RL Project

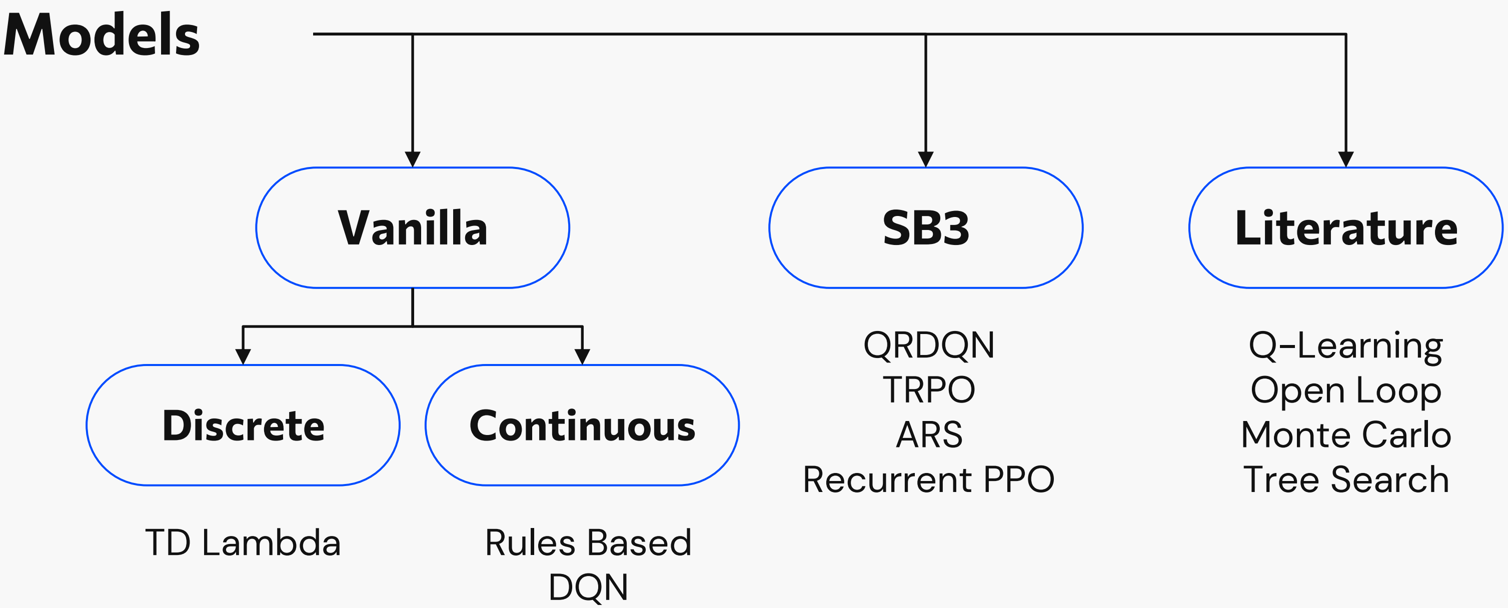
Group 8:
Chia Dehan
Colin Jiang Kelin
Lim Wei Jie
Ng Juan Yong

Table of contents

01

Task 1 Approach

Algorithms


Models
Vanilla
SB3
Literature
Discrete
Continuous
QRDQN
TRPO
ARS
Recurrent PPO
Q-Learning
Open Loop
Monte Carlo
Tree Search
TD Lambda
Rules Based
DQN

Vanilla

These models are built from James’ templates/by ourselves

They can be further categorized into discrete and continuous state spaces with discrete action spaces

- Rules Based
- TD Lambda
- DQN

Vanilla

Rules Based

Rules are created to manage pit stop and tyre changing decisions

The Rules

- If Tyre Condition < A
    - pit stop = True

- If Weather is Dry
    - Choose 0
- elif Wet% < B
    - Choose 1
- elif Wet% < C
    - Choose 2
- else
    - Choose 3

- If Laps Cleared > D
    - Choose 4

Thresholds A,B,C,D are tuned based on GridSearch

```
Fixed weather sequence
rewards = []

For A in range(0,100):
  For B in [20,40,60,80,100]:
    For C in [20,40,60,80,100]:
      For D in range(0,162):
        g = 0
        while not done:
          ...racing...
          g += r
        rewards.append(((A,B,C,D),g))
```

Vanilla

Rules Based

Rules are created to manage pit stop and tyre changing decisions

Different thresholds for different radius

Bin radius into intervals of 50; each interval will have its own thresholds

The Rules

- If Tyre Condition < A
  - pit stop = True
- If Weather is Dry
  - Choose 0
- elif Wet% < B
  - Choose 1
- elif Wet% < C
  - Choose 2
- else
  - Choose 3
- If Laps Cleared > D
  - Choose 4

Thresholds A,B,C,D are tuned based on GridSearch

Evaluation via bootstrapping over 100 episodes for each radius bin

~~Fixed weather sequence~~

```
num_episodes = 100
rewards = {}

For radius in range(625,1200,25):
  radius_bin = radius // 50 - 12
  For A in range(0,100):
    For B in [20,40,60,80,100]:
      For C in [20,40,60,80,100]:
        For D in range(0,162):
          For _ in num_episodes:
            iteration_radius = radius + randint(0,49) - 25
            g = 0
            while not done:
              ...racing...
              g += r
            rewards[radius_bin].append(((A,B,C,D),g))
```

Vanilla

Rules Based

Rules are created to manage pit stop and tyre changing decisions

Different thresholds for different radius

Bin radius into intervals of 50; each interval will have its own thresholds

Best Rules

```
{0: {'condThreshold': 0.79,
  'weatherThreshold1': 20,
  'weatherThreshold2': 60,
  'no_change_after_lap': 160},
 1: {'condThreshold': 0.85,
  'weatherThreshold1': 20,
  'weatherThreshold2': 60,
  'no_change_after_lap': 161},
 2: {'condThreshold': 0.87,
  'weatherThreshold1': 20,
  'weatherThreshold2': 60,
  'no_change_after_lap': 161},
 3: {'condThreshold': 0.82,
  'weatherThreshold1': 20,
  'weatherThreshold2': 60,
  'no_change_after_lap': 160},
 4: {'condThreshold': 0.86,
  'weatherThreshold1': 20,
  'weatherThreshold2': 60,
  'no_change_after_lap': 160},
 5: {'condThreshold': 0.82,
  'weatherThreshold1': 20,
  'weatherThreshold2': 60,
  'no_change_after_lap': 160},
```

Each key-value pair corresponds to a radius-bin to best rules pair

For example:
0 : {
'condThreshold': 0.79,
'weatherThreshold1': 20,
'weatherThreshold2': 60,
'no_change_after_lap': 160
}

The key 0 refers to the first radius bin, which is radius in range of 600-649

The values is the rules the agent will follow corresponding to the radius

Vanilla

TD-Lambda

Tyre Condition, Radius, Laps Cleared, are discretized into bins

Models created for lambdas: {0, 0.2, 0.4, 0.6, 0.8, 1.0}

Vanilla

DQN

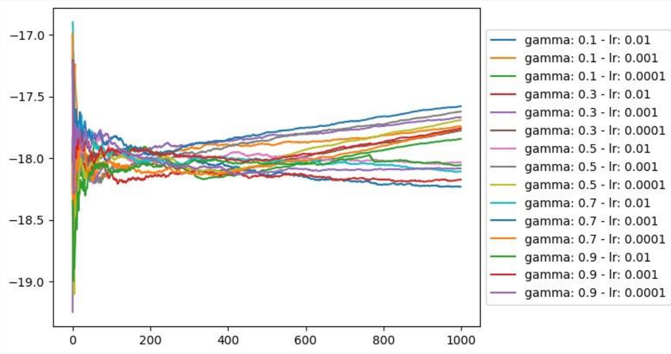
Tyre type, Weather Conditions, are one hot encoded

Hyper Parameter Tuning: Gamma, Learning Rate, Tau, Neurons

Decaying Epsilon for first 1000 episodes

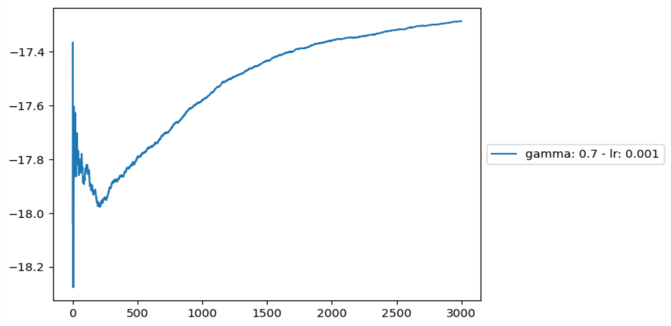
Hyper Parameter Tuning to find optimal gamma and learning rate

Evaluated using Cumulative Reward per Radius



Cumulative Reward per Radius continues to climb at 3000 episodes

However, evaluation results showed no signs of improvements after 2000 episodes



SB3

These models utilizes Stable Baseline 3 environment wrappers

They are all using continuous state spaces and discrete action space

- TRPO
- ARS
- Recurrent PPO
- QRDQN

SB3

Discrete Actions

Car Tyres

Continuous States

Tyre type, Weather Conditions, are one hot encoded

Radius, Tyre Conditions, Laps Cleared are Floats

Literature

These models are adapted from research papers

- Q-Learning Open Loop Planning

Reference:

Piccinotti, D., Likmeta, A., Brunello, N., & Restelli, M. (2021). *Online planning for F1 race strategy identification - github pages*. Online Planning for F1 Race Strategy Identification. https://prl-theworkshop.github.io/prl2021/papers/PRL2021_paper_1.pdf
Journal: Association for the Advancement of Artificial Intelligence (AAAI), www.aaai.org

Literature

Q-Learning Open-Loop Planning (Monte Carlo Tree Search)

Open loop refers to doing MCTS at every lap.

