Multi-Objective Reinforcement Learning for Power Control System with Pareto Optimization Approach
End Semester Presentation

Almog Anschel Rotem Shezaf

August 10, 2025

The Taub Faculty of Computer Science
Technion – Israel Institute of Technology

Outline

Introduction and Background

Project Objectives

- **Develop** an RL agent that controls the PCS unit inside the Energy-Net simulator.
- **Handle multiple conflicting objectives:**
  - Economic profit (energy arbitrage)
  - Battery health and lifetime
  - Grid support / stability
  - Energy autonomy
- **Deliver** a set of Pareto-optimal policies so operators can trade off objectives in real time.

Power Control System (PCS) in Energy-Net

PCS is an agent in the Energy-Net smart grid simulation.

Core Responsibilities:

- Manage battery storage: decide when to charge or discharge.
- React to ISO price signals to optimize profits.
- Perform energy arbitrage: buy low, sell high.

Challenge: Balance multiple competing objectives simultaneously.

Why Multi-Objective RL?

Single-Objective Limitations:

- Traditional RL optimizes a single scalar reward
- Real-world systems involve multiple conflicting objectives
- Fixed weights cannot adapt to changing priorities

Multi-Objective Benefits:

- Learn a set of diverse, high-quality policies
- Enable flexible selection based on real-time needs
- Provide visibility into trade-offs between objectives

Key Challenge: There is no universally 'best' policy without specifying preference information.

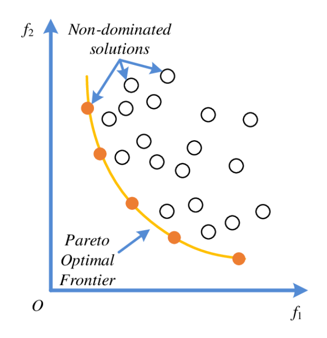
Pareto Optimality Concept

Pareto Optimality:

- A policy is **Pareto optimal** if no objective can be improved without degrading at least one other
- The **Pareto front** represents all optimal trade-off solutions

Our Goal:

- Approximate the Pareto front
- Provide operators with a diverse set of optimal policies
- Enable real-time policy selection



Baseline Evaluation and Wrapper Validation

Baseline Methodology

Challenge: No established baselines exist for multi-objective RL in EnergyNet.

Our Solution:

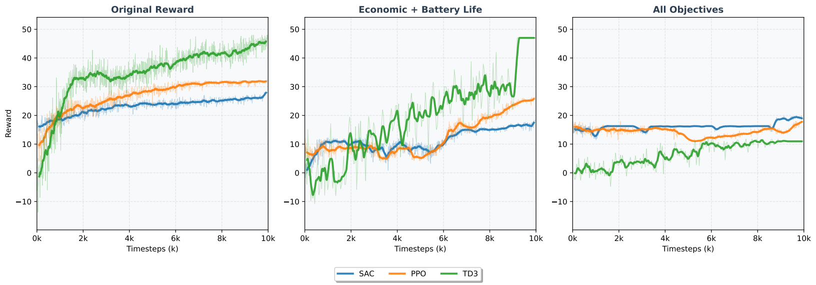
- Created a scalarization wrapper that converts vector rewards to scalar rewards
- Enables comparison between our MOSAC algorithm and standard single-objective RL algorithms

Evaluation Strategy:

- **Single-objective evaluation:** MOSAC vs. baseline algorithms (SAC, PPO, TD3) on scalarized rewards
- Validates our algorithm's competitiveness in both settings

Scalar Baseline Results

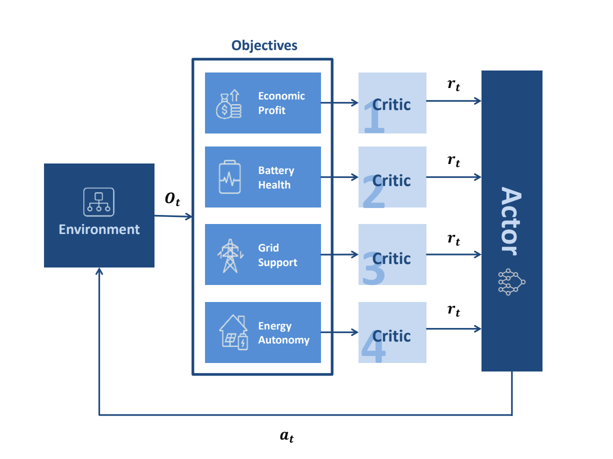
Performance of Baseline Single-Objective Algorithms



