GEM-RL: Generalized Energy Management of Wearable Devices using Reinforcement Learning

Toygun Basaklar*, Yigit Tuncel*, Suat Gumussoy†, and Umit Ogras*
*Department of Electrical and Computer Engineering, University of Wisconsin - Madison, Madison, WI, USA
†Siemens Technology, Princeton, NJ, USA

***Abstract*—Energy harvesting (EH) and management (EM) have emerged as enablers of self-sustained wearable devices. Since EH alone is not sufficient for self-sustainability due to uncertainties of ambient sources and user activities, there is a critical need for a user-independent EM approach that does not rely on expected EH predictions. We present a *generalized energy management framework* (GEM-RL) using multi-objective reinforcement learning. GEM-RL learns the trade-off between utilization and the battery energy level of the target device under dynamic EH patterns and battery conditions. It also uses a lightweight approximate dynamic programming (ADP) technique that utilizes the trained MORL agent to optimize the utilization of the device over a longer period. Thorough experiments show that, on average, GEM-RL achieves Pareto front solutions within 5.4% of the offline Oracle for a given day. For a 7-day horizon, it achieves utility up to 4% within the offline Oracle and up to 50% higher utility compared to baseline EM approaches. The hardware implementation on a wearable device shows negligible execution time (1.98 ms) and energy consumption (23.17 μJ) overhead.**

***Index Terms*—Energy harvesting, multi-objective reinforcement learning, dynamic programming, energy management**

I. Introduction

Wearable devices have become widely popular for health monitoring applications, activity recognition, smart entertainment systems, and smart fashion [1]–[3]. These devices typically include a lightweight, small, rechargeable battery to avoid user discomfort. However, small battery capacities limit the computational power and operating lifetime of the device. Energy harvesting (EH) from ambient sources has emerged as a promising solution for self-sustainable wearable devices [4], [5]. However, relying only on EH is not sufficient to achieve self-sustainability due to the uncertainties of ambient sources. Therefore, recent research has proposed energy management (EM) approaches to improve the application performance while removing the manual recharge requirements [4], [6]–[8].

The primary goal of EM is to maximize the utilization of the device by judiciously allocating the available energy according to the users' and applications' needs. EM approaches should cope with the stochastic behavior of the harvested energy and user patterns. State-of-the-art research employs dynamic optimization approaches along with a prediction method that obtains expected EH values [8], [9]. Thus, their performance depends critically on the accuracy of these predictions. The major drawback of predictive approaches is the critical dependency on user activity and location [9]. Additionally, solving an optimization problem for a specific objective in a highly dynamic environment is not practical. For example, users may want to intervene to save battery for a given day which introduces a second dimension to the optimization problem. Therefore, there is a critical need for a *prediction-free* and *user-independent* EM approach that also *enables a multi-objective optimization*.

This work presents a **g**eneralized **e**nergy **m**anagement framework using multi-objective **r**einforcement **l**earning, GEM-RL. It learns the trade-off between utilization and the battery energy level of the target device under dynamic EH patterns and battery conditions without relying on forecasts of the harvested energy. GEM-RL transforms multi-dimensional objective space into a single dimension by assigning weights (preferences) to each objective [10]. It employs a multi-objective version of the TD3 (MO-TD3) [11] algorithm to obtain a single policy network that covers the entire preference space. The *inputs* of the policy network consist of battery and harvested energy-related values (*state*) and the preferences. This formulation enables an optimal policy (energy allocations) for any user-specified preference at run-time. We randomly sample an EH pattern, an initial battery energy level, and a preference vector for each episode during training. This enables the GEM-RL agent to generalize its policy for different EH patterns and battery energy conditions. GEM-RL also uses a lightweight approximate dynamic programming (ADP) approach that utilizes the trained agent to optimize the utilization of the device over a longer horizon. We also implemented an offline Oracle and two baseline EM approaches [4], [8] as comparison points. Extensive experiments show GEM-RL achieves Pareto front solutions within 5.4% of the Oracle for a given day. For a horizon of 7 days, it achieves utility up to 4% within the Oracle and up to 50% higher utility compared to baseline approaches. Hardware implementation of GEM-RL shows the execution time and energy consumption overhead per inference are 1.98 ms and 23.17 μJ, while the memory footprint of the framework is only 118 KB. *Our major contributions are:*

- GEM-RL, *a **g**eneralized **e**nergy **m**anagement framework* using *multi-objective **r**einforcement **l**earning* that learns the trade-off between any multiple objectives under various energy harvesting patterns and battery conditions,
- A broadly usable wearable device environment that can be used as a benchmark for evaluations of RL algorithms,
- Extensive evaluations that show near-optimal results within 5.4% of the Oracle, on average, for a given day and utility up to 4% within the Oracle and up to 50% higher utility compared to baseline approaches for a horizon of 7 days,
- Wearable hardware implementation with 118 KB memory footprint and 23.17 μJ per inference energy consumption.