In the backpropagation step, we update the Q-function.

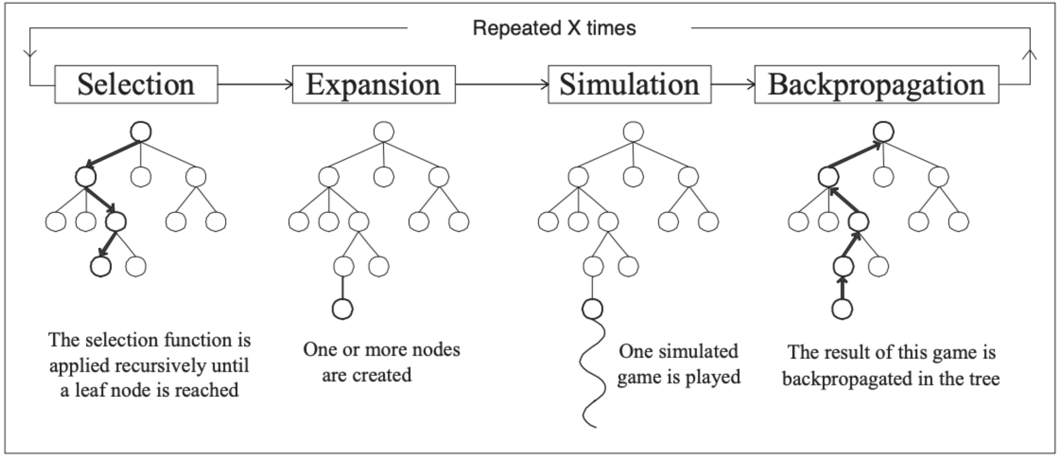
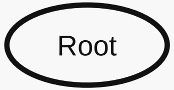


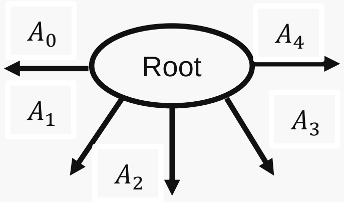
Figure 1: Outline of a Monte-Carlo Tree Search.

Monte Carlo Tree Search

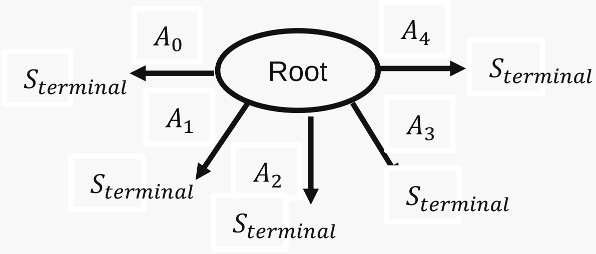
Step 1: At the ¾ mark of each lap: create root node



Step 2: Expand root node to all 5 unexplored children (i.e. 5 actions)

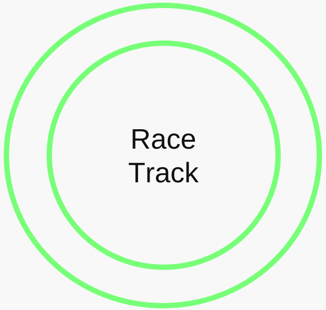


Step 3: For each child node, we rollout to the terminal state (i.e. 162 laps)



Rollout Policy – Determine Pitting Threshold

Pitstop Time = 23 seconds



$At\ ¾\ of\ each\ lap$:

Step 1: Calculate

$$Lap\ Time = \frac{Radius}{Velocity}$$

Step 2: Tune threshold

$$Lap\ Time = \frac{Radius}{Velocity} > 1.5 * 23$$

Setting a certain threshold to start considering pitstops

- Considerations:
  - For tracks with smaller radiuses, we do not want pitstops to occur too fast or too frequently
  - For tracks with larger radiuses, we do not want pitstops to be missed too frequently as the tyre degrades faster every lap.
- After several empirical testing, we find that if a car's lap time for any track hits approximately 9 times of pitstop time, it is time to consider stopping for a change of tyres.

$$Lap\ Time = \frac{2 * pi * Radius}{Velocity} > 9 * 23$$

- For simplicity, we reduce the number of calculations by using the radius as lap time proxy and reducing the RHS to 1.5 times of pitstop time

$$Lap\ Time\ Threshold\ Proxy = \frac{Radius}{Velocity} > 1.5 * 23$$

Rollout Policy – Greedy Epsilon Strategy to Account for Tyre-to-Weather Durability and Track Radius Factors

Next, we set a threshold for which tyres we should change to.

- Based on intuition and empirical testing, these tyres are most suited to the following weather conditions.

Step 1: Fix Specific Weather conditions that favors specific tyre types and set initial epsilon threshold

```
#favourable weather for specific tyre
fav_weather_tyre = {
"Ultrasoft": ["Dry", "20% Wet"],
"Soft": ["20% Wet"],
"Intermediate": ["40% Wet", "60% Wet", "80% Wet"],
"Fullwet": ["80% Wet", "100% Wet"]
}
```

$$InitialEpsilonThreshold = curr\_threshold = 0.5$$

Rollout Policy – Greedy Epsilon Strategy to Account for Tyre-to-Weather Durability and Track Radius Factors

The intuition here is that

- longer tracks will require more immediate changing of tyres as the next time to pit might be a lot longer later.
- Shorter tracks will have a lower probability to change tyres even if weather is unfavorable as the next time to change tyres is only a short while later.

<u>Step 2: Account for Track Radius by Adjusting linearly</u>

If weather is favorable to current tyre type:

- Decrease epsilon threshold linearly by factoring in track radius.
- `curr_threshold -= (0.45 * (1-(curr_state[-2]-600)/600))`

If weather is unfavorable to current tyre type:

- Increase epsilon threshold linearly by factoring in track radius.
- `curr_threshold += (0.45 * (1-(curr_state[-2]/600)-1))`

Note here that radius = `curr_state[-2]`

Rollout Policy – Greedy Epsilon Strategy to Account for Tyre-to-Weather Durability and Track Radius Factors

The intuition here is that

- longer tracks will require more immediate changing of tyres as the next time to pit might be a lot longer later.
- Shorter tracks will have a lower probability to change tyres even if weather is unfavorable as the next time to change tyres is only a short while later.

Step 3: Determine Tyres

```
epsilon_for_change = random.random()
if epsilon_for_change <= curr_threshold:
            doTyreSelection()
Else:
            doNothing
```

Rollout Policy – Tyre Selection

- To choose which tyres to fit on, we focus on the durability factor of the tyre in all race conditions.

```
possible_tyres = ["Ultrasoft", "Soft", "Intermediate", "Fullwet"]
possible_weather = ["Dry", "20% Wet", "40% Wet", "60% Wet", "80% Wet", "100% Wet"]
for radius in range(600, 1250, 50):
   for weather in possible_weather:
      doMeasurement()
```

- The durability factor is measured by:
  - Lap at which tyres reach close to 0 for a specific constant weather
  - Linear rate of deceleration across time: `(velocities[0]-velocities[-1])/lap_count`
- We weigh the 2 subfactors equally to get the most durable tyre compound for a specific radius and specific weather condition.

