Improving choice model specification using reinforcement learning

Gabriel Nova*[1], Sander van Cranenburgh[1], Stephane Hess[2]

[1]Transport and Logistics group, Delft University of Technology

[2] Choice Modelling Centre - Institute for Transport Studies, University of Leeds

What is Discrete Choice Modelling (DCM)?

People make choices every day and across dimensions.

- Transport mode choice (e.g., car, bicycle, public transport), destination choice (e.g., tourist destinations, workplaces), route choice, etc.

Choice modellers use choices

- to understand the factors that lead people to choose alternatives.
- to analyse policy and forecast demand.

Discrete choice modelling as an art

- *Requires specifying utility functions* by selecting a combination of variables, transformations, and behavioural assumptions that capture decision-making behaviours [1].

[1] van Cranenburgh et al., (2022)

Model specification problem

Motivation

Modellers must define a specification by making several interrelated decisions[1]:

1. Select attributes: Which variables influence choice? (e.g., time, cost)
2. Allow alternative-specific taste parameters: Generic or alternative-specific?
3. Try to accommodate for non-linearities and interactions
4. Estimate and evaluate

$$i = 1 : Bus \ \rightarrow \ V_1 = \beta_1 \log(x_{11}) + \beta_{12\_female}(sex == 1)x_{12} + \cdots + \beta_K x_{1K} + \varepsilon_1$$

$$i = 2 : Metro \ \rightarrow V_2 = \beta_1 \log(x_{21}) + \quad \beta_3 x_{23} + \cdots + \beta_K x_{2K} + \varepsilon_2$$

$$i = 3: Car \rightarrow \mathrm{V}_3 = \beta_1 \log(x_{31}) + \beta_{32\_female}(sex == 1)x_{32} + \cdots + \beta_K x_{3K} + \varepsilon_3$$

Metaheuristics: traditional assistance

Combinatorial, optimization-based, and hypothesis-driven metaheuristics

1. Simulated Annealing
2. Automatic relevance determination through Bayesian inference
3. Variant Neighborhood search
4. Bi-level optimization framework that integrates prior constraints
5. Grammatical Evolution
6. Bi-level optimization framework integrating GE and singular value decomposition

1. Paz et al., (2019); 2. Rodriguez et al., (2020); 3. Ortelli et al., (2021); 4. Beeramoole et al., (2023); 5. Yahai et al., (2024); 6. Ghorbani et al., (2025)

Research gap

Motivation

Model specification is not a static task -> it's a learning process!!!

Metaheuristics

- automate part of the process
- *static, lack memory, poor knowledge transfer*
- *fail to capture learning-driven nature of the modelling process*

→ Reinforcement learning is a promising alternative to automate the model specification search process

How can we leverage RL algorithms to automate the model specification search process?

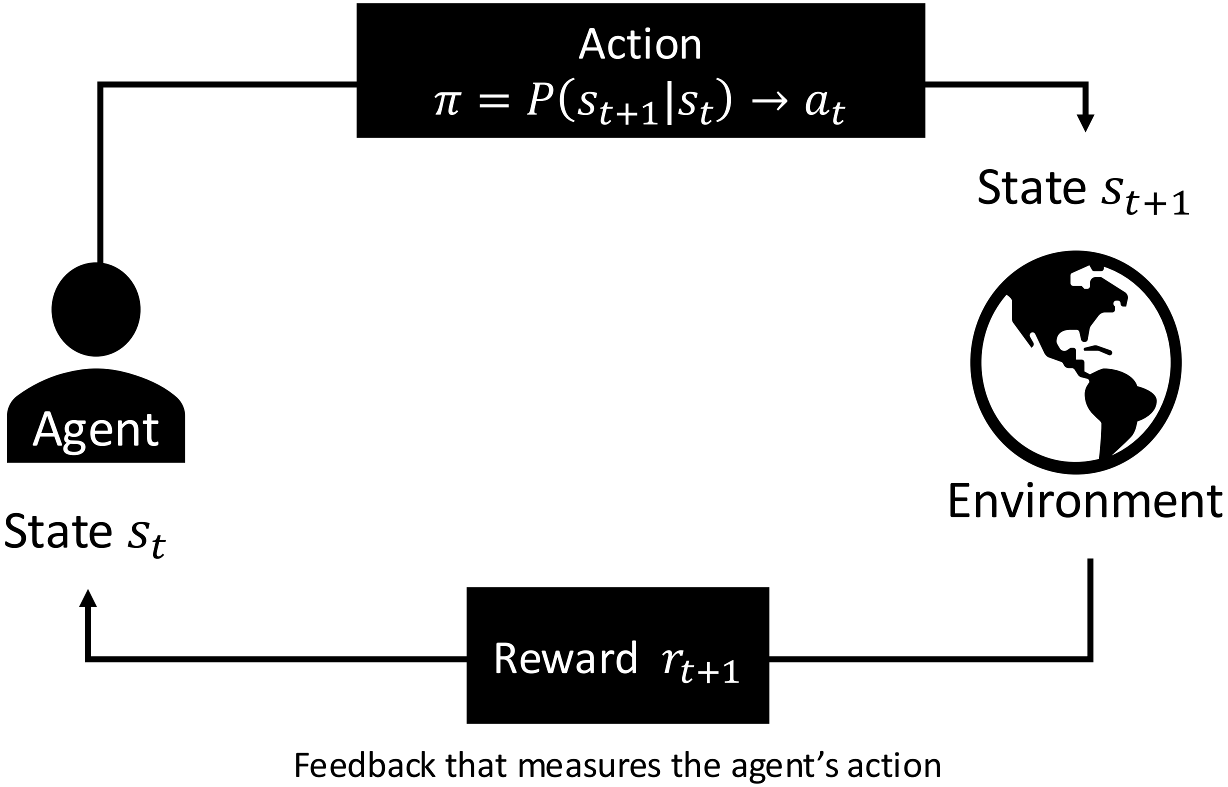
↳ How to frame the MS process as an adaptative-learning process

