Supplementary Material for "An MDP-based Dynamic Optimization Methodology for Wireless Sensor Networks"

Arslan Munir and Ann Gordon-Ross

Abstract—Wireless sensor networks (WSNs) are distributed systems that have proliferated across diverse application domains (e.g., security/defense, health care, etc.). One commonality across all WSN domains is the need to meet application requirements (i.e., lifetime, responsiveness, etc.) through domain specific sensor node design. Techniques such as sensor node parameter tuning enable WSN designers to specialize tunable parameters (i.e., processor voltage and frequency, sensing frequency, etc.) to meet these application requirements. However, given WSN domain diversity, varying environmental situations (stimuli), and sensor node complexity, sensor node parameter tuning is a very challenging task. In this paper, we propose an automated Markov Decision Process (MDP)-based methodology to prescribe optimal sensor node operation (selection of values for tunable parameters such as processor voltage, processor frequency, and sensing frequency) to meet application requirements and adapt to changing environmental stimuli. Numerical results confirm the optimality of our proposed methodology and reveal that our methodology more closely meets application requirements compared to other feasible policies.

1 Introduction

This document presents additional details supplementing our IEEE Transactions on Parallel and Distributed (TPDS) paper with the title *"An MDP-based Dynamic Optimization Methodology for Wireless Sensor Networks"*. This supplementary material document is organized as follows. A review of supplemental related work is given in Section 2. Section 3 presents the formulation of our proposed methodology as an MDP. Section 4 provides implementation guidelines and our proposed methodology's computational complexity. Section 5 provides model extensions to our proposed policy. Numerical results for an ambient conditions monitoring application are presented in Section 6.

2 Related Work

This section presents additional related work not summarized in the main paper. There is a lot of research in the area of dynamic optimizations [1][2][3][4], but however, most previous work focuses on the processor or memory (cache) in computer systems. Whereas these endeavors can provide valuable insights into WSN dynamic optimizations, they are not directly applicable to WSNs due to different design spaces, platform particulars, and a sensor node's tight design constraints.

Stevens-Navarro et al. [5] applied MDPs for vertical handoff decisions in heterogeneous wireless networks. Although our work leverages the reward function idea from their work, our work, for the first time to the best of our knowledge, applies MDPs to dynamic optimizations for WSNs.

Little previous work exists in the area of application specific tuning and dynamic profiling in WSNs. Sridharan et al. [6] obtained accurate environmental stimuli by dynamically profiling the WSN's operating environment, however, they did not propose any methodology to leverage these profiling statistics for optimizations. Tilak et al. [7] investigated infrastructure (referred to as sensor node characteristics, number of deployed sensors, and deployment strategy) tradeoffs on application requirements. The application requirements considered were accuracy, latency, energy efficiency, fault tolerance, goodput (ratio of total number of packets received to the total number of packets sent), and scalability. However, the authors did not delineate the interdependence between low-level sensor node parameters and high-level application requirements. Kogekar et al. [8] proposed an approach for dynamic software reconfiguration in WSNs using dynamically adaptive software. Their approach used tasks to detect environmental changes (event occurrences) and adapt the software to the new conditions. Their work did not consider sensor node tunable parameters. Kadayif et al. [9] proposed an automated strategy for data filtering to determine the amount of computation or data filtering to be done at the sensor nodes before transmitting data to the sink node. Unfortunately, the authors only studied the effects of data filtering tuning on energy consumption and did not consider other sensor node parameters and application requirements.

Some previous and current work investigates

• *Arslan Munir and Ann Gordon-Ross are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, 32611 USA. Ann Gordon-Ross is also affiliated with the NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida. e-mail: {amunir@ufl.edu, ann@ece.ufl.edu}*

WSN operation in changing application (mission) requirements and environmental stimuli. Marrón et al. [10] presented an adaptive cross-layer architecture *TinyCubus* for TinyOS-based sensor networks that allowed dynamic management of components (e.g., caching, aggregation, broadcast strategies) and reliable code distribution considering WSN topology. *TinyCubus* considered optimization parameters (e.g., energy, communication latency, and bandwidth), application requirements (e.g., reliability), and system parameters (e.g., mobility). The system parameters selected the best set of components based on current application requirements and optimization parameters. Vecchio [11] discussed adaptability in WSNs at three different levels: communication-level (by tuning the communication scheme), application-level (by software changes), and hardware-level (by injecting new sensor nodes). The international technology alliance in network and information science (ITA), sponsored by the UK ministry of defense (MoD) and US army research lab (ARL), investigates task reassignment and reconfiguration (including physical movement of sensor nodes or reprocessing of data) of already deployed sensor nodes in the sensor field in response to current or predicted future conditions to provide the expected sensed information at a sufficient quality [12]. However, ITA current projects, to the best of our knowledge, do not consider sensor node parameter tuning and our MDP-based parameter tuning can optimize WSN operation in changing application requirements.

Several papers explore DVFS for reduced energy consumption. Pillai et al. [13] proposed real-time dynamic voltage scaling (RT-DVS) algorithms capable of modifying the operating systems' real-time scheduler and task management service for reduced energy consumption. Childers et al. [14] proposed a technique for adjusting supply voltage and frequency at run-time to conserve energy. Their technique monitored a program's instruction-level parallelism (ILP) and adjusted processor voltage and speed in response to the current ILP. Their proposed technique allowed users to specify performance constraints, which the hardware maintained while running at the lowest energy consumption.