Rollout Policy – Tyre Selection

- The durability factor is measured by:
    - Lap at which tyres reach close to 0 for a specific constant weather
    - Linear rate of deceleration across time: (velocities[0]-velocities[-1])/lap_count
- We weigh the 2 subfactors equally to get the most durable tyre compound for a specific radius and specific weather condition.
- Top 25 most durable tyre compounds across various radius and weather conditions.

```
(1200, '100% Wet', 'FullWet') 22.05095237410584
(1150, '100% Wet', 'FullWet') 21.267627390716008
(1200, 'Dry', 'Soft') 21.218196033510914
(1200, '40% Wet', 'Intermediate') 20.851773683205177
(1200, '60% Wet', 'Intermediate') 20.851773683205177
(1200, 'Dry', 'Ultrasoft') 20.563831229170095
(1200, '80% Wet', 'FullWet') 20.5466940287321
(1100, '100% Wet', 'FullWet') 20.5466829521211
(1150, 'Dry', 'Soft') 20.437901697550082
(1150, '40% Wet', 'Intermediate') 20.192200724139802
(1150, '60% Wet', 'Intermediate') 20.192200724139802
(1150, '80% Wet', 'FullWet') 19.887833140354747
(1050, '100% Wet', 'FullWet') 19.88782208389371
(1150, 'Dry', 'Ultrasoft') 19.845847518527904
(1200, '20% Wet', 'Soft') 19.83955810553273
(1100, 'Dry', 'Soft') 19.77982225253882
(1100, '40% Wet', 'Intermediate') 19.534661055489636
(1100, '60% Wet', 'Intermediate') 19.534661055489636
(1200, '20% Wet', 'Intermediate') 19.53465619889716
(1200, '80% Wet', 'Intermediate') 19.53465619889716
(1200, '20% Wet', 'Ultrasoft') 19.309062686027673
(1000, '100% Wet', 'FullWet') 19.290710073719108
(1150, '20% Wet', 'Soft') 19.24303048392042
(1200, '60% Wet', 'FullWet') 19.2311046424955
(1100, '80% Wet', 'FullWet') 19.231093534808398
```

Pseudo Code of our Rigorously Flexible Q-Learning Open Loop Planning

```
> Procedure OLSEARCH
  > Create root node N_{0,0} from state s_0
  > while within computational budget do
    > N_{d,i,s}, s ← TREEPOLICY(N_{0,0})
    > V(N_{d,i}) ← ROLLOUT(N_{d,i}, s)
    > BACKUP (N_{d,i}, s)
  > end while
> End procedure

> Procedure TREEPOLICY(N)
  > while N not terminal do
    > if N not fully expanded then
      > return EXPANDED(N)
    > end if
  > end while
  > return N
> End procedure
```

- At each lap, we create a root node.
- We then expand the root node to its unexplored actions and subsequent states.
- Thereafter, we do a rollout to terminal state based on the previously discussed rollout policy.
- The backup function computes the Q value which represents the expected returns based on the rollout strategy.

Pseudo Code of our Rigorously Flexible Q-Learning Open Loop Planning

```
> Procedure ROLLOUT(N,s)
  > Δ← 0
  > while s is non-terminal do
    > Choose a ∈ A(s) according to rollout strategy
    > Generate next state s' and reward r
    > Δ← γΔ + r
    > s ← s'
  > end while
  > return Δ
> End procedure

> Procedure EXPAND(N)
  > Choose a ∈ untried actions from N
  > s ← Expand one node down
  > Execute a in s generating s' and r
  > Add a new child N' to N
  > N'.n ← 0
  > N'.r = r store the reward obtained in first visit
  > return N', s'
> End procedure

> Procedure BACKUP(N, V)
  > C'(N) denotes explored children nodes of N
  > N'← parent of N
  > while N' is not null do
    > if N is leaf then
      > Δ← V
    > else
      > Δ← max_{a'∈C'(N)} Q(N, a')
    > end if
    > Q(N', a) ← Q(N', a) + α(N', r + γΔ − Q(N', a))
    > N ← N'
    > N' ← parent of N
  > end while
> End procedure
```

Final Selection – Combined Returns

```
#for shorter races, we want fewer pitstops, so we look more at future rewards
self.weight_future_q = ((1-(state[-2]-600)/600))
```

- We take the maximum of long-term Q-value and the short-term immediate reward of taking the next action.
- The weightage again takes into consideration of track radius. A smaller track radius would want fewer pitstops taken, thus we weigh Q-values, a longer-term view, more than immediate rewards.

```
if self.weight_future_q <= 0.25:
  self.weight_future_q = 0.15
elif self.weight_future_q > 0.25 and self.weight_future_q <= 0.75:
  self.weight_future_q = 0.35
```

- We do a further clipping of the weightage of future q-values.
- We find that as track radius get larger, the decay in the importance of future q-values is exponential but smooths eventually to an asymptote. Thus, we try to replicate the exponential decay to an asymptote with a clipping function as shown above.

02

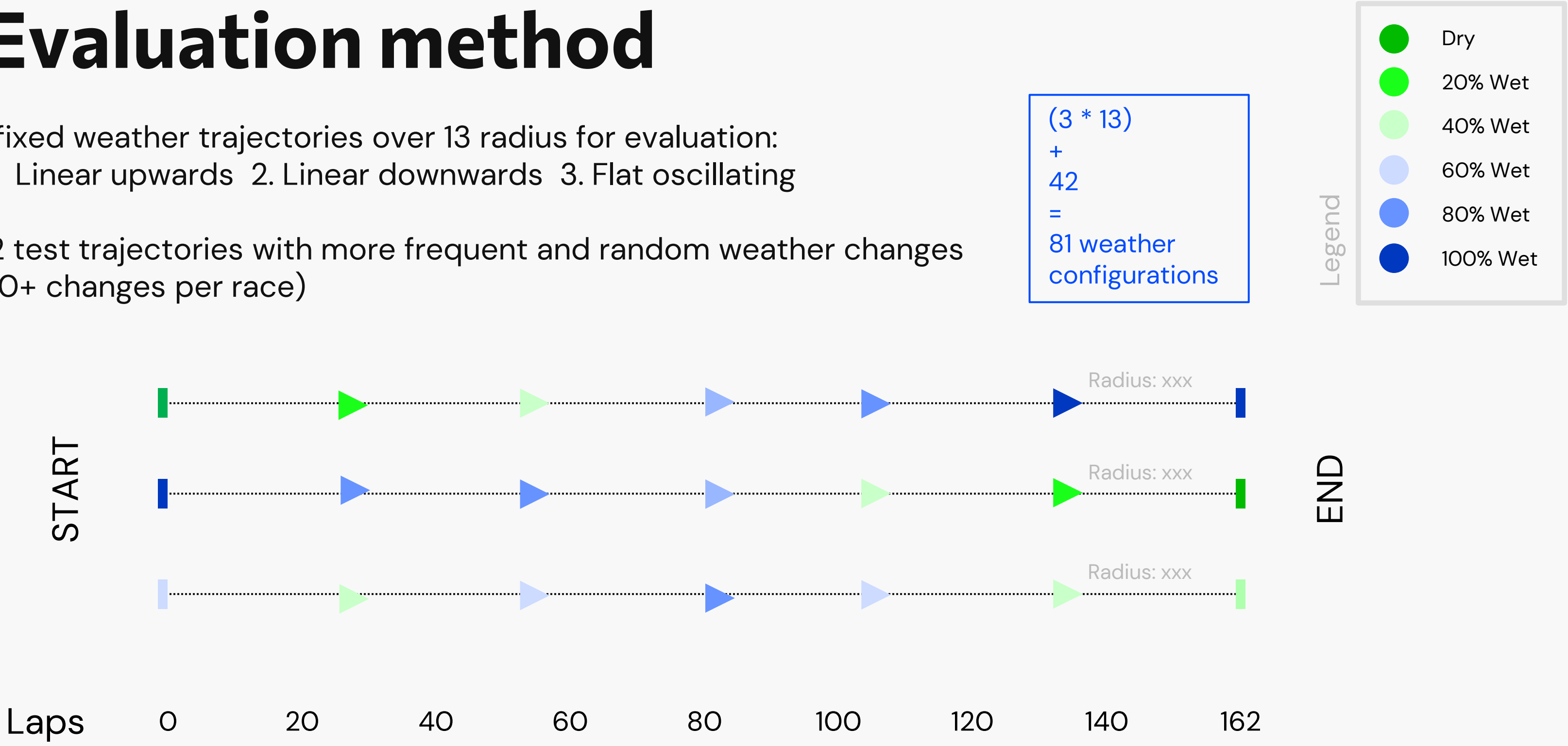
Task 1 Results

Evaluation method

3 fixed weather trajectories over 13 radius for evaluation:
1. Linear upwards 2. Linear downwards 3. Flat oscillating

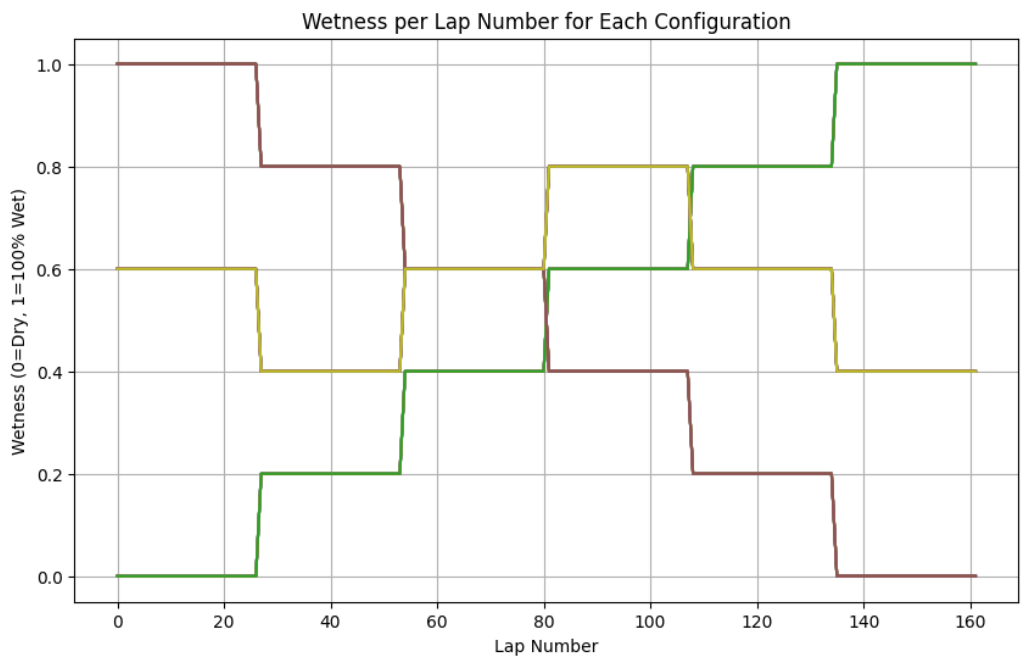
42 test trajectories with more frequent and random weather changes (20+ changes per race)

(3 * 13)
+
42
=
81 weather configurations



Evaluation method

3 fixed weather trajectories



Results – Top Models

*Not real results – For illustration only

Weather Sequence	MCTS	DQN	Rules Based
0	-11100.7	-11092.2	...
1	-11831.2	-11815	...
2	-12522.1	-12548.8	...
3	-13259.2	-13228.3	...
...	...	...	...
...	...	...	...
78	-15340.1	-15384.9	...
79	-15994.8	-16008.6	...
80	-16697.5	-16696.3	...
81	-17380.9	-17409.4	...