978-3-9819263-7-8/DATE23/© 2023 EDAA

In the rest, Section II reviews the related work, while Section III provides an overview of the problem. Section IV formulates the RL environment dynamics and presents the proposed energy manager, GEM-RL. Finally, we evaluate and discuss the results in Section V and conclude the paper in Section VI.

II. Related Work

Recent research on energy harvesting and management focuses on optimizing the allocation of harvested energy to maximize device utilization while considering the application requirements and achieving self-sustainability [4], [7]–[9]. Kansal et al. [4] propose linear programming to determine the duty cycle of the target device. Their approach uses predictions of EH using an exponentially weighted moving average to determine future energy allocations. Bhat et al. [8] solve a relaxed convex optimization that maximizes the utility while achieving self-sustainability. Their approach first decides initial energy allocations for a given period based on the predictions of future EH values. It then makes corrections to these energy allocations by considering variations between future and actual EH values. Hussein et al. propose a two-stage approach, AdaEM, that adapts the uncertainties in the EH values and activities of the user [9]. The first stage involves a supervised learning method to learn the distribution and uncertainty of the EH of a specific user based on their activities and location. The second stage solves a dynamic, robust optimization problem that takes the output from the first stage to instantiate the problem. However, their approach is not generalizable since the activity prediction is trained for a specific user or set of users, and the quality of the decision of the second stage heavily depends on the accuracy of the first stage. Recently, RL-based approaches have emerged to achieve a prediction-free and generalizable EM. RLMan [7] is a recent EM approach that utilizes RL to maximize the packet generation rate in a point-to-point communication system. However, their setting and reward function does not generalize to other applications and performance metrics such as utilization and accuracy. A more recent study, tinyMAN [12], uses RL with a generalized reward function to obtain near-optimal energy allocations while achieving self-sustainability. The main drawback of tinyMAN is that it is trained on a cluster of users that has similar EH patterns which suggests that generalization aspect is absent.

In strong contrast to prior approaches, we propose a *generalized energy management framework* using *multi-objective* reinforcement learning (MORL), GEM-RL, that learns a trade-off between multiple objectives under dynamic EH patterns and battery conditions. GEM-RL employs a general utilization function [8] and the battery energy level of the target device as its objectives. It is *prediction-free* and *user-independent* as it learns from various EH patterns of 4772 users and various battery conditions. It also uses a lightweight ADP approach that utilizes the trained MORL agent to optimize the utilization of the device over a longer horizon. We also show that GEM-RL can be embedded in a wearable device prototype.

III. Overview and Preliminaries

The proposed GEM-RL framework considers an environment that consists of an energy harvesting wearable device with processing capability, an EH source, and a rechargeable battery, as depicted in Figure 1. This section overviews these components as a background for the approach presented in Section IV.

Wearable Device and its Utility: The wearable device implements the target application, such as health monitoring. It should provide a high quality of service (QoS) to the application while maintaining the battery energy at a certain level. This QoS is modeled by a utility function, which is a non-decreasing function of the energy allocated to the application. That is, the application can perform better (e.g., provide higher accuracy, process data faster, remain active longer) when allocated more energy. The benefits can diminish and eventually saturate if the extra energy becomes redundant due to another constraint, such as the device's maximum capacity. We emphasize that *GEM-RL supports any arbitrary non-decreasing utility function* unlike prior work that requires convex functions [8], [9], [13].

EH Source and Battery Dynamics: The EH source generates energy that can be either transferred to the device or stored in the battery. GEM-RL is oblivious to the type (e.g., solar) of energy harvesting source. It also *does not* rely on any predictions of the harvested energy, GEM-RL uses only the actual harvested energy (E_t^H) at the end of a time step.

In this work, GEM-RL uses one-day long episodes ($T =$ 24 hours) divided into one-hour time steps (t). We denote the battery energy level at the start of time interval t as E_t^B, the harvested and the allocated energy in time interval t as E_t^H, E_t^A respectively. There are two physical constraints on the battery energy level. The first constraint ensures that the battery maintains a minimum energy level, E_{min}^B, to stay idle so that the device does not turn itself off. The second constraint ensures that the battery energy level cannot exceed its limits, E_{max}^B. We can define the battery energy dynamics in this environment as:

$$\begin{aligned} &E_{t+1}^B = E_t^B + E_t^H - E_t^A,\ t \in T \\ &E_{min}^B \leq E_t^B \leq E_{max}^B,\ \ t \in T \end{aligned} \tag{1}$$

EH Dataset: GEM-RL can be trained with any available EH data. *To provide reproducible results that can be compared with prior work*, we use the publicly available *American Time Use Survey* [14] and solar/motion EH models from literature [5]. Our environment randomly draws the EH on each time step from this dataset. We emphasize that the proposed GEM-RL framework is general to work with any dataset, but we are not aware of any other publicly available data.

