

MADS: A Framework for Design and Implementation of Adaptive Digital Predistortion Systems

Lin Li*, Peter Deaville[†], *Student Member, IEEE*, Adrian Sapio*, Lauri Anttila[‡], *Member, IEEE*,
Mikko Valkama[‡], *Senior Member, IEEE*,

Marilyn Wolf[§], *Fellow, IEEE*, Shuvra S. Bhattacharyya*, *Fellow, IEEE*

*University of Maryland, College Park, ECE Department, College Park, MD 20742, USA

Email: {lli12311, asapio, ssb}@umd.edu

[†]Princeton University, Electrical Engineering, Princeton, NJ 08544, USA

Email: deaville@princeton.edu

[‡]Tampere University, Electronics and Communications Engineering, Finland

Email: {lauri.anttila, mikko.valkama}@tuni.fi

[§] University of Nebraska-Lincoln, Lincoln, USA

Email: mwolf@unl.edu

Abstract—Digital predistortion (DPD) has important applications in wireless communication for smart systems, such as, for example, in Internet of Things (IoT) applications for smart cities. DPD is used in wireless communication transmitters to counteract distortions that arise from nonlinearities, such as those related to amplifier characteristics and local oscillator leakage. In this paper, we propose an algorithm-architecture-integrated framework for design and implementation of adaptive DPD systems. The proposed framework provides energy-efficient, real-time DPD performance, and enables efficient reconfiguration of DPD architectures so that communication can be dynamically optimized based on time-varying communication requirements. Our adaptive DPD design framework applies Markov Decision Processes (MDPs) in novel ways to generate optimized runtime control policies for DPD systems. We present a GPU-based adaptive DPD system that is derived using our design framework, and demonstrate its efficiency through extensive experiments.

Keywords—Smart systems, dataflow modeling, digital predistortion, Markov decision processes.

I. INTRODUCTION

Smart systems are capable of sensing the environment and making decisions based on the available data in an adaptive manner. To gather useful information about the environment from sensors in real-time, smart systems may make use of Internet of Things (IoT) technology. The sensory information acquired by IoT devices can be transmitted to a base station for aggregation and analysis to make decisions.

IoT devices are often equipped with wireless interfaces. However, the quality of the wireless communication in smart systems can suffer from non-idealities. In particular, non-linearity of the power amplifier (PA) is a notorious source of signal distortion. To address this issue, digital predistortion (DPD) can be utilized to counteract PA non-linearities [1]. The implementation of an adaptive DPD system involves a complex optimization problem that affects the wireless communications quality, energy efficiency, and real-time performance of the associated devices.

Due to changes in the environment such as temperature and voltage, PA characteristics in general vary at run-time. Thus, for most efficient operation, a DPD system should be able to change the predistortion coefficients dynamically to ensure that its underlying model matches accurately with its operating environment. Moreover, for integration in smart systems with many types of connected devices and communication modes, it is critical for DPD systems to support efficient predistortion across time-varying operational requirements and modulation schemes. Thus, dynamic system control and reconfiguration are desirable compared to static design for DPD systems.

In this paper, we develop a general framework for deploying DPD implementations that address the need for efficient adaptivity in wireless communications devices. Implementations that are deployed using this framework are designed to autonomously make run-time decisions on DPD system configurations based on the current system state and operational state. The reconfiguration is driven by policies for dynamic system management that are derived at design time using Markov decision processes (MDPs) [2]. The key novelty of this paper is the development of a general framework for MDP-based design and implementation of adaptive DPD systems.

We refer to the proposed framework as the *MDP framework for Adaptive DPD Systems (MADS)*. Dynamic reconfiguration in MADS is performed with the objective of jointly optimizing Adjacent Channel Power Ratio (ACPR), system throughput, and power efficiency. ACPR is often viewed as the most critical metric for assessing the quality of DPD systems. ACPR is defined as the ratio of the mean power centered on the adjacent channels to the mean power centered on the desired channel.

The MADS framework consists of two parts: (1) an MDP subsystem that utilizes MDP methods hierarchically to derive complex system configuration parameters that optimize specified objectives at run-time; and (2) a dataflow-based approach for DPD implementation, which facilitates efficient system

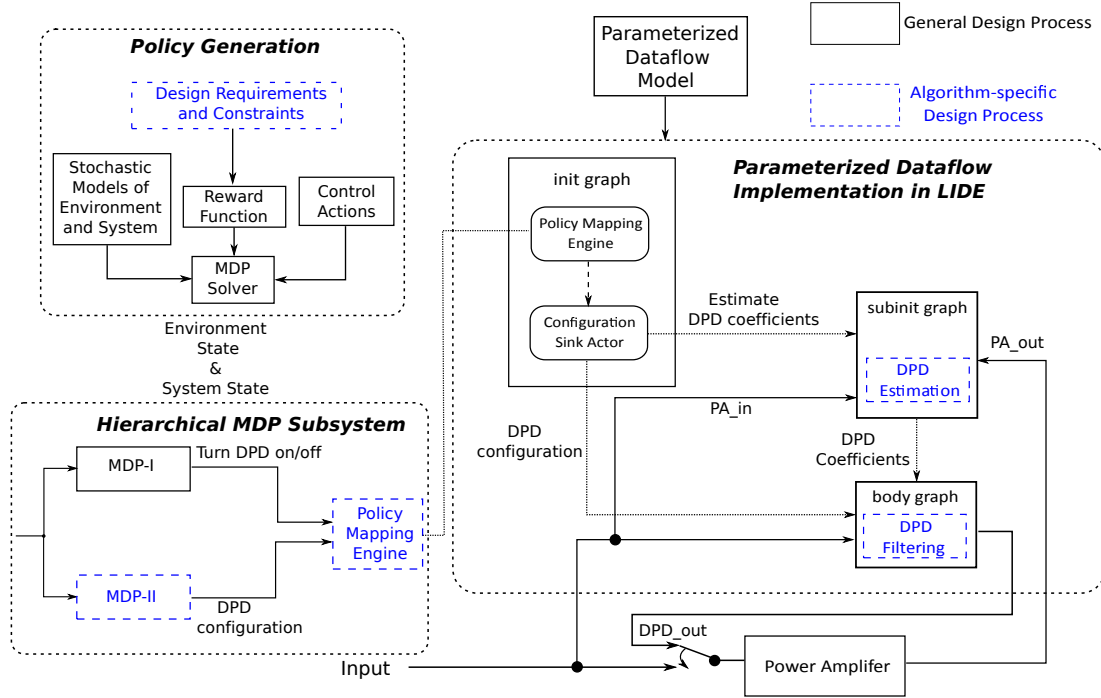


Fig. 1: An illustration of the MADS Framework.

reconfiguration. The hierarchical MDP approach in MADS consists of two cooperating MDPs — a top-level MDP, and a lower-level MDP that is controlled by the top-level one. The top-level MDP is general (can be used across different underlying DPD algorithms), and the lower-level MDP is application-specific — that is, it is customized to the specific DPD algorithm that the given MADS-based architecture applied to. Through this hierarchical MDP approach, the MADS framework systematically extends the given DPD algorithm with capabilities for efficient run-time reconfiguration.

To demonstrate the MADS Framework, we design an adaptive version of a state-of-the-art DPD algorithm from the literature — the DPD algorithm presented by Anttila in [1], which is shown to be more effective than the DPD architectures proposed in [3] and [4]. In particular, we apply the MADS Framework to develop an adaptive architecture that dynamically selects strategic configurations from the design space defined by Anttila’s algorithm. We develop a hybrid CPU/GPU implementation of this adaptive architecture, and demonstrate its efficiency through extensive experiments.

