDP with learned rewards and $f_{\text{EMI}} = 233.33$ MHz and $f_{\text{BIT}} = 300$ MHz. The gain with random rewards at that same SIR level is 8.90%.

Additionally, Fig. 7 shows the difference in PER and gain depending on the ratio between the EMI frequency and the bit rate. When the EMI frequency is an integer multiple of the bit frequency, for low SIR disturbances, the codewords can be easily transformed into all-one or all-zero data words, which introduces multiple bit errors that the Hamming code cannot detect. Thus, the PER decreases, resulting in a reduction of the obtained gain with the algorithm. This behaviour can be seen in Fig. 7d, for $f_{\text{EMI}} = 500$ MHz and $f_{\text{BIT}} = 250$ MHz. These results show the capability of the algorithm to operate with different EMI frequencies and bit rates and the improvement in terms of gain when the rewards are learned. The results also show the algorithm's limitations in harsh EM environments.

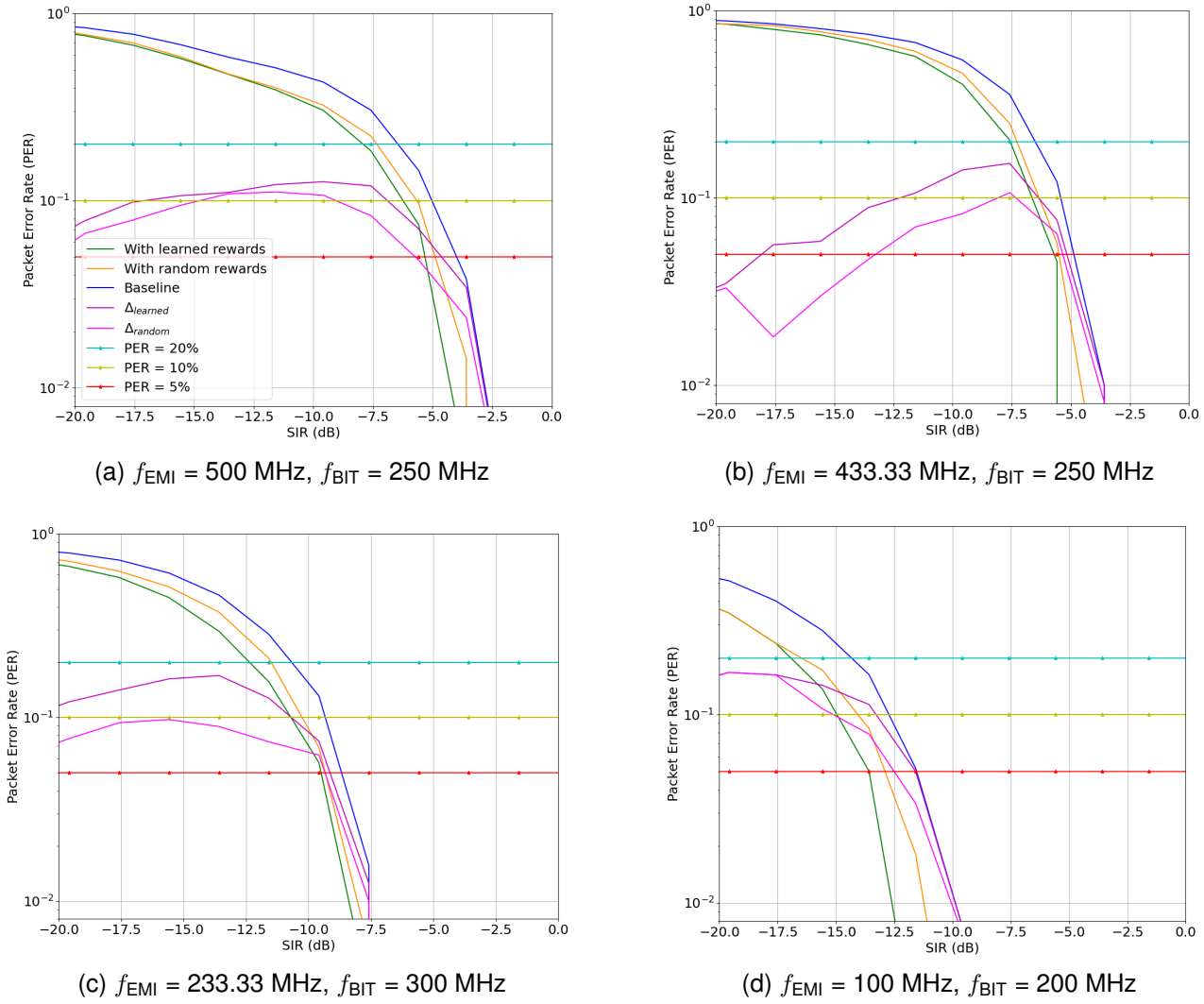
C. Latency

The latency of the algorithm is determined by the maximum number of resends allowed. Fig. 8 shows the amount of resends used for the algorithm (with and without learned rewards) at each SIR level when the maximum number of resends is set to 8, for $f_{\text{EMI}} = 500$ MHz and $f_{\text{BIT}} = 350$ MHz. Therefore, the performance of the algorithm highly depends on this parameter. As we can see, when the rewards are learned (Fig. 8b), the amount of resends (for a SIR between -11 dB and -5 dB) is more significant than the resends used for the algorithm when the rewards are set as random (Fig. 8a). However, when the rewards are learned through the process, as will be explained in Section V-D, each time the algorithm chooses to stop resending at a given SIR and the probabilities of receiving correct data at the next time step are less than in the present time (wrong decision), the rewards are modified, and the new decision is to continue resending until these probabilities become higher again. When the rewards are optimal, the algorithm employs a higher number of resends (which decreases the PER, as seen in Fig. 7).

As shown in Fig. 8, the decision algorithm decides to resend only once 100% of the times from -30 to -16 dB, in both cases. This is because the amount of Channel True Negatives and Channel False Negatives increases due to the harsh EMI environment at these SIR levels. This causes the algorithm never to trust the first received data and ask for a resend. As soon as the EM environment is less harsh, from -9 dB to -5 dB, we observe an increase in the number of resends used. Finally, when the amount of True Positives is considerably high (more than 96%), the algorithm uses the first received data without resending, assuming it can trust the data (from -5 to 5 dB).

