Multiobjective Reinforcement Learning for Cognitive Satellite Communications Using Deep Neural Network Ensembles

Paulo Victor Rodrigues Ferreira, Randy Paffenroth, Alexander M. Wyglinski, *Senior Member, IEEE*, Timothy M. Hackett, Sven G. Bilén, *Senior Member, IEEE*, Richard C. Reinhart, and Dale J. Mortensen

***Abstract*— Future spacecraft communication subsystems will potentially benefit from software-defined radios controlled by artificial intelligence algorithms. In this paper, we propose a novel radio resource allocation algorithm leveraging multiobjective reinforcement learning and artificial neural network ensembles able to manage available resources and conflicting mission-based goals. The uncertainty in the performance of thousands of possible radio parameter combinations and the dynamic behavior of the radio channel over time producing a continuous multidimensional state–action space requires a fixed-size memory continuous state–action mapping instead of the traditional discrete mapping. In addition, actions need to be decoupled from states in order to allow for online learning, performance monitoring, and resource allocation prediction. The proposed approach leverages the authors' previous research on constraining decisions predicted to have poor performance through "virtual environment exploration." The simulation results show the performance for different communication mission profiles, and accuracy benchmarks are provided for the future research reference. The proposed approach constitutes part of the core cognitive engine proof-of-concept delivered to the NASA John H. Glenn Research Center's SCaN Testbed radios on-board the International Space Station.**

***Index Terms*— Satellite communication, machine learning, artificial intelligence, reinforcement learning, neural networks, cognitive radio, space communication, SCaN Testbed, NASA GRC.**

I. Introduction

In 2012, a research project led by NASA John H. Glenn Research Center's Space Communications and Navigation (SCaN) group [1] delivered a communications research platform to the International Space Station (ISS). Known as SCaN Testbed [2]–[6], it is comprised of three software-defined radios with the aim of facilitating on-orbit communications research for future aerospace applications. The next frontier of space-based communication systems is to develop and test cognitive engines (CEs) leveraging the potential of on-orbit software-defined radios (SDRs) for future space exploration missions [7].

Next-generation space-based communication systems are expected to offer more flexibility in order to operate more efficiently in challenging environments, which include orbital dynamics and atmospheric and/or space weather, or when the spacecraft is requested to operate during unpredicted conditions. Therefore, there is need for a CE that efficiently allocates resources in order to achieve several goals, each with a certain level of priority, while the communications channel dynamically changes. The CE should consider the effects of resource consumption by the communications systems on other spacecraft subsystems, while allocating multiple different resources in an attempt to achieve multiple objectives.

A cognitive radio (CR) [8] has a CE that utilizes environmental awareness across different network layers [9] and is capable of autonomously performing perception, learning, and reasoning activities in order to optimize resource allocation based on the current node's hardware and software capabilities, channel conditions, and operational requirements.

Several simple adaptive technologies have been deployed in real-world applications as steps towards future fully cognitive systems. For instance, adaptive coding and modulation (ACM) schemes [10]–[12] adapt radio parameters during signal fading events on satellite-based television receivers as part of the DVB-S2 standard [13]. Spectrum sensing (SS) for dynamic channel access is another example, in which temporarily unused spectrum is re-used for different applications [14]–[16].

Several machine learning (ML) techniques have been proposed in the past using CR as a case study. ML enables online learning, a core feature for any CE. He *et al.* [17] and Abbas *et al.* [18] have focused on the learning problem, ML-based terrestrial CR [19], and SS [14], [15]. Several ML-based techniques have focused on CR resource allocation, such as genetic algorithms (GAs) [20].

These adaptive technologies work very well individually. ACM helps mitigate fading and SS allows secondary users to

Manuscript received June 23, 2017; revised December 30, 2017; accepted April 4, 2018. Date of publication May 3, 2018; date of current version July 23, 2018. This work was supported in part by the NASA John H. Glenn Research Center under Grant NNC14AA01A, in part by the NASA Space Technology Research Fellowship under Grant NNX15AQ41H, and in part by CAPES Science Without Borders Scholarship under Grant BEX 18701/12-4. *(Corresponding author: Paulo Victor Rodrigues Ferreira.)*

P. V. R. Ferreira and A. M. Wyglinski are with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: prferreira@wpi.edu; alexw@wpi.edu).

R. Paffenroth is with the Department of Mathematical Sciences, Department of Computer Science and Data Science Program, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: rcpaffenroth@wpi.edu).

T. M. Hackett and S. G. Bilén are with the School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, PA 16802 USA (e-mail: tmh5344@psu.edu; sbilen@psu.edu).

R. C. Reinhart and D. J. Mortensen are with Space Communications and Navigation, NASA John H. Glenn Research Center, Cleveland, OH 44135 USA (e-mail: richard.c.reinhart@nasa.gov; dale.mortensen@nasa.gov).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2018.2832820

share frequency bands temporarily. ML-based algorithms for CR deal with multi-objective problems [17]–[20]. However, these methods usually consider less than five adaptable radio transmitter parameters and conflicting objectives, and assume that performance functions are independent of the communications channel, *i.e.*, the operational environment. One of the most popular ML-based algorithms, the GA, possesses limitations for online learning requirements since it operates in batch mode. Bernardo *et al.* [21] describe a distributed solution for decentralized spectrum allocation agent in which RL is enhanced by artificial neural networks (NNs) [22] that use only one performance function as input and output values for the same resource for multiple users.

Therefore, to the best of the authors' knowledge, none of the aforementioned techniques have tackled the problems related to multiple radio resource allocation and conflicting goals for space-based communication systems considering online learning and more than five adaptable parameters and performance functions coupled to the dynamics of the environment described above while using the solution proposed in this paper. Seeking to solve the learning problem for CR for space applications, Ferreira *et al.* [23] have proposed a solution based on reinforcement learning (RL) [24], [25], and Ferreira *et al.* [26] proposed a hybrid solution that adds NNs. However, these solutions are limited by making the following assumptions:

- non-fading channels;
- discrete fixed-size state and action spaces; and
- available memory to store state–action states.

These assumptions result in the ML-based solution having very limited real-world applications. Depending on the carrier frequency, spacecraft orbital dynamics, atmospheric conditions, space weather conditions, and available memory onboard, multi-objective resource allocation represents a big challenge, requiring a new solution to allow operations during unpredicted conditions.

In this paper, we present a novel CE design for space-based communication systems that solves the aforementioned limitations. The proposed CE autonomously selects multiple radio transmitter settings while attempting to achieve multiple conflicting goals in a dynamically changing communications channel. It leverages the RL structure proposed in [23], provides an upgraded version of "virtual exploration" proposed in [26], and integrates it with a novel deep NN ensemble design and two new algorithms to implement the exploitation part of multi-objective reinforcement learning (MORL). Thus, the proposed CE enables all the following:

- table-free state–action mapping with fixed memory size;
- operation over dynamically changing channels;
- decoupling of states from actions; and
- usage of continuous action and state spaces.

These features are achieved at the cost of processing requirements to train the NNs and make predictions with them. In summary, the usage of NNs eliminates the RL state–action table and Q-values, which consequently decouples them allowing one action to be mapped into several other states, enabling operation over dynamic channels. Finally, continuous spaces lead to near-optimal solutions.

The remaining sections of this paper are organized as follows: Section II provides a brief overview of the ML concepts used in this work and points the reader to more detailed readings. Section III describes the proposed solution. Section IV discusses the simulation results. Concluding remarks are given in Section V.

II. Machine Learning Overview

ML is a term used to describe several theories and algorithms that aim to automate computational decision-making tasks. Two noticeable ML research advancements were: 1) in 2015 the deep Q-network system was used to play vintage Atari games better than a human [27], and 2) in 2016 the AlphaGo [28] system won the Go World Championship. Both systems, and the ones that have been built upon them, have been pushing the research boundaries of ML in all different fields of application, with the primary driver being computer vision systems to assist self-driving cars [29].

All these systems leverage both deep NNs and RL, briefly described in this section. This recent technological revolution has inspired the proposal of a satellite-based communication system control that makes use of both these concepts. Nevertheless, the requirements are different and, to the best of the authors' knowledge, lead to the research and development of algorithms that are not currently available in the literature. With the basic principles of RL and NN, the proposed hybrid approach design is presented, followed by simulation results and a brief discussion of the findings and trade-offs.

A. Neural Networks Overview

An artificial NN is a method for mapping inputs to outputs, usually employed in pattern recognition problems, or in function-fitting problems [22]. NNs consisting of three or more layers are referred to as "deep NNs" [30]. For instance, in this paper, deep NNs are used to approximate the non-linear environment effects by mapping actions into rewards and states into actions.

