

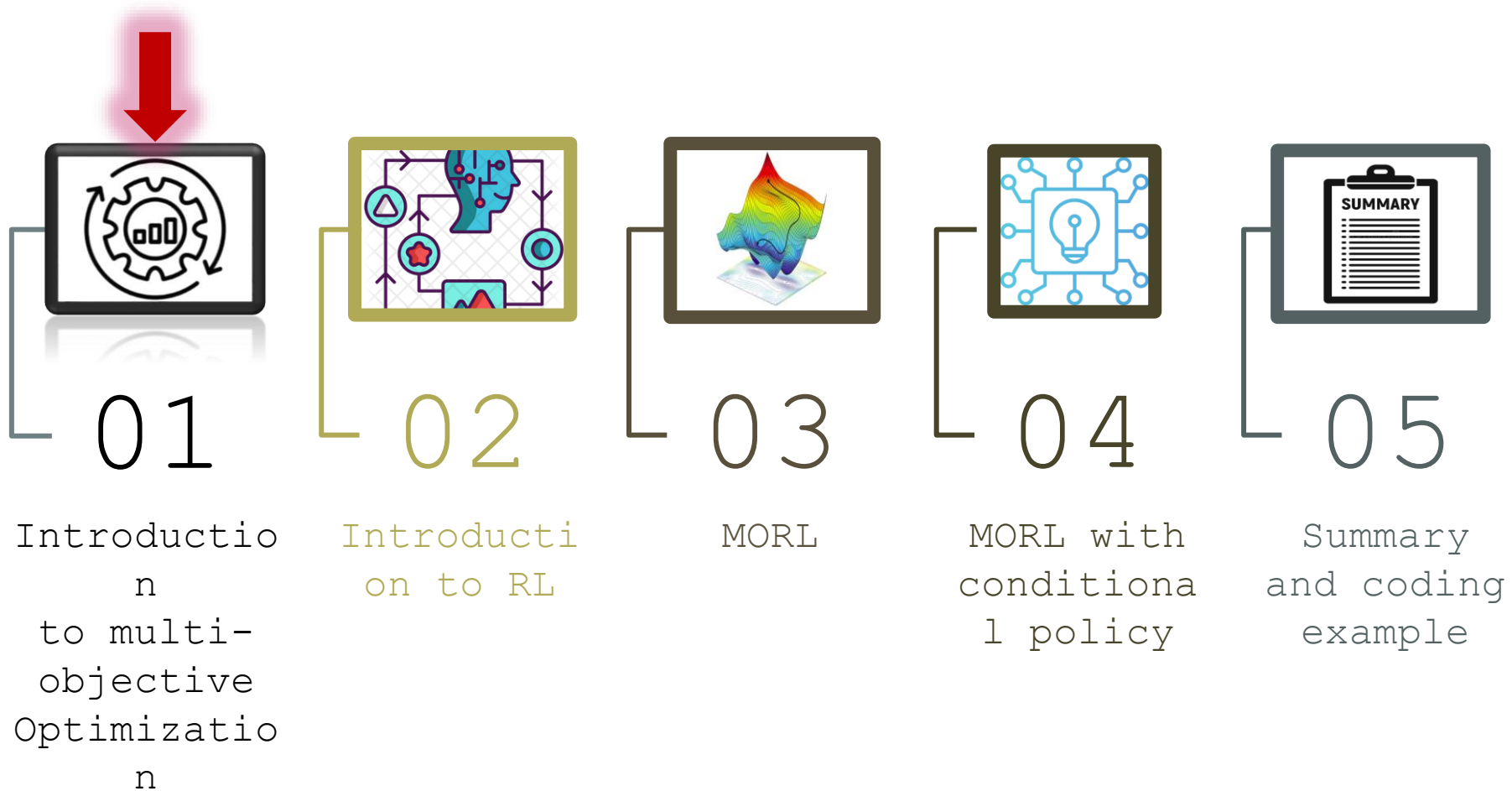
# Multi-Objective Reinforcement Learning

**Kishansingh Rajput**

Jefferson Lab, Newport News, VA, USA

1

# Outline



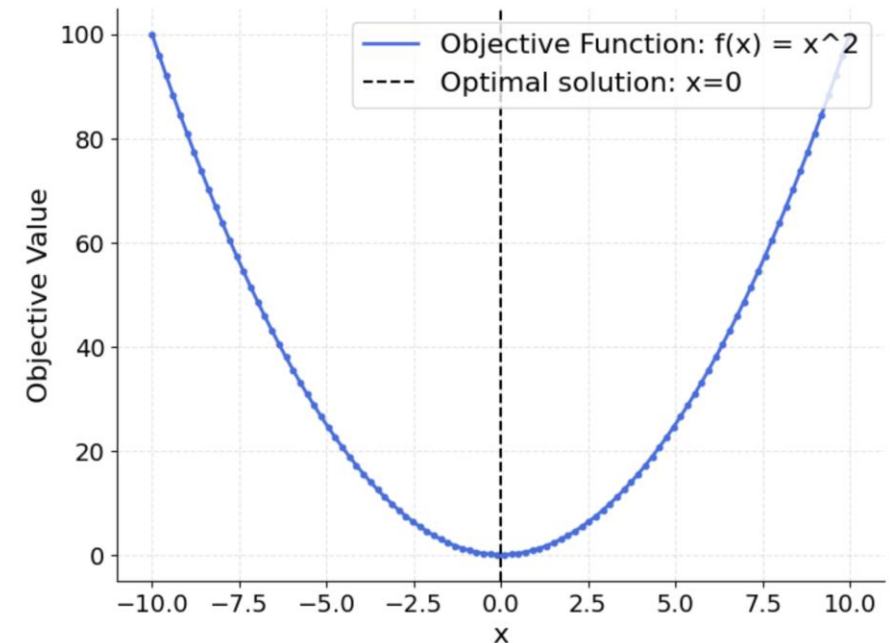
# Introduction

## What is Optimization?

- Finding the best solution to a problem, given certain constraints
- Can be maximization or Minimization problem
- For example: minimize RF heat load

## Key Components of Optimization

- **Objective Function:** The function to maximize or minimize
  - Example: heat loss function
- **Decision Variables:** The variables you can control or adjust
  - Example: RF cavity gradient set-points
- **Constraints:** The restrictions or limits that the solution must satisfy
  - Example: Operational limits on individual cavity gradients, total beam energy requirement



Objective function: minimize  $x^2$   
Decision Variables:  $x$   
Constraints:  $-10 \leq x \leq 10$