IV. Proposed Generalized Energy Management Framework - GEM-RL

A. Wearable Device Environment Dynamics for RL

RL techniques enabled breakthrough results in a wide range of topics, including autonomous driving, robotics, and gaming [11]. Impressive research progress in applying RL to these areas is primarily due to the broadly used open-source ML environments. For example, Google DeepMind's StreetLearn [15] environment and Microsoft Research's AirSim [16] lead to several novel approaches for autonomous driving. The first major contribution of this work is a general wearable device environment for RL. This environment enables us to apply the proposed MORL

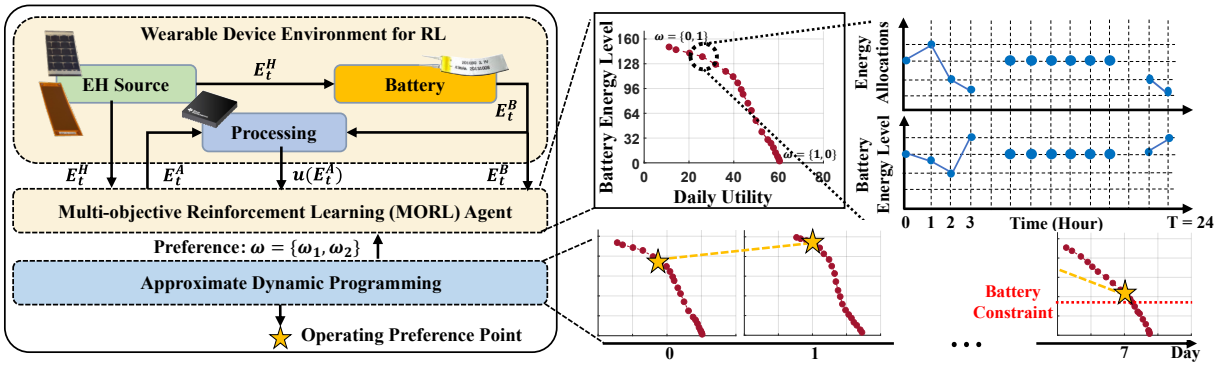

Fig. 1: Overview of GEM-RL

techniques and ADP to wearable device energy management. We also plan to release it to the public to catalyze research in this area and enable the broader RL community to use our wearable device environment as a benchmark.

The proposed wearable device environment is designed with generality in mind to enable any reinforcement learning technique. The **state space** is defined as $\mathcal{S} \subseteq \mathbb{R}^6$:

- E_t^B**:** The battery energy level at time t.
- E_{t-1}^H**:** The EH during previous time $t-1$.
- $\sum_{\tau=0}^{t-1} E_\tau^A$**:** The cumulative energy allocations until time t
- E_0^B**:** Initial battery energy at the beginning of a episode.
- t**:** The current time step.
- $\sum_{\tau=0}^{t-1} E_\tau^H$**:** The cumulative EH until time t.

The environment either generates random EH patterns (E^H) and initial battery energy conditions (E_0^B) during training or takes these as inputs to the system for evaluation purposes.

The one-dimensional **action space** corresponds to the allocated energy at every time step ($E_t^A \in \mathbb{R}$). It is bounded by E_{min}^A from below such that the device can stay in an idle state.

The agent can have one or **multiple objectives** to consider. In this work, we consider two competing objectives: the utility of the target device and the battery energy level. The goal of the agent is to learn the trade-off between the utility of the device and the battery energy level under certain constraints. As mentioned before, GEM-RL supports any arbitrary utility function. To provide a fair comparison with prior work, our experimental evaluations use the following utility function $u(E_t^A) = \ln(\frac{E_t^A}{E_{min}^A})$, used in literature [8].

In the MORL setting, the objectives generally have different scales. To alleviate this problem and to impose the constraints on the battery, we define our *reward function* as:

$$r_t = \begin{cases} [\alpha\gamma^t u(E_t^A),\ \beta\gamma^t \ln(E_t^B)] & E_t^B \geq E_{min}^B \\ [\left|E_t^B - E_{min}^B\right|,\ \left|E_t^B - E_{min}^B\right|] & E_t^B \leq E_{min}^B \\ [\alpha\gamma^t u(E_t^A),\ \left|E_t^B - E_{max}^B\right| + \beta\gamma^t \ln(E_t^B)] & E_t^B > E_{max}^B \end{cases} \tag{2}$$

For our experiments, we use the coefficients $\alpha = 1$ and $\beta = 3$ and the discount factor as $\gamma = 0.95$. We impose the minimum battery energy level constraint using $\left|E_t^B - E_{min}^B\right|$ and the maximum battery energy level constraint using $\left|E_t^B - E_{max}^B\right|$ and assigning $E_t^B = E_{max}^B$ if the battery energy level exceeds the maximum capacity of the battery.

An episode terminates if time $T = 24$ is reached or the battery is completely exhausted, $E_t^B \leq 0$. We develop our environment in Python and register it as an OpenAI's Gym [17] environment such that any RL algorithm can further adopt it. We plan to release it to the public under a GNU license.