NN algorithms are composed of two main steps: training and prediction. Initially, examples containing input and output data are preprocessed and provided to the NN for training. After meeting some minimum performance requirements, the trained NN can be used as a predictor. For a detailed description of NN basics, the interested reader is referred to [30, Ch. 6].

Currently, there are no general guidelines available in the literature on how to select the NN architecture elements, although [31] provides some useful comments and advice on what to consider when making these decisions. In this paper, we consider the standard multi-layer fully-connected NN trained by a backpropagation-based algorithm. More details on the chosen NN architectures are provided in Section III.

B. Reinforcement Learning Overview

An RL algorithm is designed to learn through interactions with the environment in a trial-and-error fashion [24].

Based on predefined goals, the RL agent looks for actions that optimize the system performance in achieving those goals. In a traditional RL algorithm, the agent alternates between exploration and exploitation of actions according to the exploration probability value ε_k computed by the exploration probability function $f(\varepsilon)$ at the discrete time instant k.

RL problems can be modeled as state-transition problems, where the state transition itself is modeled as a Markov decision process (MDP) [32]. In this paper deterministic state transitions are assumed, and an action is referred to as a set of radio transmitter parameters and a state is assumed to be a set of communications system performance values. More detailed description regarding actions and states are provided in the sections below.

Usually, control problems require the computation of a policy that maps observed states into actions. The work presented in this paper concerns controlling radio parameters such that performance is kept at an optimum level for the entire time while the channel conditions changes. The environment is comprised of the satellite communications channel through which propagating signals are affected by the dynamic geometry of the line-of-sight between transmitter and receiver and its surroundings (buildings in the vicinity of ground stations or structures in the vicinity of the antennas onboard the spacecraft), as well as dynamics of the atmosphere and space weather. Therefore, both state-transition and action–state models must account for all the variables involved with these highly complex dynamic processes. For this reason, these models are assumed to be unknown due to being too complex or difficult to obtain and a learning method that balances exploration of new actions and exploitation of known actions is expected to recommend actions, considering conflicting goals.

An action–value function $Q_\pi(s,a)$ representing the value of a certain action a taken when in state s while following policy π, should be evaluated for all actions possible from state s through a greedy policy given by

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a), \tag{1}$$

where, for every state $s \in S$, an action $a \in A$ with maximal $Q(s,a)$ is chosen, given the state space S and action space A. For problems with either continuous or discrete A containing thousands of actions a, it may not be feasible to evaluate all $Q(s,a)$ when in state s. This is the case for radio communications, for which exploring each action a over the air may cost time and force the radio receiver to experience degraded performance levels for an extended period of time. The practical alternative is to ensure that the agent keeps exploring actions.

As mentioned in [23], the CR is interested in the immediate reward, and any action can be taken from any state without the need for planning, for on-line applications. These assumptions result in a modified version of the Bellman's Q-value function, given by

$$Q_{k+1}(s_k,a_k) = Q_k(s_k,a_k) + \alpha[r_k - Q_k(s_k,a_k)], \tag{2}$$

where Q_{k+1} is the updated value after getting the reward r_k, by taking action a_k while in the state s_k, and α is the learning rate. Even though the state-transition and state–action models are unknown, $f(\varepsilon)$ and the reward function ρ, used in $r_k = \rho(s_k,a_k)$ still need to be defined. The state–action policy function h used in $a_k = h(s_k)$ is given by $\pi(s)$ in Eq. (1).

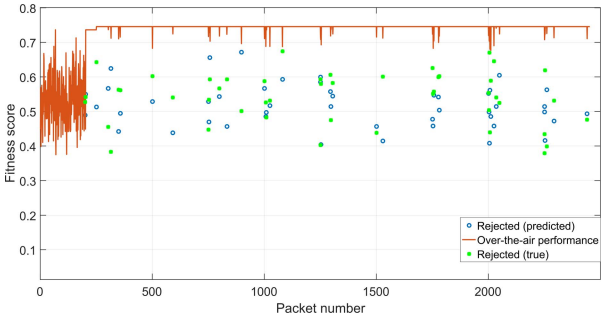

Fig. 1. Example of a 50-s time series RLNN using only virtual exploration with action rejection probability equal to 1. During the first 200 packets, the NN collects training data. Clearly, exploration of actions with predicted performance values below the threshold are avoided.

C. RLNN Overview

Previous work [23], [26] presented novel MORL algorithms. The NN-based RL (RLNN) [26], applied to satellite communications that assumed a non-changing AWGN channel as a downlink from a geostationary (GEO) satellite to a fixed ground station, was composed of the RL structure proposed in [23] and NNs [22]. The RLNN uses one NN to virtually test different actions and measure their performance allowing "virtual exploration" of the environment.

After training, the RLNN predicts all actions' multi-objective fitness score, *i.e.*, weighted sum of different objectives. A communication mission profile is a set of weight values (more details are given in Section IV). Next, actions are classified into "good" or "bad", *i.e.,* optimal and non-optimal settings of radio parameters, based on a performance threshold value, defined to be a certain percentage of the maximum performance value. Next, one action from one class was randomly chosen using an "action rejection probability" value. Fig. 1 illustrates an example of a time series performance of actions rejected by the RLNN exploration algorithm that were predicted to perform below the threshold of 0.7. Instead, it suggests actions predicted to perform above that threshold value.

One of the great benefits of the RLNN is to know *a priori*, *i.e.*, offline, each action's expected performance value, allowing them to be classified. Another benefit that follows is the ability to control the amount of good or bad actions to be explored through the "action rejection probability", improving the overall system performance by filtering out "bad" actions, which means the agent can avoid spending time on exploring actions predicted to result in poor performance.

One of the main limitations of the basic RL [23] and the RLNN algorithm [26] is that both consider only a non-dynamic channel. Fig. 2 shows a time series example of the multi-objective fitness score for each action assuming a non-dynamic channel. Depending on the orbital dynamics, carrier frequency, and atmospheric and/or space weather, fast and/or slow fading might be introduced to the channel, making it change dynamically.

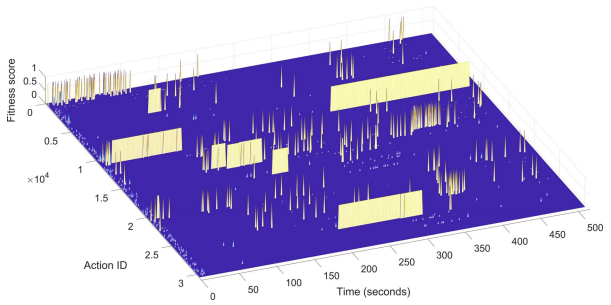

Fig. 2. RL algorithm searching for better actions to be exploited (constant amplitude bands) over a GEO satellite communications channel. Before exploiting the RL decision-maker considers performance values of actions when they were last explored.

Each discrete action used had its fitness score value stored in a Q-vector (assuming the rewards to be equal to the RL states, see [23], [26]), which received new values for new positions during exploration and updated existing ones during exploitation.

However, there are three main limitations in using this approach when the channel dynamically changes: (i) the performance value for an action taken in the present might be different from the one measured for that same action taken in the past; (ii) performance values regarding a certain action must be stored for each different multi-dimensional RL state, with each RL state being represented by a continuous variable; and (iii) each action–RL state performance value must be stored for a specific channel condition that changes dynamically, which is also a continuous variable.

Limitation (i) results in outdated performance values leading to erroneous decision-making when exploiting a certain action. A possible solution to this issue would be to update that specific action performance. Although this update must be done for all other actions previously explored, it would be just another exploration round, still making their performance unknown during the exploitation round. For an environment that does not change over time this does not seem to be an issue.

However, for dynamically changing channels, one certain action might have different performance for each different environment condition, making previous explorations' fitness scores outdated. This is well illustrated by Fig. 3, which shows the simulated performance values of two different actions a_{A}[1] and a_{B}[2] (action structure is described in Section IV), which were kept fixed while the communications channel changed. Some actions' performances show minimal difference over time, as shown by the left-side panel, even for different communication mission profiles (described in Section IV). On the other hand, there are actions that have performance changes over time, sometimes non-linear performance behavior, as is the case shown on the right-side panel. This fact represents one of the main drivers for the research presented in this article.