This remainder of this paper is organized as follows. Section II reviews related work from the literature on multi-objective DPD systems. Section III presents the proposed MADS framework along with a brief introduction to the dataflow tools that are employed in our work to prototype framework. Section IV develops formulations of multi-objective optimization in MADS, and the general, top level of the hierarchical MDP that is employed. Section V demonstrates the MADS framework by applying it to Anttila’s algorithm, as described above. The lower-level MDP for this demonstration system is presented, along with experimental results that evaluate the performance of the system. Finally, Section VI draws conclusions and summarizes directions for

future work.

II. RELATED WORK

Optimization problems for DPD systems have been widely studied over the years. For example, the work in [5]–[8] applies genetic algorithms to optimize DPD filter coefficients while assuming fixed filter and polynomial orders. In [9], both filter coefficients and polynomial orders are jointly optimized via particle swarm optimization; however, this optimization is performed with respect to only a single objective — ACPR. Ghazi et al. propose a data-parallel implementation of reconfigurable DPD on a mobile Graphics Processing Unit (GPU) [10]. The implementation allows the DPD parameters to fit various transmission scenarios by selecting a set of candidate profiles based on desired linearization performance. However, this work does not provide any control policy for optimized run-time reconfiguration. In [11], Wang et al. utilize a hill-climbing algorithm to search for an effective DPD configuration. Wang’s approach jointly considers two objectives — accuracy and the number of DPD coefficients. In [12], Pareto-optimized DPD parameters are derived subject to multi-dimensional constraints to support dynamically reconfigurable DPD systems that are adaptive to changes in the employed modulation schemes and operational constraints. However, this work does not take into account reconfiguration costs nor statistics of the environment and system states.

In comparison to the related work referenced above, our contribution in this paper is novel in its development of a general framework for integrating dynamic reconfiguration systematically into a broad class of DPD algorithms. To the best of our knowledge, the proposed framework is the first that integrates MDP algorithms for the derivation of dynamic DPD system parameters, and optimizes multiple objectives

including ACPR, power consumption and throughput. Also, this work

A preliminary version of this paper was presented at AICAS 2019 [13]. This new version of the paper goes beyond this preliminary version in the following ways. First, we elaborate on the methods to obtain transmit powers from nearby devices and we demonstrate the method to compute the transmit power transition matrix, which is a key step in the proposed MADS framework. Second, we tabulate detailed simulation and measurement results, including results on ACPR, power consumption, and throughput. The results are provided both from simulations as well as from performance measurements on a GPU-based implementation. Moreover, we report on several new experiments to demonstrate the flexibility provided to the designer by the MADS Framework in configuring trade-offs that are important for DPD system operation. Finally, we present new experiments that evaluate the performance of MADS when the environment changes as well as when the reward function changes. We also show results for the use case where a stringent ACPR performance is enforced, which is common in wireless communication standards. These experiments provide further insight into the utility of the reconfiguration capabilities provided by MADS.

III. MDP FRAMEWORK FOR ADAPTIVE DPD SYSTEMS

The MADS Framework is illustrated in Fig. 1. The framework is designed so that many kinds of DPD algorithms can be plugged in to generate MDP-integrated, adaptive systems that are based on those algorithms. When the MADS Framework is applied to a DPD algorithm X , we refer to X as the *base algorithm*, and the adaptive system produced by the MADS Framework is referred to as MADS- X . The base algorithm is assumed to have two stages: (1) an estimation stage, where the DPD coefficients are estimated according to the input and output signals of the PA, and (2) a filtering stage, where the input signal is filtered based on the coefficients estimated from the estimation stage.

In Fig. 1, boxes with black-colored borders represent general design processes that are involved in applying the MADS Framework, while boxes with blue-colored borders represent design processes that are specific to the base algorithm.

MADS consists of three major components: the policy generation subsystem, hierarchical MDP subsystem, and parameterized dataflow subsystem. The policy generation subsystem provides policies for the hierarchical MDP subsystem, while the hierarchical MDP subsystem captures the environmental and system states and maps these states into actions based on the policies. These actions in turn are used to modify dataflow graph parameters within the parameterized dataflow subsystem. In particular, the parameterized dataflow subsystem adapts DPD parameters based on actions that are produced from the hierarchical MDP subsystem. Details of the three components are elaborated on below.

The block in Fig. 1 labeled Policy Generation, which corresponds to the policy generation subsystem described above, illustrates the application of an MDP solver that is used to derive reconfiguration policies from MDP-I and MDP-II. As with the base algorithm, the MADS Framework is

not specialized to any specific MDP solver. In our current implementation of the framework, we use the MDPSOLVE solver [14]. The output of the Policy Generation block is the derived policy, which maps states to actions for MDP-I and MDP-II respectively. The policy generation subsystem is executed offline.

Based on both general features of DPD applications and architecture-specific system behaviors, the MADS Framework models environmental and system dynamics in the form of *hierarchical MDPs* [15] which is shown to be efficient for solving the multi-objective optimization problem [16]. This modeling approach is illustrated in the block in Fig. 1 that is labeled Hierarchical MDP Subsystem. The Hierarchical MDP Subsystem consists of two smaller MDPs, MDP-I and MDP-II. MDP-I generates a policy that determines when to turn the DPD system on and off, while MDP-II determines key DPD parameter configurations that should be used at a given time when the DPD is on. MDP-I may turn predistortion off, for example, if due to current channel conditions or quality of service requirements, predistortion is expected to not be needed for some significant amount of time. Using the policies produced in the Policy Generation block, the MDP subsystem determines the optimal action on-the-fly given the environmental and system state, which are encapsulated in the policy mapping engine.

The key aspect of the policy mapping engine is to provide a table lookup, which is similar to many works in literature in which tables that store the optimal DPD configuration are generated. The main difference between our work and conventional approaches to DPD adaptation is in the methods used for determining the filter configurations for different states, and for formulating the state space itself: conventional works only consider short-term DPD performance while this work takes wireless environment changes into consideration and considers long-term rewards.

DPD parameters that can be configured by MDP-II include the polynomial orders, filter orders, and filter coefficients. The exact set of parameters that MDP-II optimizes in general differs between different choices of the base algorithm. Thus, when applying the MADS design methodology, MDP-II should be formulated specifically for the given base algorithm.

As shown in Fig. 1, a reconfigurable DPD application system developed in MADS is modeled as a *parameterized dataflow* graph. Parameterized dataflow is a graph-based form of model-based design that is well-suited to design and implementation of signal processing systems that have dynamic parameters [17]. To implement the dataflow graph, we apply the *lightweight dataflow environment (LIDE)*, which is a design tool for dataflow-based design and implementation of signal processing systems [18]. LIDE provides a compact set of application programming interfaces (APIs) for implementing signal processing applications as dataflow graphs. A useful feature of LIDE is that it facilitates the retargeting of designs to different implementation languages, such as C, CUDA, and Verilog/VHDL, and different platforms [19]. MADS inherits this useful feature of retargetability from LIDE.

A parameterized dataflow specification consists of three distinct graphs — the *init*, *subinit*, and *body* graphs. Intuitively,

the init and subinit graphs are used to compute parameter updates for the body graph. The init graph can also modify parameters of the subinit graph. In the remainder of this section, we describe how these component graphs are applied in the MADS Framework; for general definitions on these and other parameterized dataflow concepts, we refer the reader to [17].

In our LIDE-based implementation of MADS, the init graph computes system hyperparameters that affect the overall DPD system architecture (i.e., the DPD specific subsystems that are used and their interconnections). We refer to these hyperparameters as *architecture configuration parameters*. The parameterized dataflow runtime system in LIDE propagates the parameter updates computed by the init graph to the subinit and body graphs.

