Multi-Objective and Energy Efficient Reinforcement Learning for Edge AI Applications

Toygun Basaklar
Ph.D. Oral Defense
29 May 2024
Motivation

PDMORL: Preference Driven Multi-Objective Reinforcement Learning Algorithm

A Comprehensive Multi-Objective Energy Management Approach for Wearable Devices with Dynamic Energy Demands

A Self-Sustained CPS Design for Reliable Wildfire Monitoring

DTRL: Decision Tree-based Multi-Objective Reinforcement Learning for Runtime Task Scheduling in Domain-Specific System-on Chips

Conclusions, Future Directions and Closing Remarks
What is Reinforcement Learning?

Image Sources:
What is Reinforcement Learning?

Image Sources:
Applicability to Real-World Problems

- Many real-world tasks involve multiple, possibly conflicting, objectives
Applicability to Real-World Problems

- Energy-Awareness
- Hardware Friendliness
- Multi-Objective Functionality
### Applicability to Real-World Problems

- **Energy-Awareness**
  - Critical for *resource-constrained environments*
  - Extend the operational life and reduce overall energy consumption
  - *Sustainable and cost-effective*

- **Hardware Friendliness**

- **Multi-Objective Functionality**
### Applicability to Real-World Problems

<table>
<thead>
<tr>
<th>Energy-Awareness</th>
<th>Hardware Friendliness</th>
<th>Multi-Objective Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Critical for <strong>resource-constrained environments</strong></td>
<td>• Ensures RL algorithms can be efficiently implemented on various hardware platforms</td>
<td></td>
</tr>
<tr>
<td>• Extend the operational life and reduce overall energy consumption</td>
<td>• <strong>Considers runtime requirements of the application</strong></td>
<td></td>
</tr>
<tr>
<td>• Sustainable and cost-effective</td>
<td>• <strong>Computationally efficient</strong> and can operate within the constraints of the application</td>
<td></td>
</tr>
</tbody>
</table>
Applicability to Real-World Problems

**Energy-Awareness**
- Critical for resource-constrained environments
- Extend the operational life and reduce overall energy consumption
- Sustainable and cost-effective

**Hardware Friendliness**
- Ensures RL algorithms can be efficiently implemented on various hardware platforms
- Considers runtime requirements of the application
- Computationally efficient and can operate within the constraints of the application

**Multi-Objective Functionality**
- Allows RL algorithms to simultaneously consider and optimize multiple objectives
- Provides solutions that are more aligned with real-world needs.
Challenges in Multi-objective RL (MORL)

• **Balancing Trade-offs Between Multiple Objectives**
  • Performance is measured using multiple-objectives
  • **Multiple Pareto-optimal solutions**
  • Diverse and representative solutions
Challenges in Multi-objective RL (MORL)

- **Balancing Trade-offs Between Multiple Objectives**
  - Performance is measured using multiple-objectives
  - *Multiple Pareto-optimal solutions*
    - Diverse and representative solutions

- **Scalability and Efficiency**
  - Repetitive training for different preference settings
  - Computational complexity
Challenges in Multi-objective RL (MORL)

• **Balancing Trade-offs Between Multiple Objectives**
  - Performance is measured using multiple-objectives
  - **Multiple Pareto-optimal solutions**
  - Diverse and representative solutions

• **Scalability and Efficiency**
  - Repetitive training for different preference settings
  - Computational complexity

• **Real-World Applicability**
  - Feasible and efficient solutions
  - Wearable devices – Limited battery life and computational power - Energy-efficient algorithms
Contributions

Energy-Awareness

Hardware Friendliness

Multi-Objective Functionality

A Comprehensive Multi-Objective Energy Management Approach for Wearable Devices with Dynamic Energy Demands

A Self-Sustained CPS Design for Reliable Wildfire Monitoring

DTRL: Decision Tree-based Multi-Objective Reinforcement Learning for Runtime Task Scheduling in Domain-Specific System-on Chips

PDMORL: Preference Driven Multi-Objective Reinforcement Learning Algorithm
Agenda

Motivation

PDMORL: Preference Driven Multi-Objective Reinforcement Learning Algorithm

A Comprehensive Multi-Objective Energy Management Approach for Wearable Devices with Dynamic Energy Demands

A Self-Sustained CPS Design for Reliable Wildfire Monitoring

DTRL: Decision Tree-based Multi-Objective Reinforcement Learning for Runtime Task Scheduling in Domain-Specific System-on Chips

Conclusions, Future Directions and Closing Remarks
Multi-Objective Reinforcement Learning (MORL)

Agent

Environment

Optimal state-action value function

\[ Q^*(s, a) = \max_{\pi} Q^\pi(s, a) \]

Optimal vectorized state-action value function

\[ MQ^*(s, a) = \max_{\pi} MQ^\pi(s, a) \]

s.t. \[ MQ^\pi(s, a) = [Q_1^\pi(s, a), ..., Q_N^\pi(s, a)]^T \]

Objective 1

Objective 2

Non-optimal

Pareto front
• Many real-world tasks involve **multiple, possibly conflicting**, objectives.
• There are **multiple Pareto-optimal solutions as a function of the preference between objectives** and performance of the algorithm is measured using multiple objectives.
• Multi-objective reinforcement learning (MORL) approaches **maximize a vector rewards** depending on the preferences for each objective.
Pareto-Optimality and Quality of the Pareto Front

- **Pareto-optimality**: A policy \( \pi \) is Pareto optimal if there is no other policy \( \pi' \) that improves its expected return for an objective without degrading the expected return of any other objective.

- **Quality of the Pareto front**:
  - **Hypervolume**: Area/Volume between a reference point and solutions on Pareto front.
  - **Sparsity**: The average square distance between consecutive solutions on the Pareto front.

