Politecnico di Milano
Gaia Biasolo, Matteo Giuseppe Cigada, Federico D'Agostini

# Lab 12: Reinforcement Learning

12/12/2025

## Abstract

This study applies Reinforcement Learning (RL), specifically the MORL-DB algorithm, to the multi-objective optimization of a double wishbone suspension system. The goal was to adjust nine hardpoint coordinates to minimize kinematic errors in camber and toe variations. Two reward computation strategies were evaluated: a strict Pareto dominance approach and a rank-based approach. Results indicate that the strict dominance approach excelled at minimizing camber error but limited the solution set , while the rank-based approach provided better coverage for toe variation.

# Contents

# 1. Introduction

The aim of this lab is to get familiar with the fundamental aspects of reinforcement learning by applying it to a multi objective optimization problem.

The backbone of the algorithm has been provided, only the computation of the "reward" (that is going to be described successively) needs to be implemented

## 1.1 Description of the problem

The goal is to perform the optimal kinematic design of a double wishbone suspension system through multi body simulations and reinforcement learning.

The formulation of the problem has been simplified to a case where only the coordinates of three hardpoints are modified and two targets need to be accomplished:

- 9 design variables (Figure 1):
    - **x,y,z coordinates** of the connection point between the lower arm and the hub carrier (**point 3**)
    - **x,y,z coordinates** of the connection point between the upper arm and the hub carrier (**point 6**)
    - **x,y,z coordinates** of the connection point between the tie rod to the hub carrier (**point 10**)
- 2 objective functions
    - **Camber variation with vertical wheel travel error**: difference between the camber variation curve and the target curve given by the assignment
    - **Toe variation error with vertical wheel travel error**: difference between the toe variation curve and the target curve given by the assignment

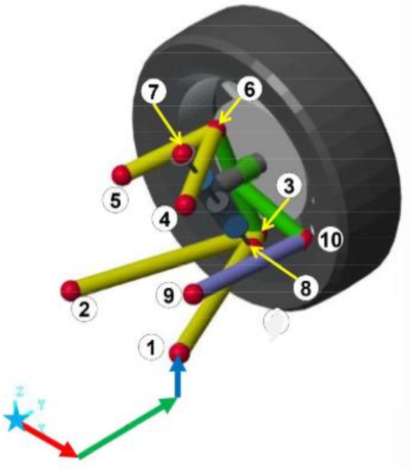

*Figure 1*

# 2. MORL – DB Algorithm

The algorithm that was used in this assignment was reinforcement learning one. Reinforcement learning is based on the interaction of an agent with the environment. The general scheme is the following

- The agent observes the environment. It is said that the algorithm works on a dynamic dataset, since no data is provided to the algorithm a priori.
- The agent performs an action on the environment that perturbs its state. The action that is taken is casual at the beginning but becomes more specific going on; the rule followed by the agent is called policy.
- The environment returns a numerical reward that is perceived (both with the environment state) by the agent.
- The agent can determine if the action was either good or bad and if it should be repeated or not

To sum up, the cycle observation-action-reward is repeated as the agent becomes progressively smarter.

Specifically, the MORL-DB algorithm is model free (it focuses on the reward, no model is provided) and it has the characteristics of both Q-Learning and Policy Optimization algorithms. In detail, they embed respectively two roles: the actor that works to maximize the reward while the critic judges the actions of the actor and helps to choose the future actions. The scheme of the algorithm is shown in