- All three baselines perform reasonably well on the original single-objective reward.
- As objectives increase, performance drops due to single-objective design.
- Results shown are the average over 3 different seeds for each run.

MOSAC Algorithm

MOSAC Architecture



- **Multi-head Critics:** Four separate critics, one for each objective.
- **Vector Rewards:** Reward vector instead of scalar reward
- **Shared Actor:** Learns a single policy

Multi-Objective SAC (MOSAC) Approach

Utility-Based Approach:

- Use scalarization on the critic to calculate the reward
- Store reward as a vector and create multi-head critic
- Use only weighted scalarization so the Bellman equation holds

Key Limitation:

- The Bellman equation only holds for a linear utility function.
- User preference might be unknown

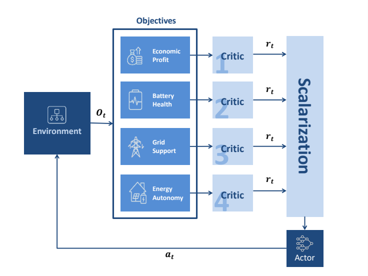


Illustration of MOSAC scalarization process

MOSAC scalar Variants

Two Implementation Approaches:

Shared Features Network:

- One network with multiple heads
- Shared feature extraction
- Faster convergence
- Higher final reward (28 vs 18.5)

Independent Critics:

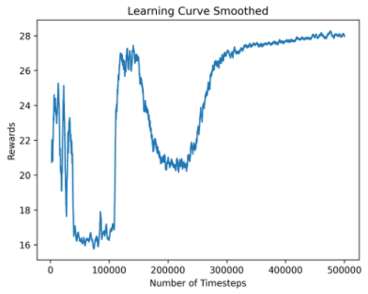
- Four separate critic networks
- Independent feature learning
- More stable but slower
- Lower final performance

Challenge: The agent failed to effectively utilize consumption and production actions.

MOSAC Results - Shared Learning Critics

Performance:

- Converges after 500,000 steps
- Achieves 28 scalarized reward



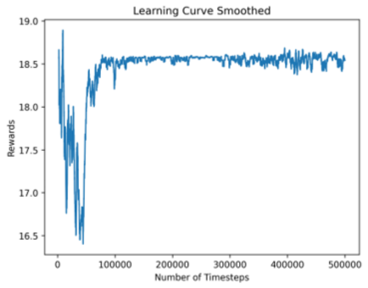
Learning Curve - Shared Feature

Parameter	Value
Learning rate	3e-4
Gamma	0.99
Train freq	1
Total timesteps	500,000
Buffer size	1000
Batch size	64

MOSAC Results - Individual Learning Critics

Performance:

- Converges after 100,000 steps
- Achieves 18.5 scalarized reward



Learning Curve - Separated Critics' networks

Parameter	Value
Learning rate	3e-4
Gamma	0.99
Train freq	1
Total timesteps	500,000
Buffer size	1000
Batch size	64

Scalar Results Comparison

Algorithm	Original Reward	Economic + Battery	All Obj.
SAC	23.10	11.25	16.22
PPO	26.92	12.48	14.30
TD3	**34.63**	**19.40**	6.58
MOSAC (Shared Critics)	–	–	**28**
MOSAC (Separate Critics)	–	–	18.50

- Values represent mean rewards over 3 seeds.
- MOSAC variants were evaluated only on the All Objectives setting, where the Shared Critics version achieved the best performance.

Hyper-MORL Algorithm

Hyper-MORL Motivation

Challenges

- The naive approach to finding the Pareto front works only with discrete action spaces.
- This is because it creates a set of Pareto-optimal policies and greedily chooses between them.
- Creating a discrete approximation of the Pareto front is inefficient for continuous action spaces.