In contrast to the init graph, the subinit graph computes system parameters that are used to configure a given DPD architecture. We refer to these architecture-specific parameters as *DPD coefficients* since they are constrained to being values associated with digital filter coefficients. The subinit graph encapsulates the base-algorithm-specific DPD estimation subsystem as a main component. The estimation subsystem is used to estimate new values for filter coefficients that are to be used in subsequent executions of the body graph. It manifests itself as a training phase that uses the indirect learning method presented in [1]. On the other hand, the body graph encapsulates the set of available DPD architectures, and performs signal predistortion based on the most-recently selected architecture (selected by the init graph) and its most-recently configured coefficients (from the subinit graph). Execution control changes iteratively among the init, subinit, and body graphs based on coordination rules that are defined as part of parameterized dataflow semantics [17].

The parameterized dataflow graph takes the policy generated from the hierarchical MDP subsystem as input and executes the given actions in real-time to update relevant dataflow graph parameters.

IV. ACTIVATION CONTROL

In this section, we present the formulation of MDP-I. This MDP is designed in a general way to operate with arbitrary base algorithms. In general, the formulation of an MDP requires specification of four key components: the state space (SS), action space (AS), state transition matrix (STM), and reward function (RF). In the remainder of this section, we describe the formulation of these components for MDP-I. For general background on MDPs, including the role of their four general components, we refer the reader to [2].

SS: The SS for MDP-I is represented as: $s_1 = (T_x, on)$, where T_x is the current transmission power level, and on is a binary value representing whether the system is turned on (1) or off (0). The subscript 1 is used to indicate correspondence with MDP-I. We quantize the continuous values of transmission power to obtain a discrete SS.

AS: The AS consists of two actions: one to turn on (activate) the DPD system and the other to turn it off. We denote the action by a_1 .

STM: We define two STMs, which specify the state transition probabilities for each of the two actions. The definition of

these STMs exploits two properties: (1) T_x is independent of the action and the other state variable, and (2) the DPD system is fully under the control of the action, so the system transitions to the on/off state as requested by the action with probability 1. The STMs are defined by:

$$\begin{aligned} P(s_1(t+1) = (j, y) | s_1(t) = (i, x), a_1) \\ = P(T_x(t+1) = j, on(t+1) = y | T_x(t) = i, on(t) = x, a_1) \\ = P(T_x(t+1) = j | T_x(t) = i) P(on(t+1) = y | a_1), \end{aligned} \quad (1)$$

where $P(on(t+1) = y | a_1)$ is 1 if a_1 is to turn on the DPD and 0 otherwise.

RF: We formulate the RF R_1 as a linear combination of four competing metrics: the averaged DPD power consumption M_P , ACPR M_A , switching cost M_S , and throughput M_T , incurred by turning on the DPD system from an off status.

$$R_1(s_1(t), a_1) = c_1 M_P(s_1(t)) + c_2 M_A(s_1(t)) + c_3 M_S(a_1, s_1(t)) + c_4 M_T(s_1(t)), \quad (2)$$

where c_1, c_2, c_3 and c_4 are the weights of the optimization objectives. Determination of these weights is a design issue that influences operational trade-offs of the DPD system. The values of c_1, c_2 , and c_3 should be negative and c_4 should be positive since the MDP is formulated to *maximize* the reward function. For brevity, we use $\mathbf{c} = [c_1, c_2, c_3, c_4]$ to represent the weight vector. Also, we impose a constraint on \mathbf{c} such that $\sum_{i=1}^4 |c_i| = 1$.

V. DEMONSTRATION AND EXPERIMENTS

In this section, we demonstrate the MADS Framework by applying it to Anttila's algorithm [1] as the base algorithm. We refer to the integration of MADS with Anttila's algorithm as *MADS-A*.

In Anttila's algorithm, the DPD system takes the form of the Hammerstein architecture [20] and is split into two parts (*branches*): direct and conjugate predistortions. We denote the maximum orders of the polynomials for the direct and conjugate branches by p and q , respectively. The parameters p and q influence trade-offs among throughput, ACPR, and power consumption. Strategic control of these parameters is the target of MDP-II in MADS-A. In Anttila's algorithm, only odd-order polynomials are used. Thus, the sets of polynomial orders are $I_p = \{1, 3, 5, \dots, p\}$ for the direct branches and $I_q = \{1, 3, 5, \dots, q\}$ for the conjugate branches. For more details about Anttila's algorithm, we refer the reader to [1].

A. MDP-II Formulation

As discussed in Section III, the MDP-II component of MADS is application-specific in that it needs to be specialized to the base algorithm. In this section, we present our MDP formulation for MDP-II in MADS-A. In MADS-A, the base algorithm parameters that are controlled by MDP-II are the polynomial orders p and q . The components of MDP-II are summarized as follows.

SS: The SS consists of the current transmission power level (as in MDP-I) and the deployed DPD configuration (p - q combination), where $p \in \{1, 3, 5, 7, 9\}$ and $q \in \{1, 3, 5, 7, 9\}$. In MDP-II, we represent the state as $s_2 = (T_x, p, q)$.

AS: An action in MDP-II corresponds to determining the p - q combination for the next MDP time step. Thus, the AS can be represented as the set of all possible p - q combinations, and the AS contains $5 \times 5 = 25$ elements. We denote the AS by a_2 . As discussed in Section III, the DPD of the proposed system has a learning phase when MDP-II decides that a new (p, q) configuration should be used. In this case, the system would calculate DPD coefficients for the new (p, q) setting using the indirect learning approach presented in [1].

STM: We define 25 STMs corresponding to the 25 actions in MDP-II. The state transitions for transmission power are controlled by MDP-I, and are independent from both the action and the system configuration. Similar to our development of MDP-I, we assume that given a particular action, there is a deterministic transition to the target configuration. The STMs can be expressed in a form similar to that of Equation 1. We omit the details due to space limitations.

RF: Similar to MDP-I, the reward function is a linear combination of M_P , M_A , M_T , and a switching cost M_S . The metrics M_P , M_A , M_T are the same as in MDP-I and have the same weighting coefficients. However, the switching cost M_S for MDP-II is different. For MDP-II, M_S refers to the cost of reconfiguration from the p - q combination in the current time step to the combination to be used in the next time step (based on a_2). The weight of M_S for MDP-II is generally determined separately from the corresponding weight for MDP-I. In this paper, we use the same weight for M_S to simplify the simulation and analysis.

B. Experiment

We implemented the MADS-A system using LIDE on a hybrid CPU/GPU platform composed of an Intel i7-2600K (CPU) running at 3.4 GHz, and an NVIDIA GeForce GTX 1080 (GPU). The body graph (see Fig. 1) of MADS-A is mapped to the GPU and the subunit and init graphs are mapped to the CPU. The system throughput and power consumption data that we collected to calibrate the reward functions for the MDP formulations in MADS-A are based on the NVIDIA GeForce GTX 1080.

In Fig. 2, we present the flowchart for the simulations. First of all, since the transition matrix of the transmission power is a crucial component for the calculation of STMs, we collect WiFi packets in two different environment to derive the transmit power transition matrix (TPTM). To calculate the reward functions defined in (2), we collect ACPR metrics via MATLAB simulations and throughput, power consumption, switching cost using CPU/GPU testbed under different states. These metrics are fed into the MDP solver Together with the derived STM which leads to optimal policies under different states. Finally, we run simulations to obtain the average rewards using the policies generated by the MDP solver. Details of each step are elaborated below.

1) Measurement of Transmit Power Transition Matrix:

To obtain TPTM, we conduct experiments within the main building that houses the Department of Electrical and Computer Engineering at the University of Maryland, College Park. After that, we conduct experiments in an office room of the Kim Building at the same campus. In our context, the main

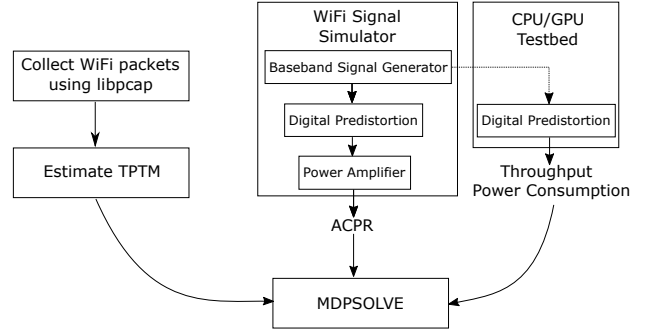


Fig. 2: Experimental setup.