Liu et al. [15] investigated reducing processor speed by varying the supply and threshold voltages for low power consumption in complementary metal-oxide-semiconductor (CMOS) VLSI (very-large-scale integration). Results showed that an optimized threshold voltage revealed an 8x power savings without any negative performance impacts. In addition, significantly greater energy savings could be achieved by reducing processor speed in tandem with threshold voltage. Burd et al. [16] presented a microprocessor system that dynamically varied its supply voltage and clock frequency to deliver high throughput during critical high-speed execution periods and extended battery life during low-speed execution periods. Results revealed that dynamic voltage scaling (DVS) could improve energy efficiency by 10x for battery-powered processor systems without sacrificing peak throughput.

Min et al. [17] demonstrated that dynamic voltage scaling in a sensor node's processor reduced energy consumption. Their technique used a voltage scheduler, running in tandem with the operating system's task scheduler, to adjust voltage and frequency based on a priori knowledge of the predicted sensor node's workload. Yuan et al. [18] studied a DVFS system for sensor nodes, which required sensor nodes sending data to insert additional information into a transmitted data message's header such as the packet length, expected processing time, and deadline. The receiving sensor node used this message information to select an appropriate processor voltage and frequency to minimize the overall energy consumption.

Some previous works in WSN optimizations explore greedy and simulated annealing (SA)-based methods. Lysecky et al. [19] proposed an SA-based automated application specific tuning of parameterized sensor-based embedded systems and found that automated tuning can improve WSN operation by 40% on average. Verma [20] studied SA and particle swarm optimization (PSO) methods for automated application specific tuning and observed that SA performed better than PSO because PSO often quickly converged to local minima. In prior work, Munir et al. [21] proposed greedy- and simulated annealing (SA)-based algorithms for parameter tuning. Whereas greedy- and SA-based algorithms are lightweight, these algorithms do not ensure convergence to an optimal solution.

There exists previous work related to DVFS and several initiatives towards application-specific tuning were taken. Nevertheless, literature presents no mechanisms to determine an optimal dynamic tuning policy for sensor node parameters in accordance with changing application requirements. To the best of our knowledge, we propose the first methodology to address WSN dynamic optimizations with the goal of meeting application requirements in a dynamic environment.

3 Dynamic Optimization Formulation as an MDP

In this section, we describe the formulation of our WSN dynamic optimization as an MDP. We formulate MDP-based policy constructs (i.e., state space, decision epochs, actions, state dynamics, policy, performance criterion, and reward function) for our system. We also introduce optimality equations and the policy iteration algorithm.

3.1 State Space

The state space for our MDP-based tuning methodology is a composite state space containing the Cartesian product of sensor node tunable parameters' state spaces. We define the state space S as:

$$S = S_1 \times S_2 \times \cdots \times S_M \quad : \quad |S| = I \tag{1}$$

where $\times$ denotes the Cartesian product, M is the total number of sensor node tunable parameters, S_k denotes the state space for tunable parameter k where $k \in \{1, 2, \ldots, M\}$, and $|S|$ denotes the state space S cardinality (the number of states in S).

The tunable parameter k's state space ($k \in \{1, 2, \ldots, M\}$) S_k consists of n values:

$$S_k = \{s_{k_1}, s_{k_2}, s_{k_3}, \ldots, s_{k_n}\} \quad : \quad |S_k| = n \tag{2}$$

where $|S_k|$ denotes the tunable parameter k's state space cardinality (the number of tunable values in S_k). S is a set of n-tuples where each n-tuple represents a sensor node state s. Each state s_i is an n-tuple, i.e., $s_i = (v_1, v_2, \ldots, v_M) : v_k \in S_k$. Note that some n-tuples in S may not be feasible (e.g., all processor voltage and frequency pairs are not feasible) and can be regarded as *do not care* tuples.

Each sensor node state has an associated power consumption, throughput, and delay. The power, throughput, and delay for state s_i are denoted by p_i, t_i, and d_i, respectively. Since different sensor nodes may have different embedded processors and attached sensors, each node may have node specific power consumption, throughput, and delay information for each state.

3.2 Decision Epochs and Actions

Sensor nodes make decisions at decision epochs, which occur after fixed time periods. The sequence of decision epochs is represented as:

$$T = \{1, 2, 3, ..., N\}, \quad N \leq \infty \tag{3}$$

where the random variable N corresponds to the sensor node's lifetime.

At each decision epoch, a sensor node's *action* determines the next state to transition to given the current state. The sensor node action in state $i \in S$ is defined as:

$$A_i = \{a_{i,j}\} \in \{0, 1\} \tag{4}$$

where $a_{i,j}$ denotes the action taken at time t that causes the sensor node to transition to state j at time $t+1$ from the current state i. A *policy* determines whether an action is taken or not. If $a_{i,j} = 1$, the action is taken and if $a_{i,j} = 0$, the action is not taken. For a given state $i \in S$, a selected action can not result in a transition to a state that is not in S. The action space can be defined as:

$$\begin{aligned} A \quad = \quad & \Big\{a = [a_{i,j}] : \{a_{i,j}\} \in \{0, 1\}, \\ & i = \{1, 2, 3, ..., I\}, j = \{1, 2, 3, ..., I\}\Big\} \end{aligned} \tag{5}$$