Limitation (ii) results in an exponential increase in memory size requirement due to the different performance levels for each action when the channel conditions change and are assumed to be discrete. Eq. (2) would then have the RL state and action to be a function of continuous time instant t, computed by

$$Q_{k+1}(s_k(t), a_k) = Q_k(s_k(t), a_k) + \alpha[r_k(t) - Q_k(s_k(t), a_k)], \tag{3}$$

which represents the dynamic behavior of a given action's fitness score over time.

In addition to the limitations imposed by (ii), limitation (iii) adds another dimension to the performance values, which is their variation over time, making the data storage problem even worse. Thus, from the implementation perspective of online system operations it is impractical and impossible to save all that information.

III. Proposed Solution

Before proceeding with the proposed algorithm, it is worth mentioning the distinction between two important terms: RL state and environment state levels. RL states are the observed system performance measured and/or monitored by the RL agent. The RL states can change in response to the execution of different RL actions and to the current environment state. As described in Section IV, these RL states are features of the fitness function. In satellite communications, the channel itself is assumed to convey all of the environment changes, represented and measured by the SNR levels at the receiver. Thus, environment state levels are also referred to as channel conditions.

For dynamically changing environments, the RLNN [26] now takes into consideration the E_{s}/N_0 levels on its inputs and is referred to as "Exploration NN" (NN1), as shown in Fig. 4. The NN1 block is actually an ensemble of several deep NNs in parallel trained with the same training dataset and used to generate multi-objective performance values from the same inputs that are averaged to become the resultant prediction value, as shown in Fig. 4. Regarding the NN architecture used for exploration, each deep exploration NN from the ensemble has a feedforward structure trained by the Levenberg–Marquardt backpropagation algorithm [33], [34]. It is composed of three fully-connected layers without bias: two hidden layers contain 7 and 50 neurons each (resulting in 449 weight parameters per NN), both layers using log-sigmoid transfer function, and the output layer with one neuron using the standard linear transfer function. It uses the mean-squared error (MSE) as the performance function with two different early stop training conditions: minimum error gradient equal to 10^{-12} and maximum validation checks equal to 20. During training the training dataset was randomly split into 70% for training, 15% for testing, and 15% for validation, with all data scaled to the $[-1, 1]$ range. The ensemble consists of $\mathrm{NN1}_m = 20$ NNs that are trained with the same training dataset. The NN1 ensemble prediction outputs are used to classify the input actions according to the performance threshold (described in Section IV) by an "action selector agent." The "action rejection probability" controls if an action is randomly chosen from the good or bad sets, which in turn is used to choose the radio parameters.

[1] $a_{\mathrm{A}} = (4.16 \cdot 10^5, 0, 5, 0.35, 32, 0.9)$
[2] $a_{\mathrm{B}} = (4.16 \cdot 10^5, 0, 5, 0.20, 32, 0.75)$

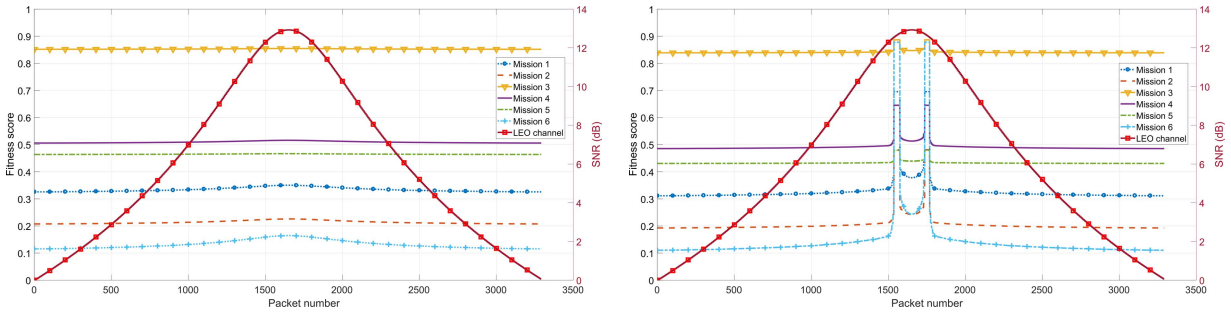

Fig. 3. Multi-objective performance for two different actions a_{A} and a_{B}, shown in panels (a) and (b), respectively. Each action was kept fixed during the entire time the environment changed, following the SNR profile, also shown in both panels. The performance of action a_{A} does not seem to change too much over time for each different mission. However, the performance of action a_{B} changes abruptly in all missions as the environment changes.

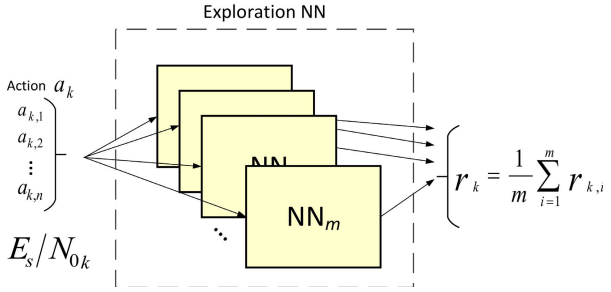

Fig. 4. Deep NNs used by RL exploration receive the same multi-dimensional input. Ensemble learning trains several NNs in parallel, and averages their prediction outputs into a single multi-objective performance value.

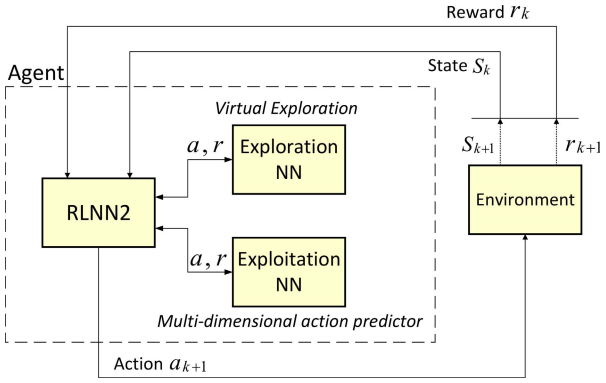

Fig. 5. Novel MORL for dynamic satellite communication channel environments. The proposed RLNN2 algorithm interacts with the proposed exploration and exploitation deep neural network ensembles.

We describe our proposed hybrid MORL algorithm, called RLNN2, that uses two different NNs, one for exploration and another for exploitation, as illustrated in Fig. 5. The RLNN2 combines the NN1 algorithm with a new one, the "Exploitation NN" (NN2). This novel algorithm is capable of dealing with dynamically changing attenuation levels within the satellite communications channel. The RLNN2's agent interacts with the environment and either explores different actions through "virtual exploration," thus preventing communications systems from spending time exploring combinations of radio parameters that would result in poor performance, or exploits actions already tried before, predicting actions best suitable for a dynamically changing environment using the NN2 through a novel technique called "multi-dimensional action predictor," described in the following subsection, as shown in Fig. 5.

Exploiting Reinforced Multi-Dimensional Actions: Seeking to solve the limitations mentioned in Section II, the computation of the Q-values was dropped and instead it was proposed to use another NN to predict which action should be exploited, given the current environment state level, *i.e.*, the communications channel conditions. The proposed NN2 features include: (i) it solves the problem of using outdated performance values for deciding which action to exploit, (ii) it does not require storage of all action–state performance values for all environment conditions, and (iii) it allows for the RL state to be decoupled from the action. This feature allows exploitation of different actions in an attempt to achieve the same previous performance level while the environment dynamically changes.

Therefore, the proposed NN2 architecture is composed of an array of ensembles of NNs, each predicting one dimension of the multi-dimensional action vector. All exploitation NNs have the same shallow architecture and receive the same input multi-dimensional performance vector, as shown in Fig. 6. The NN2 output vector is the average output of NN2$_m = 10$ parallel NNs for each output vector element. The choice of this architecture was made based on the one that showed the smallest MSE averaged over 100 simulation runs, and a brief performance analysis is provided in Section IV. Each NN is composed of two fully-connected layers without bias: the hidden layer contains 20 neurons (resulting in 160 parameters) and uses the log-sigmoid transfer function, and the output layer uses the linear transfer function. All the other functions are the same as the ones used for the individual exploration NNs mentioned in Section II-C.

The logic used to decide the NN2 input, the multi-objective performance values, are quite complex, requiring several conditional statements. Algorithm 1 describes the general operational procedure of the novel proposed hybrid architecture, called RLNN2, including the interactions between the RL, NN1, and NN2. Algorithm 2 describes the logic used to choose the NN2 inputs, required within Algorithm 1. All the required parameters are described and defined in Section IV.