We find our top model by summing the count of weather sequence which the model scored highest in

The top 3 models are:
Open loop (monte carlo tree search)
DQN (vanilla)
Rules Based

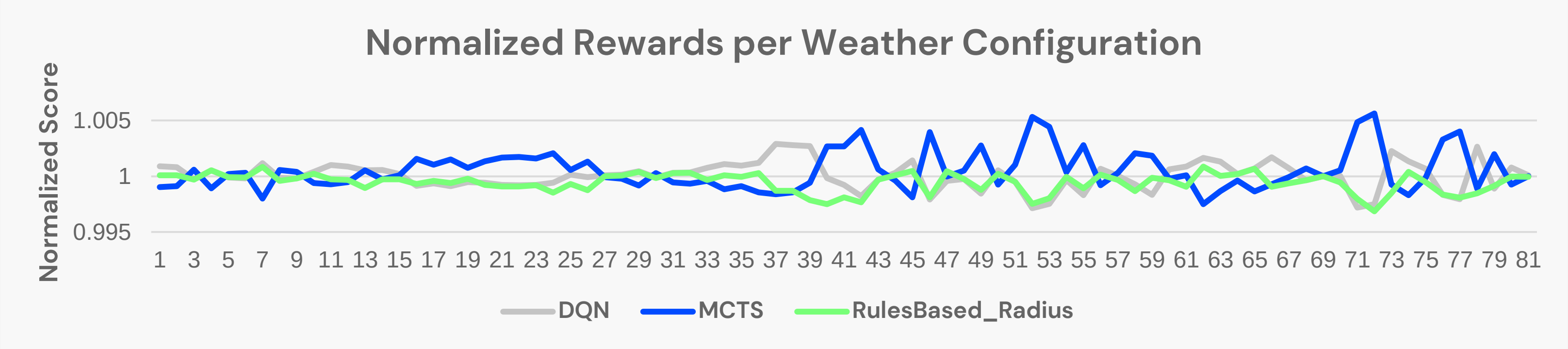
Results – Top Models

Models	Number of wins out of 81 weather configs
DQN	21
Rulesbased	32
Open loop MCTS	28

RulesBased outperforms other 2 models

Also shows more consistent results

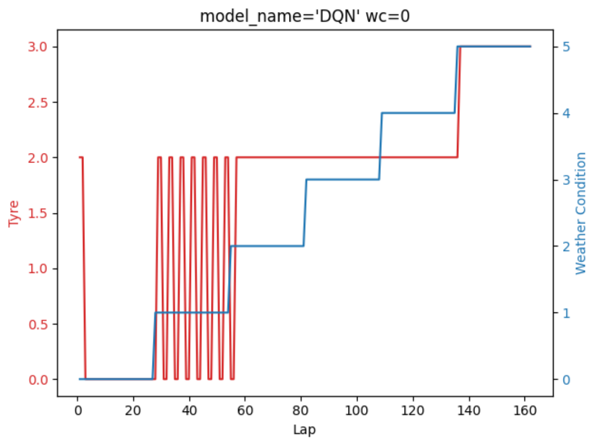
This chart demonstrates the consistency of RulesBased approach.
Lower Normalized Score = Higher Rewards

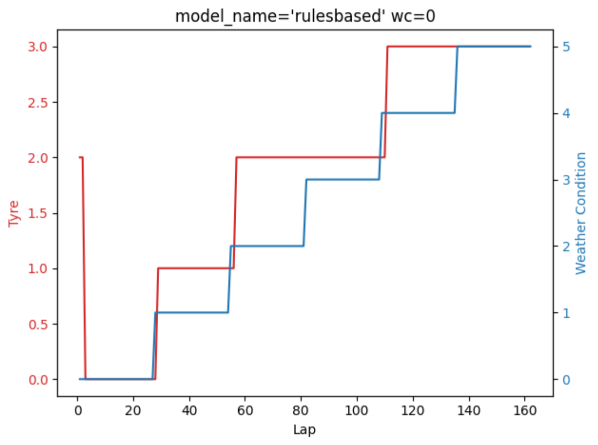


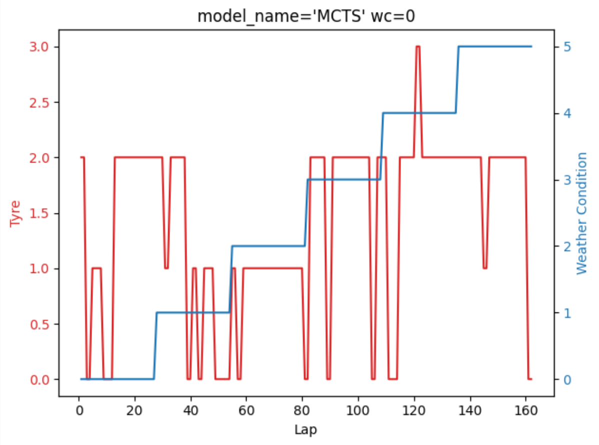
Model Behaviour: WC=0

The six weather states:
0. Dry (eg. sunny day)
1. 20% Wet (eg. drizzle which just started)
2. 40% Wet
3. 60% Wet
4. 80% Wet
5`. 100% Wet (eg. heavy rain)

The four possible tyre choices:
0. Ultrasoft
1. Soft
2. Intermediate
3. Fullwet







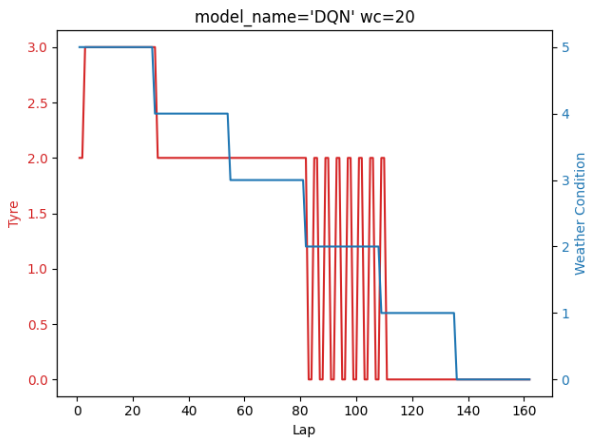
<u>Notable behavior</u>
1. Rulesbased will act immediately after valid weather changes
2. DQN shows heavy oscillation but is similar to rulesbased behaviour.
3. MCTS behaviour erratic but seems to react appropriately to weather changes

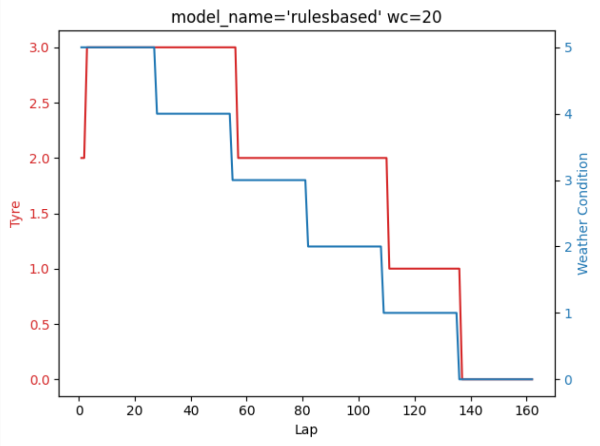
Model	Reward
DQN	-11102
Rules-based	-11093
MCTS	-11087

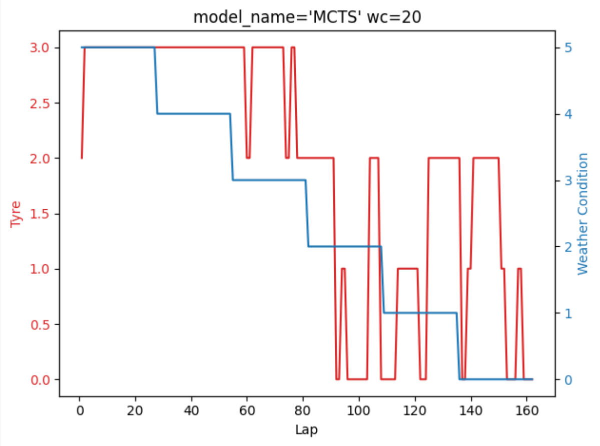
Model Behaviour: WC=20

The six weather states:
0. Dry (eg. sunny day)
1. 20% Wet (eg. drizzle which just started)
2. 40% Wet
3. 60% Wet
4. 80% Wet
5`. 100% Wet (eg. heavy rain)