# Multi-objective Optimization

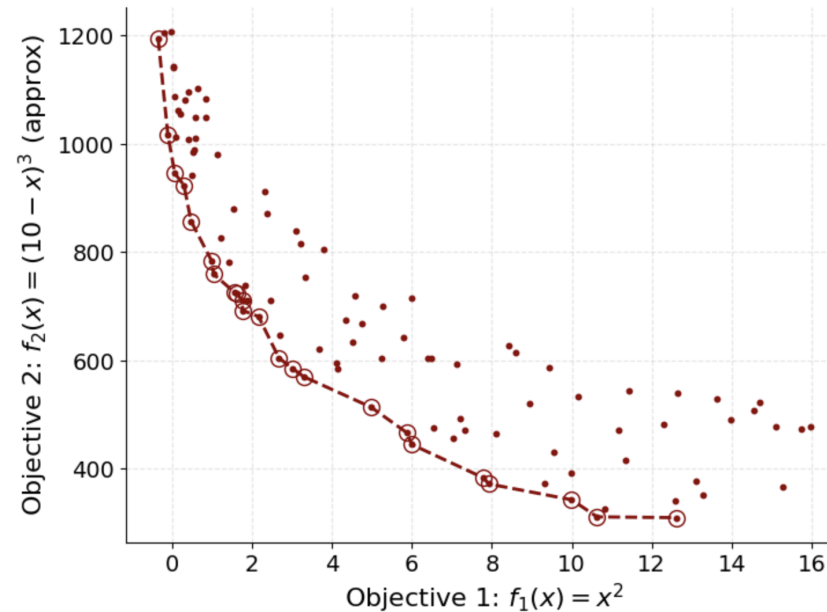
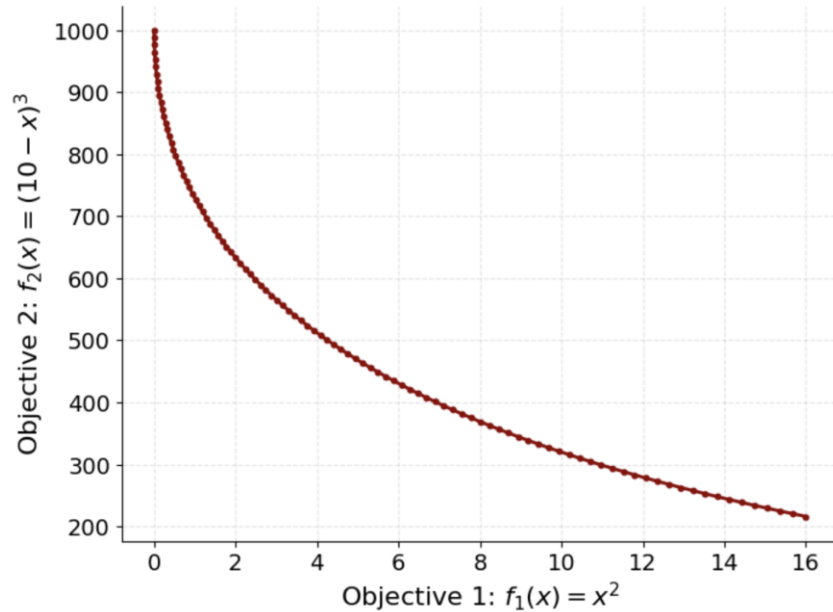
## Multi-Objective Optimization (MOO)

- Optimizing two or more conflicting objectives simultaneously
- MOO seeks to balance trade-offs between multiple objectives

## Key Concepts

- **Multiple Objective Functions:** Two or more objective functions
  - Example: heat loss function, and trip rates
- **Decision Variables:** The variables you can control or adjust
  - **All or subset of decision variables may affect two or more objectives**
  - Example: RF cavity gradient set-points
- **Constraints:** The restrictions or limits that the solution must satisfy
  - **This may also include minimum/maximum thresholds on one or more objectives**
  - Example: Operational limits on individual cavity gradients, total beam energy requirement

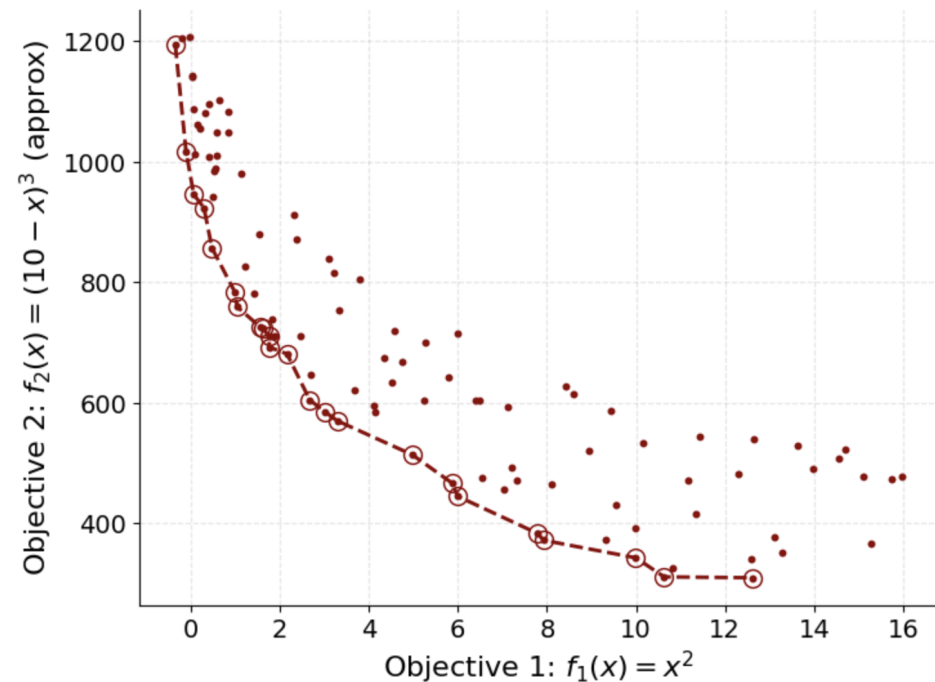
# Multi-objective Optimization



Objective functions: minimize  $x^2$  and  $(10 - x)^3$   
Decision Variables:  $x$   
Constraints:  $-4 \leq x \leq 0$

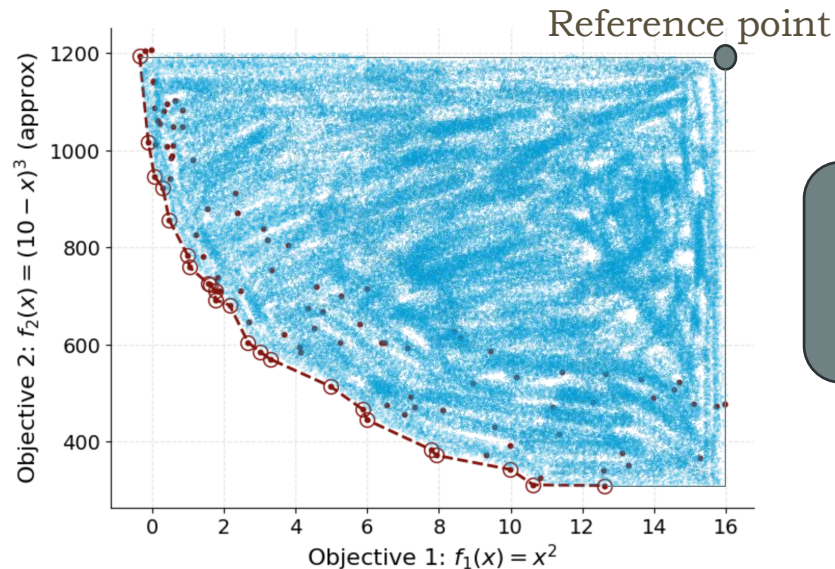
# Pareto optimality

- The **Pareto front** represents the set of **Pareto optimal solutions** in a multi-objective optimization problem
  - There is no other solution that improves one objective without sacrificing another
  - Cannot be "dominated" by any other solution (no other solution is better in all objectives)
- 
- Pareto-front represent trade-offs between multiple objectives
  - Provides multiple optimal solutions, allowing selection based on preferences/requirements
  - Tricky to visualize higher dimensional Pareto fronts
  - Hypervolume can be used to quantify the quality of Pareto fronts