TABLE I: Measured metrics for different DPD configurations.

p, q	P_{TX}					Throughput (Mps)	Power Con. (mW)
	L_1	L_2	L_3	L_4	L_5		
1,1	71.57	69.87	64.83	59.84	49.20	6130.5893	67
1,3	71.53	70.00	64.79	59.78	49.14	4083.2399	89
1,5	71.54	69.90	64.74	59.78	49.05	3076.8075	108
1,7	71.52	69.92	64.73	59.77	48.98	2348.9606	103
1,9	71.43	69.91	64.74	59.75	48.92	1888.6455	140
3,1	71.71	71.52	71.66	67.48	52.00	4103.6944	85
3,3	71.64	71.85	71.70	67.54	51.99	3081.1472	89
3,5	71.70	71.75	71.58	67.55	51.89	2359.1073	145
3,7	71.69	71.76	71.53	67.52	51.87	1901.2475	135
3,9	71.74	71.75	71.66	67.46	51.85	1572.3608	179
5,1	71.80	71.66	71.62	71.61	69.49	3091.3208	84
5,3	71.71	71.70	71.82	71.71	70.23	2363.3610	110
5,5	71.68	71.71	71.68	71.73	71.52	1904.5626	145
5,7	71.76	71.86	71.84	71.80	71.44	1581.8489	179
5,9	71.76	71.74	71.62	71.67	71.65	1363.3451	148
7,1	71.68	71.67	71.58	71.65	69.53	2374.4928	113
7,3	71.68	71.79	71.72	71.74	70.31	1905.6703	136
7,5	71.79	71.75	71.65	71.81	71.63	1582.9952	148
7,7	71.76	71.75	71.85	71.77	71.57	1371.0460	150
7,9	71.79	71.87	71.83	71.79	71.57	1173.8492	174
9,1	71.70	71.73	71.81	71.55	69.59	1901.2475	150
9,3	71.78	71.78	71.72	71.86	70.49	1582.9952	150
9,5	71.69	71.83	71.57	71.82	71.87	1365.6178	177
9,7	71.78	71.65	71.83	71.74	71.62	1175.5336	180
9,9	71.65	71.60	71.72	71.67	71.73	1019.2224	175

difference between these two buildings is that the wireless environment in the former is more complicated due to the presence of more wireless devices. These two buildings are selected as representative wireless environments within which the experimental system under investigation could operate.

For the experiments in each building, we place a laptop computer with WiFi capability in the building. The *libpcap* utility, a commonly used software utility that can monitor WiFi packets over the air, is used on the laptop. The WiFi card on the laptop works at 2.4 GHz with a bandwidth of 20 MHz. Each measurement is taken for 10 minutes. The transmit power from the laptop to the wireless device is extracted, which leads to a time series of transmit powers. To reduce the dimension of the TPTM, we classify the transmit powers into five buckets L_1, L_2, \dots, L_5 , with ranges for the buckets given as:

- L_1 : < 5 dBm
- L_2 : $[5 \sim 10)$ dBm
- L_3 : $[10 \sim 15)$ dBm
- L_4 : $[15 \sim 20)$ dBm
- L_5 : ≥ 20 dBm

We refer to L_1 and L_5 , respectively, as the low and high transmit power ranges. An example consisting of 10 bucketized transmit powers is shown as follows:

$$L_5, L_2, L_3, L_1, L_5, L_3, L_2, L_1, L_4, L_4. \quad (3)$$

The transition probability $P_{T_x}(L_j|L_i)$ can be computed as follows:

$$P_{T_x}(L_i|L_j) = \frac{\# \text{ of times when } T_x[n+1] = L_i \text{ and } T_x[n] = L_j}{\# \text{ of times when } T_x[n] = L_j}, \quad (4)$$

where $T_x[n]$ represents the n th element in the sequence of sampled transmit power levels. For the example given in (3), we can compute the following transmit power transition matrix $\mathbf{P}_{T_x}^{\text{Example}}$, where the (i, j) th entry represents $P_{T_x}(L_i|L_j)$.

$$\mathbf{P}_{T_x}^{\text{Example}} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

In (6) and (7), we show the TPTMs that resulted from our measurements for the university and office environments, respectively. Clearly, the TPTMs are very different in these two environment in that the transmit power for the office environment tends to be lower while the transmit power for the university environment has a higher probability to stay in the high transmit power regime.

$$\mathbf{P}_{T_x}^{\text{University}} = \begin{bmatrix} 0.17 & 0 & 0 & 0 & 0 \\ 0.17 & 0 & 0.02 & 0.06 & 0.05 \\ 0.17 & 0 & 0.02 & 0.02 & 0.04 \\ 0.50 & 0.53 & 0.52 & 0.62 & 0.19 \\ 0.00 & 0.47 & 0.44 & 0.29 & 0.71 \end{bmatrix} \quad (6)$$

$$\mathbf{P}_{T_x}^{\text{Office}} = \begin{bmatrix} 0.58 & 0.13 & 0.16 & 0.17 & 0 \\ 0.21 & 0.67 & 0.24 & 0.29 & 0 \\ 0.19 & 0.18 & 0.58 & 0.19 & 1 \\ 0.02 & 0.01 & 0.02 & 0.35 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (7)$$

2) *Measurement Results of ACPR, Power Consumption, and Throughput:* Metrics under different DPD configurations are obtained from a wireless transmission simulator identical to that used in [1]. The simulator that we adopt consists of a WiFi signal generator, pulse shaping filter, DPD, and Wiener PA with the same parameters as [1]. The signal bandwidth is 20 MHz with 64 subcarriers and the modulation scheme is Quadrature Phase Shift Keying (QPSK). For each system configuration (p, q, P_{TX}) , the simulator is executed and the results are used later by MDPSOLVE.

Simulations are carried out with MATLAB using reward functions that are computed based on profiled execution time and power consumption characteristics. To measure the power consumption as well as throughput, we generate simulated WiFi packets and feed them into the body graph (see Fig. 1), which encapsulates the DPD filtering functionality mapped onto the GPU. Then we leverage the NVIDIA Visual Profiler (NVVP) to measure the power consumption and throughput.

In Table I, we present simulation results for ACPR and measurements of throughput and power consumption under different (p, q) configurations. The switching cost is approximated as 1/10 of the power consumption. From Table I, we can observe that there exist clear tradeoffs among the different metrics. More specifically, the results expose and quantify the following important trends.

- For low transmit power levels, good ACPR performance can be maintained even with low p and q values, while for high transmit power levels, high p and q values are desirable.
- High transmit power leads to worsened ACPR performance. One can apply high p and q values to recover the ACPR performance. As a trade-off, using high p and q values results in degradation of throughput and increased power consumption.

Considering the different ranges of the metrics, we normalize all metrics so that the normalized forms range from $[0, 1]$. For MDP-I, we use the median value of each metric to represent the corresponding factor — namely, M_P , M_A , M_S , or M_T — in the reward function of Equation 2. The medians are calculated from the measured values tabulated in Table I. For ACPR, the median is calculated across all numbers in the five columns labeled L_1, L_2, L_3, L_4, L_5 . For throughput and power consumption, the medians are computed across the numbers shown in the columns labeled Throughput and Power Consumption, respectively. This provides a general approach for calibrating the MDP-I reward function based on a set of measurements in the form illustrated in Table I.