- **Desired behavior**: Large hypervolume, low sparsity.
• **Scalarization → Standard RL algorithms** → A policy optimized for the given preference
  • Deciding the preferences requires application domain expertise
  • A single solution for a given set of goals and constraints (preferences)

• MORL approaches:
  • **Scalable** to various domains
  • Cover the entire preference space with a single universal policy

• A novel MORL algorithm, **Preference-Driven Multi-Objective Reinforcement Learning (PD-MORL)**
  • Single policy network
  • Covers entire preference space
  • Scalable to complex application domains such as continuous robotics tasks

![Objective](image)

<table>
<thead>
<tr>
<th>Energy Efficiency</th>
<th>Forward Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \star )</td>
<td>a)</td>
</tr>
<tr>
<td>( \star )</td>
<td>b)</td>
</tr>
<tr>
<td>( \star )</td>
<td>c)</td>
</tr>
</tbody>
</table>
Motivation - Preference-Driven MORL

- Solution vectors should be aligned with the preference vectors
  - Alignment: Cosine similarity, Angle to the origin
- In RL, Q-values are approximations for the actual solutions
Motivation - Preference-Driven MORL

- Solution vectors should be aligned with the preference vectors
  - Alignment: Cosine similarity, Angle to the origin
- In RL, Q-values are approximations for the actual solutions
- Bellman Optimality Operator

\[
TQ(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s, a)} \max_{a'} Q(s', a')
\]

- Preference-Driven Multi-objective Bellman Optimality Operator

\[
TQ(s, a, \omega) := r(s, a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s, a)} \max_{a'} \left( S_c(\omega, Q(s', a', \omega)) \cdot (\omega^T Q(s', a', \omega)) \right), \omega
\]

where \( S_c(\omega, Q(s', a', \omega)) \) denotes the cosine similarity between preference vector \( \omega \) and Q-value
• Extend double deep Q-network (DDQN) to a multi-objective version (MO-DDQN) with the preference-driven optimality operator

• Minimize the loss:

\[ L_k(\theta) = \mathbb{E}_{(s,a,r,s',\omega) \sim D} \left[ (y - Q(s, a, \omega; \theta))^2 \right] \]

• \( y = r + \gamma Q \left( s', \max_a \left( S_c(\omega, Q(s', a', \omega)) \cdot (\omega^T Q(s', a', \omega)) \right), \omega; \theta' \right) \) is the preference driven target value
Other Novelties

- **Hindsight Experience Replay Buffer (HER)
  - For each episode during training, a preference vector is sampled from a uniform distribution.
  - Every transition is also stored with $N_\omega$ randomly sampled preferences different than the original preference of the transition.
  - Provides *efficient exploration* and *generalizability* to the agent.

- **Exploration in Parallel
  - Divide preference space into sub-spaces and assign child processes to these subspaces.

![Diagram showing progression in objectives](image-url)
Results for Deep Sea Treasure (DST)

- PD-MORL learns a single universal policy that can work with any preference vector specified at run-time
• Extend the TD3 algorithm to a **multi-objective version (MO-TD3)**

• We add a directional angle term to both actor's and critic's loss function:

$$L_{critic_k}(\theta_i) = \mathbb{E}_{(s,a,r,s',\omega) \sim D} \left[ \left( y - Q(s, a, \omega; \theta_i) \right)^2 \right] + \mathbb{E}_{(s,a,\omega) \sim D} \left[ g(\omega_p, Q(s, a, \omega; \theta_i)) \right]$$

$$\nabla_{\phi} L_{actor_k}(\phi) = \mathbb{E}_{(s,a,r,s',\omega) \sim D} \left[ \nabla_{a} \omega^T Q(s, a, \omega; \theta_1)|_{a=\pi(s,\omega;\phi)} \nabla_{\phi} \pi(s, \omega; \phi) \right] + \alpha \mathbb{E}_{(s,a,\omega) \sim D} \left[ \nabla_{a} g(\omega_p, Q(s, a, \omega; \theta_1)|_{a=\pi(s,\omega;\phi)} \nabla_{\phi} \pi(s, \omega; \phi) \right]$$

• The directional angle term:

$$g(\omega, Q(s, a, \omega; \theta)) = \cos^{-1} \left( \frac{\omega^T Q(s, a, \omega; \theta)}{||\omega|| \cdot ||Q(s, a, \omega; \theta)||} \right)$$

denotes the **angle between preference vector and multi-objective Q-values**
MORL Benchmarks with Continuous Control Tasks

- **MO-Walker2d-v2**: $S \subseteq \mathbb{R}^{17}, A \subseteq \mathbb{R}^{6}$.
  - **Agent**: Two-dimensional legged figure.
  - **Reward**: Two-element vector: \{Forward Speed, Energy Efficiency\}

- **MO-HalfCheetah-v2**: $S \subseteq \mathbb{R}^{17}, A \subseteq \mathbb{R}^{6}$.
  - **Agent**: Two-dimensional robot that resembles a cheetah.
  - **Reward**: Two-element vector: \{Forward Speed, Energy Efficiency\}

- **MO-Ant-v2**: $S \subseteq \mathbb{R}^{27}, A \subseteq \mathbb{R}^{8}$.
  - **Agent**: Three-dimensional robot that resembles an ant.
  - **Reward**: Two-element vector: \{X-axis speed, Y-axis speed\}

- **MO-Swimmer-v2**: $S \subseteq \mathbb{R}^{8}, A \subseteq \mathbb{R}^{2}$.
  - **Agent**: Two-dimensional robot.
  - **Reward**: Two-element vector: \{Forward Speed, Energy Efficiency\}

- **MO-Hopper-v2**: $S \subseteq \mathbb{R}^{11}, A \subseteq \mathbb{R}^{3}$.
  - **Agent**: Two-dimensional one-legged figure.
  - **Reward**: Two-element vector: \{Forward Speed, Jumping Height\}
Results for Continuous Control Tasks

- Compare PD-MORL to prior work in terms of **hypervolume and sparsity**
- **Desired** Pareto front approximation should have **high hypervolume** and **low sparsity** metrics
- PD-MORL **outperforms** the current SOTA approaches
- PD-MORL trains **only a single network**, while the other methods use customized policy networks for each Pareto optimal solution
  - **PD-MORL (This work):** # of trainable parameters: $3.4 \times 10^5$

\[
\begin{array}{cccccc}
\text{MO-Walker2d-v2} & \text{MO-HalfCheetah-v2} & \text{MO-Ant-v2} & \text{MO-Swimmer-v2} & \text{MO-Hopper-v2} \\
0.2 & 0.4 & 0.6 & 0.8 & 1 \\
1.2 & \text{HV} & \text{Sparsity} & \text{HV} & \text{Sparsity} & \text{HV} & \text{Sparsity} & \text{HV} & \text{Sparsity} \\
\end{array}
\]
Results for Continuous Control Tasks