Solutions

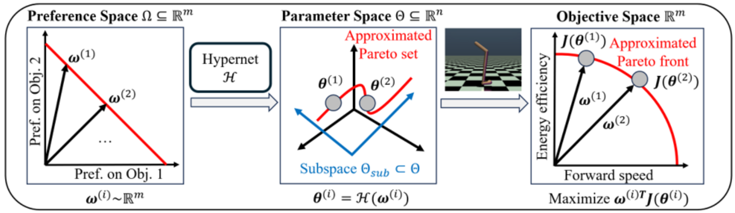
- To condition the policy output on the preference.
- To train a hyper-network to represent the Pareto front.

Hyper-MORL Algorithm

Hyper-net is an algorithm that approximates the Pareto front. [1]

- Ω is the preference space
- Θ is policy network parameter space.
- The model matches each $\omega \in \Omega$ to $\theta \in \Theta$ such that $\omega^T J(\theta)$ is maximized.

  $H_\phi(\omega) = Wf_u(\omega) + b$

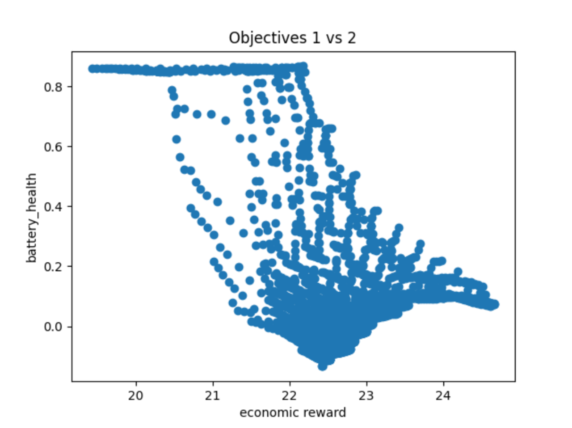


[1] Shu, T., Shang, K., Gong, C., Nan, Y., and Ishibuchi, H. (2023). Learning Pareto Set for Multi-Objective Continuous Robot Control.

Department of Computer Science and Engineering, Southern University of Science and Technology.

Hyper-MORL Algorithm

- We used the source code from [1].
- We adapted the code for our Energy-Net environment.
- We modified the code to support more than three objectives.

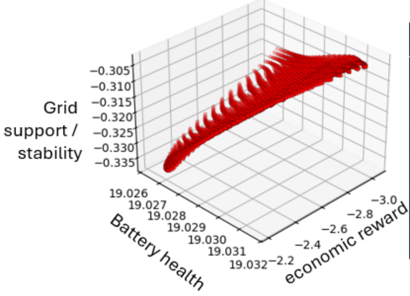


Pareto front projection on the first two objectives.

Hyper-MORL results

Performance:

- Converges after 100,000 steps
- The 3D hypervolume is zero since the Pareto front lies in four dimensions.
- The three-dimensional hypervolume is 758.65.

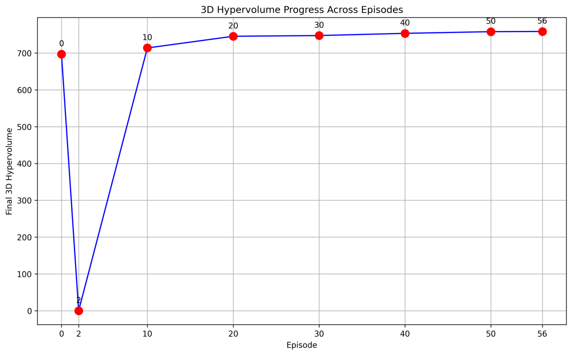


Pareto front — projection on the first three components.