# Hypervolume

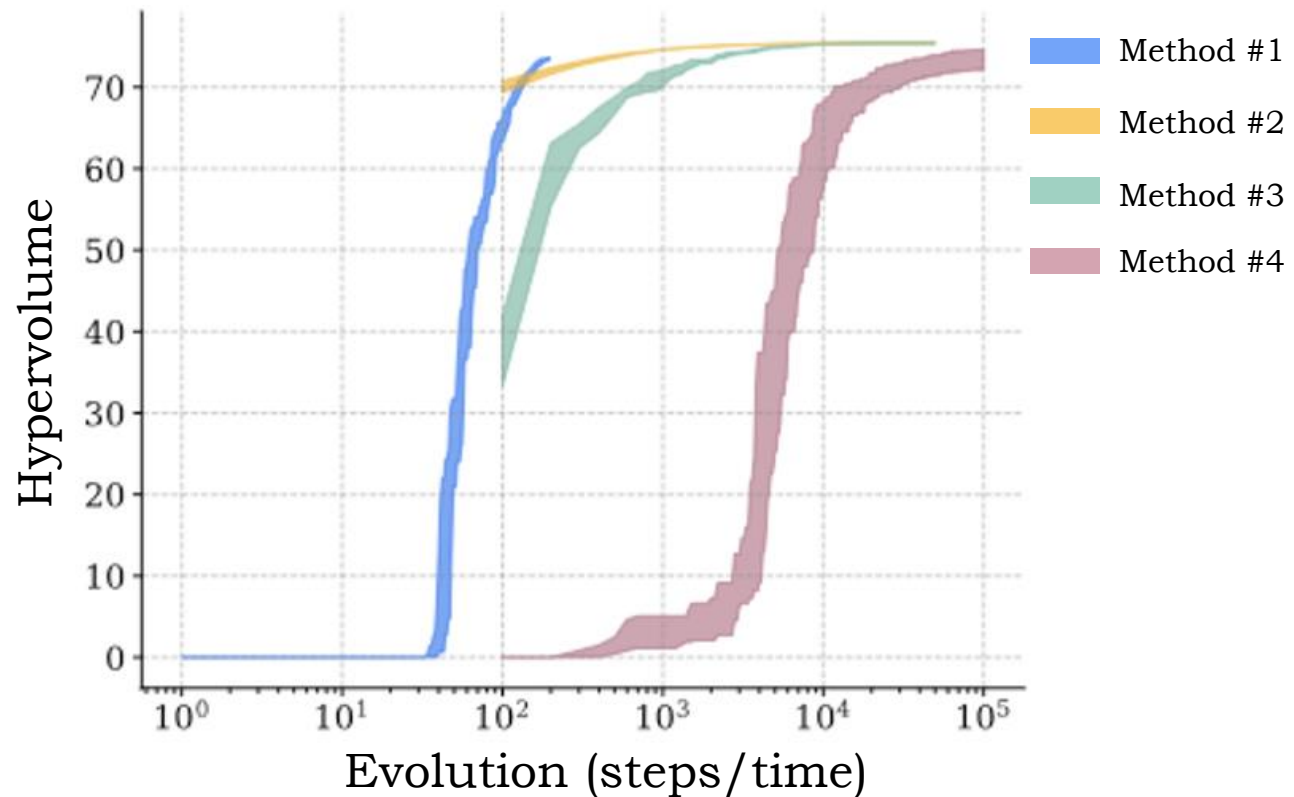
- Hypervolume measures the volume of the objective space that is dominated by a set of Pareto optimal solutions
  - with respect to a reference point (usually a worst-case point)
- **Quality Metric:** Hypervolume provides a clear measure of how well the solutions approximate the Pareto front and how large the dominated region is.
- **Diversity & Convergence:** It captures both the **convergence** (how close solutions are to the ideal) and **diversity** (how well solutions cover the objective space).