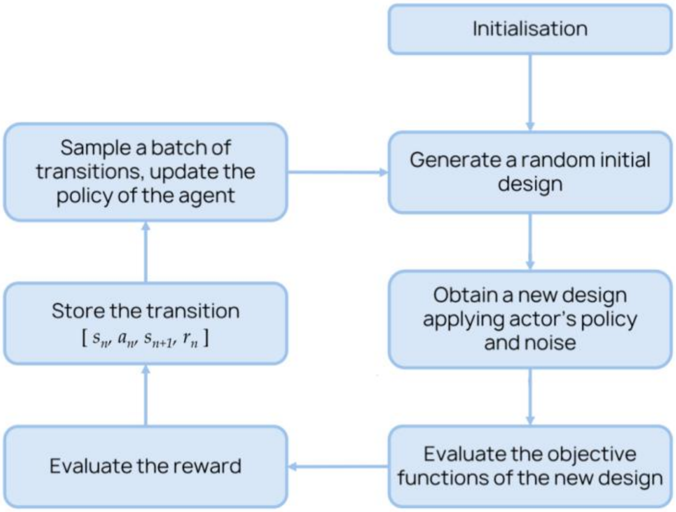
Initialisation
Sample a batch of transitions, update the policy of the agent
Generate a random initial design
Store the transition
[ $s_n$, $a_n$, $s_{n+1}$, $r_n$ ]
Obtain a new design applying actor's policy and noise
Evaluate the reward
Evaluate the objective functions of the new design

*Figure 2*

Using reinforcement learning in optimization design can have several advantages. Firstly, the reward function can be customized (and it can contain also constraint information). Moreover, the agent learns autonomously and no previous knowledge is needed. Moreover, thanks to transfer learning, it can be applied to new or slightly modified problems.

# 3. Results

In this report, two approaches are used to implement the reward computation and the pareto point selection.

## 3.1 First approach

### *Characteristics*

- Reward computation
  - the reward varies between -1 and +1 (arbitrarily chosen)
  - when the newest point computed is part of the pareto set (it is not dominated)

    $$Reward = 1$$
  - when the newest point is not part of the pareto set (it is dominated)

    $$Reward = -\frac{N_{dominatingPoint}}{N_{paretoPoints}}$$
    - $N_{dominatingPoints}$: points in the pareto optimal set that dominate it
    - $N_{paretoPoints}$: number of total points in the pareto set
- The definition of the reward was driven by the following principles
  - Every Pareto optimal solution is good in the same way, but not every bad solution is bad in the same way → reward based on the number of points dominating the solution proposed by the agent accounts for this
  - The number of points that dominate the solution proposed by the agent needs to be evaluated while considering the total number of points on the Pareto optimal set
    - Example 1: the solution proposed by the agent is dominated by 10 points on the Pareto optimal set and there are 15 points on it → "severe" dominance
    - Example 2: the solution proposed by the agent is dominated by 10 points on the Pareto optimal set and there are 1000 points on it → "less severe" dominance
- Pareto set
  - At each iteration, when the newest point is not dominated, the pareto set is recomputed completely. Consequently, only the proper pareto set points are kept at each step. So, the pareto set considered in this issue is the proper pareto set.