B. Proposed MORL Framework

The goal of the MORL algorithm is to obtain a policy that yields a solution on the Pareto front for a given preference, as shown in Figure 1. The Pareto front is a set of solutions that are non-dominated by each other but are superior to the rest of the solutions in the solution space. In this work, we employ a recent MORL algorithm [11], MO-TD3, to learn to trade-off between utility and battery energy level. In this algorithm, the network takes preference vectors of the objectives as inputs along with the state vectors. It also employs novel approaches for efficient exploration of the preference space.

Algorithm 1 describes the training of the MORL agent for our framework. We measured the idle energy consumption of the target wearable device for an hour as 0.64 J. Therefore, we set E_{min}^A to this value. Similarly, the initial battery energy level (E_0^B) is set as 1 J to allow our MORL agent to cover the corner cases where the battery is drained.

At each episode, we randomly choose an EH distribution from the dataset, an initial battery energy level, and a preference vector, $\boldsymbol{\omega}$. The agent then interacts with the environment using its actor-network (π_ϕ). The transitions collected $(s_t, a_t, \mathbf{r}_t, s', \boldsymbol{\omega}'_j, done)$ are then stored in the experience replay buffer $\mathcal{D}$. Then, the algorithm samples a minibatch (B) of transitions from $\mathcal{D}$ and updates both the actor and critic networks. The training terminates when N number of time steps is reached. We implement GEM-RL in Python. The modified hyperparameters from [11] are given in Table I.

C. Proposed Approximate Dynamic Programming Technique

The MO-TD3 algorithm presented in the previous section determines near-optimal energy allocations in a single day for a given preference vector. Next, we need to determine the

TABLE I: Definition of the hyperparameters and their values.

Hyperparameter	Description	Value
N	Number of Time Steps	6×10^6
$\mathbb{D}$	Experience Replay Buffer Size	1×10^6
B	Minibatch size	128
η	Learning Rate	1×10^{-4}
N_L	Number of hidden layers	3
N_N	Number of hidden neurons	64

Algorithm 1: GEM-RL: MO-TD3

Initialize: Replay buffer $\mathcal{D}$, Critic networks $\mathbf{Q}_{\theta_1}$,$\mathbf{Q}_{\theta_2}$ and actor network π_ϕ with parameters θ_1, θ_2, and ϕ, Target networks $\mathbf{Q}_{\theta_1'} \leftarrow \mathbf{Q}_{\theta_1}$, $\mathbf{Q}_{\theta_2'} \leftarrow \mathbf{Q}_{\theta_2}$, $\pi_{\phi'} \leftarrow \pi_\phi$.
for n = 0: N **do**
  if $done = True$ **then**
    Reset the environment to random E_0^B and EH pattern.
    Sample a preference vector ω.
  Observe state s_t and select action a_t.
  Interact with the environment and store the transition $(s_t, a_t, \mathbf{r}_t, s', \boldsymbol{\omega}, done)$ in $\mathcal{D}$.
  Sample random B transitions from $\mathcal{D}$; $\tilde{a} \leftarrow \pi_{\phi'}(s', \boldsymbol{\omega})$
  $\mathbf{y} \leftarrow \mathbf{r} + \gamma \arg_{\mathbf{Q}} \min_{i=1,2} \boldsymbol{\omega}^T \mathbf{Q}_{\theta_i'}(s', \tilde{a}, \boldsymbol{\omega})$
  $Loss_{critic_k}(\theta_i) = \mathbb{E}\Big[\Big(\mathbf{y} - \mathbf{Q}_{\theta_i}(s, a, \boldsymbol{\omega})\Big)^2\Big]$
  $\nabla_\phi Loss_{actor_k}(\phi) = \mathbb{E}\Big[\nabla_a \ \boldsymbol{\omega}^T \mathbf{Q}_{\theta_1}(s, a, \boldsymbol{\omega}) \nabla_\phi \pi_\phi(s, \boldsymbol{\omega})\Big]$
  Update critic, actor, and target network parameters.

preferences that maximize the utility over a longer period of time, such as a week or month. This section presents our second major contribution, a lightweight approximate dynamic programming (ADP) algorithm that *maximizes the device utility over a longer horizon (H) while maintaining a certain battery energy level $E_{T_H}^B$ at the end of the horizon.*

Algorithm 2 outlines the proposed novel ADP algorithm. Suppose the time horizon H is divided in the days h, $0 \leq h \leq H$. The state $S_h = \{E_{0_h}^B, E_{T_h}^B\}$ consists of the initial battery energy level ($E_{0_h}^B$) and the target battery energy level $E_{T_h}^B$ for step h (Line 1). The action/decision corresponds to a preference vector at the beginning of each day $\boldsymbol{\omega} \in \Omega$ (Line 1). We define $U_h(\boldsymbol{\omega}, S_h) = \sum_{t=0}^{T-1} u(E_t^A)$ as the total utility obtained at the end of day h, starting from state S_h for the preference $\boldsymbol{\omega}$. Using these definitions, our DP formulation becomes maximizing the total utility while meeting a minimum battery constraint, E_{Cstr}^B, at the end of the horizon (Line 2):

$$\max_{\boldsymbol{\omega} \in \Omega} \sum_{h=0}^{H} U_h(\boldsymbol{\omega}, S_h), \quad st \quad E_{T_H}^B \geq E_{Cstr}^B \tag{3}$$