3.3 State Dynamics

The state dynamics of the system can be delineated by the state transition probabilities of the embedded Markov chain. We formulate our sensor node policy as a deterministic dynamic program (DDP) because the choice of an action determines the subsequent state with certainty. Our sensor node DDP policy formulation uses a *transfer function* to specify the next state. A transfer function defines a mapping $\tau_t(s, a)$ from $S \times A_s \rightarrow S$, which specifies the system state at time $t+1$ when the sensor node selects action $a \in A_s$ in state s at time t. To formulate our DDP as an MDP, we define the *transition probability function* as:

$$p_t(j|s, a) = \begin{cases} 1 & \text{if } \tau_t(s, a) = j \\ 0 & \text{if } \tau_t(s, a) \neq j. \end{cases} \tag{6}$$

3.4 Policy and Performance Criterion

For each given state $s \in S$, a sensor node selects an action $a \in A_s$ according to a policy $\pi \in \Pi$ where Π is a set of admissible policies defined as:

$$\Pi = \{\pi : S \rightarrow A_s | d_t(s) \in A_s, \ \ \forall \ s \in S\} \tag{7}$$

A *performance criterion* compares the performance of different policies. The sensor node selects an action prescribed by a policy based on the sensor node's current state. If the random variable X_t denotes the state at decision epoch t and the random variable Y_t denotes the action selected at decision epoch t, then for the deterministic case, $Y_t = d_t(X_t)$.

As a result of selecting an action, the sensor node receives a reward $r(X_t, Y_t)$ at time t. The *expected total reward* denotes the expected total reward over the decision making horizon given a specific policy. Let $v^\pi(s)$ denote the expected total reward over the decision making horizon when the horizon length N is a random variable, the system is in state s at the first decision epoch, and policy π is used [5][22]:

$$v^\pi(s) = E_s^\pi \left[E_N \left\{ \sum_{t=1}^{N} r(X_t, Y_t) \right\} \right] \tag{8}$$

where E_s^π represents the expected reward with respect to policy π and the initial state s (the system state at the time of the expected reward calculation), and E_N denotes the expected reward with respect to the probability distribution of the random variable N. We can write (8) as [22]:

$$v^\pi(s) = E_s^\pi \left\{ \sum_{t=1}^{\infty} \lambda^{t-1} r(X_t, Y_t) \right\} \tag{9}$$

which gives the *expected total discounted reward*. We assume that the random variable N is geometrically distributed with parameter λ and hence the distribution *mean* is $1/(1-\lambda)$ [5]. The parameter λ can be interpreted as a *discount factor*, which measures the present value of one unit of reward received one period in the future. Thus, $v^\pi(s)$ represents the expected total present value of the reward (income) stream obtained using policy π [22]. Our objective is to find a policy that maximizes the expected total discounted reward i.e., a policy π^* is *optimal* if

$$v^{\pi^*}(s) \geq v^\pi(s) \quad \forall \ \ \pi \in \Pi \tag{10}$$

3.5 Reward Function

The reward function captures application metrics and sensor node characteristics. Our reward function characterization considers the power consumption (which affects the sensor node lifetime), throughput, and delay application metrics. We define the reward function $f(s,a)$ given the current sensor node state s and the sensor node's selected action a as:

$$f(s,a) = \omega_p f_p(s,a) + \omega_t f_t(s,a) + \omega_d f_d(s,a) \tag{11}$$

where $f_p(s,a)$ denotes the power reward function, $f_t(s,a)$ denotes the throughput reward function, and $f_d(s,a)$ denotes the delay reward function; ω_p, ω_t, and ω_d represent the *weight factors* for power, throughput, and delay, respectively. The weight factors' constraints are given as $\sum_m \omega_m = 1$ where $m = \{p,t,d\}$ such that $0 \leq \omega_p \leq 1$, $0 \leq \omega_t \leq 1$, and $0 \leq \omega_d \leq 1$. The weight factors are selected based on the relative importance of application metrics with respect to each other, e.g., a habitat monitoring application taking camera images of the habitat requires a minimum image resolution to provide meaningful analysis that necessitates a minimum throughput and therefor throughput can be assigned a higher weight factor than the power metric for this application.

We define linear reward functions for application metrics because an application metric reward (objective function) typically varies linearly, or piecewise linearly, between the minimum and the maximum allowed values of the metric [5][19]. However, a non-linear characterization of reward functions is also possible and depends upon the particular application. We point out that our methodology works for any characterization of reward function. The reward function characterization only defines the reward obtained from operating in a given state. Our MDP-based policy determines the optimal reward by selecting an optimal operating state given the sensor node design space and application requirements for any reward function characterization. We consider linear reward functions as a typical example from the space of possible reward functions (e.g., piecewise linear, non-linear) to illustrate our MDP-based policy. We define the power reward function (Fig. 1(a))in (11) as:

$$f_p(s,a) = \begin{cases} 1, & 0 < p_a \leq L_P \\ (U_P - p_a)/(U_P - L_P), & L_P < p_a < U_P \\ 0, & p_a \geq U_P. \end{cases} \tag{12}$$

where p_a denotes the power consumption of the current state given action a taken at time t and the constant parameters L_P and U_P denote the minimum and maximum allowed/tolerated sensor node power consumption, respectively.