### *Training process*

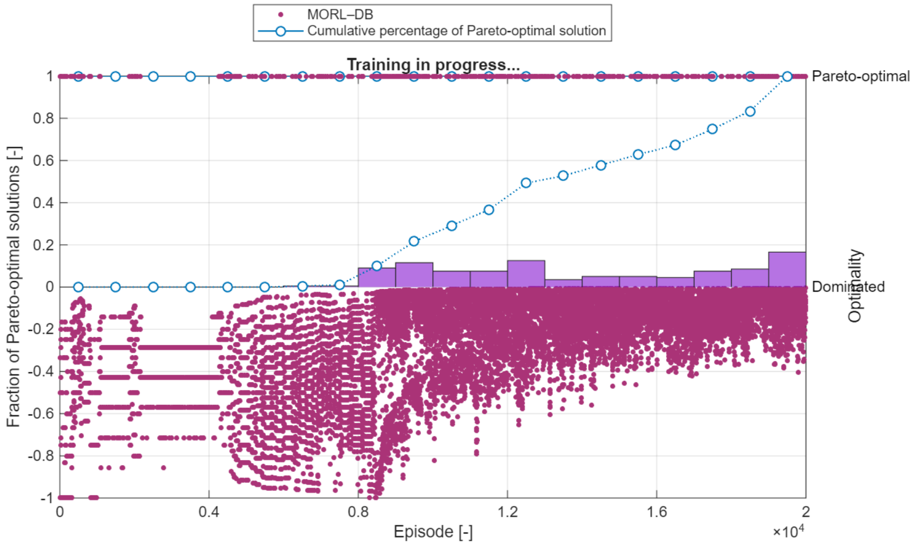

*Figure 3*

- The average reward increases throughout the training process
- At the end of the training process, the average reward is still negative
    - **The selected approach to compute the reward does not seem to be effective (poor results are expected)**
- The height of the histograms represents the ratio between the number of Pareto optimal points found by the algorithm in that set of 1000 episodes (width of each histogram) and the total number of Pareto optimal points

### *Advantages and disadvantages*

At each iteration of the training procedure, the Pareto optimal set is computed with a sorting algorithm

- **Disadvantage**: increased computational time because a sorting algorithm is performed each time only to verify if the newest solution provided by the agent belongs to the Pareto optimal set or not. This sorting procedure could have been performed once at the end of the learning process
- **Advantage**: this ensures that if the newest solution provided by the agent dominates a point belonging to the Pareto optimal set of the previous iteration, this latter point is discarded immediately
    - The size of the Pareto optimal set is kept small, and the sorting algorithm operates on a small set of individuals

### *Results: pareto set*

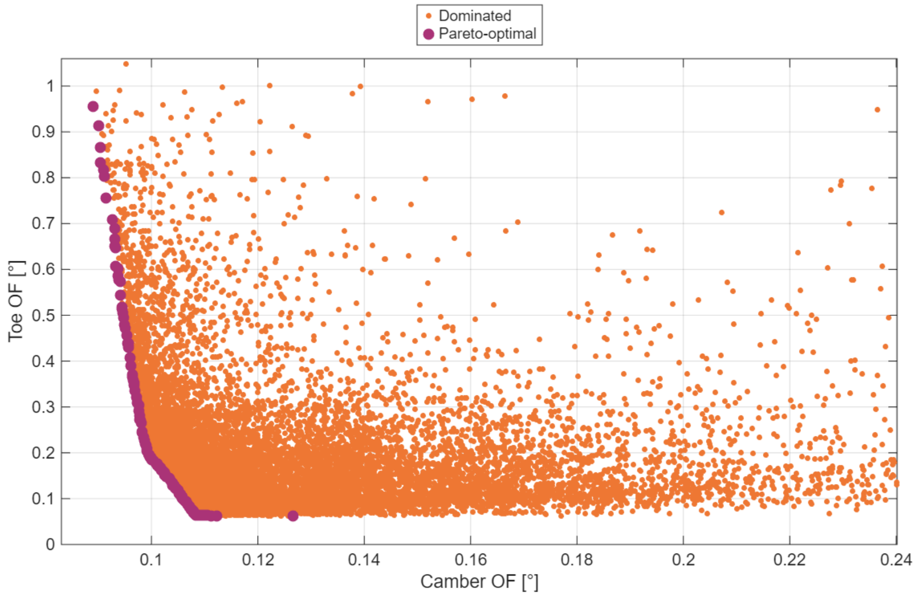

*Figure 4*

- As expected, the number of Pareto optimal points identified by the algorithm is very low (199 points), this confirms the poor quality of the training
- **Almost all the points identified by the algorithm are on the left boundary of the domain**

## 3.2 Second approach

### *Characteristics*

- Reward computation
    - The reward varies between 0 and +1 (arbitrarily chosen)
    - when the newest point computed is part of the pareto set (it is not dominated)

$$Reward = 1$$

    - when the newest point is not part of the pareto set (it is dominated)

$$Reward = \frac{1}{rank}$$

        - $rank$: it is the depth of the point compared to the pareto set, see the definition of dominance-based ranking used in **Lab07**
- Pareto set
    - At each iteration, when the newest point is not dominated by the pareto set points, it is added to the pareto set itself. All the points that have been part of the pareto set are kept until the end, regardless of that they are dominated by the newest point.