Next, we express the principle of optimality [18] as: $\max_{\boldsymbol{\omega} \in \Omega} \ U_{h=H}(\boldsymbol{\omega}, S_h) + U_{h-1}^*(\boldsymbol{\omega}, S_{h-1})$. Here, the $*$ symbol denotes the optimality of a given function. Then, we solve this equation using a backward recursion since we know $E_{T_H}^B \geq E_{Cstr}^B$ should be satisfied at the end of the horizon. However, an exact DP solution is not feasible on a wearable device due to the limited computational resources. Therefore, we approximate $U_{h-1}^*(\boldsymbol{\omega}, S_{h-1})$ at each step except the first step ($h = 0$). To this end, we first generate an energy battery level set by dividing the battery energy level range, $[0, 160]$, into M equal values (Line 4). We then use these M levels as the initial battery energy levels in state vector $S_{h,m} = \{E_{0_{h,m}}^B, E_{T_h}^B\}$ $\forall m \in M$. Then, we use uniformly distributed energy allocations based on the expected EH, and actual initial energy battery level, $E_{0_{h=0}}^B$:

$$E_{t,m}^A = \frac{E_{0_{h=0}}^B - E_{0_{h,m}}^B + \sum_{h=0}^{h-1} \sum_{\tau=0}^{T} E_\tau^H}{T(h-1)}, t = \{0, T(h-1)\} \tag{4}$$

Using these energy allocations, we then obtain the previous utility values as $U_{h-1}^*(\boldsymbol{\omega}, S_{h-1,m})$.

Algorithm 2: GEM-RL: DP Technique

State: $S_h = \{E_{0_h}^B, E_{0_h}^B\}$, **Action:** $\boldsymbol{\omega} \in \Omega$
Formulation: $\max_{\boldsymbol{\omega} \in \Omega} \ \sum_{h=0}^{H} U_h(\boldsymbol{\omega}, S_h)$ **s.t** $E_{T_H}^B \geq E_{Cstr}^B$
$\max_{\boldsymbol{\omega} \in \Omega} \ U_{h=H}(\boldsymbol{\omega}, S_h) + U_{h-1}^*(\boldsymbol{\omega}, S_{h-1})$
Initialize: $E_{0_m}^B$ $m \in M$ and $E_{Cstr}^B = 80 + \Delta$
for h = H: 0 **do**
  if $h = 0$ **then**
    $\arg\max_{\boldsymbol{\omega}} \ U_h(\boldsymbol{\omega}, S_h)$ **subject to** $E_{T_h}^B \geq E_{Cstr}^B$
  else
    $E_{t,m}^A$ given by Equation 4
    Find $U_{h-1}^*(\boldsymbol{\omega}, S_{h-1,m})$ $\forall$ $\boldsymbol{\omega} \in \Omega, m \in M$ with $E_{t,m}^A$.
    Find $U_h(\boldsymbol{\omega}, S_{h,m})$ $\forall \boldsymbol{\omega} \in \Omega, m \in M$ using MORL.
    $\arg\max_{\boldsymbol{\omega}, m} \ U_h(\boldsymbol{\omega}, S_{h,m}) + U_{h-1}^*(\boldsymbol{\omega}, S_{h-1})$
    subject to $E_{T_h}^B \geq E_{Cstr}^B$
    Assign $E_{Cstr}^B = E_{T_{h-1}}^B$

We run our trained MORL agent for each $\{\boldsymbol{\omega}, S_{h,m}\}$ pair to calculate the utility for day h. Then, we simply select the pair $\{\boldsymbol{\omega}, S_{h,m}\}$ that achieves the maximum of the $U_h(\boldsymbol{\omega}, S_{h,m}) + U_{h-1}^*(\boldsymbol{\omega}, S_{h-1,m})$ while achieving the E_{Cstr}^B constraint. We then use this state's initial battery energy ($E_{0_{h-1,m}}^B$) as the target battery energy E_{Cstr}^B of the previous step, $h-1$. We recursively apply this to decide the battery constraints E_{Cstr}^B for the previous steps until we reach the first step, $h = 0$. As a result, the proposed ADP approach optimizes the utility of the device by determining the preference vector $\boldsymbol{\omega}$ at the beginning of each day over a longer horizon. It also calculates the difference between the actual battery energy level at the end of the day and the expected E_{Cstr}^B for step $h = 0$ and feeds this difference (Δ) back to the algorithm to correct the E_{Cons}^B at step $h = H$ for the next day.

V. Experimental Evaluations

A. Experimental Setup

Optimal Oracle: We designed an offline Oracle using CVX [19]. It solves the following optimization problem for a given preference and finite horizon:

$$\begin{aligned} maximize \quad & \sum_{t=0}^{T-1} \alpha \gamma^t \omega_1 u(E_t^A) + \beta \gamma^t \omega_2 \ln(E_{t+1}^B) \\ subject\ to \quad & E_t^A > E_{min}^A \text{ and } E_{min}^B \leq E_t^B \leq E_{max}^B \end{aligned} \tag{5}$$

Since the Oracle uses the actual harvested energy values and allocations, it is infeasible, but it provides optimal results.