↳ How to include modelling outcomes as part of the reward function

Reinforcement learning paradigm

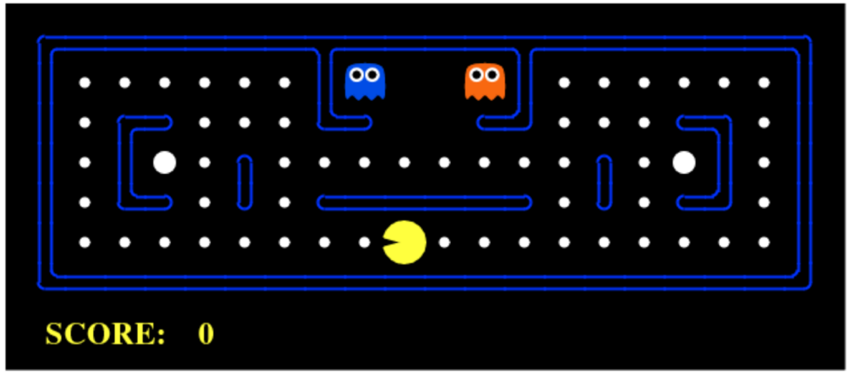
Reinforcement Learning

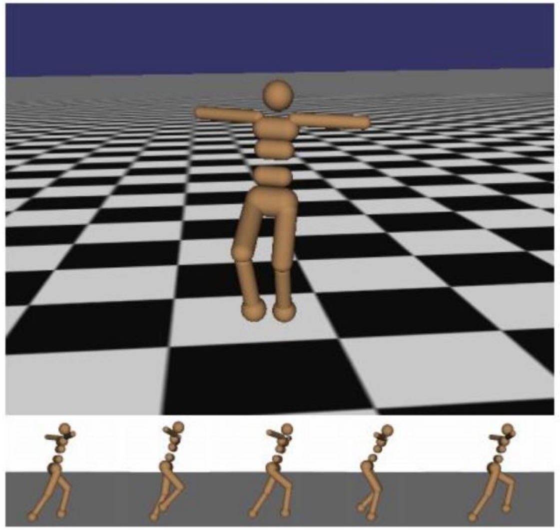
- Goal: Learn how to take actions (π) to maximize total discounted rewards (R)
- Data: Obtained by interacting with an environment (state, action, reward, next state)
- Markov decision process defined by $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}, \boldsymbol{\pi}, \boldsymbol{\gamma})$