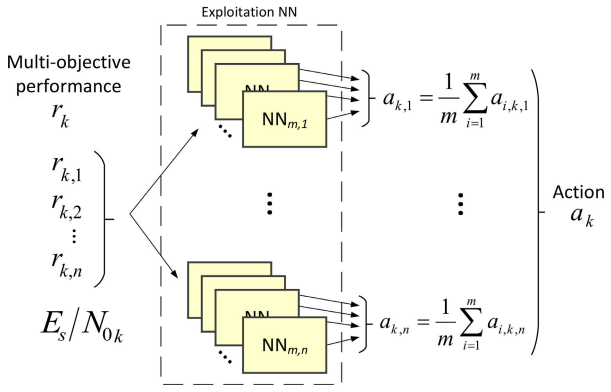

Fig. 6. Ensemble of NNs for action prediction by performance exploitation. Different sets of NNs predict a specific action element by averaging their outputs. All NNs receive the same input.

TABLE I
RLNN2 ADAPTABLE PARAMETERS

Parameter	Variable	Value range
Modulation order	$\bar{M}$	$[4, 8, 16, 32]$
Bits per symbol	$\bar{k}$	$[2, 3, 4, 5]$
Encoding rate	$\bar{c}$	$[1/4 - 9/10]$
Roll-off factor	$\bar{\beta}$	$[0.2, 0.3, 0.35]$
Bandwidth	$\overline{\mathrm{BW}}$	$[0.5 - 5]$ MHz
Symbol rate	$\bar{R}_\mathrm{s}$	$[0.41 : 0.1 : 3.7]$ MS/s
Additional Tx E_s/N_0	$\bar{E}_\mathrm{s}$	$[0 : 1 : 10]$ dB

IV. RESULTS

Before presenting any results achieved by the proposed hybrid solution RLNN2, some terms and other additional requisites that are part of the simulation need to be defined.

In order to make the system simulations compliant with the DVB-S2 standard [13] adaptable parameters, such as modulation scheme and encoding rate, were assumed to be used by both space-based transmitter and ground-based receiver radios. Other parameters[3] were also considered to be adaptable, with ranges described in Table I.[4] Following the operational sequence described in Algorithm 1, a multi-dimensional action composed of six parameters $\bar{a} = (R_\mathrm{s}, E_\mathrm{s}, k, \beta, M, c)$ is used by the transmitter over a dynamically changing environment.

For the simulation results presented below, a slow fading communications channel was assumed with the attenuation time-series given in Fig. 3 for clear-sky conditions. This profile is an example of path loss, obtained from the STK orbit simulator [35], during a single low Earth orbit (LEO) satellite pass at an orbit similar to the one flown by the ISS. This represents the environment variations through the natural attenuation experienced by the transmitted signal due to the satellite flight trajectory over the ground station.

Optimum action adaptation is the ultimate goal of the proposed RLNN2 algorithm. Since NN training is not perfect,

[3]All the simulations presented by this paper assume that the satellite transmitter amplifier operates in the close-to-linear region.

[4]Different modulation schemes use different encoding rate sets.

Algorithm 1 RLNN2 Operational Routine

Require: NN1 and NN2 setup, and $\bar{R}_\mathrm{s}$, $\bar{E}_\mathrm{s}$, $\bar{k}$, $\bar{\beta}$, $\bar{M}$, $\bar{c}$, NN_bs, $\mathrm{NN}_\mathrm{dump}$, $f(\varepsilon)$, tr_a, $\mathrm{min}_{\mathrm{good}\%}$

U $\leftarrow$ all combinations of $(\bar{R}_\mathrm{s}, \bar{E}_\mathrm{s}, \bar{k}, \bar{\beta}, \bar{M}, \bar{c})$
loop
  if $(\bar{R}_\mathrm{s}, \bar{E}_\mathrm{s}, \bar{k}, \bar{\beta}, \bar{M}, \bar{c})$ has changed **then**
    U $\leftarrow$ all combinations of $(\bar{R}_\mathrm{s}, \bar{E}_\mathrm{s}, \bar{k}, \bar{\beta}, \bar{M}, \bar{c})$
  end if
  while NN training data buffer not full **do**
    'Forced exploration': only explore
  end while
  z $\leftarrow$ uniform random number $[0, 1]$
  if z $< f(\varepsilon) = \varepsilon_k$ **then** ▷ with prob. ε_k (Explore)
    Predict actions using NN1
    Classify actions into good or bad using $\mathrm{min}_{\mathrm{good}\%}$
    u $\leftarrow$ uniform random number $[0, 1]$
    if u$<$ tr_a **then** ▷ Action rejection
      Randomly select one good action a
    else
      Randomly select one bad action a
    end if
  else ▷ with probability $1 - \varepsilon_k$ (Exploit)
    Predict action to be exploited using NN2
    Execute selected a
    Measure and/or read RL states $\bar{s}$
    Compute multi-objective fitness function $f_\mathrm{obs}(x)$
    Select next $\mathrm{NN2}_\mathrm{input}$ ▷ see Algorithm 2
    Update NN training data buffer
    Build NN1 and NN2 training datasets
    if NN training data buffer is full **then**
      Remove $\mathrm{NN}_\mathrm{dump}$ entries from NN training dataset
    end if
  end if
end loop

NN outputs represent a best effort towards predicting an action as close as possible to the optimum action, given the current available knowledge in the training buffers. These adaptations occur by the communications system selecting different actions during either exploration or exploitation phases of the RL algorithm. Throughout transmissions, the RL agent seeks to optimize the fitness function score composed of conflicting multi-objective functions. The communications mission target is a vector comprised of six metrics: bit error rate (BER), throughput (Thrp), bandwidth (BW), spectral efficiency (Spc_eff), additional consumed power (Pwr_con), and power efficiency (Pwr_eff), with all values scaled to the range of $[0, 1]$. The additional power is assumed to be a variable power amount added to the constant power rated for the worst case condition already used by the satellite.

For proof-of-concept purposes, the multi-objective considered in this work is composed of functions that compute the throughput

$$f_\mathrm{Thrp} = R_\mathrm{s} k c, \tag{4}$$

TABLE II
WEIGHT VALUES FOR COMMUNICATIONS MISSIONS

Mission	w_1	w_2	w_3	w_4	w_5	w_6
1 - Launch/reentry	.2	.4	.1	.1	.1	.1
2 - Multimedia	.5	.3	.05	.05	.05	.05
3 - Power saving	.05	.05	.05	.05	.3	.5
4 - Normal	1/6	1/6	1/6	1/6	1/6	1/6
5 - Cooperation	.05	.05	.4	.4	.05	.05
6 - Emergency	.1	.8	.025	.025	.025	.025

the bandwidth

$$f_{\mathrm{BW}} = R_{\mathrm{s}}(1+\beta), \tag{5}$$

the spectral efficiency

$$f_{\mathrm{Spc\_eff}} = kc/(1+\beta), \tag{6}$$

the power efficiency

$$f_{\mathrm{Pwr\_eff}} = (kc)/(10^{(E_{\mathrm{s}}/N_0)/10}\ R_{\mathrm{s}}), \tag{7}$$

and the additional power consumed is computed by

$$f_{\mathrm{Pwr\_con}} = E_{\mathrm{s}} \cdot R_{\mathrm{s}}. \tag{8}$$

The BER[5] is computed by interpolating the curve functions acquired via simulations of the DVB-S2 communications system in MATLAB [37]. Although MATLAB provides the end-to-end modulators, demodulators, and LDPC and BCH encoders, both 16-APSK and 32-APSK modulation scheme constellations and all other encoding rates had to be added to the simulator. All the frames were assumed to have a length of $64{,}800$ bits, representing the DVB-S2 long-frame type.

The fitness score, *i.e.,* the reward r_k, previously defined by function ρ, is computed by the multi-objective fitness function $f_{\mathrm{obs}}(x)$, given by the weighted sum similar to the approach used in [20], computed by:

$$f_{\mathrm{obs}}(x) = w_1 f_{\mathrm{Thrp}} + w_2 f_{\mathrm{BER}} + w_3 f_{\mathrm{BW}} + w_4 f_{\mathrm{Spc\_eff}} + w_5 f_{\mathrm{Pwr\_eff}} + w_6 f_{\mathrm{Pwr\_con}}, \tag{9}$$

where w_i are the weights for each performance parameter according to the communications mission selected, defined by the user. Examples of mission profiles used in the simulations are given by Table II, and x represents all w_i values and all the other variables that the multi-objective functions depend on.

Algorithm 1 defines the usage of an NN training data buffer, containing a total of $\mathrm{NN_{bs}}$ entries each composed by the action, the RL states s, and the fitness function value, collected during exploration. Whenever this buffer gets filled up, both NN1 and NN2 are re-trained and the oldest $\mathrm{NN_{dump}}$ entries are removed from the buffer.