**MO-Walker2d-v2**

- a) Prioritize energy efficiency
- b) Balance objectives
- c) Prioritize forward speed

**MO-Ant-v2**

- a) Prioritize y-axis speed
- b) Balance objectives
- c) Prioritize x-axis speed
Agenda

Motivation

PDMORL: Preference Driven Multi-Objective Reinforcement Learning Algorithm

A Comprehensive Multi-Objective Energy Management Approach for Wearable Devices with Dynamic Energy Demands

A Self-Sustained CPS Design for Reliable Wildfire Monitoring

DTRL: Decision Tree-based Multi-Objective Reinforcement Learning for Runtime Task Scheduling in Domain-Specific System-on Chips

Conclusions, Future Directions and Closing Remarks
Self-Sustained Wearable Devices

- IoT Devices have limited battery life
  - Bulky batteries are inflexible, while flexible batteries have low capacity
  - Small form factor limits the battery capacity
    - Oura Ring 3: 22 mAh @ 3.7V battery

- **Compliance challenge:** Users do not want another device to manage and maintain!
- **The remedy:** Reduce dependency on battery with **Self-sustained operation**
The Three Pillars of Self-Sustained Operation

1 – Wearable Energy Harvesting

2 – Optimal Energy Management

3 – Energy Consumption Optimizations: SW and HW
• **Achieving Self-Sustained Operation is challenging**
  • Uncertainty in harvested energy ~ Quality-of-Service requirements of the application
  • Application’s energy demand is dynamic
    • e.g. Activity tracker needs more energy when user is active

• **Energy management is a sequential decision-making problem**
  • Decision at one time step affects future decisions
  • Harvested energy is unknown ahead of time, demand is dynamic
    • Problem space is huge and *time-varying*

• **Need for a runtime solution!**
Two Conflicting Objectives:

- Allocate energy to meet energy demand by the application
- Maximize the average battery level during the day

Generalizes to various energy harvesting and demand patterns and battery conditions
Problem Formulation – Demand and Utility

- **Activity Tracker**
  - Demand is a function of activity level

- **ATUS Activity Level → Demand**
  - Daily Activity Patterns for 4772 Users
  - Activity Level → Energy Consumption
    - Oura Ring – *Not Active*
    - Amazon Halo - *Daily*
    - Whoop Strap - *Active*

- **Maximizing utility = Meeting demand**

\[
U(E^A_t, D_t) = \begin{cases} 
\frac{E^A_t}{D_t} & E^A_t < D_t \\
1 & E^A_t \geq D_t 
\end{cases}
\]
Battery energy dynamics:

\[ E_{t+1}^B = E_t^B + E_t^H - E_t^A, \quad 0 \leq t \leq T - 1 \]

- \( E_t^B \): Battery energy
- \( E_t^H \): Harvested energy
- \( E_t^A \): Allocated energy

Physical constraints of the battery (empty, full):

\[ E_{max}^B \geq E_t^B \geq E_{min}^B, \quad 0 \leq t \leq T - 1 \]
**Battery energy dynamics:**

\[
E_{t+1}^B = E_t^B + E_t^H - E_t^A, \quad 0 \leq t \leq T - 1
\]

- \( E_t^B \): Battery energy;
- \( E_t^H \): Harvested energy;
- \( E_t^A \): Allocated energy;

**Physical constraints of the battery (empty, full):**

\[
E_{\text{max}}^B \geq E_t^B \geq E_{\text{min}}^B, \quad 0 \leq t \leq T - 1
\]

**State Space:** \( \{E_t^B, D_{t-1}, E_{t-1}^H, E_0^B, t, \sum_{\tau=1}^{t-1} E_{\tau}^H\} \)

- \( E_t^B \): Current battery energy level
- \( D_{t-1} \): Energy demand during the previous time step (hour)
- \( E_{t-1}^H \): Energy harvested during the previous time step (hour)
- \( E_0^B \): Battery energy level at the beginning of the episode (day)
- \( t \): The current time step (hour)
- \( \sum_{\tau=1}^{t-1} E_{\tau}^H \): Cumulative EH in the previous time steps
Battery energy dynamics:
\[ E_{t+1}^B = E_t^B + E_t^H - E_t^A, \quad 0 \leq t \leq T - 1 \]

- \( E_t^B \): Battery energy; \( E_t^H \): Harvested energy;
- \( E_t^A \): Allocated energy;

Physical constraints of the battery (empty, full):
\[ E_{\text{max}}^B \geq E_t^B \geq E_{\text{min}}^B, \quad 0 \leq t \leq T - 1 \]

State Space: \( \{ E_t^B, D_{t-1}, E_{t-1}^H, E_0^B, t, \sum_{t=1}^{t-1} E_t^H \} \)
- \( E_t^B \): Current battery energy level
- \( D_{t-1} \): Energy demand during the previous time step (hour)
- \( E_{t-1}^H \): Energy harvested during the previous time step (hour)
- \( E_0^B \): Battery energy level at the beginning of the episode (day)
- \( t \): The current time step (hour)
- \( \sum_{t=1}^{t-1} E_t^H \): Cumulative EH in the previous time steps

Action Space: Allocated energy at every time step
\[ E_t^A \in [E_{\text{min}}^A, E_t^B] \]
- \( E_{\text{min}}^A \): Minimum energy level to stay in the idle state
Battery energy dynamics:

\[ E_{t+1}^B = E_t^B + E_t^H - E_t^A, \quad 0 \leq t \leq T - 1 \]

- \( E_t^B \): Battery energy; \( E_t^H \): Harvested energy;
- \( E_t^A \): Allocated energy;