The four possible tyre choices:
0. Ultrasoft
1. Soft
2. Intermediate
3. Fullwet







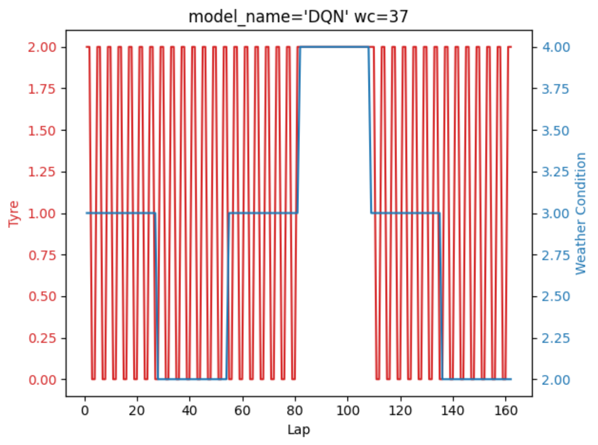
Notable behavior
1. Rulesbased will act immediately after valid weather changes
2. DQN shows heavy oscillation but is similar to rulesbased behaviour.
3. MCTS behaviour erratic but seems to react appropriately to weather changes

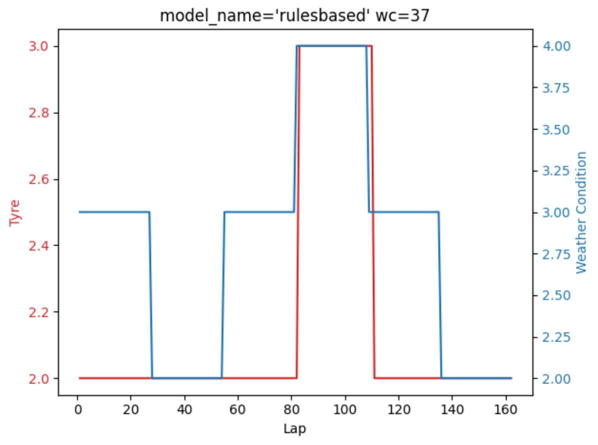
Model	Reward
DQN	-16009
Rules-based	-16007
MCTS	-16014

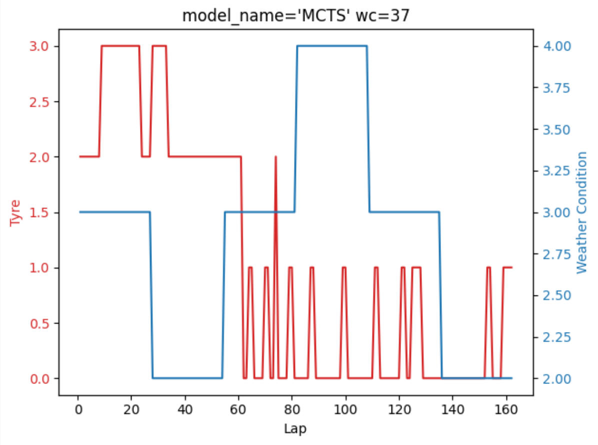
Model Behaviour: WC=37

The six weather states:
0. Dry (eg. sunny day)
1. 20% Wet (eg. drizzle which just started)
2. 40% Wet
3. 60% Wet
4. 80% Wet
5`. 100% Wet (eg. heavy rain)

The four possible tyre choices:
0. Ultrasoft
1. Soft
2. Intermediate
3. Fullwet







<u>Notable behavior</u>

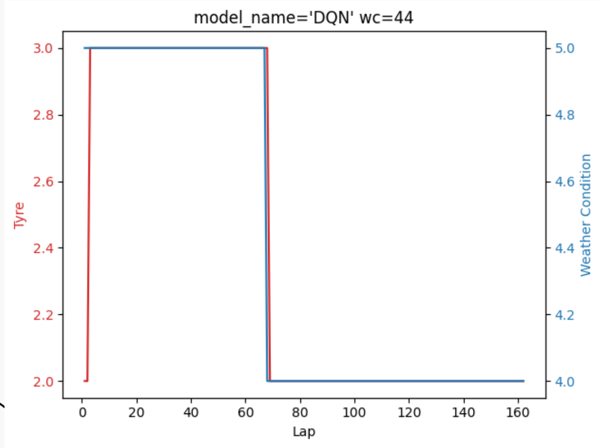
1. Rulesbased responds to 60% Wet weather condition
2. DQN shows heavy oscillation in tyres, which is hard to interpret.
3. MCTS takes a different tyre trajectory compared to rulesbased, using all 4 tyre types.

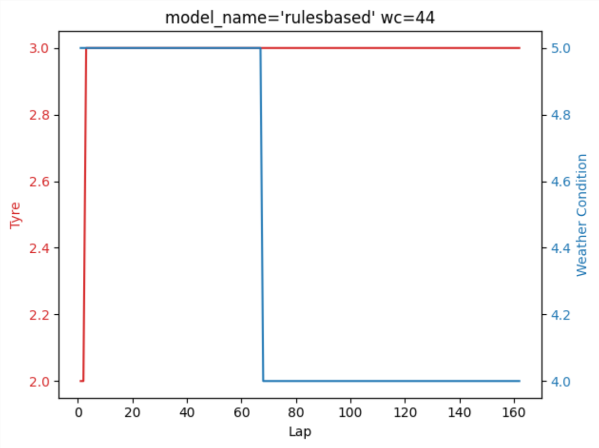
Model	Reward
DQN	-18770
Rules-based	-18694
MCTS	-18588

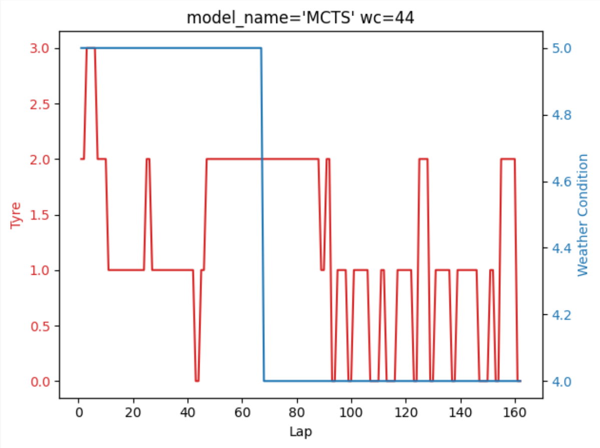
Model Behaviour: WC=44

The six weather states:
0. Dry (eg. sunny day)
1. 20% Wet (eg. drizzle which just started)
2. 40% Wet
3. 60% Wet
4. 80% Wet
5`. 100% Wet (eg. heavy rain)

The four possible tyre choices:
0. Ultrasoft
1. Soft
2. Intermediate
3. Fullwet







Notable behavior

1. Rulesbased chooses a constant tyre
2. DQN similar to rulesbased, but changes tyre to Intermediate upon weather change.
3. MCTS takes a different tyre trajectory compared to rulesbased, using all 4 tyre types.

Model	Reward
DQN	-11139
Rules-based	-11128
MCTS	-11078

03

Task 2

Key Addition

Multiple Racers – each aiming for best race ranking

- Rewards based on lap position
- Option to drive recklessly or conservatively, but risk getting into an accident

Each Racer follows different behaviors

Multi-Agent?

Not Really. Racers take action based on Lap ranking, but not on other Racers' actions

Additonal Elements

Driving style

Reckless or Conservative driving styles.

Driving recklessly increases chances of accidents

Complexity

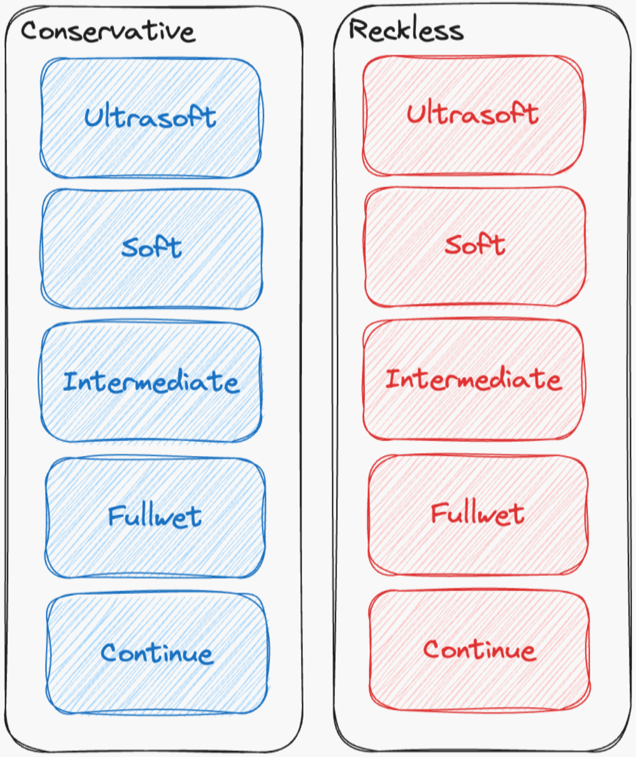
Track complexity.