Before each packet transmission, the RL agent decides between exploration and exploitation by computing ε_k through $f(\varepsilon)$. Then, if a random number z is less than ε_k, the agent virtually explores the environment through NN1, draws another random value u, and checks it against the "action rejection" value tr_a in order to pick an action from either the good or bad sets. These sets are defined by a performance threshold value, defined by the percentage of the maximum fitness score achieved during virtual exploration, defined by $\mathrm{min_{good\%}}$.

[5]Plots of BER curves and MATLAB functions are available in [36].

Algorithm 2 RLNN2 NN Exploitation Input Algorithm

Require: $\mathrm{max\_}f_{\mathrm{obs}} = 0$, m_{reset}, $\mathrm{Exploit_{score}} = [\,]$
while Training data buffer not full **do**
  if 'Forced exploration' **then**
    $\mathrm{Exploit_{score}} = \max(f_{\mathrm{obs}}(x))$
  end if
end while
if $f_{\mathrm{obs}} > \mathrm{max\_}f_{\mathrm{obs}}$ **then**
  $\mathrm{max\_}f_{\mathrm{obs}} = f_{\mathrm{obs}}(x)$
  if z$< \varepsilon_k$ **then** ▷ with probability ε_k (Explore)
    $\mathrm{NN2_{input}} = \bar{s}$
  else ▷ with probability $1-\varepsilon_k$ (Exploit)
    if $f_{\mathrm{obs}}(x) < \mathrm{Exploit_{score}}$ **then**
      if $\mathrm{Exploit_{score}} - f_{\mathrm{obs}}(x) > m_{\mathrm{reset}}$ and $a_k = a_{k-1}$ **then**
        Reset NN training data buffer
        Start 'Forced exploration'
      else if $f_{\mathrm{obs}}(x) < 0.9\ \mathrm{Exploit_{score}}$ **then**
        $\mathrm{NN2_{input}}$ receives $\bar{s}$ from NN training data buffer
      else if $f_{\mathrm{obs}}(x) > 0.9\ \mathrm{Exploit_{score}}$ **then**
        $\mathrm{Exploit_{score}} = f_{\mathrm{obs}}(x)$
      else
        $\mathrm{NN2_{input}} = \mathrm{last\_NN2_{input}}$
      end if
    else
      $\mathrm{Exploit_{score}} = f_{\mathrm{obs}}(x)$
      $\mathrm{last\_NN2_{input}} = \bar{s}$
    end if
  end if
end if

For the simulation results presented below, it was assumed $\mathrm{NN_{bs}} = 200$, $\mathrm{NN_{dump}} = 50$, $\mathrm{tr}_a = 0.95$, and $\mathrm{min_{good\%}} = 0.9$. $f(\varepsilon)$ varies with time step k, assumed to be equal to $\varepsilon_k = 1/k$, starting at 1 and reset every time $\varepsilon_k = 10^{-4}$.

Algorithm 2 defines the logic to select the input for NN2 through two main mechanisms: slow recovery and fast recovery. The former relates to the exploitation of performance values stored in the NN training buffer in a descending order of fitness score values. The latter resets the NN training buffer, which results in retraining of both NNs after enough data are collected while in the forced exploration mode. Since the fast recovery forces continuous exploration for the duration of $\mathrm{NN_{bs}}$ packets, a threshold value m_{reset} controls when it is triggered. Although it is an operator parameter, the results used $m_{\mathrm{reset}} = 0.5$ for all missions.

Simulations were run for the missions presented in Table II. Because of the time duration required by each iteration for each mission (MATLAB simulation runs varied between 4 and

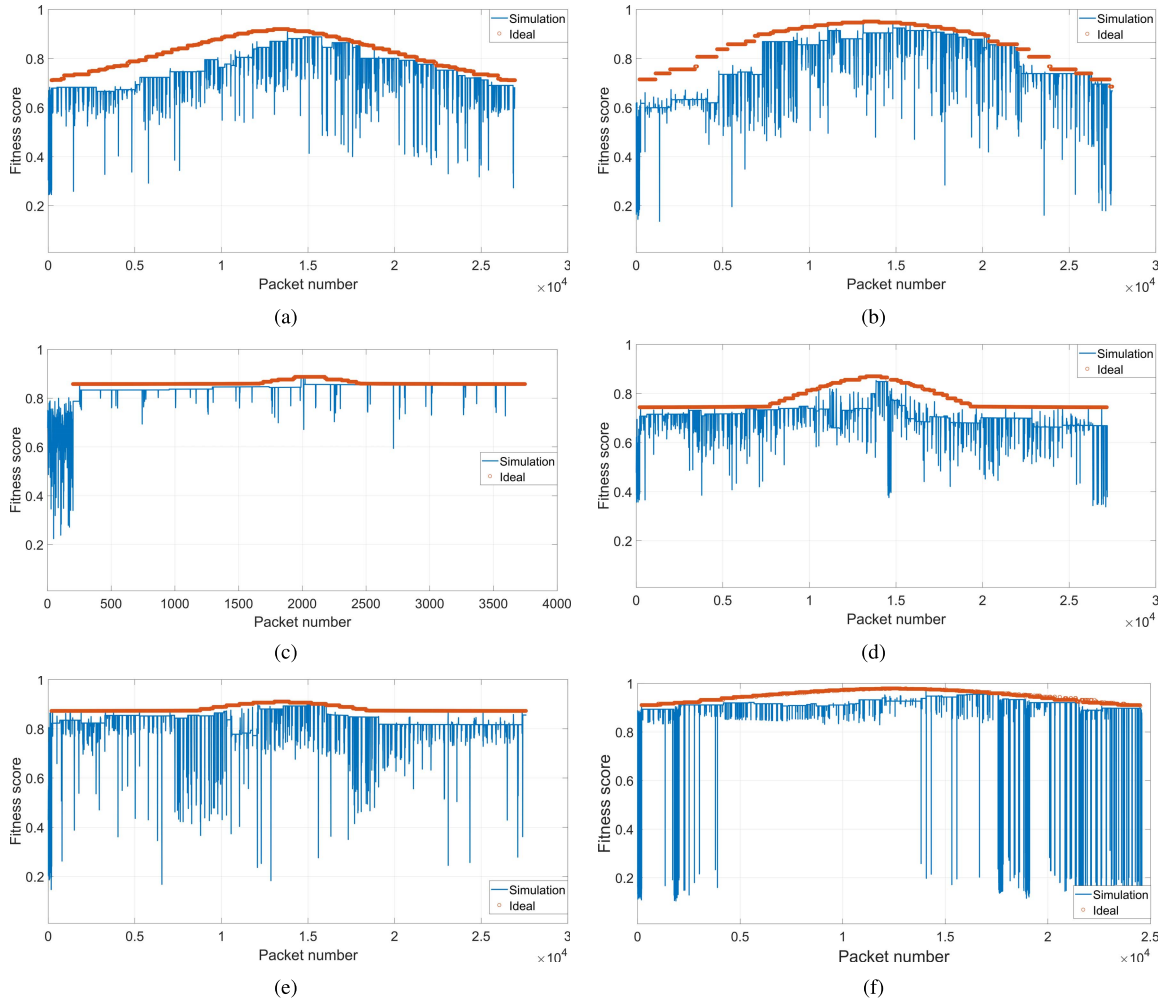

Fig. 7. Examples of fitness score value time series per packet. Panels (a)–(f) show performances for Missions 1 to 6, respectively. In all missions, the majority of simulation packets experienced performances close to the ideal. Constant score levels represent exploitation, while spikes below and above them represent exploration.

8 hours each, a total of 320 simulation hours using two Intel Xeon 16-core 2.3-GHz processors), the results are averaged over 10 runs.

For the interested reader, Fig. 7 provides examples of fitness score time series for each mission, with the duration of a LEO satellite pass over the ground station mentioned in previous sections, *i.e.,* 512 seconds. These time series include performance values during both exploration and exploitation periods.

A. Remarks on RLNN2 Performance

As shown in Fig. 8, panels (a) to (f) present the normalized packet count distribution over fitness score values for Missions 1 to 6, respectively. Even though each mission has a different profile, in all panels the performance is concentrated at high fitness score values. Since the goal is to have the majority of packets experiencing the fitness scores as close as possible to the ideal scores, computed by exhaustive search, evaluating the performance of all actions at each instant, the plots show that the RLNN2 was able to select good actions, learning the relationship between rewards and actions while the channel dynamically changed.