Physical constraints of the battery (empty, full):

\[ E_{\text{max}}^B \geq E_t^B \geq E_{\text{min}}^B, \quad 0 \leq t \leq T - 1 \]

Reward Function:

\[ r_t = \begin{cases} 
U(E_t^A, D_t), \frac{E_t^B}{E_{\text{max}}} & E_t^B \geq E_{\text{min}}^B \\
U(E_t^A, D_t), 1 - \frac{E_t^B}{E_{\text{max}}} & E_t^B \geq E_{\text{max}}^B \\
[-|E_t^B - E_{\text{min}}|, -|E_t^B - E_{\text{min}}|] & E_t^B < E_{\text{min}}^B 
\end{cases} \]

State Space: \( \{E_t^B, D_{t-1}, E_t^H, E_0^B, t, \sum_{t=1}^{T-1} E_t^H\} \)

- \( E_t^B \): Current battery energy level
- \( D_{t-1} \): Energy demand during the previous time step (hour)
- \( E_t^H \): Energy harvested during the previous time step (hour)
- \( E_0^B \): Battery energy level at the beginning of the episode (day)
- \( t \): The current time step (hour)
- \( \sum_{t=1}^{T-1} E_t^H \): Cumulative EH in the previous time steps

Action Space: Allocated energy \( E_t^A \) at every time step

\[ E_t^A \in [E_{\text{min}}^A, E_t^B] \]

- \( E_{\text{min}}^A \): Minimum energy level to stay in the idle state
tinyMAN - RL Framework Training

- **At each episode, choose a random**
  - User from the American Time Use Survey dataset
    - Different EH and demand
  - Initial battery level
  - Preference vector

- **At each step t**
  - Take actions according to state and preference inputs $\rightarrow E_t^A$
  - Calculate the new battery level $\rightarrow E_t^B$
  - Calculate the reward using $E_t^A$ and $E_t^B$
  - Update state vector for the next step

- **Episode terminates if**
  - Battery depletes or step reaches $T$ (i.e., $t=24$)

- **tinyMAN employs PD-MORL [1] algorithm for training.**

---

Experimental Evaluation

- Exclude 10% of the users from training for test purposes
- Optimal solution by an offline solver (CVX)
- Four different initial battery energy levels
  - $E_0^B = \{16, 48, 112, 144\} J$
- <10% MAPE with the Optimal Pareto-frontier
• Exclude 10% of the users from training for test purposes
• Optimal solution by an offline solver (CVX)
• Four different initial battery energy levels
  • \( E^B_0 = \{16, 48, 112, 144\} \) J
• <10% MAPE with the Optimal Pareto-frontier

• Deployability:
  • TI CC2652R Microcontroller:
    • ARM Cortex M4F @ 48 MHz
    • 352 KB Flash Memory
    • 80 KB SRAM
  • Evaluation of deployability:
    • Execution time per inference: 1.98 ms
    • Energy consumption per inference: 23.18 \( \mu \)J
    • Memory utilization: 118 KB
Increasingly frequent and severe wildfires challenge the resilient operation of power systems.

Example: The 2018 Camp Fire in California
- Killed 85 people
- Incurred more than $16B damage
- Led the liable power utility to file for bankruptcy

Electric power infrastructure can both ignite devastating wildfires and be crippled by them.
Wildfire Prevention and Detection Techniques

- Current preventive techniques are based on various risk indexes
  - Wildland Fire Potential Index (WFPI), National Fire Danger Rating System (NFDRS), others
- These indexes have low temporal and spatial resolution
- Wildfire detection capabilities are limited
  - Existing satellite, airborne, and infield sensing systems are either intermittent
  - Or unaware of local ambient risk factors such as high wind speeds or low humidity
A new suite of data-informed, comprehensive sensing frameworks is needed to assess wildfire risks more accurately and enable strategic decisions.
Motivation

- A new suite of data-informed, comprehensive sensing frameworks is needed to assess wildfire risks more accurately and enable strategic decisions.
The proposed CPS design considers an energy-harvesting sensor suit with
- Sensing, processing, and communication capabilities
- An energy-harvesting source
- A rechargeable battery

**Objective:** Maximize the accuracy of sensor readings by altering the sampling rates while always keeping the device operational (i.e., the battery is not depleted)

**Approach:** Pose this problem as a sequential decision-making problem and solve it using reinforcement learning
Problem Formulation

• **Sensor Energy Consumption**

• **Energy Harvesting and Battery Dynamics**
  • We assume solar energy harvesting using PV-cells
    • We use a realistic PV-cell model with real-life solar irradiance data
  • The evolution of battery energy:
    \[ E_{t+1}^B = E_t^B + E_t^H - E_t^C(\alpha), \ t \in T \]

• **Sensor readings during wildfire**
  • Sensor readings are exponentially correlated with the distance to fire (a)
  • The variance in sensor readings increase as fire approaches (b)
  • **Increase sample rate** to obtain a better estimate of the actual value
Problem Formulation

\[
\text{minimize } \sum_{i} |\hat{S}_i(a_i, d) - \mathbb{E}[S_i(d)]|
\]

subject to \[ E_{t+1}^B = E_t^B + E_t^H - E_t^C(a) \]
\[ E_{\text{min}}^B \leq E_t^B \quad E_t^B \leq E_{\text{max}}^B \quad E_T^B \geq E_0^B \]

The optimal solution:
→ Minimizes the absolute error in sensor readings
→ while satisfying battery constraints and self-sustained operation
→ by optimizing the sampling rates of the sensors
**Cell2fire [6]**
- Employs cellular-automata networks to model wildfires at a higher level
- Uses real-world temperature, wind measurements, and terrains
- Validated using several real-world fires

**Dogrib Creek**
- Grid of 357x223 cells, each cell is 100 m x 100 m
- Each cell has an elevation and vegetation value

**Generate a burn trace**
- Random 150-hour segment from an hourly temperature and wind dataset
- Randomly pick an ignition point