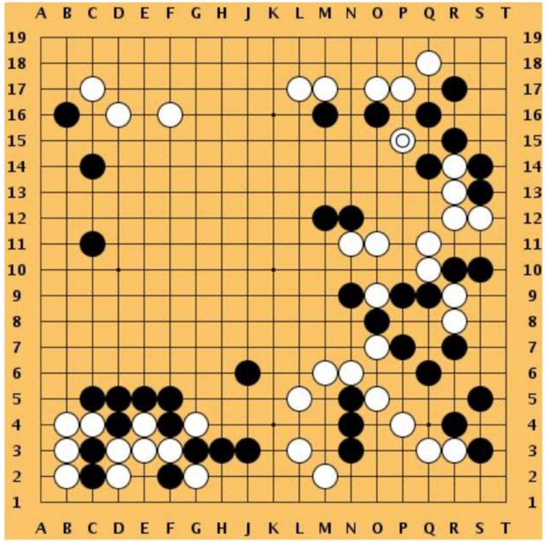


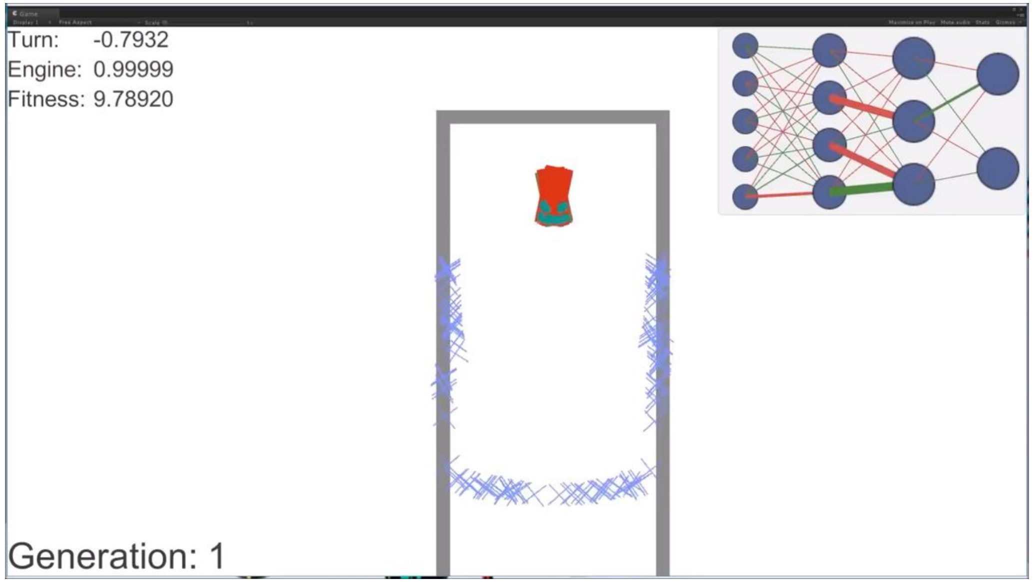
Agent-environment iteration. Adapted from Sutton and Barto (1999)

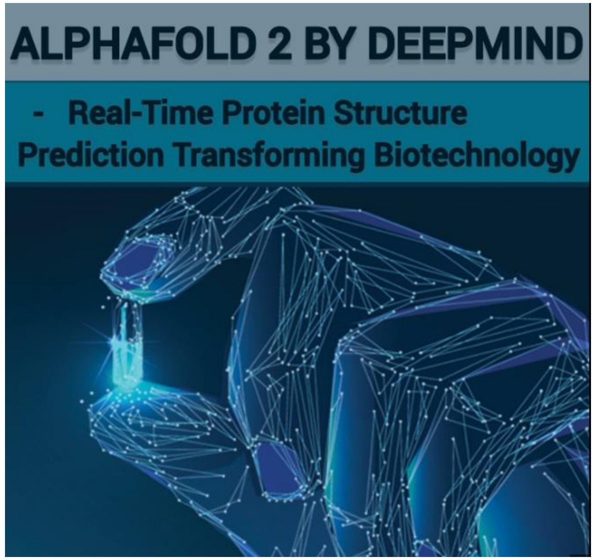
Examples


SCORE: 0




A B C D E F G H J K L M N O P Q R S T
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1


Turn: -0.7932
Engine: 0.99999
Fitness: 9.78920
Generation: 1


ALPHAFOLD 2 BY DEEPMIND
- Real-Time Protein Structure Prediction Transforming Biotechnology