Higher complexity increases probability of accidents

Accidents

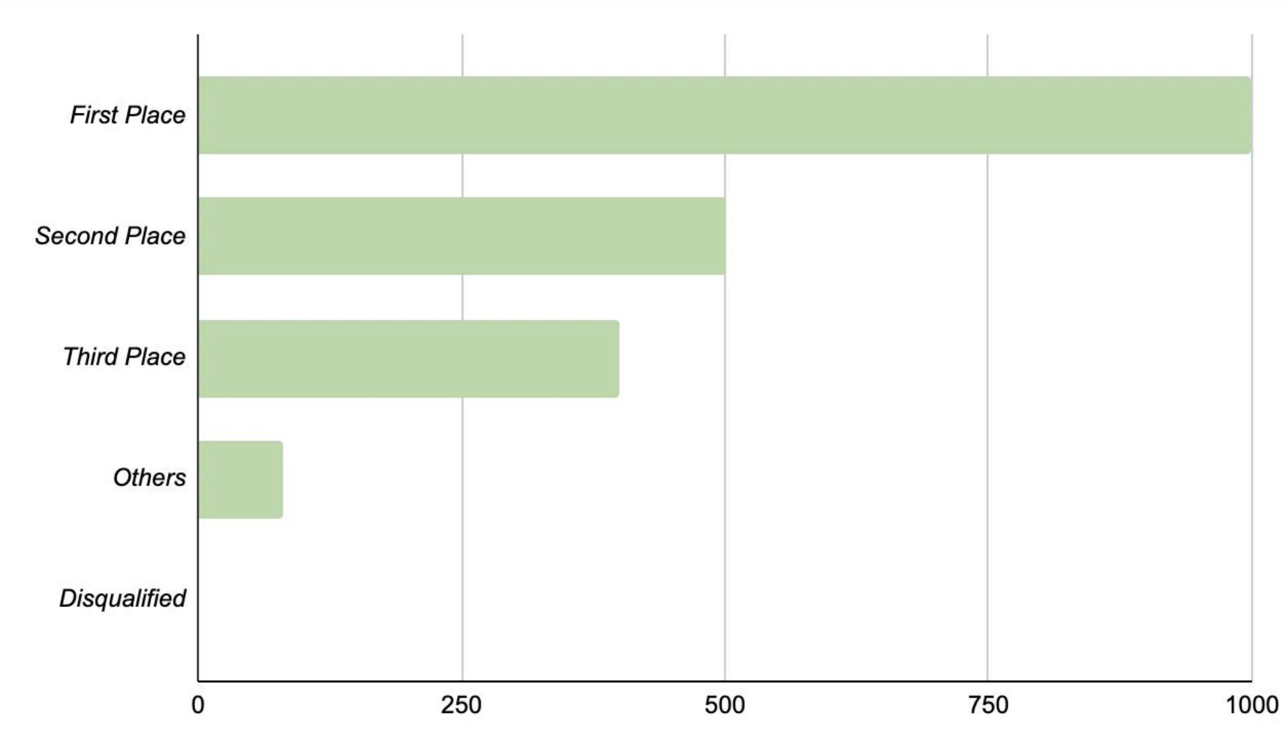
Accident = Game Over

Action Space



Rewards

Rewards



Decreasing reward for lower placing.

Same rewards for placing fourth and below.

Zero reward for not completing the race.

Algorithms

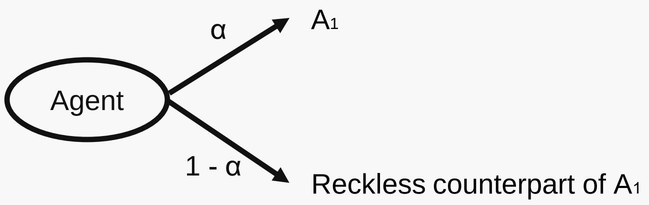
Models

- **Multiple Models**
  - Split models
  - Stack models
- **Probabilistic**
  - DQN from Task 1
- **SB3**
  - DQN

Probabilistic Models

Risk Level

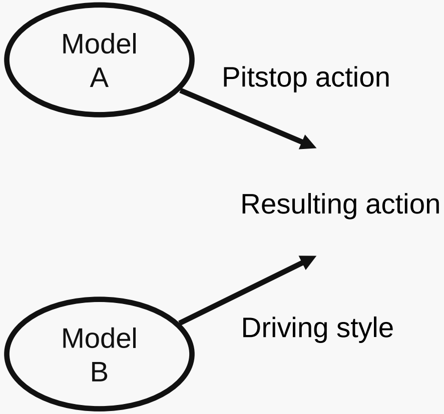
Agents with higher risk level has a higher probability to adopt a reckless driving style for a specific lap.



Multiple Models

Split Models

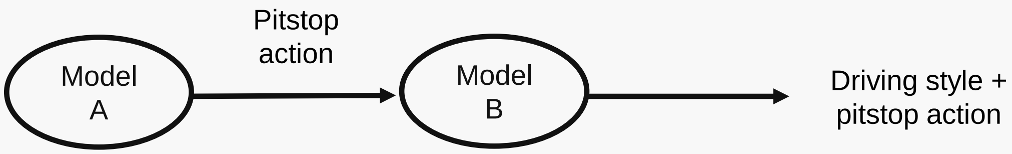
One model determines the pitstop action, while the other determines the driving style to be adopted.



Multiple Models

Stack Models

One model determines the pitstop action and the other determines resulting action by taking the model's output into account.



Evaluation method

Participants

10 participants – probabilistic models (p = [0, 0.5]), DQN, split models and stack models.

Rounds

50 races for each complexity level of 0, 0.5 and 1.0.