3) *Experimental Results:* In the remainder of this section, we present experimental results for MDP-I, MDP-II and the hierarchical combination of both MDP-I and MDP-II. We compare the three resulting MDP-generated policies among themselves to assess the utility of using the proposed hierarchical MDP. We also compare the MDP approaches with a number of simple, static policies for configuration management. The hierarchical combination represents the MADS-A implementation, while the other reconfiguration policies are implemented using the same CPU/GPU testbed by adding/disabling appropriate functionality.

In each of the experiments, we simulate the DPD application system for 10,000 MDP time steps (*reconfiguration rounds*), where the interval between steps is 10 milliseconds (ms). This is the average reception interval between two packets that was measured in the WiFi experiments described above.

4) *Detailed Results for a Specific Weight Vector:* In this section, we present detailed results for $\mathbf{c} = (-0.4, -0.3, -0.2, 0.1)$, which is selected as a representative weight vector. These results help to validate and concretely demonstrate the operation of the MADS framework. The results are summarized in Fig. 3(a)–3(c). Here, the curves labeled “Maximum Reward Mapping” represent the policy that selects the action with highest reward in the current state without considering the potential impact of the action on the future. The curve labeled “Thresholding” represents the performance of a policy that turns off the system when the transmission power falls into L_1 — i.e., is smaller than 5 dBm. The curves labeled with the prefix “Fixed Policy”

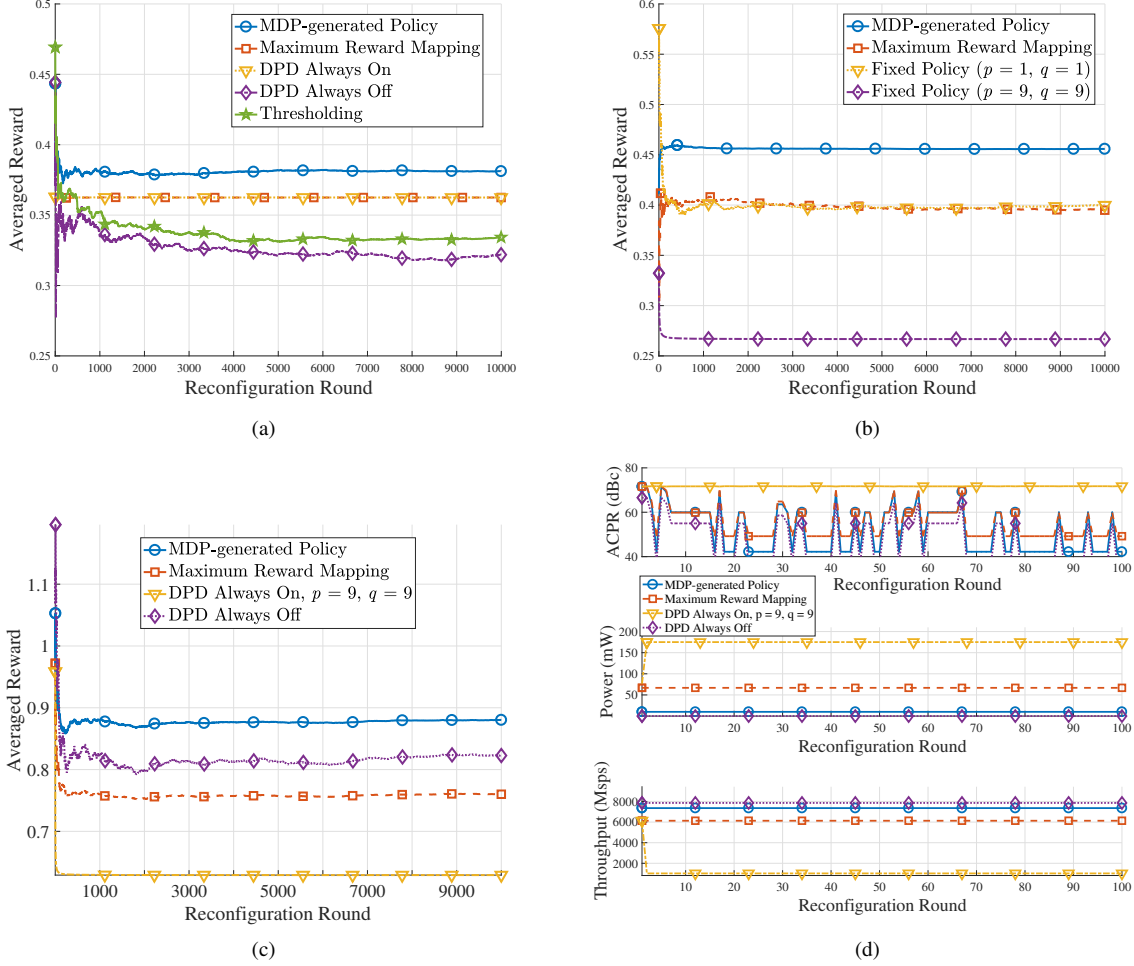


Fig. 3: Measured results for (a) MDP-I, (b) MDP-II, (c) MADS-A (Hierarchical MDPs), (d) absolute value of ACPR (dBc), power (mW) and throughput (MSPs) of hierarchical MDPs.

represent policies that simply fix the DPD configuration as specified.

The results in Fig. 3(a)–3(c) clearly demonstrate the capability of MADS-A to significantly outperform the individual MDPs used in isolation as well the more conventional (static) configuration management schemes. A similar observation can be made for different values of \mathbf{c} ; this is demonstrated in Section V-B5.

In Fig. 3(d), we demonstrate the ACPR, power consumption, and throughput under different policies. These results show the effectiveness of the proposed framework — in particular, its ability to strike a balance among different DPD metrics.

5) *Configuring Trade-Offs Using the Weight Vector*: In this section, we demonstrate optimized trade-offs derived by MADS-A under several different settings for the weight vector \mathbf{c} . The results help to demonstrate the flexibility provided to the designer for configuring operational trade-offs by adjusting the value of \mathbf{c} .

In Fig. 4(a), we demonstrate the performance of MADS-A when $\mathbf{c} = [0.0, -1.0, 0.0, 0.0]$ — i.e., when we are optimizing for ACPR only. In this case, MADS-A adopts the deterministic setting of always turning on DPD with $(p, q) = (9, 9)$ in order

to achieve the best ACPR performance.

On the other hand, when \mathbf{c} is configured as $[0.0, 0.0, 0.0, 1.0]$, MADS-A with $T = +\infty$ achieves the same performance as the case of not turning on DPD. The same observation can be made when tuning \mathbf{c} to optimize for power consumption and switching power. Finally, when all four metrics are considered, MADS-A with $T = +\infty$ outperforms the other settings, as can be seen from Fig. 4(d).

In Table II, we demonstrate the optimal policy given $T = +\infty$ for different settings of \mathbf{c} . To focus the optimization on ACPR performance ($\mathbf{c} = [0.0, -1.0, 0.0, 0.0]$), MADS-A turns on DPD for all states. On the other hand, when either power consumption or throughput is considered as the only important metric, MADS-A turns off DPD in all states. Finally, when all metrics are considered, MADS-A turns off DPD if the transmit power is low and turns on DPD otherwise. This can be justified because the non-linearity of the PA is not severe for low levels of transmit power, and thus, it is unnecessary to turn on DPD. The results for MDP-II are omitted for brevity.