### *Training*

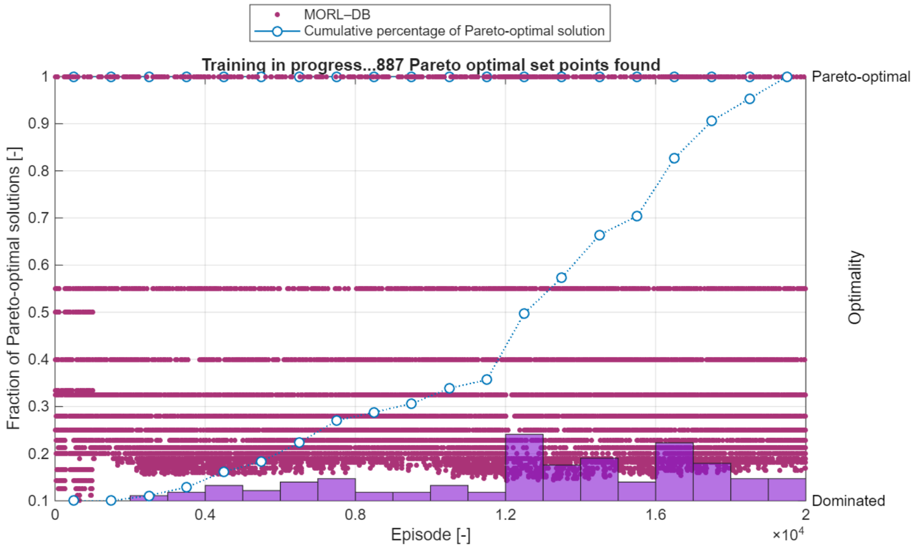

*Figure 5*

- Mind that the “887 points” in the title of Figure 5 does not represent the proper size of the Pareto set. In this second approach it counts also the points that were Pareto when they were found and are no longer part of it.
  - The final number of Pareto optimal points found by the algorithm is 249 (still relatively low considering the potential of this tool but better than before)

### *Results: pareto set*

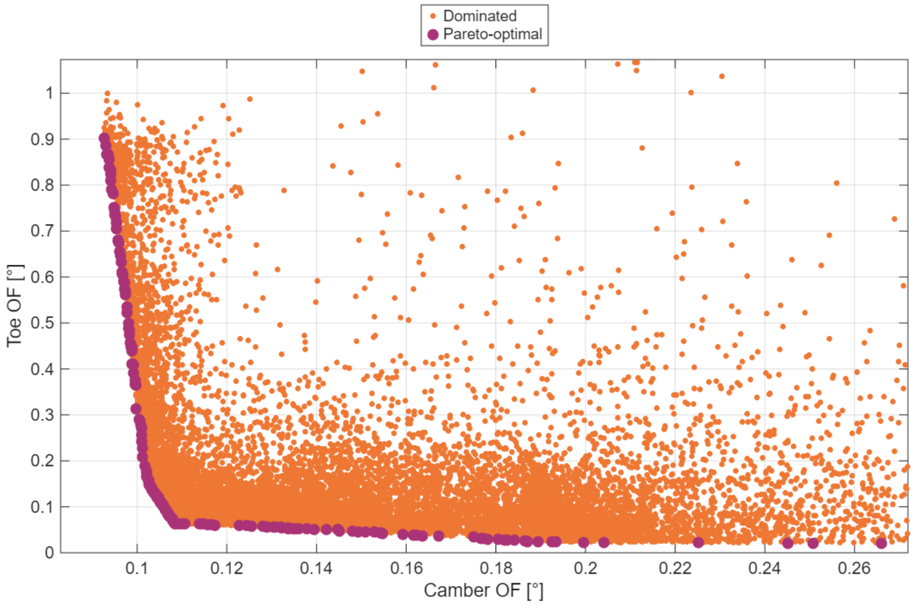

*Figure 6*

- The Pareto optimal points identified by the algorithm are spread across the left and the lower bounds of the domain