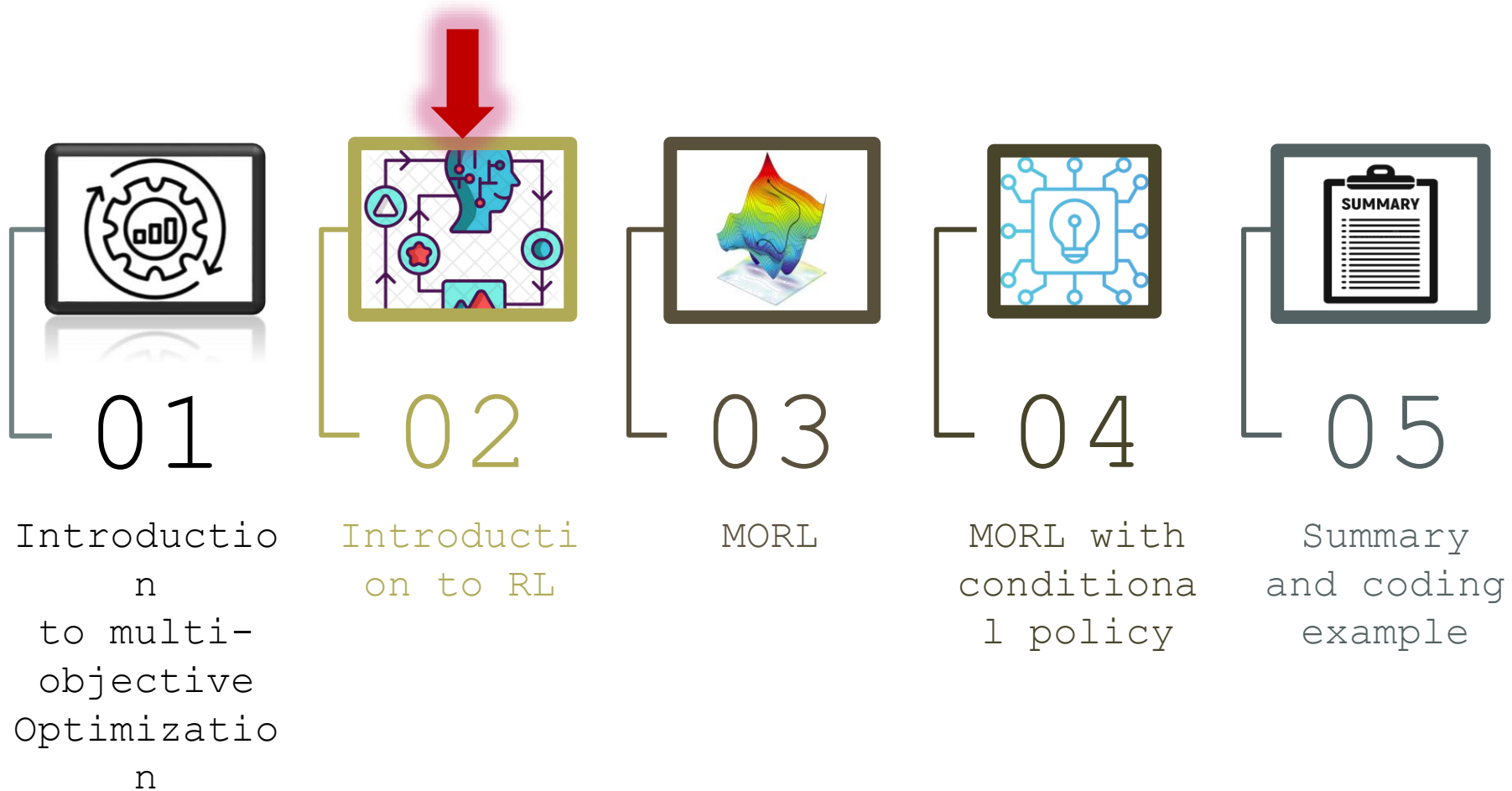
2D → Area under/over the curve

Higher Dim → Volume

# Hypervolume as a metric

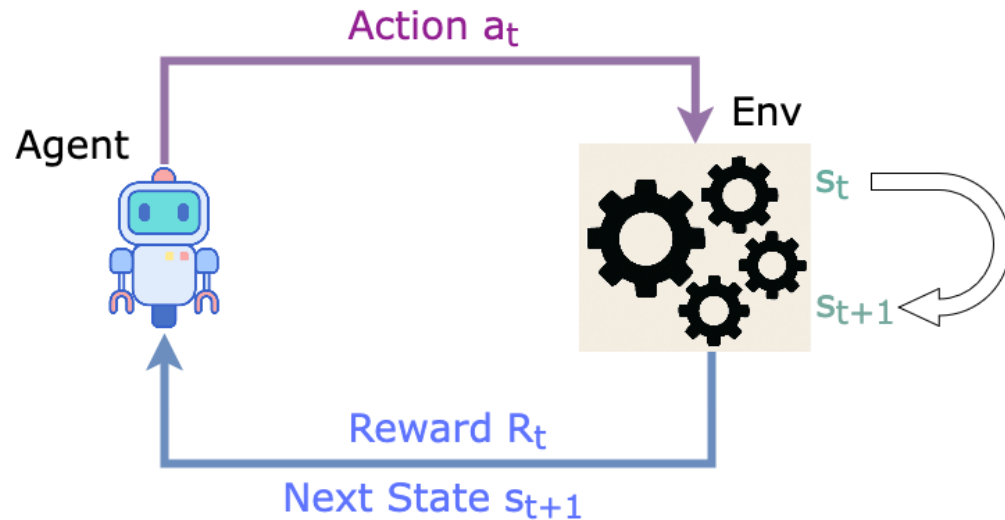


# Outline



# Reinforcement Learning

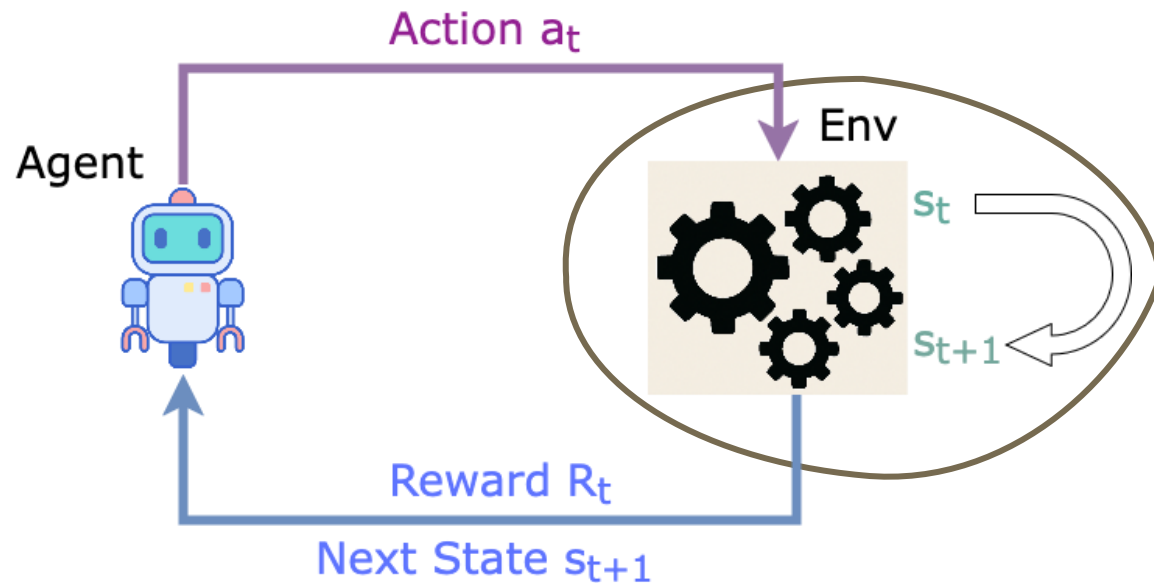
- Reinforcement Learning (RL) is a sequential controls algorithm by design
- Two parts – Agent, and environment



At a given time step  $t$ ,

- the agent observes the current state of the env  $s_t$
- The agent provides action  $a_t$
- The env receives  $a_t$  and provides reward  $R_t$  to the agent
- At the same time the env transitions from state  $s_t$  to  $s_{t+1}$

# Reinforcement Learning



At time step  $t$ , the environment observes incoming action  $a_t$  and does the following

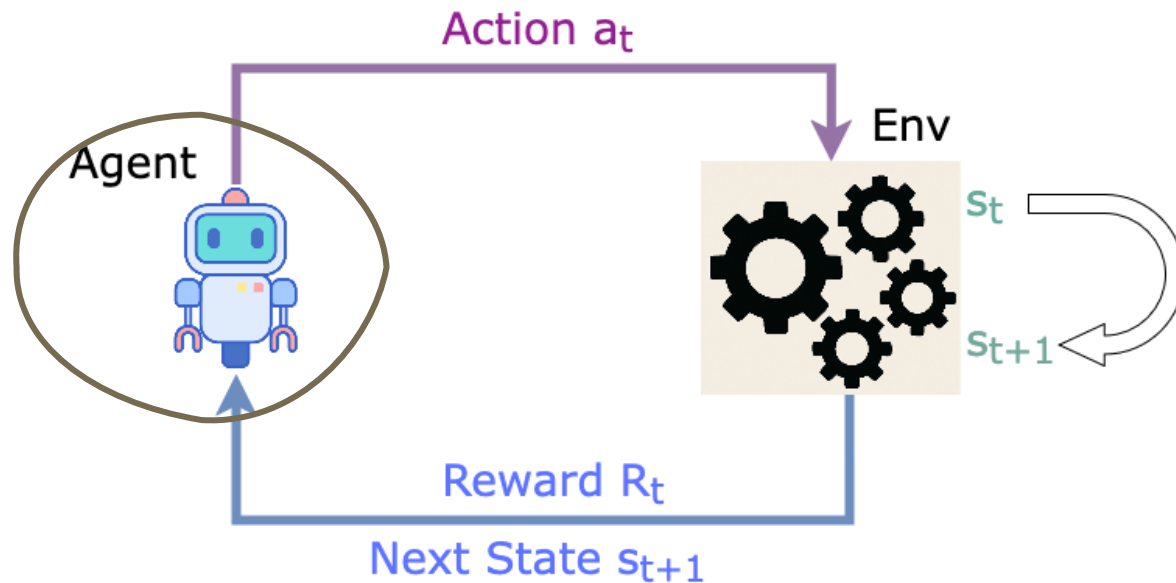
- Transition function: Move to a new state ( $s_{t+1}$ ) based on current state  $s_t$  and action  $a_t$  combination