D. Learned Rewards

The decision algorithm formulation highly depends on the rewards assigned to the different data categories and the penalty resend. For this reason, an experiment was performed in which we started with rewards chosen randomly in the range of $[-1, 1]$, and then, for a specific packet at a specific SIR level, we determined if the algorithm has made the right decision (use the data, resend or go to a minimum risk state) or the



(a) f_{EMI} = 500 MHz, f_{BIT} = 250 MHz

(b) f_{EMI} = 433.33 MHz, f_{BIT} = 250 MHz

(c) f_{EMI} = 233.33 MHz, f_{BIT} = 300 MHz

(d) f_{EMI} = 100 MHz, f_{BIT} = 200 MHz

Fig. 7: Gain with the decision algorithm at different PER and SIR levels, with and without optimal rewards.

wrong decision. We measure that by setting a threshold for the received data (see Table IV). Then, when the algorithm makes the wrong decision, we adjust the rewards (subtracting/adding a fixed amount) until the decision is correct.

Table V shows the performance metrics obtained with the decision algorithm if the rewards are set as random or learned. Accuracy is calculated as the number of right/wrong decisions divided by the total number of decisions. For each decision ("Resend", "Use data received at t_n", or "Go to minimum risk state"), we set a criterion to measure how accurate these decisions are. Table IV shows the criterion for evaluating the algorithm's accuracy. We set a threshold for the probability of the received data being correct (96%), which serves as a metric to evaluate whether the decisions are right or wrong. The process to calculate the accuracy is as follows:

1) The probabilities of receiving a TP, FP, CTN, DTN or CFN are determined for each SIR level. This is shown in Fig. 4.
2) The algorithm works with these probabilities and determines its final decisions for each packet using the modified Bellman equation, as explained in Section IV-A.
3) We evaluate how accurate these final decisions are (see Table IV). For example, for the decision "Resend", the following criterion is used: if the algorithm has decided to resend at t_n, and the probability of receiving a True Positive at that same SIR level is higher than 96%, we consider that the algorithm made a "right decision".
4) For each packet and each SIR level, we count how often each decision is "right" or "wrong". We do that for the two cases considered in this paper: with and without learned rewards, finally obtaining the results shown in Table V.