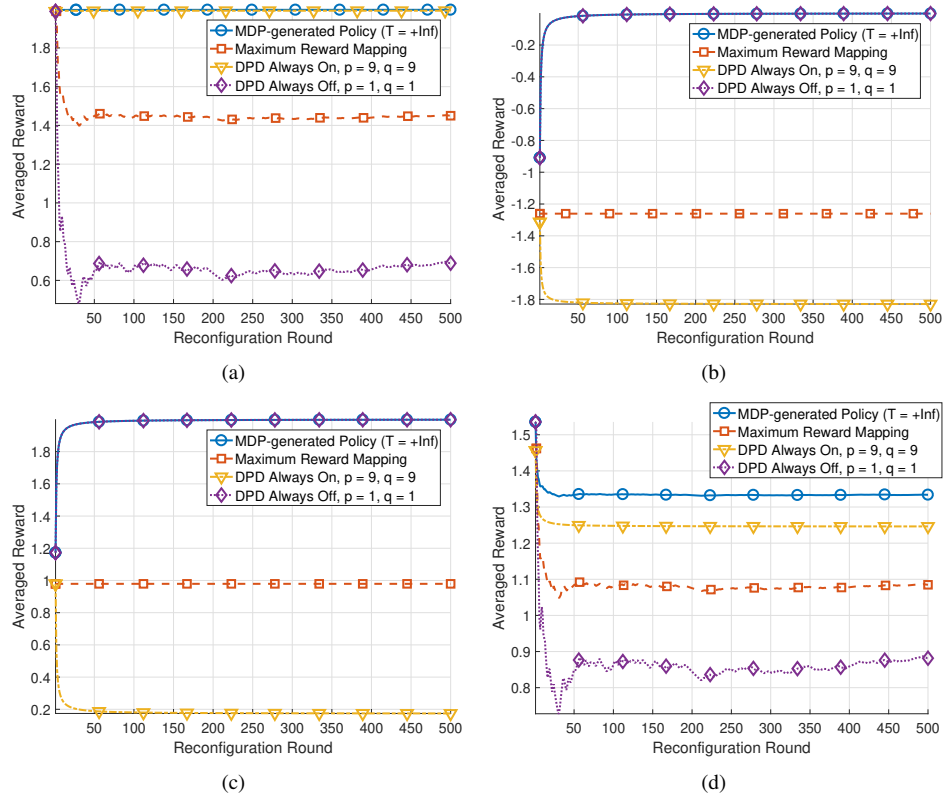


Fig. 4: Measured results for (a) $\mathbf{c} = [0.0, -1.0, 0.0, 0.0]$; (b) $\mathbf{c} = [-0.5, 0.0, -0.5, 0.0]$; (c) $\mathbf{c} = [0.0, 0.0, 0.0, 1.0]$; (d) $\mathbf{c} = [-0.05, -0.7, -0.05, 0.2]$.

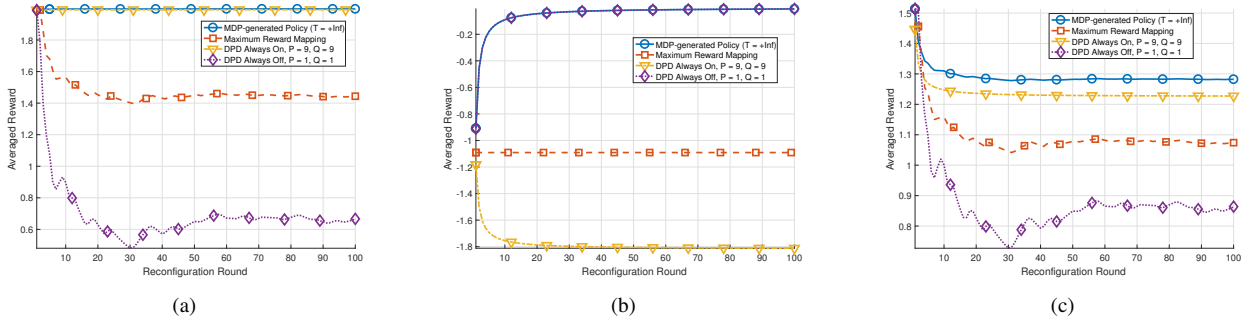


Fig. 5: Measured results with first-order throughput and power consumption models under different weight vectors: (a) $\mathbf{c} = [0.0, -1.0, 0.0, 0.0]$; (b) $\mathbf{c} = [-0.5, 0.0, -0.5, 0.0]$; (c) $\mathbf{c} = [-0.05, -0.7, -0.05, 0.2]$.

C. Performance with First-Order Reward Function Modeling

Throughput and power consumption can be modeled as a function of the complexity of the DPD system. Such modeling is useful if measurements of throughput and power are infeasible to obtain. Additionally, even if measurement is feasible, evaluating the performance of MADS before making extensive measurements can help designers gain insight into the possible outcomes of applying MADS. Such analysis can be very useful in early stages of the design process.

In this section, we present the performance of MADS based on the first-order modeling approach for DPD throughput and power consumption presented in [1]. The metrics can be written as

$$M_T[p, q] = \frac{1}{K_T \max(p, q)} \quad (8)$$

$$M_P[p, q] = K_P(p + q) \quad (9)$$

where $M_T[p, q]$ and $M_P[p, q]$ stand for the throughput and power consumption under DPD configuration (p, q) , respectively, and K_T and K_P are two constants to scale the metrics to the appropriate ranges of values.

With this first-order modeling approach, we reformulate the reward functions and execute simulations with the new functions. Here, we use $K_T = 0.0002 \text{ Msps}^{-1}$ and $K_P = 10 \text{ mW}$. This leads to the results shown in Fig. 5(a), Fig. 5(b), and Fig. 5(c). Similar to results shown in Section V-B5, we

TABLE II: Optimal policy with infinite horizon ($T = +\infty$) for MDP-I.

	Optimal Action ($T = +\infty$)			
	$\mathbf{c} = [0.0, -1.0, 0.0, 0.0]$	$\mathbf{c} = [-0.5, 0.0, -0.5, 0.0]$	$\mathbf{c} = [0.0, 0.0, 0.0, 1.0]$	$\mathbf{c} = [-0.05, -0.7, -0.05, 0.2]$
(DPD on, L1)	Turn on DPD	Turn off DPD	Turn off DPD	Turn off DPD
(DPD on, L2)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD
(DPD on, L3)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD
(DPD on, L4)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD
(DPD on, L5)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD
(DPD off, L1)	Turn on DPD	Turn off DPD	Turn off DPD	Turn off DPD
(DPD off, L2)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD
(DPD off, L3)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD
(DPD off, L4)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD
(DPD off, L5)	Turn on DPD	Turn off DPD	Turn off DPD	Turn on DPD

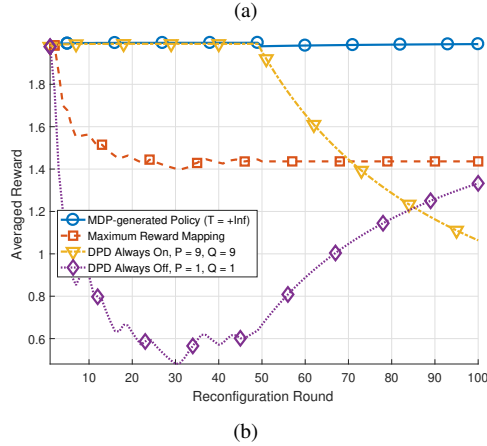
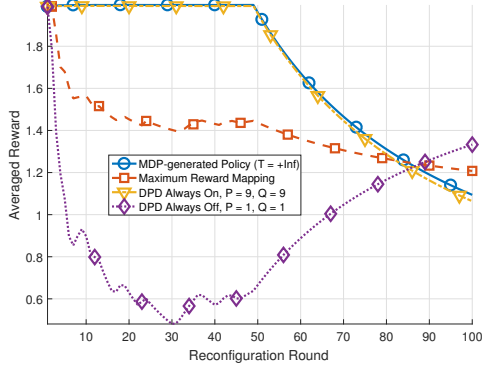


Fig. 6: Measured results with weight vector change: (a) without policy adjustment (b) with policy adjustment.