We define the throughput reward function (Fig. 1(b))in (11) as:

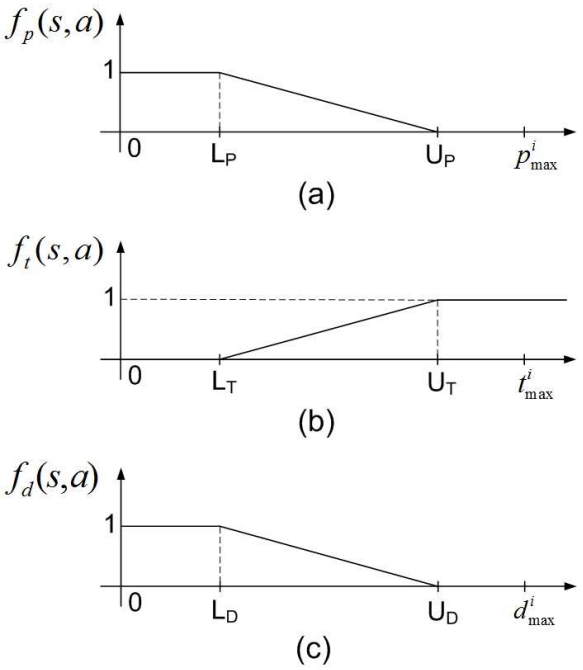

Fig. 1. Reward functions: (a) Power reward function $f_p(s,a)$; (b) Throughput reward function $f_t(s,a)$; (c) Delay reward function $f_d(s,a)$.

$$f_t(s,a) = \begin{cases} 1, & t_a \geq U_T \\ (t_a - L_T)/(U_T - L_T), & L_T < t_a < U_T \\ 0, & t_a \leq L_T. \end{cases} \tag{13}$$

where t_a denotes the throughput of the current state given action a taken at time t and the constant parameters L_T and U_T denote the minimum and maximum allowed/tolerated throughput, respectively.

We define the delay reward function (Fig. 1(c)) in (11) as:

$$f_d(s,a) = \begin{cases} 1, & 0 < d_a \leq L_D \\ (U_D - d_a)/(U_D - L_D), & L_D < d_a < U_D \\ 0, & d_a \geq U_D. \end{cases} \tag{14}$$

where d_a denotes the delay in the current state and the constant parameters L_D and U_D denote the minimum and maximum allowed/tolerated delay, respectively.

State transitioning incurs a cost associated with switching parameter values from the current state to the next state (typically in the form of power and/or execution overhead). We define the transition cost function $h(s,a)$ as:

$$h(s,a) = \begin{cases} H_{i,a} & \text{if } i \neq a \\ 0 & \text{if } i = a. \end{cases} \tag{15}$$

where $H_{i,a}$ denotes the transition cost to switch from the current state i to the next state as determined by action a. Note that a sensor node incurs no transition cost if

action a prescribes that the next state is the same as the current state.

Hence, the overall reward function $r(s,a)$ given state s and action a at time t is:

$$r(s,a) = f(s,a) - h(s,a) \tag{16}$$

which accounts for the power, throughput, and delay application metrics as well as state transition cost.

We point out that many other application metrics (e.g., security, reliability, and lifetime) are of immense significance to WSNs. For example, WSNs are vulnerable to security attacks such as distributed denial of service and Sybil attacks for which a security reward function can be included. A reliability reward function can encompass the reliability aspect of WSNs since sensor nodes are often deployed in unattended and hostile environments and are susceptible to failures. Similarly, considering sensor nodes' constrained battery resources, power optimization techniques exist that put sensor nodes in sleep or low-power mode (where communication and/or processing functions are disabled) for power conservation when less activity is observed in the sensed region as determined by previously sensed data. These power optimizations can be captured by a lifetime reward function. These additional metrics incorporation in our model is the focus of our future work.

3.6 Optimality Equation

The optimality equation, also known as Bellman's equation, for expected total discounted reward criterion is given as [22]:

$$v(s) = \max_{a \in A_s} \left\{ r(s,a) + \sum_{j \in S} \lambda p(j|s,a) v(j) \right\} \tag{17}$$

where $v(s)$ denotes the maximum expected total discounted reward. The salient properties of the optimality equation are: the optimality equation has a unique solution; an optimal policy exists given conditions on states, actions, rewards, and transition probabilities; the value of the discounted MDP satisfies the optimality equation; and the optimality equation characterizes stationary policies.

The solution of (17) gives the maximum expected total discounted reward $v(s)$ and the MDP-based optimal policy π^* (or π^{MDP}), which gives the maximum $v(s)$. π^{MDP} prescribes the action a from action set A_s given the current state s for all $s \in S$. There are several methods to solve the optimality equation (17) such as value iteration, policy iteration, and linear programming, however in this work we use the policy iteration algorithm.

3.7 Policy Iteration Algorithm

The policy iteration algorithm can be described in four steps:

1) Set $l = 0$ and choose any arbitrary decision rule $d_0 \in D$ where D is a set of all possible decision rules.
2) *Policy evaluation* - Obtain $v^l(s) \ \forall \ s \in S$ by solving the equations:
$$v^l(s) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v^l(j) \tag{18}$$
3) *Policy improvement* - Select $d_{l+1} \ \forall \ s \in S$ to satisfy the equations:
$$d_{l+1}(s) \in \arg\max_{a \in A_s} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v^l(j) \right\} \tag{19}$$
and setting $d_{l+1} = d_l$ if possible.
4) If $d_{l+1} = d_l$, stop and set $d^* = d_l$ where d^* denotes the optimal decision rule. If $d_{l+1} \neq d_l$, set $l = l + 1$ and go to step 2.