The table shows that for the decision "resend", in 77.63% of the cases, it was the right decision, while in 22.37% of the cases this was the wrong decision. When we allow the algorithm to learn those rewards, the number of correct decisions increases to 98.01%, while the wrong decisions decrease to 1.99%. That demonstrates the dependence of the algorithm on the chosen rewards. The same behavior can be

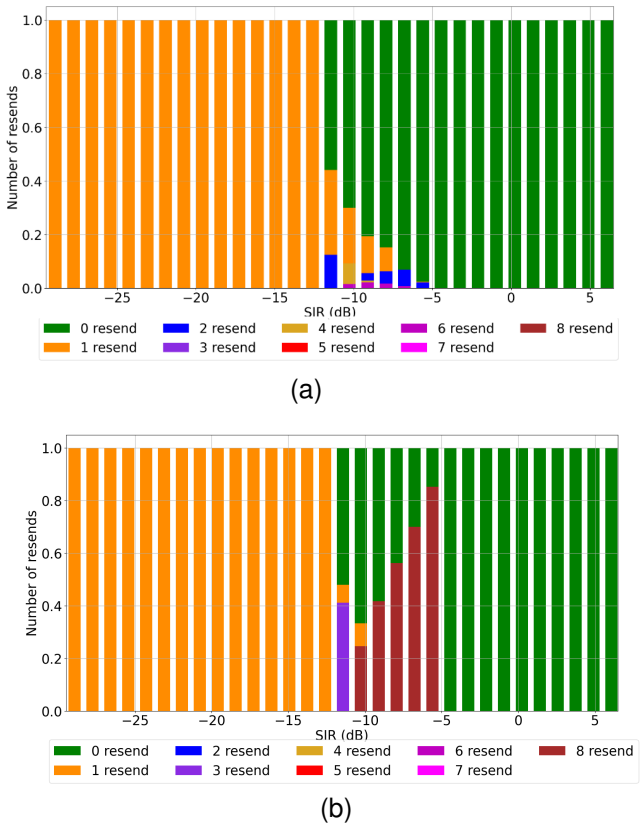


Fig. 8: Number of resends used by the decision algorithm per SIR level, with random rewards (Fig. 8a) and with learned rewards (Fig. 8b), for $f_{\text{EMI}} = 500$ MHz and $f_{\text{BIT}} = 350$ MHz.

TABLE IV: Criterion to evaluate the algorithm decisions

Decision	**State of data**	**Effect** (Right/wrong decision)
Resend	$P_{\text{data after resend}}(TP) \geq 96\%$	Right
Resend	$P_{\text{data after resend}}(TP) < 96\%$	Wrong
Use data received at t_n	$P_{\text{data at } t_n}(TP) \geq 96\%$	Right
Use data received at t_n	$P_{\text{data at } t_n}(TP) < 96\%$	Wrong
Go to minimum risk state	$P_{\text{data at } t_{n-1}}(TP) < 96\%$ and $P_{\text{data at } t_n}(TP) < 96\%$	Right
Go to minimum risk state	$P_{\text{data at } t_{n-1}}(TP) < 96\%$ and $P_{\text{data at } t_n}(TP) \geq 96\%$	Wrong

observed with the decisions "Use data received at t_n" and "Go to minimum risk state".

TABLE V: Performance metrics with and without learned rewards, for $f_{\text{EMI}} = 500$ MHz and $f_{\text{BIT}} = 350$ MHz

Decision	Accuracy	
	Initial rewards	Learned rewards
Resend (**Right decision**)	77.63%	98.01%
Resend (**Wrong decision**)	22.37%	1.99%
Use data received at t_n (**Right decision**)	54.93%	85.29%
Use data received at t_n (**Wrong decision**)	45.07%	14.71%
Go to minimum risk state (**Right decision**)	6.90%	98.17%
Go to minimum risk state (**Wrong decision**)	93.10%	1.83%

VI. Conclusion

This paper assessed the performance of a decision-making algorithm based on the Bellman Equation used in Markov Decision Processes to deal with EMI-induced bit errors. A detailed explanation of the decision algorithm was presented, and a comparison was made with the case in which no decision-making has been implemented. This comparison in terms of PER shows that the proposed technique decreases the PER by 16.94 % at low SIR (-13.58 dB) when $f_{\text{EMI}} = 233.33$ MHz and $f_{BIT} = 300$ MHz. This leaves a higher percentage of data that is error-free or has been successfully corrected. As such, the overall safety and availability of a system operating in harsh electromagnetic environments increases.

Furthermore, two different cases are studied to show the influence of the rewards and the maximum amount of resends on the algorithm's performance. When the rewards are learned, the number of resends used by the algorithm increases, increasing the certainty of the received data.