$$s_{t+1} = T(s_t, a_t, \beta)$$

Here  $\beta$  represents system dynamics

- Reward function: Quantify the quality of action  $a_t$  and produce  $R_t$ ;  $R_t = R(s_t, a_t, s_{t+1})$ 
  - A scalar reward function is a result of a quality metric or complex combination of many such metrics

# Reinforcement Learning



At time step  $t$

- The agent observes current state  $s_t$  of the env and provides an action  $a_t$  based on its policy  $\pi$ ;

$$a_t = \pi(s_t)$$

- Once the action is applied to the env, it observes incoming reward  $R_t$  to quantify the quality of its action

- Deep RL algorithms typically stores combinations of state, action, reward, next state and any additional relevant information to train its policy

# Reward

- Reward is a scalar (or vector in MOO) feedback  $R_t$
- RL is based on reward hypothesis
  - Reward hypothesis
    - All goals can be described by the maximization of expected cumulative reward
- Agent's job is to maximize cumulative reward
  - RL models sequential decision making
  - Agent need to select actions to maximize total future reward
  - Actions may have long term consequences
  - Sacrifice immediate reward to gain more in longer term

# Bellman equation

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma \cdot R(s_{t+1}, a_{t+1}) + \gamma^2 \cdot R(s_{t+2}, a_{t+2}) + \dots$$

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma (R(s_{t+1}, a_{t+1}) + \gamma \cdot R(s_{t+2}, a_{t+2}) + \dots)$$

$$Q^*(s_t, a_t) = E[R(s_t, a_t) + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})]$$

Modeled by a neural network in DRL

- $R(s_t, a_t)$  is an immediate reward for action  $a_t$  at state  $s_t$
- This indicates how good the action  $a_t$  is at state  $s_t$
- The goal of the agent is to maximize the cumulative reward

- RL builds history
- Exploration vs exploitation
- Sample efficiency

# Outline



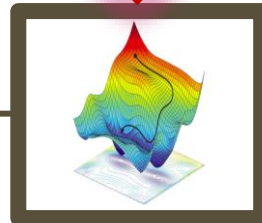
01

Introduction  
to multi-  
objective  
Optimization



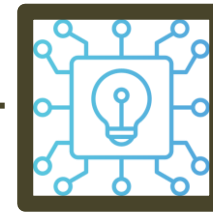
02

Introduction to RL



03

MORL



04

MORL with  
conditional  
policy

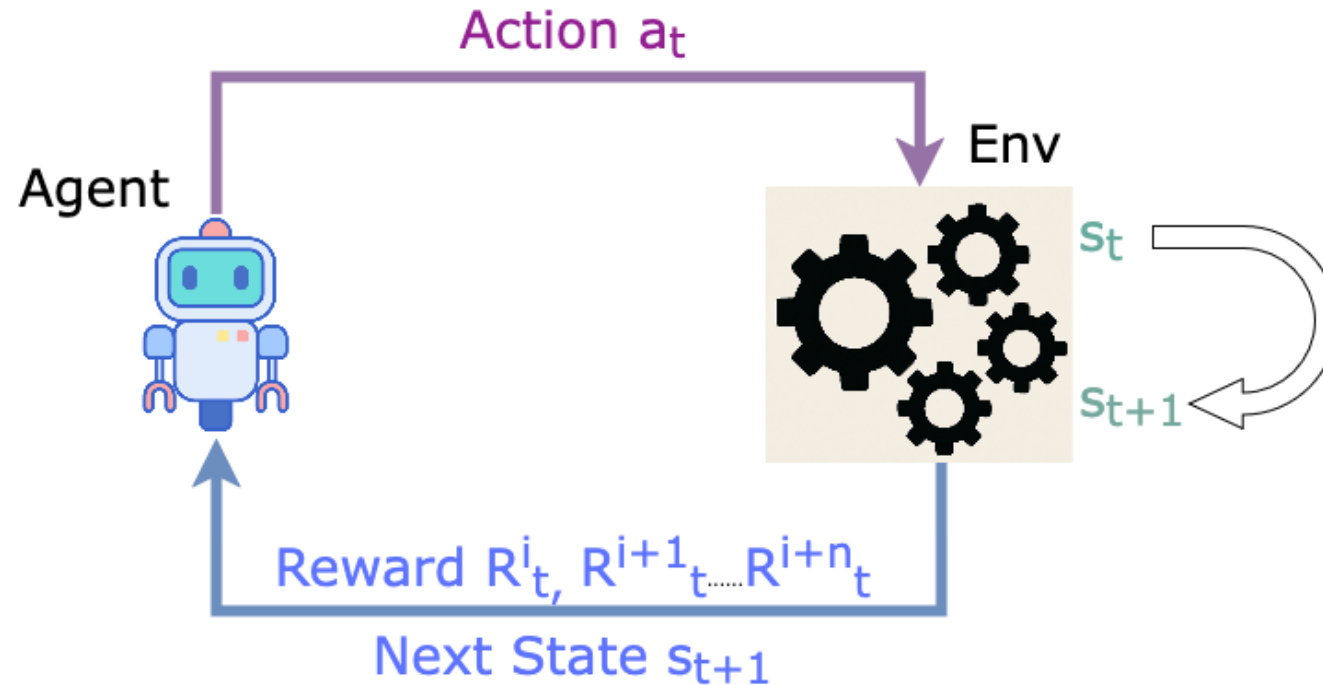


05

Summary  
and coding  
example

# Multi-Objective RL

- Multi-Objective RL (MORL) extends traditional RL to handle problems with multiple conflicting objectives
- MORL involves optimizing a vector of rewards, each corresponding to different objectives



# MORL vs Multi goal RL

- Multi goal RL setup is different from MORL
- Multi goal RL has multiple states that are optimal
  - Number of objectives can be one or more
- MORL explicitly have multiple objectives

S				
				G
				G

Multiple goal states

S				
				G

Single goal with multiple objectives to balance

# MORL Strategies: Scalarization

---

MORL has two main strategies

1. Scalarization approach: find a single policy that optimizes a weighted sum of rewards
  - Simplifies the problem by **combining multiple objectives into a single scalar** value
  - Transforms a multi-objective problem into a **single-objective** problem
  - making it easier to solve using traditional reinforcement learning techniques