Step 2 is referred to as policy evaluation, because by solving (18), we obtain the expected total discounted reward for decision rule d_l. Step 3 is referred to as policy improvement, because this step selects a v^l-improving decision rule. In step 4, $d_{l+1} = d_l$ quells cycling, because a decision rule is not necessarily unique.

4 Implementation Guidelines and Complexity

In this section, we describe the implementation guidelines and computational complexity for our proposed MDP-based optimal policy. The implementation guidelines describe the mapping of MDP-specifics (e.g., state space, reward function) in our problem formulation (Section 3) to actual sensor node hardware. The computational complexity focuses on the convergence of the policy iteration algorithm and the data memory analysis for our MDP-based dynamic tuning methodology. The prototype implementation of our MDP-based tuning methodology on hardware sensor platforms is the focus of our future work.

4.1 Implementation Guidelines

In order to implement our MDP-based optimal policy, particular values must be initially defined. The reward function (16) uses the power, throughput, and delay values offered in a sensor node state s_i (Section 3.1). An application manager specifies the minimum and maximum power, throughput, and delay values required by (12), (13), and (14), respectively, and the power, throughput, and delay weight factors required in (11) according to application specifications.

A sensor node's embedded processor defines the transition cost $H_{i,a}$ as required in (15), which is dependent on a processor's particular power management and switching techniques. Processors have a set of available voltage and frequency pairs, which defines the V_p and F_p values, respectively, in a sensor

node state tuple (Section 3.1). Embedded sensors can operate at different defined sensing rates, which define the F_s value in a sensor node state tuple (Section 3.1). The embedded processor and sensor characteristics determine the value of I, which characterizes the state space in (1) (i.e., number of allowable processor voltage, processor frequency, and sensing frequency values determine the total number of sensor node operating states, and thus the value of I).

The sensor nodes perform parameter tuning decisions at decision epochs (Section **??**). The decision epochs can be guided by the dynamic profiler module to adapt to the environmental stimuli. For instance, for a target tracking application and a fast moving target, the decision epoch period should be small to better capture the fast moving target. On the other hand, for stationary or slow moving targets, decision epoch period should be large to conserve battery energy. However, since the exact decision epoch period is application specific, the period should be adjusted to control the sensor node lifetime.

Both the MDP controller module (which implements the policy iteration algorithm to calculate the MDP-based optimal policy π^{MDP}) and the dynamic profiler module (Section **??**) can either be implemented as software running on a sensor node's embedded processor or custom hardware for faster execution.

One of the drawbacks for MDP-based policy is that computational and storage overhead increases as the number of states increases. Therefore, WSN designer would like to restrict the sensor states (e.g., 2, 4, or 16, etc.) to reduce the computational and storage overhead. If state restriction is not a viable option, the WSN configuration could be augmented with a back end base station node to run our MDP-based optimization and the sensor node operating states would be communicated to the sensor nodes. This communication of operating state information to sensor nodes by the base station node would not consume enough power resources given that this state information is transmitted periodically and/or aperiodically after some minimum duration determined by the agility of the environmental stimuli (e.g., more frequent communication would be required for a rapidly changing environmental stimuli as opposed to a slow changing environmental stimuli). This WSN configuration could also consider *global optimizations*, which are optimizations that take into account sensor node interactions and dependencies and is a focus of our future work. We point out that global optimization storage and processing overhead increases rapidly as the number of sensor nodes in WSN increases.

4.2 Computational Complexity

Since sensor nodes have limited energy reserves and processing resources, it is critical to analyze our proposed MDP-based optimal policy's computational complexity, which is related to the convergence of the policy iteration algorithm. Since our problem formulation (Section 3) consists of finite states and actions, [22] proves a theorem that establishes the convergence of the policy iteration algorithm for finite states and actions in a finite number of iterations. Another important computational complexity factor is the algorithm's convergence rate. [22] shows that for a finite number of states and actions, the policy iteration algorithm converges to the optimal value function at least quadratically fast. Empirical observations suggest that the policy iteration algorithm can converge in $\mathcal{O}(\ln|S|)$ iterations where each iteration takes $\mathcal{O}(|S|^3)$ time (for policy evaluation), however, no proof yet exists to verify these empirical observations [23]. Based on these empirical observations for convergence, policy iteration algorithm can converge in 4 iterations for $|S|$ = 64 .