In the second case, it is shown that the process of learning the rewards increases the accuracy of the final decisions made by the algorithm. This study also shows that these parameters can be tuned to maximize the algorithm's performance.

Harsh EM environments show the limits of the proposed algorithm. As can be seen from the obtained results, at low SIR, the algorithm's effectiveness significantly drops due to the high amount of incorrect data words received, leading the algorithm to "go to a minimum risk state". Therefore, further research is needed to protect the transmitted data from corruption due to EMI and improve the decision algorithm.

In future work, the proposed decision algorithm will be evaluated using different coding techniques, such as Pulse Amplitude Modulation or Huffman coding instead of Hamming code. In addition, we will explore other decision algorithms, for instance, based on Bayesian theory. Finally, the robustness of the algorithm will be tested against multi-harmonic EM disturbances.

This paper demonstrated the feasibility of using a decision-making algorithm based on a modification of the Bellman equation in harsh electromagnetic environments, to increase the certainty of the received data.

REFERENCES

[1] J. Van Waes, D. Vanoost, J. Vankeirsbilck, J. Lannoo, D. Pissoort, and J. Boydens, "Resilience of error correction codes against harsh electromagnetic disturbances: Fault mechanisms," *IEEE Transactions on Electromagnetic Compatibility*, vol. 62, no. 4, pp. 1017–1027, 2020.

[2] J. Van Waes, J. Vankeirsbilck, J. Lannoo, D. Vanoost, D. Pissoort, and J. Boydens, "Complementary fault models for assessing the effectiveness of hamming codes," in *2019 IEEE XXVIII International Scientific Conference Electronics (ET)*, 2019, pp. 1–4.

[3] H. Jin, L. Zhang, H.-Z. Ma, S.-C. Yang, X.-L. Yang, and E.-P. Li, "Machine learning for complex EMI prediction, optimization and localization," in *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, 2017, pp. 1–3.

[4] P. Eliardsson and P. Stenumgaard, "Artificial intelligence for automatic classification of unintentional electromagnetic interference in air traffic control communications," in *2019 International Symposium on Electromagnetic Compatibility - EMC EUROPE*, 2019, pp. 896–901.

[5] H. Chen and S. Ye, "Modeling and optimization of EMI filter by using artificial neural network," *IEEE Transactions on Electromagnetic Compatibility*, vol. 61, no. 6, pp. 1979–1987, 2019.

[6] D. Zupan, N. Czepl, and D. Kircher, "Framework for developing neural network regression models predicting the influence of EMI on integrated circuits," in *2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2022, pp. 1–4.

[7] W. Belloumi, A. Bréard, O. Hajji, C. Vollaire, and J. B. H. Slama, "Automatic PCB layout optimization of a DC-DC converter through genetic algorithm regarding EMC constraints," *IEEE Access*, vol. 9, pp. 149 870–149 882, 2021.

[8] Z. Tianci, W. Yongyong, and L. Renfei, "Research on behavior mathematical modeling of cgf based on reinforcement learning under complex electromagnetic environment," in *2021 IEEE International Conference on Data Science and Computer Application (ICDSCA)*, 2021, pp. 781–784.

[9] L. Zeng, F. Yao, J. Zhang, and M. Jia, "Dynamic spectrum access based on prior knowledge enabled reinforcement learning with double actions in complex electromagnetic environment," *China Communications, 2022*, vol. 19, no. 7, pp. 13–24, 2022.

[10] R. A. H. William Beranek, "Dynamic programming and Markov processes," *Technometrics*, vol. 3, no. 1, pp. 120–121, 1961. [Online]. Available: https://doi.org/10.1080/00401706.1961.10489934

[11] S. Balachandran and E. M. Atkins, "A constrained Markov decision process for flight safety assessment and management," 2015.

[12] A. Zacharaki, I. Kostavelis, and I. Dokas, "Decision making with stpa through Markov decision process, a theoretic framework for safe human-robot collaboration," *Appl. Sci.*, vol. 11, p. 5212, 2021. [Online]. Available: https://doi.org/10.3390/app11115212

[13] Y. Li, L. Ru, and J. Niu, "Power structure optimization for grid-connected photovoltaic system based on Markov decision processes," in *2010 5th IEEE Conference on Industrial Electronics and Applications*, 2010, pp. 620–623.

[14] G. Suseendran and D. Balaganesh, "Cattle movement monitoring and location prediction system using Markov decision process and IoT sensors," in *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, 2021, pp. 188–192.