## 3.3 Comparison

### *Evaluation of the dominance in between the Pareto fronts found*

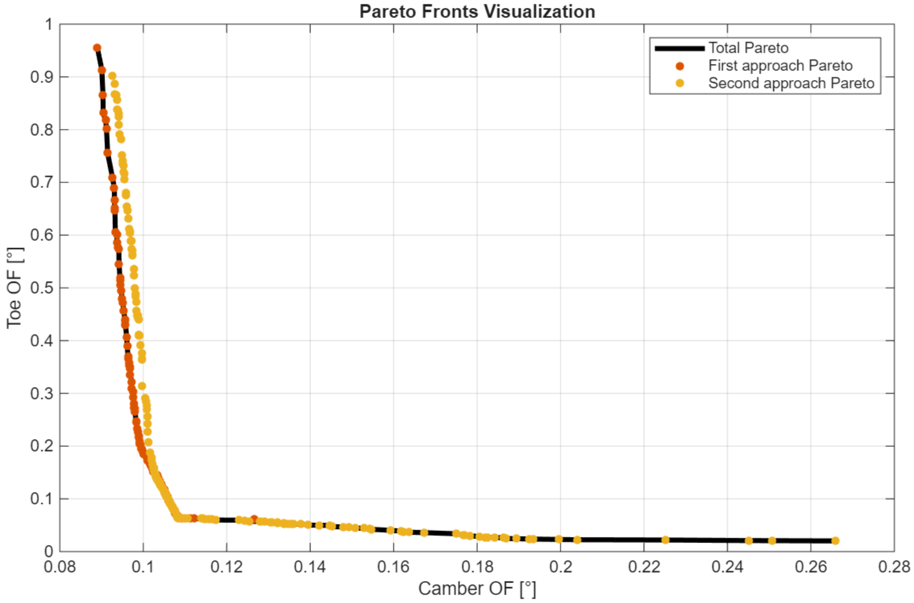

*Figure 7*

- The first approach correctly evaluates the best points that minimize the camber angle objective function**, the pareto set from the first approach clearly dominates the second one in that zone** (expected result considering the definition of the reward).
- The second approach correctly evaluates the best points that minimize the toe angle objective function, whereas the first one is not even able to find many Pareto optimal points in that region.
- In the zone closest to the origin the two approaches give similar results.

Depending on the specific case different approaches may be chosen, it is not possible to conclude which one is best from this analysis, some further trials would be needed.

# 4. Comparison with DOE

### *Comparison between DOE approach and RL approach*

- In the DOE laboratory less design variables were considered but the goal was still to minimize the camber and toe variation during a parallel wheel travel maneuver
- In the DOE lab, a quadratic model was used to fit the behavior of the vehicle, and the optimization was performed using the reduced order model
- In this lab, the agent interacts directly with the multibody model and through the algorithm learns how to modify the hardpoints coordinates to find the optimal solution. The quality of the modification proposed is evaluated in terms of reward
- The RL agent can be used to identify new optimal configurations (inference) or transferred to new optimization problems (transfer learning)
    - **Inference**: the RL is queried to determine new optimal configurations in addition to those determined during the training. The agent's knowledge can be exploited to refine the optimization at a later stage.
    - **Transfer learning**: an agent previously trained in a specific environment is used and trained in a new environment without starting from scratch; this can be used to optimize model variants or when considering a new load case (if a DOE approach was used, the reduced order model would have to be characterized from scratch)

# 5. Conclusion

The laboratory results confirm that the RL agent can successfully optimize suspension kinematics without prior knowledge of the system model. The choice of reward function proved critical: Approach 1 (dominance count) concentrated solutions on the camber-minimizing boundary, whereas Approach 2 (ranking) achieved a broader distribution of Pareto optimal points, particularly for toe targets. Although the RL training yielded a negative average reward initially, the method offers distinct advantages over DOE by enabling the agent's trained policy to be transferred to new load cases or model variants without re-characterization