4.3 Data Memory Analysis

We performed data memory analysis using the 8-bit Atmel ATmega128L microprocessor [24] in XSM sensor nodes [25]. The Atmel ATmega128L microprocessor contains 128 KB of on-chip in-system reprogrammable flash memory for program storage, 4 KB of internal SRAM data memory, and up to 64 KB of external SRAM data memory. Integer and floating point data types require 2 and 4 bytes of storage, respectively. Our data memory analysis considers all storage requirements for our MDP-based dynamic tuning formulation (Section 3) including state space, action space, state dynamics (transition probability matrix), Bellman's equation (17), MDP reward function calculation (reward matrix), and policy iteration algorithm. We estimate data memory size for three sensor node configurations:

- 4 sensor node states with 4 allowable actions in each state (16 actions in the action space) (Fig. **??**)
- 8 sensor node states with 8 allowable actions in each state (64 actions in the action space)
- 16 sensor node states with 16 allowable actions in each state (256 actions in the action space)

Data memory analysis revealed that 4, 8, and 16 sensor node state configurations required approximately 1.55 KB, 14.8 KB, and 178.55 KB, respectively. Thus, currently available sensor node platforms contain enough memory resources to implement our MDP-based dynamic tuning methodology with 16 sensor nodes states or fewer. However, the memory requirements increase rapidly as the number of states increases due to the transition probability matrix and reward matrix specifications. Therefore, depending upon available memory resources, an application developer could restrict the number of states accordingly or otherwise would have to resort to back-end base station based computational policy as outlined in Section 4.1 to conserve power and storage.

5 Model Extensions

Our proposed MDP-based dynamic tuning methodology for WSNs is highly adaptive to different WSN characteristics and particulars, including additional sensor node tunable parameters (e.g., radio transmission power) and application metrics (e.g., reliability). Furthermore, our problem formulation can be extended to form MDP-based stochastic dynamic programs. Our current MDP-based dynamic optimization methodology provides a basis for MDP-based stochastic dynamic optimization that would react to changing environmental stimuli and wireless channel conditions to autonomously switch to an appropriate operating state. This stochastic dynamic optimization would provide a major incentive to use an MDP-based policy (because of the capability of MDP to formulate stochastic dynamic programs) as opposed to using lightweight heuristic policies (e.g., greedy- or simulated annealing-based) for parameter tuning that can determine an appropriate operating state out of a large state space without requiring large computational and memory resources [26].

To exemplify additional tuning parameters, we consider a sensor node's transceiver (radio) transmission power. The extended state space can be written as:

$$S = V_p \times F_p \times F_s \times P_{tx} \tag{20}$$

where P_{tx} denotes the state space for a sensor node's radio transmission power.

We define the sensor node's radio transmission power state space P_{tx} as:

$$P_{tx} = \{P_{tx_1}, P_{tx_2}, P_{tx_3}, \ldots, P_{tx_m}\} \;:\; |P_{tx}| = m \tag{21}$$

where $P_{tx_i} \in P_{tx} \;\forall\; i \in \{1, 2, 3, ..., m\}$ denotes a radio transmission power, m denotes the number of radio transmission power values, and $|P_{tx}| = m$ denotes the radio transmission power state space cardinality.

To exemplify the inclusion of additional application metrics, we consider reliability, which measures the reliability of sensed data, such as the total number of sensed data packets received at the sink node without error in an arbitrary time window. The reliability can be interpreted as the packet reception rate, which is the complement of the packet error rate (PER) [27]. The factors that affect reliability include wireless channel condition, network topology, traffic patterns, and the physical phenomenon that triggered the sensor node communication activity [27]. In general, the wireless channel condition has the most affect on the reliability metric, because sensed data packets may experience different error rates depending upon the channel condition. A sensor node may maintain application specified reliability in different wireless channel conditions by tuning/changing the error correcting codes, modulation schemes, and/or transmission power. The dynamic profiler module in our proposed tuning methodology helps estimating the reliability metric at runtime by profiling the number of packet transmissions from each sensor node and the number of packet receptions at the sink node.

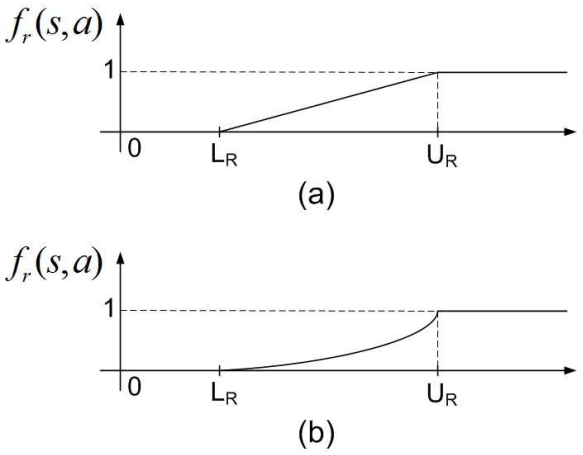

Fig. 2. Reliability reward functions: (a) linear variation; (b) quadratic variation.

The sensor node's reliability can be added to the reward function and the extended reward function can be written as:

$$\begin{aligned} f(s,a) \;\; = \;\; & \omega_p f_p(s,a) + \omega_t f_t(s,a) \\ & + \omega_d f_d(s,a) + \omega_r f_r(s,a) \end{aligned} \tag{22}$$

where $f_r(s,a)$ denotes the reliability reward function, ω_r represents the weight factor for reliability, and the remainder of the terms in (22) have the same meaning as in (11). The weight factors' constraints are given as $\sum_m \omega_m = 1$ where $m = \{p, t, d, r\}$ such that $0 \leq \omega_p \leq 1$, $0 \leq \omega_t \leq 1$, $0 \leq \omega_d \leq 1$, and $0 \leq \omega_r \leq 1$.