In order to provide a better understanding of the accuracy achieved by the proposed solution, the normalized error is subtracted from one. The normalized error is the difference between the simulation fitness score values of each mission and their respective ideal performance values. The resultant accuracy distribution curves, shown in Fig. 9, are expected to serve as benchmarks for future research on CE for space-based communications systems. For all missions, the RLNN2 algorithm resulted in the majority of packets being concentrated around very high accuracy values, as desired.

Another metric is to compute the area under the distribution curves shown in Figs. 8 and 9, shown in Table III. In terms of performance, the average integrals in Fig. 8 present a minimum equal to 0.72 for Mission 4 and a maximum equal to 0.88 for Mission 6. In terms of accuracy, the integral values for the error distributions computed for Fig. 9 have a minimum equal to 0.01 for Mission 3 and a maximum equal to 0.06 for Mission 2. Thus, because of the values chosen for the fitness score function weights, each mission has its own ideal performance profile, and the error becomes a better metric to assess the efficiency of the proposed solution in choosing good actions during each different channel condition.

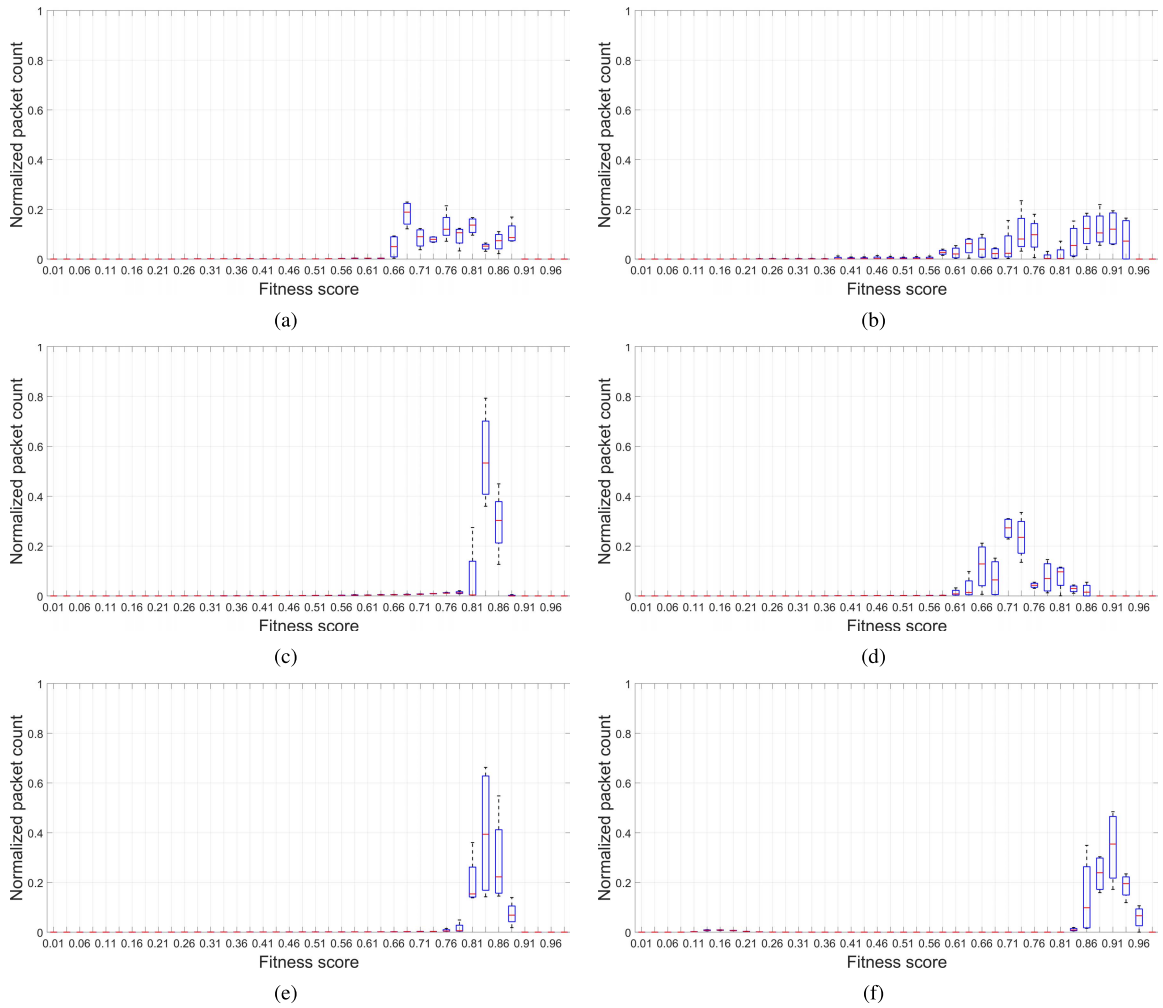

Fig. 8. Boxplots of normalized packet count distribution over fitness score values obtained using RLNN2. Panels (a)–(f) represent performances for Missions 1 to 6, respectively. More packets concentrated around larger score values is better.

TABLE III

INTEGRAL VALUES FOR AVERAGE FITNESS SCORE AND ERROR DISTRIBUTION CURVES

Mission	Integral values			
	$NN1_m = 20$ $NN2_m = 10$		$NN1_m = 1$ $NN2_m = 1$	
	Fitness score	Error	Fitness score	Error
1 - Launch/reentry	0.7605	0.0525	0.7631	0.1278
2 - Multimedia	0.7878	0.0643	0.7643	0.158
3 - Power saving	0.8316	0.0177	0.7894	0.1055
4 - Normal	0.7237	0.0491	0.7049	0.0927
5 - Cooperation	0.8382	0.0366	0.7834	0.0651
6 - Emergency	0.8836	0.0387	0.8931	0.044

Table III also shows the integral values for both NN1 and NN2 ensemble size equal to one, $NN1_m = NN2_m = 1$. This ensemble size choice directly impacts the processing time of both training and prediction for the NN, with the former being the most sensitive for the case being studied. Even though for simulations running on MATLAB the processing time decreased by a factor of 12, this gain in terms of execution time has yet to be defined for implementations using dedicated hardware, such as FPGAs, and is out of the scope of this work. This analysis was aimed at showing the trade-offs involved with accelerating the simulation time versus the algorithm performance errors.

Therefore, for all missions, the integral of the performance distribution is similar when a larger ensemble size is chosen. However, with the exception of Mission 6, the error increased for all other mission performances. Mission 3 experienced an error almost 6 times greater, with all other errors being around 2 or more times greater. In addition, under the same channel conditions and mission profile, the same fitness score value given by Eq. (9) can be achieved by different actions. This leads to different performance values for each parameter in the communications mission target vector, and is the cost of control flexibility by setting multiple parameters while trying to achieve multiple goals at the same time. The potential user is expected to consider this effect when designing a mission profile.

B. RLNN2 Performance Trade-Off

The simulation results in Section IV verify that the proposed solution is compliant with the upper-bound, established by the exhaustive search, since the results never achieve a