[15] M. A. Frey, J. Attmanspacher, and A. Schulte, "A dynamic Bayesian network and Markov decision process for tactical UAV decision making in mum-t scenarios," in *2022 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, 2022, pp. 47–54.

[16] M. H. Ngo and V. Krishnamurthy, "On optimality of monotone channel-aware transmission policies: A constrained Markov decision process approach," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 3, 2007, pp. III–621–III–624.

[17] T. Erseghe, A. Zanella, and C. G. Codemo, "Markov decision processes with threshold based piecewise linear optimal policies," *IEEE Wireless Communications Letters*, vol. 2, no. 4, pp. 459–462, 2013.

[18] J. Zheng and A. S. Namin, "Defending SDN-based IoT networks against ddos attacks using Markov decision process," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 4589–4592.

[19] N. S. Monteiro, V. M. Gonçalves, and C. A. Maia, "Motion planning of mobile robots in indoor topological environments using partially observable Markov decision process," *IEEE Latin America Transactions*, vol. 19, no. 8, pp. 1315–1324, 2021.

[20] M. Sivadas and A. Mohan, "Robot navigation in an uncertain environment using dynamic programming via Markov decision process," in *2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT)*, 2018, pp. 190–194.

[21] D. Hill, "Plane wave integral representation for fields in reverberation chambers," *IEEE Transactions on Electromagnetic Compatibility*, vol. 40, no. 3, pp. 209–217, 1998.

[22] M. Magdowski, S. V. Tkachenko, and R. Vick, "Coupling of stochastic electromagnetic fields to a transmission line in a reverberation chamber," *IEEE Transactions on Electromagnetic Compatibility*, vol. 53, no. 2, pp. 308–317, 2011.

[23] F. Vanhee, D. Pissoort, J. Catrysse, G. A. E. Vandenbosch, and G. G. E. Gielen, "Efficient reciprocity-based algorithm to predict worst case induced disturbances on multiconductor transmission lines due to incoming plane waves," *IEEE Transactions on Electromagnetic Compatibility*, vol. 55, no. 1, pp. 208–216, 2013.

[24] J. Lannoo, J. Van Waes, A. Degraeve, D. Vanoost, J. Boydens, and D. Pissoort, "Effectiveness of time diversity to obtain EMI-diverse redundant systems," in *2018 International Symposium on Electromagnetic Compatibility (EMC EUROPE)*, 2018, pp. 288–292.

[25] H. Tirmizi, J. Lannoo, D. Vanoost, T. Claeys, G. A. E. Vandenbosch, and D. Pissoort, "Resilience of time diversity against multiharmonic electromagnetic disturbances under reverberation conditions: An overview of fault mechanisms," *IEEE Transactions on Electromagnetic Compatibility*, vol. 64, no. 3, pp. 631–639, 2022.

[26] Keysight. [Online]. Available: https://www.keysight.com/zz/en/lib/resources/software-releases/pathwave-em-design-empro-2020.html

[27] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, pp. 147–160, 1950.

[28] M. Dipperstein, *Hamming (7,4) Code Discussion and Implementation*. [Online]. Available: https://michaeldipperstein.github.io/hamming.html

[29] T. Claeys, H. Tirmizi, H. Habib, D. Vanoost, G. A. Vandenbosch, and D. Pissoort, "A system's perspective on the use of EMI detection and correction methods in safety critical systems," in *2021 IEEE International Joint EMC/SI/PI and EMC Europe Symposium*, 2021, pp. 905–910.

[30] J. Torres, *The Bellman Equation*. [Online]. Available: https://towardsdatascience.com/the-bellman-equation-59258a0d3fa7

[31] J. Van Waes, D. Vanoost, J. Vankeirsbilck, J. Lannoo, D. Pissoort, and J. Boydens, "Mitigating false negatives in harsh electromagnetic environments by combining complementary error detection codes," *IEEE Letters on Electromagnetic Compatibility Practice and Applications*, vol. 3, no. 1, pp. 34–37, 2021.

[32] J. Van Waes, J. Lannoo, J. Vankeirsbilck, A. Degraeve, J. Peuteman, D. Vanoost, D. Pissoort, and J. Boydens, "Effectiveness of hamming single error correction codes under harsh electromagnetic disturbances," in *2018 International Symposium on Electromagnetic Compatibility (EMC EUROPE)*, 2018, pp. 271–276.