# MORL Strategies: Scalarization

## Weighting Sum:

- Each objective is assigned a **weight** that determines its importance
- The agent aims to maximize the weighted sum

## Scalarized Reward Function:

The scalarized reward function can be represented as:

$$R(s, a) = \sum_{i=1}^m w_i \cdot R_i(s, a)$$

- Where  $R_i(s, a)$  is the reward for the  $i^{th}$  objective
- $w_i$  is the weight for the  $i^{th}$  objective with  $\sum w_i = 1$

## Other methods

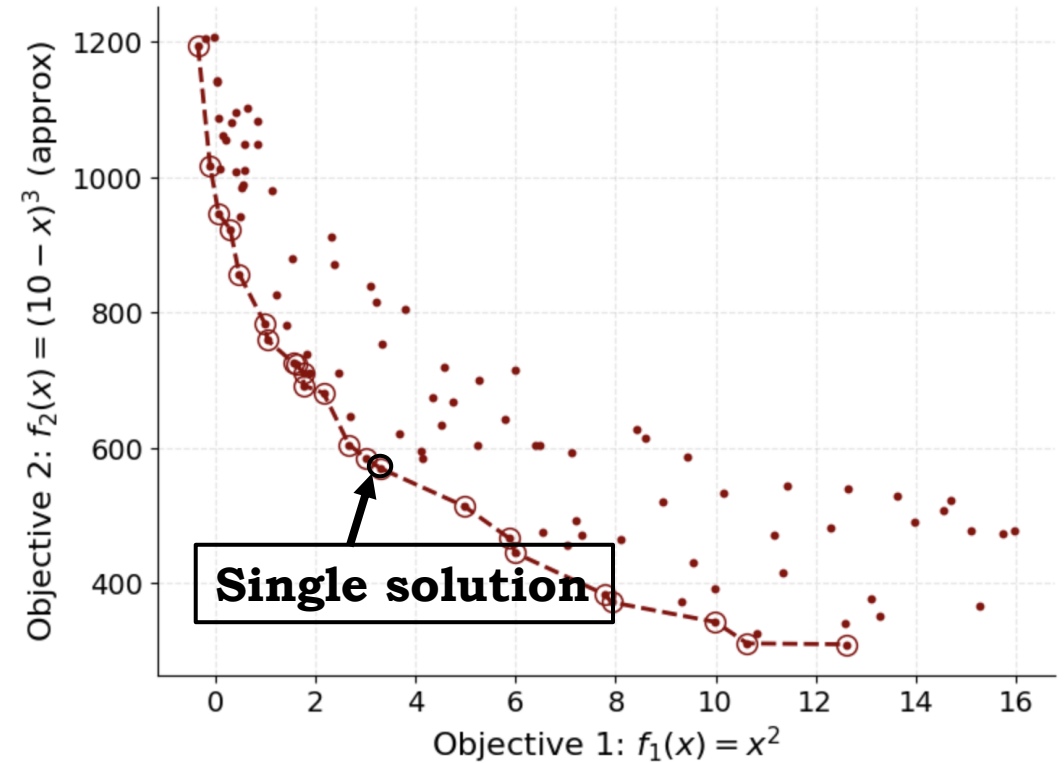
For example, Chebyshev Scalarization

Adjusts the relative importance of objectives based on the worst-case deviation from the ideal values

# MORL Strategies: Scalarization

## Challenges:

- The choice of weights significantly impacts the solution
- If the weights are not chosen properly, it can lead to suboptimal trade-offs
- Optimal Weights are not known
- Deterministic policy with a single solution on pareto front
- Do not provide full Pareto front
- Can become increasingly difficult as the number of objectives grows



# MORL Strategies: Pareto Approach

---

MORL has two main strategies

## 2. Pareto Approach: Find multiple policies that cover the Pareto front

- Instead of searching for a single optimal policy, the agent searches for a set of Pareto optimal policies
- These solutions represent different trade-offs between the objectives
- The agent explores different regions of the objective space to discover the Pareto front.
- Each policy on the Pareto front represents a unique balance between competing objectives

# MORL Strategies: Pareto Approach

- Pareto Q-Learning:
  - A modification of traditional Q-learning where multiple Q-values are maintained, each corresponding to a different objective
  - The agent learns to balance these Q-values to select a policy that approximates the Pareto front
- Pareto Policy Iteration
  - Similar to policy iteration but applied to multiple objectives
  - The algorithm iterates through policies, improving them to cover more of the Pareto front
- Scalarization Methods:
  - **Although the Pareto approach does not focus on scalarizing the rewards, scalarization techniques like weighted sums can be used to guide the search toward the Pareto front - Conditional Policy**
  - This method balances trade-offs between objectives by adjusting the weights dynamically

# MORL Strategies: Pareto Approach

---

## Challenges:

- Exploration Complexity:
  - Computationally expensive and time-consuming
- Diversity of Solutions:
  - Generating diverse solutions that cover the Pareto front requires careful exploration to avoid converging to suboptimal areas
- Scalability:
  - As the number of objectives increases, the size of the Pareto front grows, making the search space exponentially larger