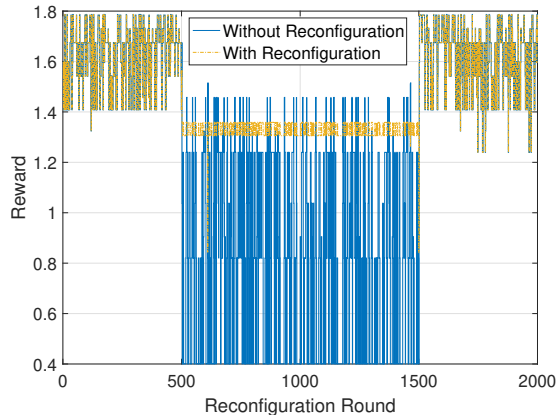


Fig. 7: Instantaneous reward with a time-varying TPTM.

find that MADS quantifies relevant trade-offs, as expected. This experiment demonstrates the versatility of the MADS framework in handling different reward functions.

D. Performance when the Reward Function Changes

In different periods of time, we might have different objectives. For instance, we may want to lower the power consumption when a device equipped with MADS loses connection to the power grid, while ACPR performance may be more important when the connection to the power grid is restored. Such scenarios lead to a time-varying reward function.

In general, if the reward function changes, we need to re-run the MDP solver to derive the optimal policy to enable reward convergence. To demonstrate this, we run simulations in which the optimization weight vector changes during the simulation, while the policy remains unchanged. Then we compare this with results obtained after adjusting the policy by re-running the MDP solver based on the new weight vector. The results are shown in Fig. 6(a) and Fig. 6(b). We adopt the setting of $\mathbf{c} = [0.0, -1.0, 0.0, 0.0]$ for the first half of the simulation and $\mathbf{c} = [-0.5, 0.0, -0.5, 0.0]$ for the second half of the simulation. Clearly, if we do not adjust the policy after the weight vector changes, then the reward drops significantly. On the other hand, when we adjust the policy based on the new weight vector, we are able to restore optimal performance.

E. Performance with Alternating TPTMs

When the wireless environment changes, the TPTM in general changes as a result. In this case, it is desirable for MADS-A to recalibrate the TPTM, and execute the MDP solver again to obtain the optimal policy based on the updated TPTM. In other words, when significant changes to the TPTM are detected, MDP policies can be recomputed dynamically. In this section, we motivate the development of such dynamic recomputation capability as a useful direction for future work on the MADS Framework.

To motivate the potential utility of dynamically recomputing the MDP policies, we examine two different wireless environments, as represented by the ECE Department and Kim Building environments described in Section V-B1. We compare the instantaneous reward with and without recalibration of the TPTM. The results are presented in Fig. 7. For the time epoch 0 to 500, the TPTM shown in (6) is used and the TPTM switches to the one shown in (7) for the time epoch 500 to 1500. Finally, the TPTM returns to

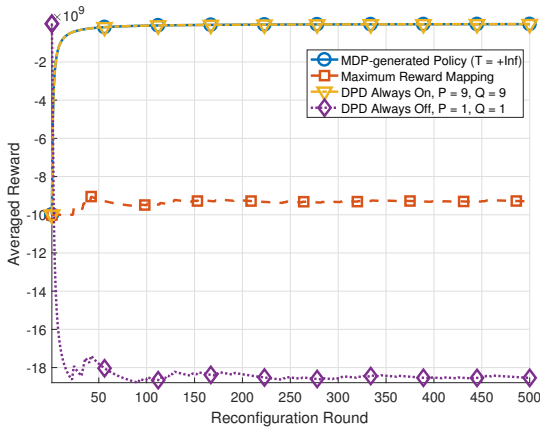


Fig. 8: Reward performance under ACPR requirement of ≥ 60 dBc.

the one in (6). As we can see from the figure, if we do not adjust the policy dynamically when the TPTM changes, the instantaneous reward degrades significantly. On the other hand, if we can keep track of changes in the TPTM and adjust the policy accordingly, the instantaneous reward improves. Efficiently integrating the required recalibration and dynamic policy recomputation capabilities into MADS-A implementations is a useful direction for future work.

F. Performance with Stringent ACPR Requirements

Many wireless standards enforce stringent ACPR requirements, which specify that ACPR levels under a given threshold are not tolerable. MADS can support this important use case by adjusting the reward functions for MDP-I and MDP-II such that the reward is set to $-\Gamma$, where Γ is a large real value, when the ACPR falls below the given threshold. In Fig. 8, we demonstrate the average reward with an ACPR threshold of 60 dBc and $\mathbf{c} = [-0.5, 0, -0.5, 0]$. The reward is set as -10^{10} (i.e., $\Gamma = 10^{10}$) when the ACPR falls below 60 dBc. Compared with Fig. 4(b), we observe that MADS prioritizes ACPR performance to avoid violating the ACPR requirement even if power consumption is considered as the most important factor as indicated from the weight vector \mathbf{c} . This verifies that MADS can support standard wireless applications where stringent ACPR requirements must be enforced.

VI. CONCLUSIONS

In this paper, we have proposed a general framework that applies Markov decision processes (MDPs) for design and implementation of adaptive DPD systems. Our framework, called the MDP framework for Adaptive DPD Systems (MADS), is designed so that many kinds of DPD algorithms can be plugged in to generate MDP-integrated, adaptive systems that are based on those algorithms. We demonstrate MADS by plugging into it a state-of-the-art DPD algorithm, and implementing the resulting adaptive DPD system on a hybrid CPU/GPU platform. Through extensive experiments, we have demonstrated the utility of the resulting implementation, and of the hierarchical MDP approach that is at the core of the MADS Framework.

An interesting direction for future work is to detect changes in the transmit power transition matrix, and incorporate associated capabilities in MADS to dynamically recompute the optimal policy. Secondly, it is useful to take into account the bit width of different DPD branches within the optimization process of the framework. This would provide MADS with finer granularity in the design optimization process. Thirdly, it would be interesting to investigate the application of the MADS framework to other DPD algorithms beyond Anttila's algorithm.

Another future research direction is to extend the scope of MADS from DPD to the entire wireless transmitter. In this case, we need to consider the power consumption of the PA in addition to that of the DPD. In this paper, we have focused only on the power consumption of the DPD. An interesting direction for future work is to develop an end-to-end system model that not only includes MADS, as proposed in this paper, but also other aspects including adaptive transmission power and modulation control. To develop such an approach, we would redefine the environment states so that they include the wireless channel condition between the base station and end device, as well as the expected transmission rate. We anticipate that the resulting system can still be formulated as a hierarchical MDP — for example, with the addition of a third MDP, which determines the optimal modulation and transmission power to apply. In this case, the transmit power will in general not be independent of the action, and the dependence would be handled optimally by the extended MDP formulation.

ACKNOWLEDGMENTS

This research was sponsored in part by the US National Science Foundation.

REFERENCES

- [1] L. Anttila, P. Händel, and M. Valkama, "Joint mitigation of power amplifier and I/Q modulator impairments in broadband direct-conversion transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 4, pp. 730–739, 2010.
- [2] O. Sigaud and O. Buffet, Eds., *Markov Decision Processes in Artificial Intelligence*. Wiley, 2010.
- [3] Y.-D. Kim, E.-R. Jeong, and Y. H. Lee, "Adaptive compensation for power amplifier nonlinearity in the presence of quadrature modulation/demodulation errors," *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4717–4721, 2007.
- [4] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Transactions on communications*, vol. 52, no. 1, pp. 159–165, 2004.
- [5] R. Sperlich, J. A. Sills, and J. S. Kenney, "Power amplifier linearization with memory effects using digital pre-distortion and genetic algorithms," in *Proceedings of the Radio and Wireless Conference*, 2004, pp. 355–358.
- [6] C. Çiflikli and A. Yarıcı, "Genetic algorithm optimization of a hybrid analog/digital predistorter for RF power amplifiers," *Analog Integrated Circuits and Signal Processing*, vol. 52, no. 1, pp. 25–30, 2007.
- [7] K. Freiberger, M. Wolkerstorfer, H. Enzinger, and C. Vogel, "Digital predistorter identification based on constrained multi-objective optimization of WLAN standard performance metrics," in *Proceedings of the International Symposium on Circuits and Systems*, 2015, pp. 862–865.
- [8] J. A. Sills and R. Sperlich, "Adaptive power amplifier linearization by digital pre-distortion using genetic algorithms," in *Proceedings of the Radio and Wireless Conference*, 2002, pp. 229–232.