**Total of 6000 different 150-hour long wildfire burn traces**

• **pvlib [7]** to simulate the performance of photovoltaic energy systems
  • Combine real hourly irradiance data with a realistic electrical model of 0.1m² PV-Cell
  • Obtain a year-long hourly energy harvesting data with seasonal changes (8760 hours in total)

• **Sensors**
  • Temperature sensor
  • Particle sensor
  • Wind speed/direction sensor

---

RL Environment – Environment Dynamics

**State Space** \( \mathcal{S} \subseteq \mathbb{R}^{14} \):

- Current battery energy \( \left( \frac{E_t^B}{E_{B_{max}}} \in [0,1] \right) \)
- Harvested energy in the previous time step \( \left( \frac{E_{t-1}^H}{E_{B_{max}}} \in [0,1] \right) \)
- Cumulative harvested energy \( \left( \sum_{\tau=0}^{t-1} \frac{E_{\tau}^H}{E_{B_{max}}} \in \mathbb{R} \right) \)
- Initial battery energy level \( \left( \frac{E_0^B}{E_{B_{max}}} \in [0,1] \right) \)
- Target battery energy level \( \left( \frac{E_T^B}{E_{B_{max}}} \in [0,1] \right) \)
- Action in the previous time step \( \left( a_{t-1} \in [-1,1]^3 \right) \)
- Sensor readings in the previous time step \( \left( S_{t-1} \in [0,1]^3 \right) \)
- Moving average of sensor readings \( \left( \frac{1}{5} \sum_{\tau=t-5}^{t-1} S_\tau \in [0,1]^3 \right) \)
State Space ($S \subseteq \mathbb{R}^{14}$):

- Current battery energy ($\frac{E_t^B}{E_{max}^B} \in [0,1]$)
- Harvested energy in the previous time step ($\frac{E_{t-1}^H}{E_{max}^B} \in [0,1]$)
- Cumulative harvested energy ($\sum_{\tau=0}^{t-1} \frac{E_{\tau}^H}{E_{max}^B} \in \mathbb{R}$)
- Initial battery energy level ($\frac{E_0^B}{E_{max}^B} \in [0,1]$)
- Target battery energy level ($\frac{E_T^B}{E_{max}^B} \in [0,1]$)
- Action in the previous time step ($a_{t-1} \in [-1,1]^3$)
- Sensor readings in the previous time step ($s_{t-1} \in [0,1]^3$)
- Moving average of sensor readings ($\frac{1}{5} \sum_{\tau=t-5}^{t-1} s_{\tau} \in [0,1]^3$)

Action Space: Assigned hourly sensor sampling rates at every time step, $a_{t-1} \in [-1,1]^3$
**State Space (S ⊆ ℝ^{14}):**

- Current battery energy \( \left( \frac{E^B_t}{E^B_{\text{max}}} \right) \in [0,1] \)
- Harvested energy in the previous time step \( \left( \frac{E^H_{t-1}}{E^B_{\text{max}}} \right) \in [0,1] \)
- Cumulative harvested energy \( \sum_{\tau=0}^{t-1} \frac{E^H_\tau}{E^B_{\text{max}}} \in \mathbb{R} \)
- Initial battery energy level \( \left( \frac{E^B_0}{E^B_{\text{max}}} \right) \in [0,1] \)
- Target battery energy level \( \left( \frac{E^B_T}{E^B_{\text{max}}} \right) \in [0,1] \)
- Action in the previous time step \( a_{t-1} \in [-1,1]^3 \)
- Sensor readings in the previous time step \( s_{t-1} \in [0,1]^3 \)
- Moving average of sensor readings \( \frac{1}{5} \sum_{\tau=t-5}^{t-1} s_\tau \in [0,1]^3 \)

**Action Space:** Assigned hourly sensor sampling rates at every time step, \( a_{t-1} \in [-1,1]^3 \)

**Reward Function:** Minimize the difference between the expected sensor readings and actual sensor readings

\[
\begin{align*}
r_t = \begin{cases} 
-100 & E^B_t \leq E^B_{\text{min}} \\
\alpha \left( \frac{E^B_t - E^B_T}{E^B_{\text{max}}} \right) - |\hat{S}(a,d) - E[S(d)]| & \text{otherwise}
\end{cases}
\end{align*}
\]
**State Space \((S \subseteq \mathbb{R}^{14})\):**

- Current battery energy \(\left(\frac{E_t^B}{E_{max}}\right) \in [0,1]\)
- Harvested energy in the previous time step \(\left(\frac{E_{t-1}^H}{E_{max}}\right) \in [0,1]\)
- Cumulative harvested energy \(\left(\sum_{\tau=0}^{t-1} \frac{E_{t-1}^H}{E_{max}}\right) \in \mathbb{R}\)
- Initial battery energy level \(\left(\frac{E_0^B}{E_{max}}\right) \in [0,1]\)
- Target battery energy level \(\left(\frac{E_T^B}{E_{max}}\right) \in [0,1]\)
- Action in the previous time step \(a_{t-1} \in [-1,1]^3\)
- Sensor readings in the previous time step \(S_{t-1} \in [0,1]^3\)
- Moving average of sensor readings \(\left(\frac{1}{5} \sum_{\tau=t-5}^{t-1} S_\tau\right) \in [0,1]^3\)

**Action Space:** Assigned hourly sensor sampling rates at every time step, \(a_{t-1} \in [-1,1]^3\)

**Reward Function:** Minimize the difference between the expected sensor readings and actual sensor readings

\[
r_t \begin{cases} 
-100 & E_t^B \leq E_{max}^B \\
\alpha \left(\frac{E_t^B}{E_{max}^B} - \frac{E_T^B}{E_{max}^B}\right) - |\hat{S}(a,d) - E[S(d)]| & \text{o. w.}
\end{cases}
\]

- **Episode**
  - Randomly chose a 150-hour slice from EH dataset, one wildfire trace, and an initial battery energy level
  - Determines the sampling rate of each sensor according to the state
  - Terminates if the battery is depleted, if the wildfire reaches the sensor suite, or if time step reaches 150.
Experimental Setup