# Outline



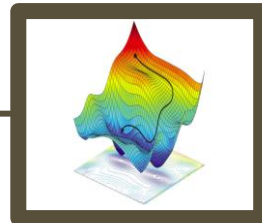
01

Introduction  
to multi-  
objective  
Optimization



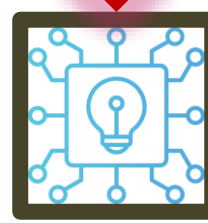
02

Introduction to RL



03

MORL



04

MORL with  
conditional  
policy

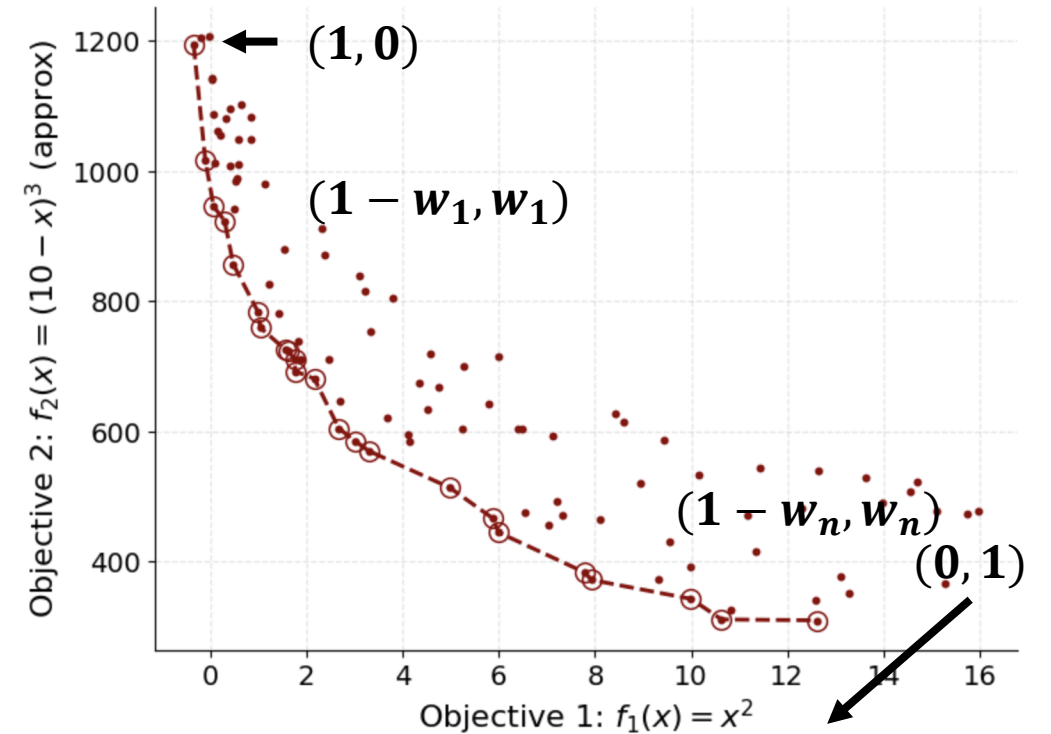


05

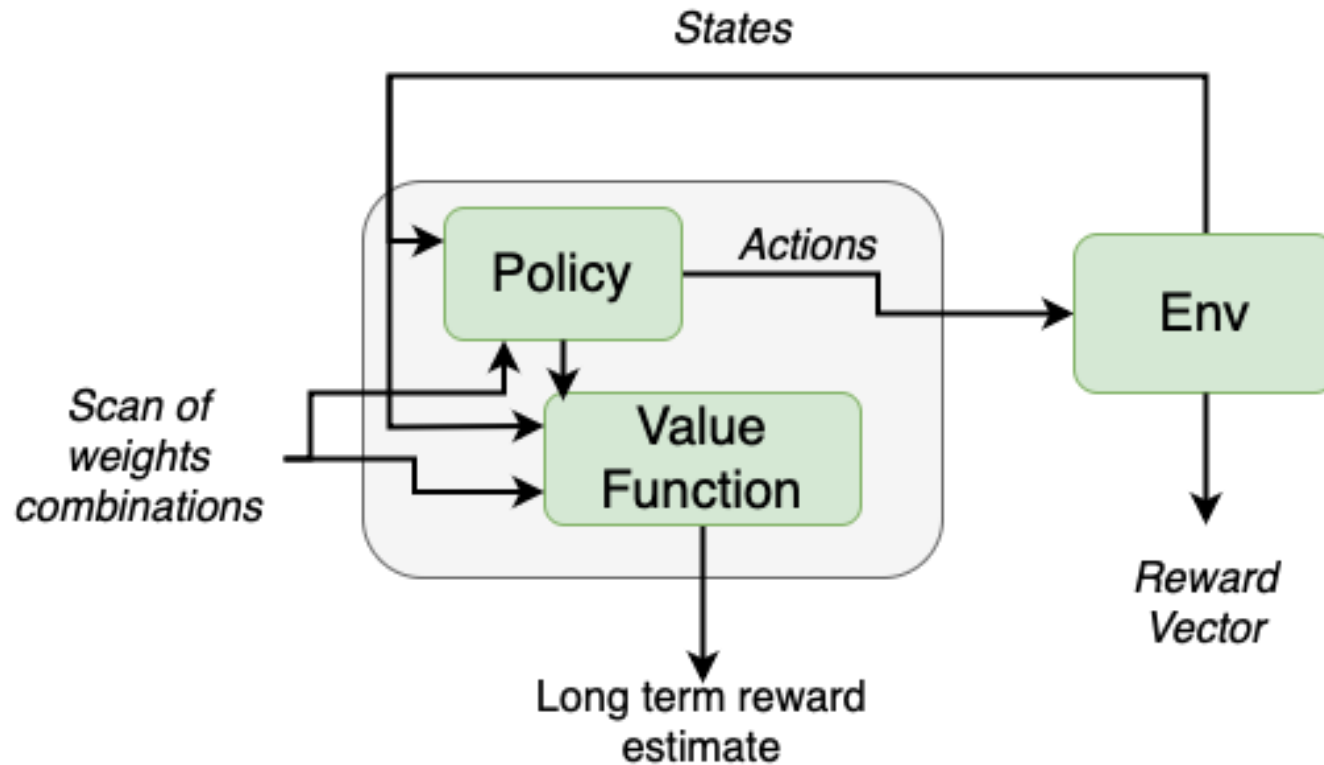
Summary  
and coding  
example

# MORL Strategies: Conditional Policy

- Produce Pareto front via reward scalarization
- Scan the possible reward weights
  - Train a conditional policy
  - Conditioned on reward scalarization weights
  - Randomly (or cleverly) switch between conditions per epoch
- Simpler to implement for lower dimensional problems
- Gets more complex as number of objectives increase



# MORL Strategies: Conditional Policy



- Easy to implement for large dimensional multi objective problems
- Works well for optimization problems
- May require extensive training and tuning on sequential control problems

# Outline



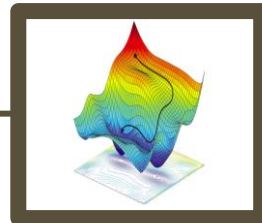
01

Introduction  
to multi-  
objective  
Optimization



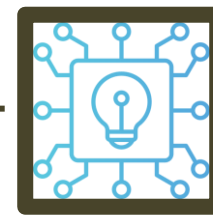
02

Introduction to RL



03

MORL



04

MORL with  
conditional  
policy



05

Summary  
and coding  
example

# Summary

---

- Multi-objective optimization can be significantly challenging compared to single-objective optimization
- RL seeks to maximize long term cumulative reward
- MORL uses a vector of rewards to optimize multiple objectives
- Scalarization can convert your multi-objective RL problem to a single-objective RL problem
- Scalarization may not provide full suite of solutions on the Pareto front
- Explicit Pareto approaches are more complex to implement and train but provides multiple solutions on a Pareto front
- Scalarization can be used to guide the agent to produce full Pareto front

Hands on examples – implements a simple env and conditional MORL agent based on TD3:

## #1 Scalarized

[https://colab.research.google.com/drive/18JoA-8je2U3QG9ro7RDoH5SwOkREC\\_Vi?usp=sharing](https://colab.research.google.com/drive/18JoA-8je2U3QG9ro7RDoH5SwOkREC_Vi?usp=sharing)