The reliability reward function (Fig. 2(a)) in (22) can be defined as:

$$f_r(s,a) = \begin{cases} 1, & r_a \geq U_R \\ (r_a - L_R)/(U_R - L_R), & L_R < r_a < U_R \\ 0, & r_a \leq L_R. \end{cases} \tag{23}$$

where r_a denotes the reliability offered in the current state given action a taken at time t and the constant parameters L_R and U_R denote the minimum and maximum allowed/tolerated reliability, respectively. The reliability may be represented as a multiple of a base reliability unit equal to 0.1, which represents a 10% packet reception rate [27].

The reward function capturing an application's metrics can be defined according to particular application requirements, and may vary quadratically (Fig. 2(b)) instead of linearly (as defined above) over the minimum and maximum allowed parameter values

and can be expressed as:

$$f_r(s,a) = \begin{cases} 1, & r_a \geq U_R \\ (r_a - L_R)^2/(U_R - L_R)^2, & L_R < t_a < U_R \\ 0, & r_a \leq L_R. \end{cases} \quad (24)$$

Thus, our proposed MDP-based dynamic tuning methodology works with any reward function formulation.

Currently, our problem formulation considers a DDP with fixed states and state transition probabilities equal to 1 (Section 3.3), however, our formulation can be extended to form stochastic dynamic programs with different state transition probabilities [28]. One potential extension could include environmental stimuli and wireless channel condition in the state space. The environmental stimuli and wireless channel condition vary with time and have a different probability of being in a certain state at a given point in time. For instance, considering the wireless channel condition along with the sensor node state, the state space vector $s(t)$ can be given as:

$$\begin{aligned} s(t) &= [s_s(t), s_c(t)] \\ &= [s_{s,1}(t), s_{s,2}(t), \ldots, s_{s,I}(t), \\ &\quad s_{c,1}(t), s_{c,2}(t), \ldots, s_{c,J}(t)] \end{aligned} \quad (25)$$

where $s_s(t)$ and $s_c(t)$ represent sensor state and wireless channel state at time t assuming that there are I total sensor states and J total wireless channel states. The state dynamics could be given by $p_t(j|t,s,a)$ which denotes the probability that the system occupies state j in t time units given s and a. If the wireless channel condition does not change state with time then $p_t(j|t,s,a) = 1 \ \forall \ t$ thus forming a DDP. The determination of $p_t(j|t,s,a)$ requires probabilistic modeling of wireless channel condition over time and is the focus of our future work.

6 Numerical Results

In this section, we present sensitivity analysis and convergence results for our MDP-based policy. Rest of the results are presented int the main paper.

6.1 Sensitivity Analysis

An application manager can assign values to MDP reward function parameters, such as $H_{i,a}$, L_P, U_P, L_T, U_T, L_D, U_D, ω_p, ω_t, and ω_d,[0] before a WSN's initial deployment according to projected/anticipated application requirements. However, the average sensor node lifetime (calculated from λ) may not be accurately estimated at the time of initial WSN deployment, as environmental stimuli and wireless channel conditions vary with time and may not be accurately anticipated. The sensor node lifetime depends on sensor node activity (both processing and communication), which varies with the changing environmental stimuli and wireless channel conditions. *Sensitivity analysis* analyzes the effects of changes in average sensor node lifetime after initial deployment on the expected total discounted reward. Thus, if the actual lifetime is different than the estimated lifetime, what is the loss in total expected discounted reward if the actual lifetime had been accurately predicted at deployment.

WSN sensitivity analysis can be carried out with the following steps: [5]:

1) Determine the expected total discounted reward given the actual average sensor node lifetime $l = 1/(1-\lambda)$, referred to as the *Optimal Reward* R_o.
2) Let $\hat{l}$ denote the estimated average sensor node lifetime and δl denote the percentage change from the actual average sensor node lifetime (i.e., $\hat{l} = (1+\delta l)l$). $\hat{l}$ results in a suboptimal policy with a corresponding suboptimal total expected discounted reward, referred to as *Suboptimal Reward* R_{so}.
3) The *Reward Ratio* r is the ratio of the suboptimal reward to the optimal reward (i.e., $r = R_{so}/R_o$), which indicates suboptimal expected total discounted reward variation with the average sensor node lifetime estimation inaccuracy.

It can be shown that the reward ratio varies from (0,2] as δl varies from (-100%, 100%]. The reward ratio's ideal value is 1, which occurs when the average sensor node lifetime is accurately estimated/predicted ($\hat{l} = l$ corresponding to $\delta l = 0$). Sensitivity analysis revealed that our MDP-based policy is sensitive to accurate determination of parameters, especially average lifetime, because inaccurate average sensor node lifetime results in a suboptimal expected total discounted reward. The dynamic profiler module (Fig. **??**) measures/profiles the remaining battery energy (lifetime) and sends this information to the application manager along with other profiled statistics (Section **??**), which helps in accurate estimation of λ. Estimating λ using the dynamic profiler's feedback ensures that the estimated average sensor node lifetime differs only slightly from the actual average sensor node lifetime, and thus helps in maintaining a reward ratio close to 1.

6.2 Number of Iterations for Convergence

The policy iteration algorithm determines π^{MDP} and the corresponding expected total discounted reward on the order of $O(ln(|S|))$ (Section 4.2). In our numerical results with four sensor node states, the policy iteration algorithm converges in two iterations on average.

Acknowledgments

This work was supported by the National Science Foundation (NSF) (CNS-0834080) and the Natural Sciences and Engineering Research Council of Canada (NSERC). Any opinions, findings, and conclusions or recommendations expressed in this material are those of

the author(s) and do not necessarily reflect the views of the NSF and NSERC.