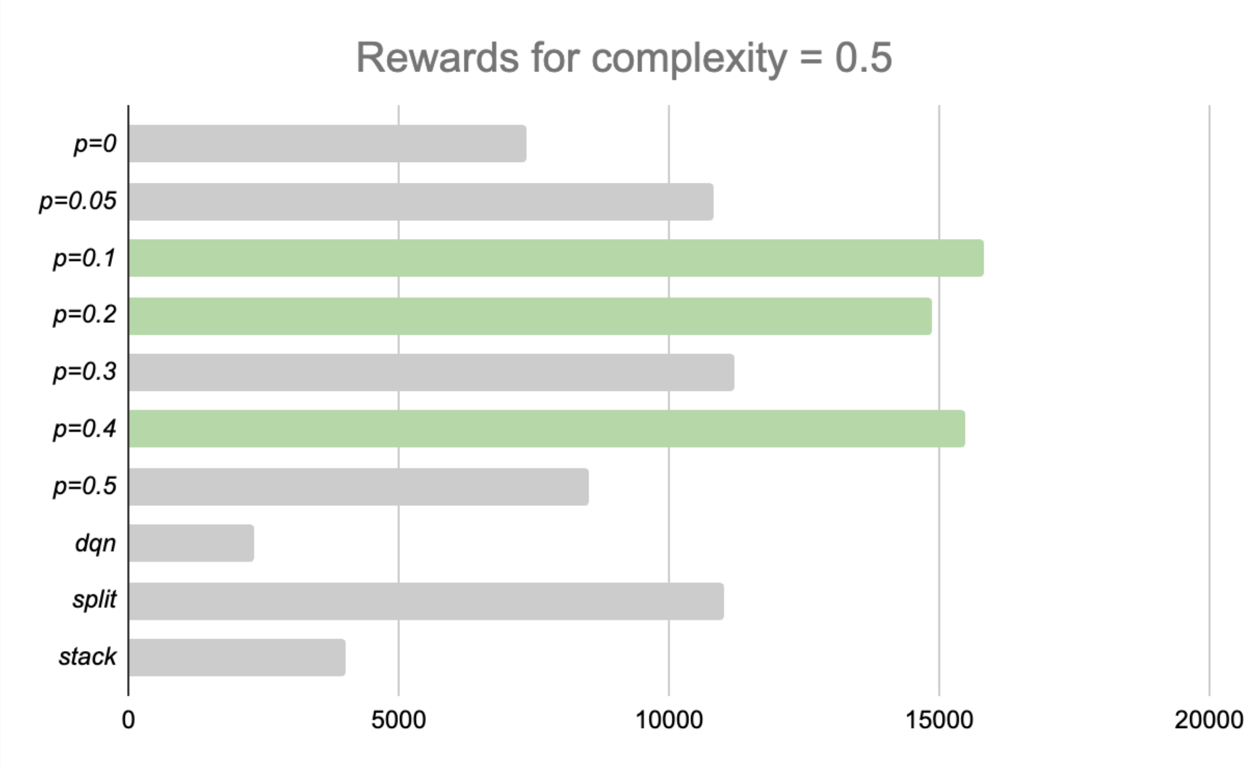
Accident Rates

Conservative - 0%
Reckless – 1.5% to 3.0%

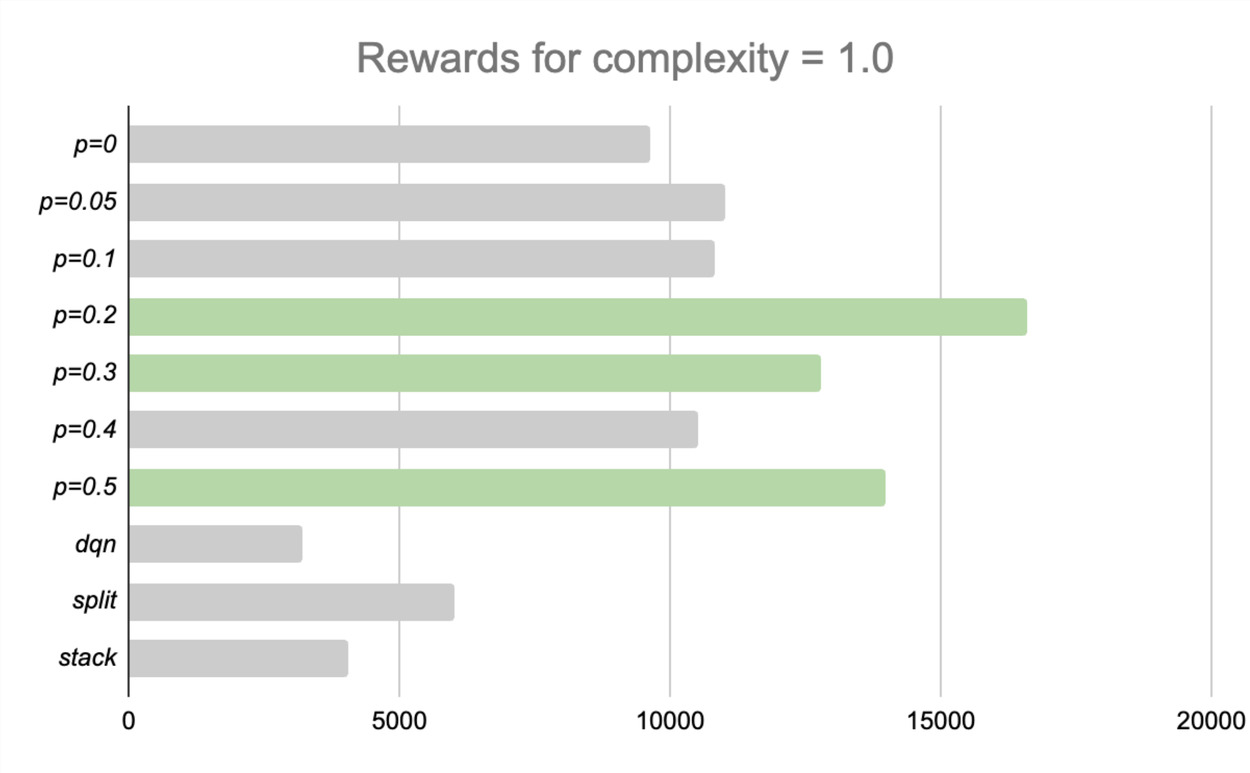
Results

Rewards for complexity = 0.0
p=0
p=0.05
p=0.1
p=0.2
p=0.3
p=0.4
p=0.5
dqn
split
stack
0
5000
10000
15000
20000
25000

Results


Rewards for complexity = 0.5
p=0
p=0.05
p=0.1
p=0.2
p=0.3
p=0.4
p=0.5
dqn
split
stack
0
5000
10000
15000
20000

Results

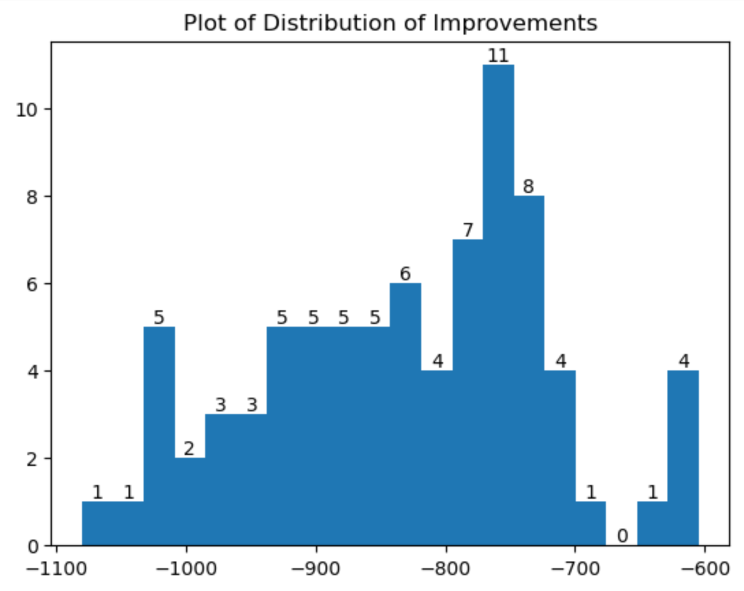

Rewards for complexity = 1.0
p=0
p=0.05
p=0.1
p=0.2
p=0.3
p=0.4
p=0.5
dqn
split
stack
0
5000
10000
15000
20000

Thanks!

Do you have any questions?

Results

- From the above histogram plot, generally, we can visualize what is the distribution of improvement that RF-QLOP Planning achieves over the single-action agent.
- Our model is able to reduce the race time with a median of 825 seconds across all types of conditions with a constraint of radius from 600m to 1200m.
- We can see that distribution is slightly skewed to the right and has the most occurrences of race time reduction by 750 seconds to 775 seconds.



	0
count	81.000000
mean	-830.525079
std	109.890237
min	-1080.216805
25%	-906.982293
50%	-825.331788
75%	-754.376902
max	-604.513900

Results

- From the breakdown in the above diagram, we can see that the model is generalizable across all types of weathers.
- Does better in more distinct weather types, such as Dry or 100% Wet conditions.
- As radius increases, performance drops.

	SingleAction	MonteCarloTreeSearch	Difference	PercentImprovement
('Intermediate', 1.0, '100% Wet', 600, 0.0, 'weather_config_44')	**-12181.70107**	**-11101.48426**	**-1080.216805**	**0.088675366**
('Intermediate', 1.0, '100% Wet', 700, 0.0, 'weather_config_50')	-13573.93895	-12533.68561	-1040.253336	0.07663607
('Intermediate', 1.0, '100% Wet', 600, 0.0, 'weather_config_13')	-12111.34518	-11093.73409	-1017.61109	0.08402131
('Intermediate', 1.0, '100% Wet', 650, 0.0, 'weather_config_14')	-12791.10971	-11821.10061	-970.0090938	0.075834632
('Intermediate', 1.0, '100% Wet', 800, 0.0, 'weather_config_56')	-14921.56224	-13959.23843	-962.3238062	0.064492162
('Intermediate', 1.0, '100% Wet', 700, 0.0, 'weather_config_15')	-13467.55908	-12560.57679	-906.982293	0.067345707
('Intermediate', 1.0, '100% Wet', 1200, 0.0, 'weather_config_80')	-20229.09034	-19323.11635	-905.9739865	0.044785701
('Intermediate', 1.0, '100% Wet', 750, 0.0, 'weather_config_16')	-14140.71714	-13260.57229	-880.1448436	0.062241882
('Intermediate', 1.0, '100% Wet', 800, 0.0, 'weather_config_17')	-14810.92508	-13972.82003	-838.1050497	0.056586948159152994
('Intermediate', 1.0, '100% Wet', 1000, 0.0, 'weather_config_68')	-17541.57526	-16704.07249	-837.5027672	0.047743874
('Intermediate', 1.0, '100% Wet', 1100, 0.0, 'weather_config_74')	-18888.84602	-18053.86485	-834.9811671	0.044204986
('Intermediate', 1.0, '100% Wet', 850, 0.0, 'weather_config_18')	-15478.01368	-14652.17548	-825.8381968	0.053355567
('Intermediate', 1.0, '100% Wet', 900, 0.0, 'weather_config_62')	-16148.22479	-15346.61489	-801.6099022	0.049640744575463636
('Intermediate', 1.0, '100% Wet', 900, 0.0, 'weather_config_19')	-16141.72935	-15357.99584	-783.7335151	0.048553256
('Intermediate', 1.0, '100% Wet', 950, 0.0, 'weather_config_20')	-16802.51768	-16048.14078	-754.3769017	0.044896659
('Intermediate', 1.0, '100% Wet', 1000, 0.0, 'weather_config_21')	-17460.25505	-16724.1903	-736.0647592	0.042156587
('Intermediate', 1.0, '100% Wet', 1150, 0.0, 'weather_config_24')	-19414.80542	-18685.32432	-729.4811026	0.037573443908452964
('Intermediate', 1.0, '100% Wet', 1050, 0.0, 'weather_config_22')	-18114.72181	-17389.26218	-725.4596251	0.040048069
('Intermediate', 1.0, '100% Wet', 1200, 0.0, 'weather_config_25')	-20060.025	-19358.91806	-701.1069385	0.034950451887044376
('Intermediate', 1.0, '100% Wet', 1100, 0.0, 'weather_config_23')	-18766.24181	-18072.14071	-694.1010966	0.036986686