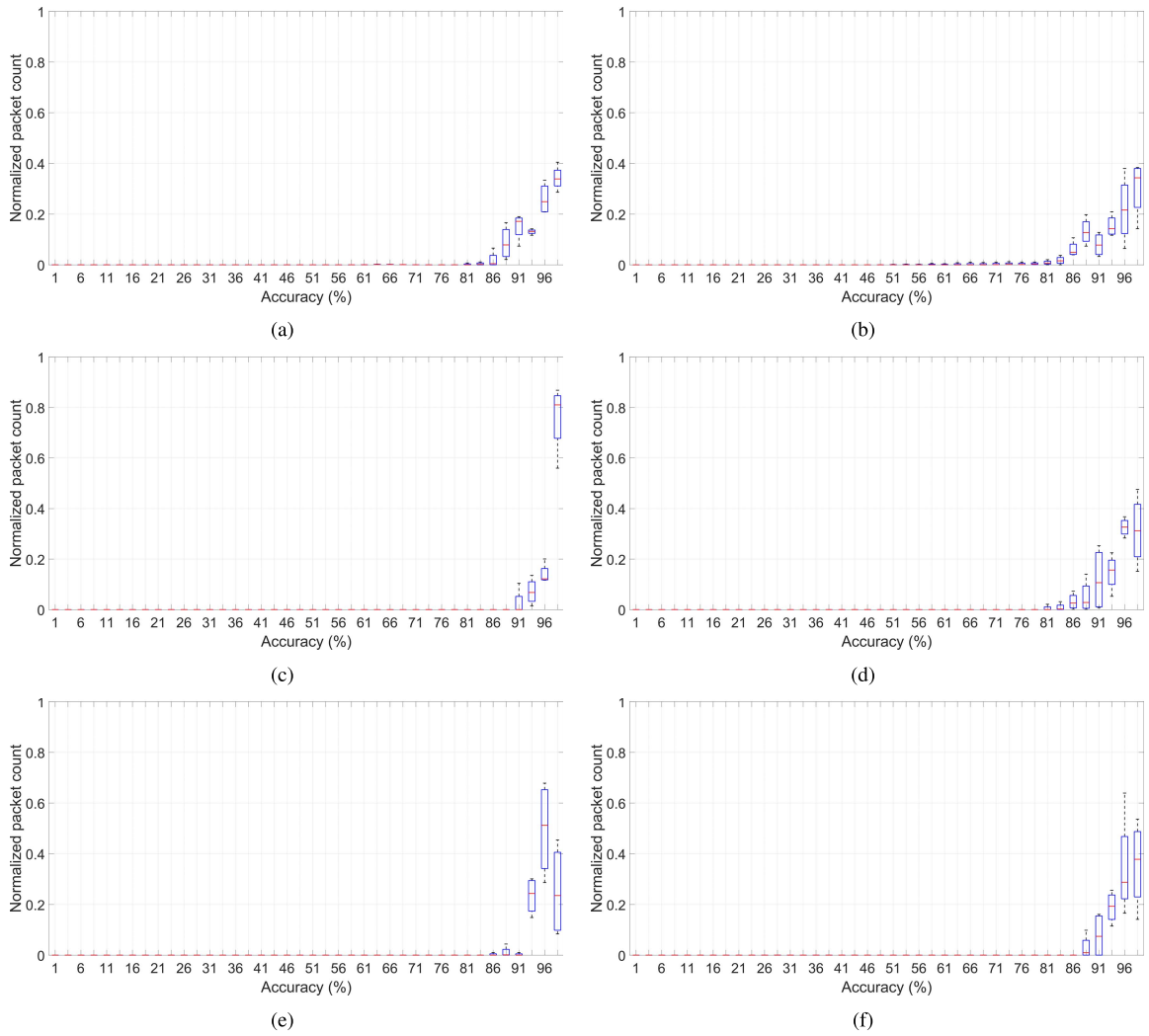

Fig. 9. Boxplots of normalized packet count distribution over accuracy values between proposed RLNN2 and ideal solution. Panels (a)–(f) represent performances for Missions 1 to 6, respectively. More packets concentrated around larger accuracy values is better.

performance higher than the ideal. Besides the limitations that inspired the NN2 proposal, mentioned in Section III, using NNs for choosing actions to be exploited provides flexibility instead of performance improvements. For instance, systems that focus on only one feature adaptation, such as the adaptive protocol used by the DVB-S2 standard are unable to optimize an entire set of radio resources at the same time.

In addition, the NN1 enables a performance control during the RL exploration phase. Systems without NN1 could achieve similar performance if the granularity provided by the state–action mapping is high enough (assuming there is enough memory to store this information, not to mention sufficient memory access time), and if ε_k is also high enough.

However, the cost for training and predicting using NNs provides flexibility in terms of enabling more parameters to be selected at the same time, while relieving the requirements of memory size by allowing the state–action mapping to be performed for continuous action and/or state dimensions, a more realistic approach in the real world. As a byproduct of using NNs for exploration and for exploitation, a fixed memory size learning capability is achieved, since the RL state–action mapping is performed by the NN input–output transfer function with fixed-size weights, resulting in a table-free MORL approach. Although the research presented in this work used the learned knowledge to predict actions, future communication systems could potentially use it to achieve more optimized performance or other tasks.

In addition, the usage of NNs also provides scalability. Future implementations can extend the NN2 design to deal with larger scales of available action parameters, and more state variables monitored, trading off memory size and access time for NN training and prediction processing requirements. This work shows that this trade-off is technically feasible by providing algorithms to implement the proposed solution, and accuracy benchmarks against ideal cases, for future research reference.

It is worth noting here that, to the best of the authors' knowledge, there is no similar approach in the literature related to space communications systems capable of finding multi-dimensional actions to achieve a multi-dimensional objective while the environment changes dynamically. Thus, it would not be fair to compare the proposed algorithms against individual approaches that aim to adapt only individual parameters, such as ACM. Other approaches that operate in

batch mode, as GA, would cause extremely high latency to the system, as compared to the incremental approach of RLNN2. Therefore, the results and accuracy distributions presented here considered the ideal solutions and allow for future research on CE for space communications systems to be compared against.

V. Conclusions

The algorithms provided in this paper constitute the CE core to be used as a baseline for the development of space communication systems in the next generation of spacecraft and satellites. A proof-of-concept solution was proposed to enable multi-objective performance through RL. It relies on exploiting and exploring radio parameter sets using NNs for dynamically changing channels.

The proposed solution consists of an ensemble of deep neural networks for exploration of action parameters that provides performance control during the exploration phase of RL, and an array of ensembles of shallow NNs to predict radio parameters for a certain known performance target. The proposed algorithms enable flexibility of several radio parameters adaptation in a continuous action–state universe, instead of improving the performance of a specific metric, while keeping the memory requirements fixed over time. The overall performance is monitored based on the chosen communications mission profile to be defined by the system operator.

Trade-off analyses were provided regarding the computational cost of the proposed solution, execution time, and its performance accuracy. Simulation results showed that very low performance errors can be achieved when compared to the ideal solutions. An average accuracy higher than 80% was achieved by all the example mission profiles. These results are expected to serve as benchmark for future research on CE for space communications systems.

References

[1] NASA Glenn Research Center. *SCaN Testbed*. Accessed: Jun. 20, 2017. [Online]. Available: http://spaceflightsystems.grc.nasa.gov/SOPO/SCO/SCaNTestbed/

[2] S. K. Johnson, R. C. Reinhart, and T. J. Kacpura, "CoNNeCT's approach for the development of three software defined radios for space application," in *Proc. IEEE Aerosp. Conf.*, Mar. 2012, pp. 1–13.

[3] R. Reinhart, T. J. Kacpura, S. K. Johnson, and J. P. Lux, "NASA's Space Communications and Navigation Test Bed aboard the International Space Station," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 28, no. 4, pp. 4–15, Apr. 2013.

[4] R. C. Reinhart, "Using International Space Station for cognitive system research and technology with space-based reconfigurable software-defined radios," in *Proc. 66th Int. Astron. Congr.*, Jerusalem, Israel, 2015, pp. 1–12.

[5] D. Chelmins, J. Downey, S. K. Johnson, and J. Nappier, "Unique challenges testing SDRs for space," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2013, pp. 1–9.

[6] S. Johnson, D. Chelmins, D. Mortensen, M. Shalkhauser, and R. Reinhart, "Lessons learned in the first year operating software defined radios in space," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Aug. 2014, pp. 1–10.

[7] *NASA Technology Roadmaps*. Accessed: Jun. 20, 2017. [Online]. Available: https://www.nasa.gov/offices/oct/home/roadmaps/index.html

[8] E. Hossain, D. Niyato, and D. I. Kim, "Evolution and future trends of research in cognitive radio: A contemporary survey," *Wiley Wireless Commun. Mobile Comput.*, vol. 15, no. 1, pp. 1530–1564, 2013.

[9] J. F. Kurose and K. W. Ross, *Computer Networking a Top-Down Approach*. London, U.K.: Pearson, 2013.

[10] A. Morello and V. Mignone, "DVB-S2: The second generation standard for satellite broad-band services," *Proc. IEEE*, vol. 94, no. 1, pp. 210–227, Jan. 2006.

[11] D. Tarchi, G. E. Corazza, and A. Vanelli-Coralli, "Adaptive coding and modulation techniques for next generation hand-held mobile satellite communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 4504–4508.

[12] D. Tarchi, G. E. Corazza, and A. Vanelli-Coralli, "A channel state-driven ACM algorithm for mobile satellite communications," *Int. J. Satellite Commun. Netw.*, vol. 34, no. 6, pp. 787–807, 2016.

[13] *DVB-S2 Standard*. Accessed: Jun. 20, 2017. [Online]. Available: https://www.dvb.org/standards/dvb-s2

[14] S. Chatzinotas, B. Ottersten, and R. D. Gaudenzi, *Cooperative and Cognitive Satellite Systems*. San Diego, CA, USA: Academic, 2015.