Parameter	Value
Learning rate	5e-4
Gamma	0.95
Train freq	1
Total timesteps	30,000,000
Number of processes	4
Warmup steps	2048

Hyper-MORL results

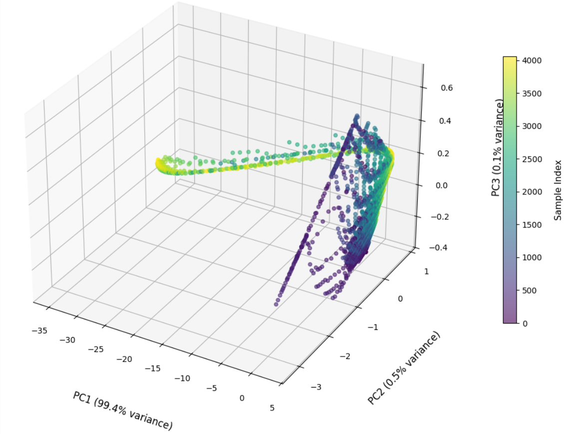
3D Hyper-volume over episodes



the hypervolume calculated on the first three objectives over the running episodes

Hyper-MORL results - PCA

We used PCA to project the Pareto front into a three-dimensional space.



Pareto Front - PCA analysis. The components, in order, are: economic reward, battery health, and grid support/stability.

Hyper-MORL results - PCA

Table 1: Principal Components Loadings and Explained Variance

	PC1	PC2	PC3	PC4
Explained Variance	0.9945	0.0050	0.0006	0.0000
economic reward	-0.057617	0.968286	0.243111	0.000000
battery health	-0.000887	-0.243565	0.969884	0.000000
grid support/stability	0.998338	0.055666	0.014893	0.000000
autonomy	-0.000000	0.000000	-0.000000	1.000000

Hyper-MORL - PCA

conclusions from PCA analysis:

- The most conflicting objectives are battery health and economic reward.
- The most influential objectives in constructing the Pareto front, in order, are: grid support, economic reward, and battery health.

Hyper-MORL Summary

Conclusions:

- Hyper-MORL converges more slowly than MOSAC
- The 3D hypervolume metric converges.
- consumption and production actions were not performed
- No formal proof of convergence for Hyper-MORL

Further work

Further work PSL

- PSL also uses a hyper-network, $\phi(\omega)$
- The algorithm samples a preference from a distribution and uses it as input to both the policy and the hyper-network Δ.
- the mixed network parameters obtained by using a parameter fusion technique on the hyper network and the policy network

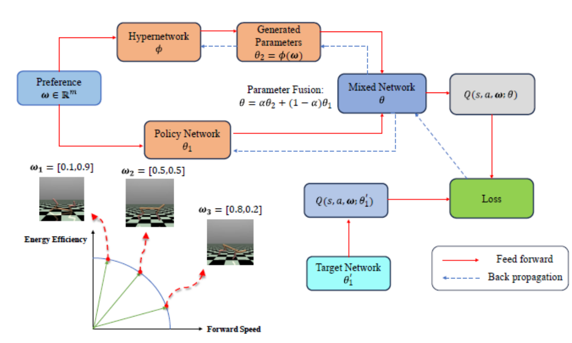


Illustration of the proposed PSL-MORL method [2]

Further work PSL

- PSL does not include warm up stage, contrary to Hyper-MORL
- We also plan to implement PSL with SAC or PPO to compare with Hyper-MORL.
- matching the git of the hyper-MORL algorithm was challenging, so we ran out of time

[2] Liu, E., Wu, Y.-C., Huang, X., Gao, C., Wang, R.-J., Xue, K., and Qian, C. (2025). Pareto Set Learning for Multi-Objective Reinforcement Learning. National Key Laboratory for Novel Software Technology, Nanjing University.

Further work - problems we faced

- We created an environment with both ISO and PCS together
- We were in the process of running the algorithms on the PCS agent alone, but didn't have enough time
- We didn't figure out why the PCS does not perform buy/sell actions
- It might be possible to run the environment with other types of rewards, such as the ISO reward

Summary

In Conclusion:

- We examined two approaches for multi-objective RL
- The utility-based approach, examined using the MOSAC algorithm
- The multi-policy, examined using the Hyper-MORL algorithm.
- MOSAC algorithm is designed for cases where the scalarization weights are known