• Evaluation Scenarios
  • 5 wildfire traces with 6 different ignition points. (30 different combinations)
  • Long term simulations (6 months, 1 year, 2 years, 3 years, 4 years, and 5 years)

• Evaluation Metrics
  • The cumulative number of inactive hours where the battery was depleted
  • The cumulative sampling error
  • The initial response time to a wildfire (Delay between the emergence of the wildfire and the first measurement after that)

• Ideal Case
  • No inactive hours, error and response time are minimized
  • Challenging since CPS does not know when the fire takes place and the stored and harvested energies are limited and varying
Baseline Heuristic Approach for Comparison

• No existing method in the literature

• Hierarchical Heuristic
  • Sorts the sensors according to their energy consumption in ascending order.
  • Uses harvested energy predictor to predict the battery energy in the next interval.
    • Predictor is obtained by averaging the energy harvested at each hour in a day over the 365 days
  • Based on the predicted battery level, assign sampling rate values to the sensors \((c_1, c_2, c_3)\)
  • Based on the sensor readings (potential wildfire occurrence), wake up the next sensor in line and assign a more aggressive sampling rate
Baseline Heuristic Approach for Comparison

- No existing method in the literature
- Hierarchical Heuristic
  - Sorts the sensors according to their energy consumption in ascending order.
  - Uses harvested energy predictor to predict the battery energy in the next interval.
    - Predictor is obtained by averaging the energy harvested at each hour in a day over the 365 days
  - Based on the predicted battery level, assign sampling rate values to the sensors ($c_1$, $c_2$, $c_3$)
  - Based on the sensor readings (potential wildfire occurrence), wake up the next sensor in line and assign a more aggressive sampling rate

<table>
<thead>
<tr>
<th></th>
<th>Conservative</th>
<th>Balanced</th>
<th>Aggressive</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>1 S/hr</td>
<td>15 S/hr</td>
<td>33 S/hr</td>
</tr>
<tr>
<td>$c_2$</td>
<td>1 S/hr</td>
<td>21 S/hr</td>
<td>39 S/hr</td>
</tr>
<tr>
<td>$c_3$</td>
<td>1 S/hr</td>
<td>27 S/hr</td>
<td>45 S/hr</td>
</tr>
</tbody>
</table>
Performance Evaluation – Short term results

- **Behavior in an episode:**
  - One-week simulations, wildfire occurs at the final day of the week
  - Heuristics start at 90%, RL starts at 10% battery energy (top right)
  - Assign 20% higher sampling rate to sensors (2\textsuperscript{nd} row →)
  - Achieves 30% lower absolute sensor reading error (3\textsuperscript{rd} row →)
Performance Evaluation – Long term results

• 1-year analysis
• All approaches start at 90%, battery energy
• Wildfire occurs in the final week of the simulation
• Heuristics lead to fluctuation in the battery levels
• RL agent never depletes the battery and has a faster response time
Energy Consumption Evaluation

- **Deployability:**
  - Total memory footprint < 200 KB
  - Execution time 2 ms/inference
  - Consumption 25 µJ/inference
  - Total daily consumption is 0.6 mJ
  - **Negligible energy overhead,** does not compromise self-sustained operation!
Agenda

- Motivation
- PDMORL: Preference Driven Multi-Objective Reinforcement Learning Algorithm
- A Comprehensive Multi-Objective Energy Management Approach for Wearable Devices with Dynamic Energy Demands
- A Self-Sustained CPS Design for Reliable Wildfire Monitoring
- DTRL: Decision Tree-based Multi-Objective Reinforcement Learning for Runtime Task Scheduling in Domain-Specific System-on Chips
- Conclusions, Future Directions and Closing Remarks
Multicore Computing Architectures

Homogeneous Architectures

✓ Improved performance, power consumption compared to single-core processors

✗ Rigid power-performance characteristics

Offer either

Low-power (or)
High-performance
## Multicore Computing Architectures

### Homogeneous Architectures
- ✓ Improved performance, power consumption compared to single-core processors
- ✗ Rigid power-performance characteristics

### Heterogeneous Architectures
- ✓ Better match resources with application needs
- ✓ Improved performance, energy-efficiency
- ✗ Significant gap with respect to special-purpose solutions

### Offer

<table>
<thead>
<tr>
<th>Low-power</th>
<th>High-performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offer either</td>
<td>Offer both</td>
</tr>
</tbody>
</table>
## Multicore Computing Architectures

### Homogeneous Architectures

- **✓** Improved performance, power consumption compared to single-core processors
- **✗** Rigid power-performance characteristics

<table>
<thead>
<tr>
<th>Arm LITTLE CPU</th>
<th>Arm LITTLE CPU</th>
<th>Offer either</th>
</tr>
</thead>
</table>

### Heterogeneous Architectures

- **✓** Better match resources with application needs
- **✓** Improved performance, energy-efficiency
- **✗** Significant gap with respect to special-purpose solutions

<table>
<thead>
<tr>
<th>Arm LITTLE CPU</th>
<th>Arm LITTLE CPU</th>
<th>Arm big CPU</th>
<th>Arm big CPU</th>
<th>Offer both</th>
</tr>
</thead>
</table>

### Domain-Specific Systems-on-Chip (DSSoC)

- **✓** Judiciously combine general-purpose, special-purpose and hardware accelerator cores
- **✓** Highly efficient for domain applications
- **✓** Flexibility to execute other domains

<table>
<thead>
<tr>
<th>Offer Programmability &amp; energy-efficiency</th>
</tr>
</thead>
</table>

- **Low-power**
- **High-performance**
- **High**
- **Specialized processing**
Harvest Full Potential of DSSoCs: Scheduling

• How to harvest the full potential of DSSoCs?
  • Optimally utilize the processing elements (PEs) at runtime

• Task scheduling:
  • Assign tasks to PEs to achieve optimization goals at runtime
    • Minimize execution time, power dissipation, energy consumption

![Application DAG](image)

![Sample Schedule](chart)
Scheduling Approaches

• **Optimization-based approaches:**
  • Integer Linear Programming, Constraint Programming
  • High complexity, high runtime overhead, optimal results