[15] S. K. Sharma, S. Maleki, S. Chatzinotas, J. Grotz, and B. Ottersten, "Implementation issues of cognitive radio techniques for Ka-band (17.7–19.7 GHz) SatComs," in *Proc. 7th Adv. Satellite Multimedia Syst. Conf. 13th Signal Process. Space Commun. Workshop*, Livorno, Italy, 2014, pp. 241–248.

[16] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 1, pp. 116–130, 1st Quart., 2009.

[17] A. He *et al.*, "A survey of artificial intelligence for cognitive radios," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 1578–1592, May 2010.

[18] N. Abbas, Y. Nasser, and K. El Ahmad, "Recent advances on artificial intelligence and learning techniques in cognitive radio networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2015, pp. 174-1–174-20, Jun. 2015.

[19] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1136–1159, Jul. 2013.

[20] S. Chen, T. R. Newman, J. B. Evans, and A. M. Wyglinski, "Genetic algorithm-based optimization for cognitive radio networks," in *Proc. IEEE Sarnoff Symp.*, Princeton, NJ, USA, Apr. 2010, pp. 1–6.

[21] F. Bernardo, R. Agusti, J. Perez-Romero, and O. Sallent, "Distributed spectrum management based on reinforcement learning," in *Proc. 4th Int. Conf. CROWNCOM*, Hannover, Germany, 2009, pp. 1–6.

[22] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.

[23] P. V. R. Ferreira *et al.*, "Multi-objective reinforcement learning for cognitive radio-based satellite communications," in *Proc. 34th AIAA Int. Commun. Satellite Syst. Conf., Int. Commun. Satellite Syst. Conf. (ICSSC)*, Cleveland, OH, USA, 2016, pp. 1–16.

[24] A. Barto and R. S. Sutton, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1988.

[25] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *J. Artif. Intell. Res.*, vol. 48, no. 1, pp. 67–113, 2013.

[26] P. V. R. Ferreira *et al.*, "Multi-objective reinforcement learning-based deep neural networks for cognitive space communications," in *Proc. IEEE Cognit. Commun. Aerosp. Appl. Workshop*, Cleveland, OH, USA, Jun. 2017, pp. 1–8.

[27] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–541, Feb. 2015.

[28] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 539, no. 7587, pp. 484–489, Jan. 2016.

[29] *Self-Driving Cars*. Accessed: Jun. 20, 2017. [Online]. Available: https://www.theguardian.com/technology/self-driving-cars

[30] Y. B. Ian Goodfellow and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[31] *Neural Networks FAQ*. Accessed: Jun. 20, 2017. [Online]. Available: http://www.faqs.org/faqs/ai-faq/neural-nets/part2/

[32] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton, FL, USA: CRC Press, 2010.

[33] M. T. Hagan and M. B. Menhaj, "Training feed-forward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[34] H. Yu and B. M. Wilamowski, "Levenberg–Marquardt training," in *The Industrial Electronics Handbook*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2012.

[35] *STK, Version 10.1.0*, Analytical Graphs, Inc., Exton, PA, USA, 2014.

[36] P. Ferreira and A. Wyglinski, "Simulated DVB-S2 BER curves for long-frames," San Francisco, CA, USA, GitHub, 2017, doi: 10.5281/zenodo.495732.

[37] *MATLAB, Version 9.2.0.538062 (R2017a)*, The MathWorks Inc., Natick, MA, USA, 2017.

Paulo Victor Rodrigues Ferreira received his Ph.D. degree in 2017 from Worcester Polytechnic Institute and his B.S. and M.S. degrees in 2010 and 2012 from Universidade Federal de Uberlandia, Brazil in Electrical Engineering with emphasis on Telecommunications and Electronics. Paulo Victor was awarded a Ph.D. scholarship from the Brazilian government program Science without Borders. He has worked as a Graduate Assistant at the Wireless Innovation Laboratory, within the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute, Worcester, MA, USA, and as an R&D intern at General Electric Global Research Headquarters. His interests include satellite communications, machine learning, cognitive radio, and space weather.

Randy Paffenroth graduated from Boston University in 1992 with degrees in both mathematics and computer science and he was awarded his Ph.D. in Applied Mathematics from the University of Maryland in June of 1999. After attaining his Ph.D., Dr. Paffenroth spent seven years as a Staff Scientist in Applied and Computational Mathematics at the California Institute of Technology. In 2006, he joined Numerica Corporation where he held the position of Computational Scientist and Program Director. Dr. Paffenroth is currently an Associate Professor of Mathematical Sciences and Associate Professor of Computer Science at Worcester Polytechnic Institute where his focus is on the WPI Data Science Program. His current technical interests include machine learning, signal processing, large scale data analytics, compressed sensing, and the interaction between mathematics, computer science, and software engineering, with a focus on applications in cyber-defense.

Alexander M. Wyglinski (SM'11) is a Professor of Electrical and Computer Engineering at Worcester Polytechnic Institute, Worcester, MA, USA and Director of the Wireless Innovation Laboratory. Dr. Wyglinski received his B.Eng. and Ph.D. degrees in 1999 and 2005 from McGill University, and his M.Sc.(Eng.) degree from Queen's University in Kingston in 2000, all in Electrical Engineering. During his academic career, Dr. Wyglinski has published over 40 journal papers, over 80 conference papers, 9 book chapters, and two textbooks. Dr. Wyglinsk's current research activities include wireless communications, cognitive radio, software-defined radio, dynamic spectrum access, spectrum measurement and characterization, electromagnetic security, wireless system optimization and adaptation, and cyber-physical systems. He is currently being or has been sponsored by organizations such as the Defense Advanced Research Projects Agency (DARPA), the Naval Research Laboratory (NRL), the Office of Naval Research (ONR), the Air Force Research Laboratory (AFRL) - Space Vehicles Directorate, The MathWorks, Toyota InfoTechnology Center U.S.A., Raytheon, the MITRE Corporation, National Aeronautics and Space Administration (NASA) and the National Science Foundation (NSF). Dr. Wyglinski is a Senior Member of the IEEE, as well as a member of Sigma Xi, Eta Kappa Nu, and the ASEE. Furthermore, Dr. Wyglinski is currently the President of the IEEE Vehicular Technology Society.

Timothy M. Hackett is a Ph.D. candidate in the Systems Design Laboratory (SDL), within the School of Electrical Engineering and Computer Science at The Pennsylvania State University in University Park, PA, USA. Timothy received his B.S. (student marshal) and M.S. degrees in 2015 and 2017 from Penn State in Electrical Engineering with an emphasis on communications and signal processing. He was awarded a NASA Space Technology Research Fellowship to fund his M.S. and Ph.D. work. He has worked as an intern at General Electric and The Boeing Company and as a graduate fellow at NASA Glenn Research Center and NASA's Jet Propulsion Laboratory. His research interests include satellite communications, schedule optimization, and machine learning.

Sven G. Bilén (S'90–M'98–SM'08) received his B.S. degree from Penn State in 1991, and M.S.E. and Ph.D. degrees from the The University of Michigan in 1993 and 1998, respectively. He is a professor of engineering design, electrical engineering, and aerospace engineering at Penn State, and head of the School of Engineering Design, Technology, and Professional Programs. His research interests include software-defined radio techniques and systems, cognitive radio, and wireless sensor systems.

Richard C. Reinhart is a senior communications engineer with NASA Glenn Research Center, located in Cleveland, OH. He is the Principal Investigator for NASA's software-defined and cognitive-radio flight experiment aboard International Space Station. He is a principal architect to define NASA's future communications relay satellite and ground station architecture. He has worked with space communications technology for over 25 years on various satellite, radio, and array antenna technologies. He received his B.S. and M.S. in Electrical Engineering from The University of Toledo and Cleveland State University, respectively. Mr. Reinhart has published a number of technical papers and conference presentations associated with the SCaN Testbed, the application of cognitive technologies to NASA, SDR technology, and the Ka-band Advanced Communications Technology Satellite (ACTS). He is a principal author of the SDR Space Telecommunications Radio System (STRS) architecture, now a NASA-wide standard.

Dale J. Mortensen has worked as an electronics engineer at the NASA Glenn Research Center since 1990. He earned his B.S. and M.S. from The Ohio State University and Case Western Reserve University in 1990 and 1995, respectively. The majority of his work at NASA has supported space communications research and flight projects in the areas of free-space laser systems, high-speed digital modems, software-defined radio, and cognitive communication systems. Dale is a co-author of the NASA's Space Telecommunications Radio System (STRS) architecture standard, and now serves as a systems lead for the Cognitive Communications Project.