	SingleAction	MonteCarloTreeSearch	Difference	PercentImprovement
('Intermediate', 1.0, 'Dry', 600, 0.0, 'weather_config_0')	-12111.82464	-11081.13392	-1030.690727	0.08509789
('Intermediate', 1.0, 'Dry', 650, 0.0, 'weather_config_1')	-12791.70289	-11804.24057	-987.462323	0.077195533
('Intermediate', 1.0, 'Dry', 600, 0.0, 'weather_config_39')	-12035.44162	-11106.1947	-929.2469193	0.077209208
('Intermediate', 1.0, 'Dry', 700, 0.0, 'weather_config_2')	-13468.3361	-12542.67413	-925.6619724	0.068728755
('Intermediate', 1.0, 'Dry', 750, 0.0, 'weather_config_3')	-14141.45017	-13218.93794	-922.5122269	0.065234627
('Intermediate', 1.0, 'Dry', 800, 0.0, 'weather_config_4')	-14811.45457	-13945.5981	-865.8564699	0.058458571
('Intermediate', 1.0, 'Dry', 900, 0.0, 'weather_config_6')	-16142.55432	-15280.63356	-861.9207587	0.053394323
('Intermediate', 1.0, 'Dry', 700, 0.0, 'weather_config_45')	-13367.42078	-12515.80405	-851.616727	0.06370838
('Intermediate', 1.0, 'Dry', 850, 0.0, 'weather_config_5')	-15478.67575	-14642.63439	-836.0413624	0.054012460482917524
('Intermediate', 1.0, 'Dry', 1050, 0.0, 'weather_config_9')	-18115.58551	-17331.44596	-784.1395492	0.043285355
('Intermediate', 1.0, 'Dry', 950, 0.0, 'weather_config_7')	-16803.48266	-16020.48126	-783.0013981	0.046597566
('Intermediate', 1.0, 'Dry', 1000, 0.0, 'weather_config_8')	-17461.21015	-16692.35996	-768.850184	0.044031895699468415
('Intermediate', 1.0, 'Dry', 1100, 0.0, 'weather_config_10')	-18767.31737	-17998.73456	-768.5828158	0.040953259
('Intermediate', 1.0, 'Dry', 1150, 0.0, 'weather_config_11')	-19416.08415	-18656.30914	-759.7750093	0.039131217
('Intermediate', 1.0, 'Dry', 1000, 0.0, 'weather_config_63')	-17410.11528	-16652.85277	-757.2625088	0.043495548
('Intermediate', 1.0, 'Dry', 900, 0.0, 'weather_config_57')	-16091.84516	-15337.70433	-754.1408296	0.046864783
('Intermediate', 1.0, 'Dry', 800, 0.0, 'weather_config_51')	-14700.85951	-13955.37905	-745.4804604	0.050709991
('Intermediate', 1.0, 'Dry', 1200, 0.0, 'weather_config_12')	-20061.49573	-19337.41551	-724.0802206	0.036093033
('Intermediate', 1.0, 'Dry', 1100, 0.0, 'weather_config_69')	-18679.3219	-17976.07795	-703.243952	0.037648259
('Intermediate', 1.0, 'Dry', 1200, 0.0, 'weather_config_75')	-19936.51838	-19310.91832	-625.6000518	0.031379604

Results

- In less distinct weather condition, we also see more variance in model performance.

	SingleAction	MonteCarloTreeSearch	Difference	PercentImprovement
('Intermediate', 1.0, '80% Wet', 600, 0.0, 'weather_config_43')	-12123.67056	-11108.41006	-1015.2605	0.083742007
('Intermediate', 1.0, '80% Wet', 700, 0.0, 'weather_config_49')	-13544.24084	-12538.01146	-1006.229385	0.074292048
('Intermediate', 1.0, '80% Wet', 800, 0.0, 'weather_config_55')	-14877.45966	-13963.15567	-914.3039822	0.061455651925579224
('Intermediate', 1.0, '80% Wet', 1000, 0.0, 'weather_config_67')	-17595.69755	-16721.67127	-874.0262847	0.049672727212092815
('Intermediate', 1.0, '80% Wet', 1100, 0.0, 'weather_config_73')	-18895.70054	-18025.80933	-869.8912034	0.046036462
('Intermediate', 1.0, '80% Wet', 900, 0.0, 'weather_config_61')	-16206.0648	-15351.62955	-854.4352447	0.052723178
('Intermediate', 1.0, '80% Wet', 1200, 0.0, 'weather_config_79')	-20053.98118	-19350.12756	-703.8536226	0.035097949686799246

	SingleAction	MonteCarloTreeSearch	Difference	PercentImprovement
('Intermediate', 1.0, '60% Wet', 600, 0.0, 'weather_config_26')	-12123.03779	-11105.95326	-1017.084526	0.083896837
('Intermediate', 1.0, '60% Wet', 600, 0.0, 'weather_config_42')	-12113.80497	-11103.53085	-1010.27412	0.083398579
('Intermediate', 1.0, '60% Wet', 650, 0.0, 'weather_config_27')	-12803.54087	-11827.80074	-975.7401353	0.076208616
('Intermediate', 1.0, '60% Wet', 700, 0.0, 'weather_config_28')	-13480.99624	-12533.33409	-947.6621482	0.070296151
('Intermediate', 1.0, '60% Wet', 750, 0.0, 'weather_config_29')	-14155.57649	-13264.80532	-890.7711675	0.062927226
('Intermediate', 1.0, '60% Wet', 800, 0.0, 'weather_config_30')	-14826.33822	-13950.64906	-875.689162	0.059063077
('Intermediate', 1.0, '60% Wet', 700, 0.0, 'weather_config_48')	-13410.97989	-12538.57715	-872.4027357	0.065051379
('Intermediate', 1.0, '60% Wet', 850, 0.0, 'weather_config_31')	-15494.26762	-14644.36488	-849.9027474	0.054852722
('Intermediate', 1.0, '60% Wet', 900, 0.0, 'weather_config_32')	-16159.09694	-15346.49972	-812.597219	0.050287291544594215
('Intermediate', 1.0, '60% Wet', 950, 0.0, 'weather_config_33')	-16820.88913	-16010.83571	-810.0534185	0.048157586
('Intermediate', 1.0, '60% Wet', 1100, 0.0, 'weather_config_72')	-18847.17141	-18051.03571	-796.1357076	0.042241655
('Intermediate', 1.0, '60% Wet', 900, 0.0, 'weather_config_60')	-16135.83992	-15343.05393	-792.7859882	0.049131994
('Intermediate', 1.0, '60% Wet', 1050, 0.0, 'weather_config_35')	-18135.30797	-17345.76656	-789.541411	0.043536145741840726
('Intermediate', 1.0, '60% Wet', 1000, 0.0, 'weather_config_34')	-17479.77344	-16692.4442	-787.3292459	0.045042302661896905
('Intermediate', 1.0, '60% Wet', 800, 0.0, 'weather_config_54')	-14752.83715	-13974.51188	-778.3252649	0.052757667
('Intermediate', 1.0, '60% Wet', 1000, 0.0, 'weather_config_66')	-17465.48089	-16701.26239	-764.2185009	0.043755938
('Intermediate', 1.0, '60% Wet', 1100, 0.0, 'weather_config_36')	-18787.09137	-18032.50019	-754.5911804	0.040165407
('Intermediate', 1.0, '60% Wet', 1150, 0.0, 'weather_config_37')	-19436.7552	-18691.16679	-745.5884132	0.038359716
('Intermediate', 1.0, '60% Wet', 1200, 0.0, 'weather_config_38')	-20083.17036	-19372.92745	-710.2429122	0.035365079287561385
('Intermediate', 1.0, '60% Wet', 1200, 0.0, 'weather_config_78')	-19963.75003	-19316.94457	-646.8054599	0.032398996

	SingleAction	MonteCarloTreeSearch	Difference	PercentImprovement
('Intermediate', 1.0, '40% Wet', 700, 0.0, 'weather_config_47')	-13480.71363	-12541.97449	-938.7391418	0.069635716
('Intermediate', 1.0, '40% Wet', 600, 0.0, 'weather_config_41')	-12029.02838	-11094.94758	-934.0808056	0.077652224
('Intermediate', 1.0, '40% Wet', 900, 0.0, 'weather_config_59')	-16228.24994	-15324.70048	-903.5494627	0.055677566330040174
('Intermediate', 1.0, '40% Wet', 800, 0.0, 'weather_config_53')	-14778.96166	-13953.62987	-825.3317884	0.055845045645072644
('Intermediate', 1.0, '40% Wet', 1000, 0.0, 'weather_config_65')	-17434.87112	-16686.90473	-747.9663878	0.042900597474922475
('Intermediate', 1.0, '40% Wet', 1200, 0.0, 'weather_config_77')	-20058.02135	-19330.06318	-727.958164	0.036292620862677816
('Intermediate', 1.0, '40% Wet', 1100, 0.0, 'weather_config_71')	-18644.31882	-18039.80492	-604.5138997	0.032423491