• **Heuristic-based approaches:**
  • Low runtime overhead, sub-optimal results

• **Machine learning-based approaches:**
  • Low runtime overhead, sub-optimal results
  • Lack to adapt to workload and platform changes, need re-training

• **Reinforcement learning-based approaches:**
  • High runtime overheads, being unable to run on heterogeneous SoCs and on high-intensity workloads, high training complexity

• **Multi-objective approaches:**
  • Genetic, evolutionary, multi-objective heuristics
  • High complexity, high runtime overhead
Scheduling Approaches

- **Optimization-based approaches:**
  - Integer Linear Programming, Constraint Programming
  - High complexity, high runtime overhead, optimal results

- **Heuristic-based approaches:**
  - Low runtime overhead, sub-optimal results

- **Machine learning-based approaches:**
  - Low runtime overhead, sub-optimal results
  - Lack to adapt to workload and platform changes, need re-training

- **Reinforcement learning-based approaches:**
  - High runtime overheads, being unable to run on heterogeneous SoCs and on high-intensity workloads, high training complexity

- **Multi-objective approaches:**
  - Genetic, evolutionary, multi-objective heuristics
  - High complexity, high runtime overhead

*Can we achieve an approach that delivers near-optimal results, low overheads, efficiently support multiple objectives?*
Proposed Framework DTRL

- **DTRL**, a decision-tree-based multi-objective reinforcement learning technique for runtime task scheduling in DSSoCs
  - Multi-objective Proximal Policy Optimization (PPO) algorithm and **differentiable decision tree (DDT) policy**
    - **DDT policy**: Low inference latency overheads
    - **Objectives**: Power consumption vs. Execution Time
  - **A novel RL environment for DSSoCs**, utilizing an open-source DSSoC simulator [8]

Multi-Objective PPO with DDT as an Actor

- **Objective:** Obtain a single policy that covers the entire preference space for multiple objectives in a runtime task scheduling problem

- **PPO → MO-PPO**

\[
\begin{align*}
\text{state } (s), \text{action } (a) & \quad \rightarrow \quad \text{Environment} \quad \rightarrow \quad r = \{r_1, r_2\} \\
\text{state } (s), \text{preference } (\omega) & \quad \rightarrow \quad \text{Value Network } (\phi) \quad \rightarrow \quad V_\phi(s, \omega) \\
\text{state } (s), \text{preference } (\omega) & \quad \rightarrow \quad \text{DDT Policy } (\theta) \quad \rightarrow \quad \pi_\theta(s, \omega)
\end{align*}
\]

Differentiable Decision Tree (DDT)

- Traditional DTs have a tendency to **overfit** and **fail to generalize well**
- DDTs combines **the interpretability of traditional DTs with the differentiability of neural networks to address these limitations**
End-to-End Training Flow with Novel RL Environment

- Enhance the capabilities of open-source DSSoC simulator [1] to integrate it into a Gym environment.
- Handshake mechanism to facilitate communication between simulator, the RL environment, and the RL agent.

RL Environment Dynamics

**State Space** ($S \subseteq \mathbb{R}^{2C+4}$):

- Task ID
- Depth of task in DAG
- Application type
- Execution time on $C$ clusters
- Application ID
- Earliest availability of $C$ clusters
RL Environment Dynamics

**State Space** \( (S \subseteq \mathbb{R}^{2C+4}) \):
- Task ID
- Depth of task in DAG
- Application type
- Execution time on \( C \) clusters
- Application ID
- Earliest availability of \( C \) clusters

**Action Space** \( (A \subseteq \mathbb{N}^{\mathbb{N}}) \): Selecting from a set of \( C \) processing clusters
RL Environment Dynamics

**State Space** \( (S \subseteq \mathbb{R}^{2C+4}) \):
- Task ID
- Depth of task in DAG
- Application type
- Execution time on \( C \) clusters
- Application ID
- Earliest availability of \( C \) clusters

**Action Space** \( (A \subseteq \mathbb{N}^{C}) \): Selecting from a set of \( C \) processing clusters

**Reward Vector:**
- **Execution Time** of the task on the chosen PE
- **Power Consumption** of the task on the chosen PE
RL Environment Dynamics

- DSSoC comprises several PEs, and similar PEs are grouped into $C$ processing clusters

**Episode:**
- User-selected number of frames are generated, with each frame containing several tasks
- A random preference vector is chosen
- Scheduling is performed for each task in the workload
- Terminates when all tasks in the workload complete execution

**State Space** ($S \subseteq \mathbb{R}^{2C+4}$):
- Task ID
- Depth of task in DAG
- Application type
- Execution time on $C$ clusters
- Application ID
- Earliest availability of $C$ clusters

**Action Space** ($A \subseteq \mathbb{N}^C$): Selecting from a set of $C$ processing clusters

**Reward Vector:**
- Execution Time of the task on the chosen PE
- Power Consumption of the task on the chosen PE
Experimental Setup

- **SoC configuration: 16 PEs**
  - Arm LITTLE (4 PEs) and Arm big (4 PEs)
  - Hardware accelerators:
    - 2 Matrix multipliers, 2 Viterbi decoders, 4 FFT

- **Domain: Wireless communications and radar systems**
  - WiFi-Transmitter and WiFi-Receiver
  - Single-carrier Transmitter and Receiver
  - Radar correlator and temporal mitigation

- **Workload: Mix of 100 frames of all 6 applications**

- **Simulation and Emulation Frameworks:**
  - Our novel OpenAI Gym environment integrated with an open-source DSSoC simulator, DS3 [1] simulator. Simulator is validated against Odroid-XU3 (Samsung Exynos 5422) and Xilinx Zynq ZCU102
  - Hardware runtime overhead of DTRL is measured by implementing it **within an open-source Linux-based emulation and runtime environment, CEDR [2]**, on the Xilinx Zynq ZCU102

---


Baseline and SOTA Approaches for Comparison

- **Optimization-based approach:** Integer linear Programming (ILP) through IBM ILOG CPLEX Optimization Studio.