## #2 MORL with conditional policy

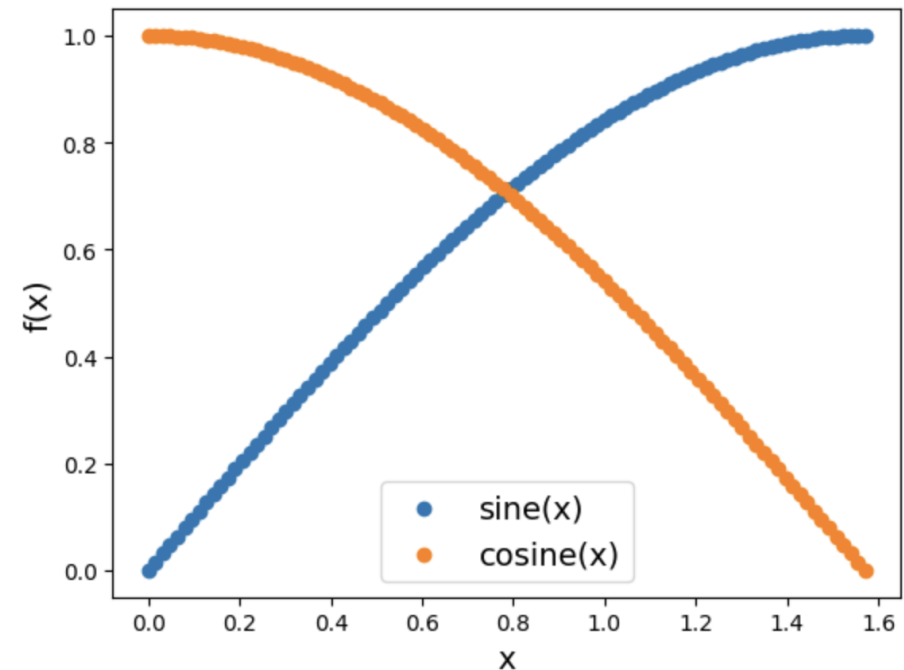
<https://colab.research.google.com/drive/10vBhNpGCbE0DL6xLe9D8gpH7AeYp3edc?usp=sharing>

Requires basic understanding of python, and reinforcement learning ☺

**Objective functions:** minimize  
 $\sin(x)$  and  $\cos(x)$

**Decision Variables:**  $x$

**Constraints:**  $0 \leq x \leq \pi/2$



Hands on examples – implements a simple env and conditional MORL agent based on TD3:

## #1 Scalarized

[https://colab.research.google.com/drive/18JoA-8je2U3QG9ro7RDoH5SwOkREC\\_Vi?usp=sharing](https://colab.research.google.com/drive/18JoA-8je2U3QG9ro7RDoH5SwOkREC_Vi?usp=sharing)

## #2 MORL with conditional policy

<https://colab.research.google.com/drive/10vBhNpGCbE0DL6xLe9D8gpH7AeYp3edc?usp=sharing>

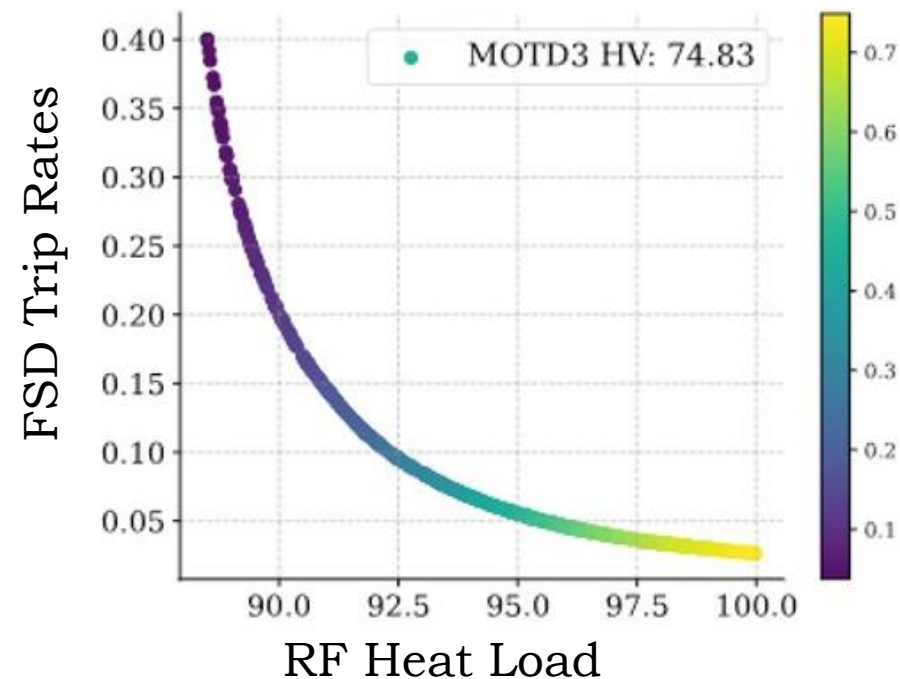
Requires basic understanding of python, and reinforcement learning ☺

The algorithm in the notebooks is similar to the one used to produce these real-life results

- May not include all the components for pareto-modeling

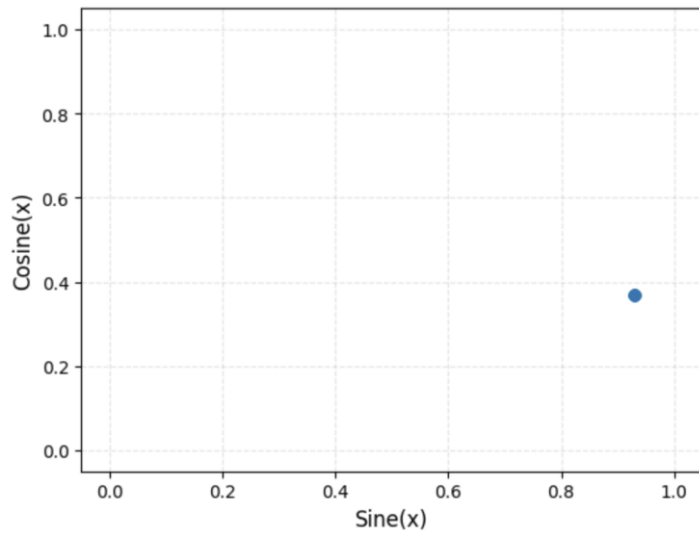
Full version will be released soon at

<https://github.com/JeffersonLab/SciOptControlToolkit>

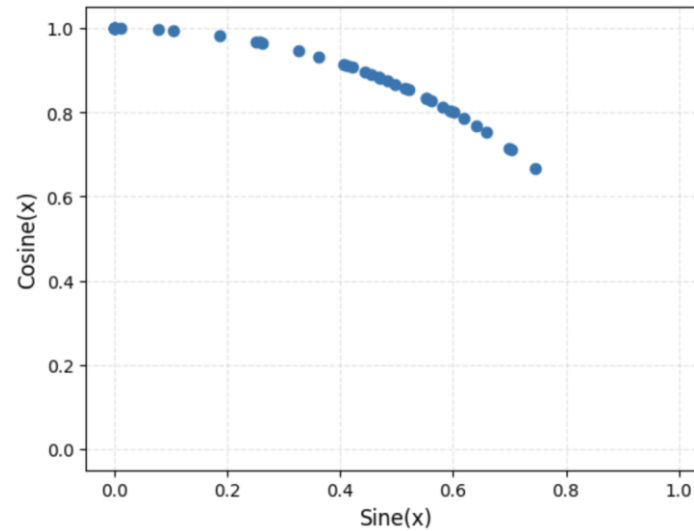


# Expected Output: Scalarized Method

With no stochasticity



With stochasticity



MORL with conditional policy