Baseline Approaches: We implemented two prior approaches [4], [8] for comparison. We use the median user's expected EH pattern in the EH dataset described in Section III, and extend it over the horizon of 7 days. Both approaches use these future EH values to determine future energy allocations.

Evaluation conditions: The training set excludes the users corresponding to 10^{th}, 30^{th}, 50^{th}, 70^{th}, 90^{th} percentile cumulative EH in the dataset. We denote these EH patterns as EH_1, EH_2, EH_3, EH_4, EH_5, respectively. We also initialize the battery (E_0^B) at three different levels: 20% (32 J), 50% (80

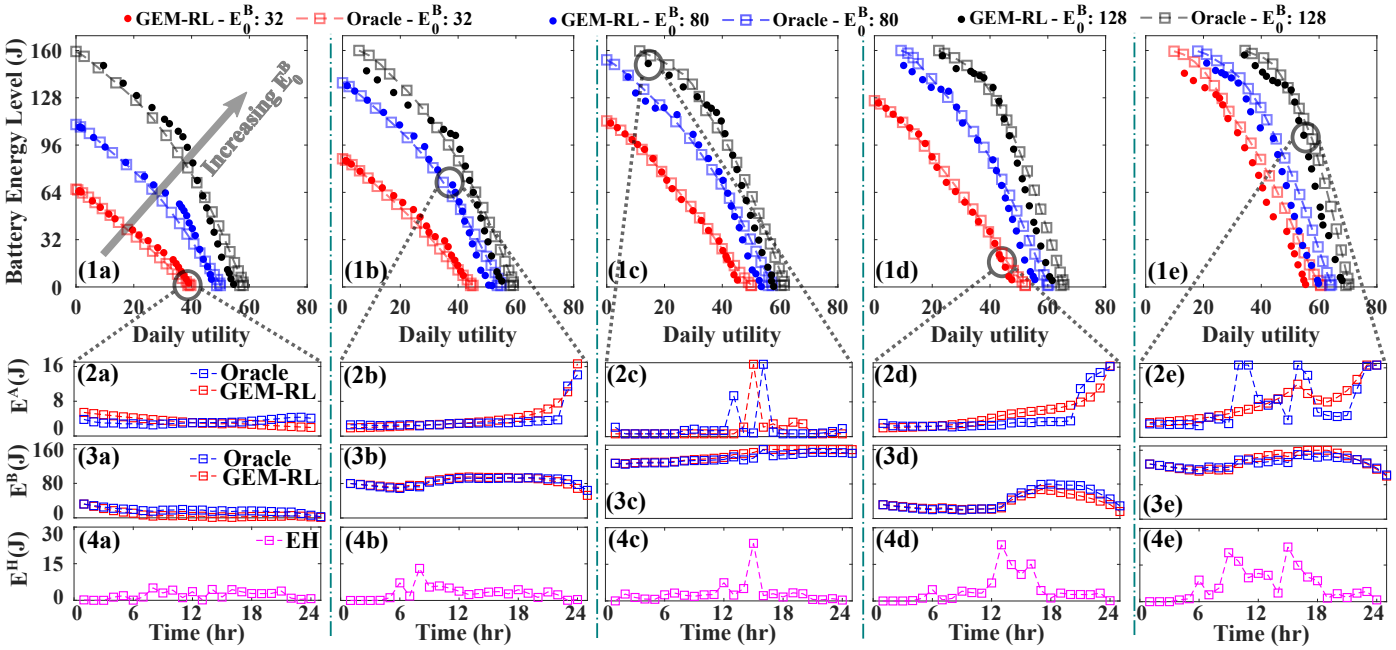

Fig. 2: (1): Comparison of Pareto front solutions between an offline Oracle and GEM-RL using EB_1, EB_2 and EB_3 (a-e) EH_1, EH_2, EH_3, EH_4, EH_5, respectively. (2-4): Energy allocations, battery energy level, harvested energy obtained by Oracle and GEM-RL for a) EB_1, $\omega = \{1, 0\}$, b) EB_2, $\omega = \{0.5, 0.5\}$, c) EB_3, $\omega = \{0, 1\}$, d) EB_1, $\omega = \{0.75, 0.25\}$, e) EB_3, $\omega = \{0.5, 0.5\}$

J), 80% (128 J) of the battery capacity (160 J) and denote these as EB_1, EB_2, EB_3, respectively. We refer the combinations of the EH patterns and battery levels as the *evaluation set*.