- **Heuristic-based schedulers:**
  - Earliest task first (ETF) makes scheduling decisions by iterating through all the ready tasks and available PEs to determine the task with the earliest finish time.
  - ETF variants that targets different objectives such as power consumption, energy consumption, energy-delay product.

- **Machine-learning-based scheduler:**
  - Imitation-learning for task scheduling (ILS [1]). It uses ETF as the Oracle and trains a regression tree to approximate the Oracle decisions.

- **MORL-based scheduler (Our own adaptation as a comparison point):**
  - Scalarized-MOPPO involves performing network updates after computing the inner product of the vectorized values and the preference vectors.
  - Trains separate policies for various preference ratios of the different objectives

---

DTRL Evaluation

- **A global DDT scheduling policy** is obtained at the end of the training.
  - Constructed with 16 input features (including the preference vector)
  - Maximum depth of 3

- Frames (including several tasks) are dynamically injected into the system based on an exponential distribution
  - **Target throughput (Injection rate) is varied between 1-50 frames per millisecond**
  - Average metrics of **10 random seeds** are used to avoid bias in the distribution
**DTRL Evaluation – Single Objective (Corner Cases)**

- **DTRL is up** of 1.03x, 1.05x, 1.25x, 1.9x, and 9x faster than ILS, Scalarized-MOPPO, ETF-EDP, ETF-Energy, and ETF-Power.
- **DTRL is trained with multiple objectives** ~ average execution time within 4% and 7% of ETF and ILP.
- ETF, ILP, and ILS are designed to optimize a single specific objective ~ DTRL is trained to handle multiple objectives.
• **DTRL achieves** 3.06x, 3.08x, 3.06x, 1.97x, and 1.05x lower energy consumption compared to ETF, ILP, ILS, ETF-EDP, ETF-Energy, and ETF-Power.

• **DTRL does not require retraining for the energy objective** since the global DDT policy dynamically generates near-optimal decisions based on the user-defined preference provided at runtime.
DTRL Evaluation – Multi-Objective Functionality

Preference vectors \( \omega \) separated by a step size of 0.1, \( \omega \in \{\{1, 0\}, \{0.9, 0.1\}, \ldots, \{0.1, 0.9\}, \{0, 1\}\}. 

![Graphs showing energy efficiency versus average execution time for different methods.](image)
ETF, ILP, and ILS have only one point on the plot since they support only one objective. ETF variants correspond to different comparison points in the objective space.

Using a single global DDT policy, DTRL:
- Scales to several throughputs
- Covers the entire preference space and outperforms other schedulers
DTRL Evaluation – Multi-Objective Functionality

- A workload with 100% target throughput denotes the maximum frame rate supported by the simulation platform.
- As the target throughput increases, the average job execution time and power consumption increase due to the increase in congestion in the SoC.
DTRL on a Runtime Emulation Platform

- DTRL is deployed in the CEDR framework [9]
  - C++ module
  - Analyze its performance on a Xilinx Zynq ZCU102 FPGA
  - **DSSoC configuration**: 1 FFT core, 1 MM core, 3 general-purpose cores
  - **Three domain application**: WiFi transmitter, range detection, and temporal mitigation

DTRL on a Runtime Emulation Platform

- **DTRL is deployed in the CEDR framework [9]**
  - C++ module
  - Analyze its performance on a Xilinx Zynq ZCU102 FPGA
  - **DSSoC configuration:** 1 FFT core, 1 MM core, 3 general-purpose cores
  - **Three domain application:** WiFi transmitter, range detection, and temporal mitigation

![Graph showing performance comparison between ETF and DTRL](image)

DTRL on a Runtime Emulation Platform

• DTRL is deployed in the CEDR framework [9]
  • C++ module
  • Analyze its performance on a Xilinx Zynq ZCU102 FPGA
  • DSSoC configuration: 1 FFT core, 1 MM core, 3 general-purpose cores
  • Three domain application: WiFi transmitter, range detection, and temporal mitigation

• Runtime overhead
  • ETF $\rightarrow$ 100-1000 ns
  • DTRL $\rightarrow$ 120 ns

Motivation

PDMORL: Preference Driven Multi-Objective Reinforcement Learning Algorithm

A Comprehensive Multi-Objective Energy Management Approach for Wearable Devices with Dynamic Energy Demands

A Self-Sustained CPS Design for Reliable Wildfire Monitoring

DTRL: Decision Tree-based Multi-Objective Reinforcement Learning for Runtime Task Scheduling in Domain-Specific System-on Chips

Conclusions, Future Directions and Closing Remarks
Conclusion

Energy-Awareness

Hardware Friendliness

Multi-Objective Functionality

A Comprehensive Multi-Objective Energy Management Approach for Wearable Devices with Dynamic Energy Demands

A Self-Sustained CPS Design for Reliable Wildfire Monitoring

DTRL: Decision Tree-based Multi-Objective Reinforcement Learning for Runtime Task Scheduling in Domain-Specific System-on Chips

PDMORL: Preference Driven Multi-Objective Reinforcement Learning Algorithm
Future Research Directions

• **Improving Training Efficiency in Reinforcement Learning**
  - Using domain-knowledge to incorporate transfer learning
  - Efficient training techniques – JAX implementation

• **Improvements in Wildfire Monitoring Framework**
  - Integrating cameras as the final level of sensors in the hierarchy
  - Multi-agent, neighbor-aware sensor-suite network.

• **Advancements to DTRL Framework**
  - Additional optimization objectives
  - Applicability of DTRL in other domains

• **On-Device Learning**
Future Research Directions

• **Improving Training Efficiency in Reinforcement Learning**
  - Using domain-knowledge to incorporate transfer learning
  - Efficient training techniques – JAX implementation

• **Improvements in Wildfire Monitoring Framework**
  - Integrating cameras as the final level of sensors in the hierarchy
  - Multi-agent, neighbor-aware sensor-suite network.

• **Advancements to DTRL Framework**
  - Additional optimization objectives
  - Applicability of DTRL in other domains

• **On-Device Learning**
References


List of Publications


List of Publications


Multi-Objective and Energy Efficient Reinforcement Learning for Edge AI Applications

Thank You!

Questions?