REFERENCES

[1] C.-Y. Seong and B. Widrow, "Neural Dynamic Optimization for Control Systems," *IEEE Trans. on Systems, Man, and Cybernatics*, vol. 31, no. 4, pp. 482–489, August 2001.

[2] K. Hazelwood and M. D. Smith, "Managing Bounded Code Caches in Dynamic Binary Optimization Systems," *ACM Trans. on Architecture and Code Optimization*, vol. 3, no. 3, pp. 263–294, September 2006.

[3] H. Hamed, A. El-Atawy, and A.-S. Ehab, "On Dynamic Optimization of Packet Matching in High-Speed Firewalls," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 10, pp. 1817–1830, October 2006.

[4] S. Hu, M. Valluri, and L. K. John, "Effective Management of Multiple Configurable Units using Dynamic Optimization," *ACM Trans. on Architecture and Code Optimization*, vol. 3, no. 4, pp. 477–501, December 2006.

[5] E. Stevens-Navarro, Y. Lin, and V. Wong, "An MDP-based Vertical Handoff Decision Algorithm for Heterogeneous Wireless Networks," *IEEE Trans. on Vehicular Technology*, vol. 57, no. 2, pp. 1243–1254, March 2008.

[6] S. Sridharan and S. Lysecky, "A First Step Towards Dynamic Profiling of Sensor-Based Systems," in *Proc. of IEEE SECON*, San Francisco, California, June 2008.

[7] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure Tradeoffs for Sensor Networks," in *Proc. of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, Georgia, September 2002.

[8] S. Kogekar, S. Neema, B. Eames, X. Koutsoukos, A. Ledeczi, and M. Maroti, "Constraint-Guided Dynamic Reconfiguration in Sensor Networks," in *Proc. of ACM IPSN*, Berkeley, California, April 2004.

[9] I. Kadayif and M. Kandemir, "Tuning In-Sensor Data Filtering to Reduce Energy Consumption in Wireless Sensor Networks," in *Proc. of ACM DATE*, Paris, France, February 2004.

[10] P. Marrón and et al., "Adaptation and Cross-Layer Issues in Sensor Networks," in *Proc. of IEEE ISSNIP*, December, Melbourne, Australia 2004.

[11] A. Vecchio, "Adaptability in Wireless Sensor Networks," in *Proc. of IEEE ICECS*, September, Malta 2008.

[12] ITA, "International Technology Alliance in Network and Information Science," December 2010. [Online]. Available: http://www.usukita.org/

[13] P. Pillai and K. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," in *Proc. of ACM SOSP*, Banff, Alberta, Canada, October 2001.

[14] B. Childers, H. Tang, and R. Melhem, "Adapting Processor Supply Voltage to Instruction-Level Parallelism," in *Proc. of Koolchips Workshop, in conjunction with MICRO-33*, Monterey, California, December 2001.

[15] D. Liu and C. Svensson, "Trading Speed for Low Power by Choice of Supply and Threshold Voltages," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 1, pp. 10–17, January 1993.

[16] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, November 2000.

[17] R. Min, T. Furrer, and A. Chandrakasan, "Dynamic Voltage Scaling Techniques for Distributed Microsensor Networks," in *Proc. of IEEE WVLSI*, Orlando, Florida, April 2000.

[18] L. Yuan and G. Qu, "Design Space Exploration for Energy-Efficient Secure Sensor Network," in *Proc. of IEEE ASAP*, San Jose, California, July 2002.

[19] S. Lysecky and F. Vahid, "Automated Application-Specific Tuning of Parameterized Sensor-Based Embedded System Building Blocks," in *Proc. of IEEE UbiComp*, Orange County, California, September 2006.

[20] R. Verma, "Automated Application Specific Sensor Network Node Tuning for Non-Expert Application Developers," *M.S. Thesis, Department of Electrical and Computer Engineering, University of Arizona*, 2008.

[21] A. Munir, A. Gordon-Ross, S. Lysecky, and R. Lysecky, "A Lightweight Dynamic Optimization Methodology for Wireless Sensor Networks," in *Proc. of IEEE WiMob*, October, Niagara Falls, Canada 2010.

[22] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., 2005.

[23] T. Lane, "MDP Policy Iteration Lecture," April 2011. [Online]. Available: http://www.cs.unm.edu/∼ terran/

[24] ATMEL, "ATMEL ATmega128L 8-bit Microcontroller Datasheet," in *ATMEL Corporation*, San Jose, California, December 2010. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[25] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events," in *Proc. of ACM IPSN*, Los Angeles, California, April 2005.

[26] A. Munir, A. Gordon-Ross, S. Lysecky, and R. Lysecky, "A Lightweight Dynamic Optimization Methodology for Wireless Sensor Networks," in *Proc. of IEEE WiMob*, October 2010.

[27] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance in Dense Wireless Sensor Networks," in *Proc. of ACM SenSys*, Los Angeles, California, November 2003.

[28] F. Yu and V. Krishnamurthy, "Optimal Joint Session Admission Control in Integrated WLAN and CDMA Cellular Networks with Vertical Handoff," *IEEE Trans. on Mobile Computing*, vol. 6, no. 1, pp. 126–139, January 2007.