- [9] A. H. Abdelhafiz, O. Hammi, A. Zerguine, A. T. Al-Awami, and F. M. Ghannouchi, "A PSO based memory polynomial predistorter with embedded dimension estimation," *IEEE Transactions on Broadcasting*, vol. 59, no. 4, pp. 665–673, 2013.
- [10] A. Ghazi, J. Boutellier, O. Silvén, S. Shahabuddin, M. Juntti, S. S. Bhattacharyya, and L. Anttila, "Model-based design and implementation of an adaptive digital predistortion filter," in *Proceedings of the IEEE Workshop on Signal Processing Systems*, 2015, pp. 1–6.
- [11] S. Wang, M. A. Hussein, O. Venard, and G. Baudoin, "A novel algorithm for determining the structure of digital predistortion models," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7326–7340, Aug 2018.
- [12] L. Li, A. Ghazi, J. Boutellier, L. Anttila, M. Valkama, and S. S. Bhattacharyya, "Evolutionary multiobjective optimization for adaptive dataflow-based digital predistortion architectures," *EAI Endorsed Transactions on Cognitive Communications*, vol. 3, no. 10, pp. 1–9, 2017.
- [13] L. Li, P. Deaville, A. Sapio, L. Anttila, M. E. Valkama, M. Wolf, and S. Bhattacharyya, "A framework for design and implementation of adaptive digital predistortion systems," in *Proceedings of the IEEE International Conference on Artificial Intelligence Circuits and Systems*, Hsinchu, Taiwan, March 2019, pp. 112–116.
- [14] P. L. Fackler, "MDPSOLVE a MATLAB toolbox for solving Markov decision problems with dynamic programming — user's guide," North Carolina State University, Tech. Rep., January 2011.
- [15] A. Jonsson and A. Barto, "Causal graph based decomposition of factored MDPs," *Journal of Machine Learning Research*, vol. 7, pp. 2259–2301, 2006.
- [16] A. Sapio, L. Li, J. Wu, M. Wolf, and S. S. Bhattacharyya, "Reconfigurable digital channelizer design using factored Markov decision processes," *Journal of Signal Processing Systems*, vol. 90, no. 10, pp. 1329–1343, 2018.
- [17] B. Bhattacharya and S. S. Bhattacharyya, "Parameterized dataflow modeling for DSP systems," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2408–2421, October 2001.
- [18] C. Shen, W. Plishker, H. Wu, and S. S. Bhattacharyya, "A lightweight dataflow approach for design and implementation of SDR systems," in *Proceedings of the Wireless Innovation Conference and Product Exposition*, Washington DC, USA, November 2010, pp. 640–645.
- [19] S. Lin, Y. Liu, K. Lee, L. Li, W. Plishker, and S. S. Bhattacharyya, "The DSPCAD framework for modeling and synthesis of signal processing systems," in *Handbook of Hardware/Software Codesign*, S. Ha and J. Teich, Eds. Springer, 2017, pp. 1–35.
- [20] L. Ding, "Digital predistortion of power amplifiers for wireless applications," Ph.D. dissertation, Georgia Institute of Technology, 2004.



Lin Li obtained her Ph.D. degree from the Department of Electrical and Computer Engineering at the University of Maryland, College Park, USA. She received the bachelor's degree in Electrical Engineering and Automation from Fudan University, Shanghai, China. Her research has focused on dataflow-based framework for design, implementation, and optimization of signal processing systems including wireless communication systems and machine learning systems.



Peter Deaville received his B.S. degree in electrical engineering from the University of Maryland, College Park, in 2018. He joined Princeton University, New Jersey, in the spring of 2019 as a Ph.D. student in the Department of Electrical Engineering, where his research focuses on exploring in-memory computing using magnetic random-access memory technology. His other research interests include circuit design for machine learning, leveraging emerging methods and technologies.



Adrian Sapio received his BS and MS degrees in Electrical Engineering from the Rochester Institute of Technology in 2004 and 2010, respectively. He is a Ph.D. candidate in Electrical Engineering at the University of Maryland, College Park, and also a Lead Engineer at Intelligent Automation, Inc. in Rockville, MD. His research interests include design methodologies for embedded systems used in wireless networks.



Lauri Anttila received the M.Sc. and the D.Sc. (Hons.) degrees in electrical engineering from Tampere University of Technology (TUT), Finland, in 2004 and 2011, respectively. Since 2016, he has been a University Researcher at the Department of Electrical Engineering, Tampere University (formerly TUT). In 2016–2017, he was a Visiting Research Fellow at the Department of Electronics and Nanoengineering, Aalto University, Finland. He has co-authored 100+ refereed articles, as well as three book chapters. His research interests are in radio communications and signal processing, with a focus on the radio implementation challenges in systems such as 5G, full-duplex radio, and large-scale antenna systems.



Mikko Valkama received the M.Sc. (Tech.) and D.Sc. (Tech.) Degrees (both with honors) in electrical engineering (EE) from Tampere University of Technology (TUT), Finland, in 2000 and 2001, respectively. In 2002, he received the Best Doctoral Thesis -award by the Finnish Academy of Science and Letters for his dissertation entitled "Advanced I/Q signal processing for wideband receivers: Models and algorithms". In 2003, he was working as a visiting post-doc research fellow with the Communications Systems and Signal Processing Institute at SDSU, San Diego, CA. Currently, he is a Full Professor and Department Head of Electrical Engineering at newly formed Tampere University (TAU), Finland. His general research interests include radio communications, radio localization, and radio-based sensing, with particular emphasis on 5G and beyond mobile radio networks.



Marilyn Wolf is Professor and Chair of the Department of Computer Science and Engineering at the University of Nebraska Lincoln. She received her BS, MS, and PhD in electrical engineering from Stanford University in 1980, 1981, and 1984, respectively. She was with AT&T Bell Laboratories from 1984 to 1989. She was on the faculty of Princeton University from 1989 to 2007 and was Farmer Distinguished Chair at Georgia Tech from 2007 to 2019. Her research interests included embedded computing, embedded video and computer vision, and VLSI systems. She has received the IEEE Computer Society Goode Memorial Award, the ASEE Terman Award and IEEE Circuits and Systems Society Education Award. She is a Fellow of the IEEE and ACM and an IEEE Computer Society Golden Core member.



Shuvra S. Bhattacharyya is a Professor in the Department of Electrical and Computer Engineering at the University of Maryland, College Park. He holds a joint appointment in the University of Maryland Institute for Advanced Computer Studies (UMIACS). He also holds a part-time position as International Research Chair, joint with INSA/IETR, and INRIA in Rennes, France. His research interests include signal processing, embedded systems, electronic design automation, machine learning, wireless communication, and wireless sensor networks. He

received the Ph.D. degree from the University of California at Berkeley. He has held industrial positions as a Researcher at the Hitachi America Semiconductor Research Laboratory (San Jose, California), and Compiler Developer at Kuck & Associates (Champaign, Illinois). He has held a visiting summer research position at AFRL in Rome, New York. From 2015 through 2018, he was a part time visiting professor in the Department of Pervasive Computing at the Tampere University of Technology, Finland, as part of the Finland Distinguished Professor Programme (FiDiPro).