B. Performance Evaluation of our MORL Agent

Figure 2 shows the Pareto front solutions of the Oracle and GEM-RL for each configuration in the evaluation set. Specifically, Figure 2(1a) plots the Pareto front solutions for three levels of E_0^B for the user with an EH pattern of EH_1. We observe that GEM-RL closely follows the Pareto front obtained by the Oracle. For example, the mean absolute percentage error (MAPE) with respect to Oracle averaged over all preferences obtained by GEM-RL is 5.5% for EB_1 (red curve). Similarly, the MAPE is 6.4% and 6.8% for EB_2 (blue) and EB_3 (black), respectively. The remaining plots (1b)–(1e) show that GEM-RL consistently performs very close to the offline Oracle. For instance, the MAPE over all preferences for EB_1, EB_2 and EB_3 is 6.0%, 3.81%, 3.88%, respectively, for EH_5 shown in Figure 2(1e). Considering all configurations in the evaluation set, the total MAPE averaged over all preferences is only 5.4%. We further investigate the behavior of our approach by presenting the energy allocations in a day for a given initial battery energy level, EH pattern, and preference in Figure 2(2a-e). Our energy allocations closely follow those generated by Oracle. For example, Figure 2(2a) and (3a) show the energy allocations and battery energy levels obtained by the Oracle and GEM-RL for EB_1 with $\omega = \{1, 0\}$. In fact, this configuration is a corner case where the initial battery energy level is 20% (EB_1), the harvested energy is low (EH_1), and the preference ($\omega = \{1, 0\}$) is set to maximize the utility without considering the battery energy level. GEM-RL closely follows the Oracle even in this condition in terms of energy allocations and battery energy level, and its utility is within 0.5% of the Oracle's, without violating any constraints. Moreover, we observe that GEM-RL responds to changes in EH similar to the Oracle. The error values and the Pareto front and individual solutions illustrated in Figure 2 clearly demonstrate that *GEM-RL successfully optimizes the trade-off between utilization and the battery energy level for any unseen EH pattern, battery condition, and preference without relying on the forecasts of harvested energy using a single extremely light-weight policy, as demonstrated in Section V-D.*

C. Performance Evaluation of GEM-RL for Longer Period

The previous section shows the performance of the MORL algorithm throughout a single day. This section evaluates GEM-RL's novel ADP approach, over 7-day horizon ($H = 7$) with $M = 25$ battery levels and $E^B_{Cstr} = 80$ J final battery level constraint. Figure 3 shows the cumulative utility, daily utility, and battery energy levels obtained by the Oracle, baseline approaches [4], [8], and GEM-RL for EB_1, EB_2 and EB_3. Since the prior approaches [4], [8] decide future allocations from the predicted EH values, they completely drain the battery on the first day for corner cases such as low initial battery and low EH as shown in Figure 3(1-3a). To analyze the performance of the approaches, we calculate the MAPE of cumulative utilities over 7 days with respect to Oracle for each EH pattern in the evaluation set. We set the MAPE to 100% if the algorithm completely fails to keep the device on. Table II shows that GEM-RL achieves cumulative utility up to 4% within the Oracle and up to 50% higher cumulative utility compared to baseline approaches for 7-day horizon. Additionally, GEM-RL does not violate any constraints, whereas the baseline approaches [4] and [8], cause violations 16% and 10% of the total instances, respectively. Thus, we conclude that GEM-RL achieves near-optimal energy allocations with various EH patterns and battery conditions for an arbitrary horizon.

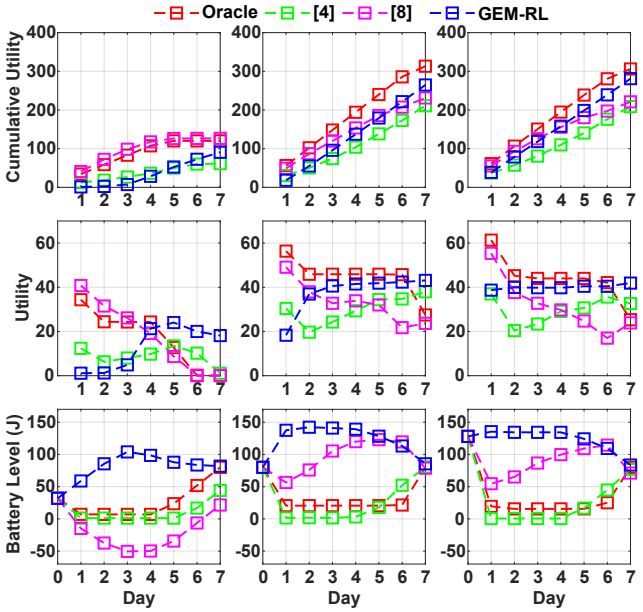

Fig. 3: Comparison of cumulative utility, daily utility, and battery energy level for a longer period of time between an offline Oracle, baseline approaches, GEM-RL for (a) EB_1, EH_1, (b) EB_2, EH_4, and (c) EB_3, EH_3.

D. Energy-Performance Evaluation of GEM-RL on Hardware

We deployed our MORL agent using TensorFlow Lite for Micro (TFLM) flow [20] and implemented the GEM-RL framework on the TI CC2652R microcontroller (MCU). The MCU has an ARM Cortex M4F running at 48 MHz and has 352 KB of flash memory and 80 KB of SRAM. The execution time and energy consumption overhead of a single policy network call are measured as 1.98 ms and 23.17 µJ, respectively. There are 24 network calls in one day for the specific preference. To determine this preference, the total execution time and energy consumption overhead of the ADP solutions are 7.32 s and 84.12 mJ, respectively. Hence, in total, the GEM-RL framework takes up to 7.37 s and consumes 84.58 mJ energy in a single day. These results are negligible compared to 24 hours and 160 J battery capacity. Moreover, considering the TFLM operators, inputs, outputs, intermediate values, and our agent's network weights, the memory footprint of GEM-RL becomes 118 KB. These results suggest that GEM-RL provides an efficient general EM framework that is deployable on a wearable device.

TABLE II: MAPE of cumulative utilities over 7-day horizon for each configuration in the evaluation set w.r.t. Oracle. For each EH pattern, the MAPE of EB_1, EB_2, EB_3 are averaged.

	EH_1	EH_2	EH_3	EH_4	EH_5
[4]	100%	100%	28%	25%	44%
[8]	48%	47%	32%	32%	22%
GEM-RL	4%	5%	8%	16%	20%

VI. Conclusion

Energy management approaches for wearable devices must be user-independent, prediction-free, and able to handle multiple objectives to be practical. This paper presented GEM-RL, a generalized energy management framework using MORL that learns trade-offs between multiple objectives under different EH patterns and battery conditions. It determines near-optimal energy allocations in a single day for a given preference. It also uses a novel dynamic programming technique to maximize the utility over longer periods. GEM-RL achieves Pareto front solutions within 5.4% of Oracle, on average, for a given day. For 7-day horizon, it achieves utility up to 4% within the Oracle and up to 50% higher utility compared to baseline approaches. Hardware measurements show that it has 1.98 ms and 23.17 µJ per inference execution time and energy consumption overhead.

Acknowledgment

This work was supported in part by NSF CAREER award (CNS-1651624) and DARPA YFA Grant (D14AP00068).

References

[1] T. Basaklar, Y. Tuncel, S. An, and U. Ogras, "Wearable devices and low-power design for smart health applications: challenges and opportunities," in *Intl. Symp. on Low Power Electronics and Design*, 2021, pp. 1–1.

[2] G. Sucharitha, B. Tannmayee, and K. Dwarakamai, "Revolution in iot: Smart wearable technology," in *Internet of Things and Its Applications*, 2022, pp. 407–425.

[3] G. Bhat, R. Deb, and U. Y. Ogras, "Openhealth: open-source platform for wearable health monitoring," *IEEE Design & Test*, vol. 36, no. 5, pp. 27–34, 2019.

[4] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embed. Comput. Syst. (TECS)*, vol. 6, no. 4, pp. 32–es, 2007.

[5] Y. Tuncel, T. Basaklar, and U. Ogras, "How much energy can we harvest daily for wearable applications?" in *Intl. Symp. on Low Power Electronics and Design*, 2021, pp. 1–6.

[6] Y. Tuncel, G. Bhat, J. Park, and U. Y. Ogras, "Eco: Enabling energy-neutral iot devices through runtime allocation of harvested energy," *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 4833–4848, 2021.

[7] F. A. Aoudia, M. Gautier, and O. Berder, "Rlman: An energy manager based on reinforcement learning for energy harvesting wireless sensor networks," *IEEE Trans. Green Commun. Netw., 2(2)*, pp. 408–417, 2018.

[8] G. Bhat, J. Park, and U. Y. Ogras, "Near-optimal energy allocation for self-powered wearable systems," in *Intl. on Conf. Comput.-Aided Des.*, 2017, pp. 368–375.

[9] D. Hussein, G. Bhat, and J. R. Doppa, "Adaptive energy management for self-sustainable wearables in mobile health," in *Proc. AAAI*, 2022.

[10] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2014.

[11] T. Basaklar, S. Gumussoy, and U. Y. Ogras, "Pd-morl: Preference-driven multi-objective reinforcement learning algorithm," *arXiv preprint arXiv:2208.07914*, 2022.

[12] T. Basaklar, Y. Tuncel, and U. Y. Ogras, "tinyman: Lightweight energy manager using reinforcement learning for energy harvesting wearable iot devices," *arXiv preprint arXiv:2202.09297*, 2022.

[13] N. Yamin, G. Bhat, and J. R. Doppa, "Diet: a dynamic energy management approach for wearable health monitoring devices," in *Design, Automation & Test in Europe Conf. & Exhibition*, 2022, pp. 1365–1370.

[14] US Department of Labor, "American Time Use Survey," 2018, https://www.bls.gov/tus/, accessed 1 March 2021.

[15] P. Mirowski *et al.*, "Learning to navigate in cities without a map," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[18] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[19] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[20] R. David *et al.*, "Tensorflow lite micro: Embedded machine learning on tinyml systems," *arXiv preprint arXiv:2010.08678*, 2020.