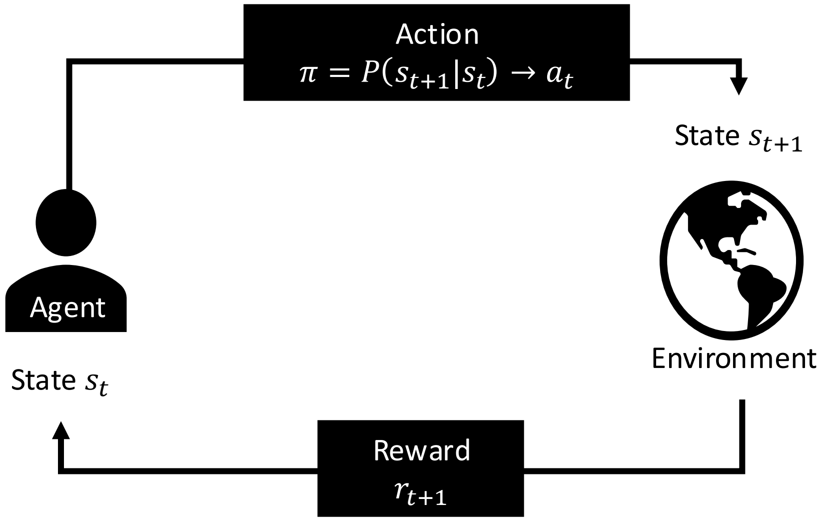
Reinforcement Learning

- Goal: Learn how to take actions (π) to maximize total discounted rewards (R)
- Data: Obtained by interacting with an environment (state, action, reward, next state)
- The agent's policy infers the best action to take at its state

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, \mathrm{Q}(s_t, a_t)$$

- Q-value captures the expected total discounted future reward

$$\mathrm{Q}(s_t, a_t) = \mathrm{E}[R_t | s_t, a_t] \text{ and } R_t = \sum_{i=t}^{\infty} \gamma^i r_i$$



How to learn the optimal policy π^*?

Reinforcement Learning algorithms

To train an RL agent, we require:

1. an optimisation algorithm

- Value-based: learn Q-values

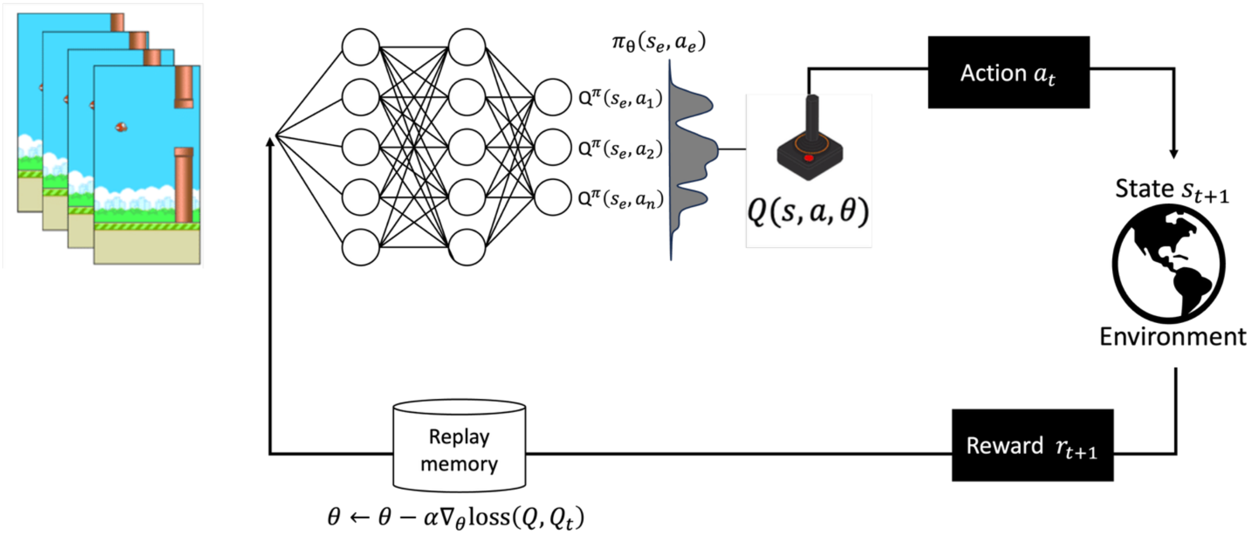
$$a^* = \underset{a}{\operatorname{argmax}}\, Q(s_t, a_t)$$

- Policy-based: learn policy directly

$$a^* \sim \pi(s)$$

2. a neural network as a function approximator (Q-values)

3. a loss function to update network parameters



Deep Q-Network architecture

State space

- Any model specification (state) is represented as lists of tuples, each of them representing a component of the model

$$S_e \quad = [(\underbrace{0}_{\text{Attribute index}}, \underbrace{linear}_{\text{Transformation}}, \underbrace{0}_{\text{Generic – Alt. specific taste parameters}}, \underbrace{2}_{\text{Covariate index}}), (1, linear, 1, 1), (2, box-cox, 0, 0), (6, log, 1, 0)]$$

- Each tuples is encoded and decoded as one-hot vectors for NN processing.

Action space

- Defines all feasible operations that the agent can apply to any encoded model specification.

  → *Add* new variables (generic-linear additive)
  → *Change* any tuple component
  → *End* model specification process

- Masks invalid operations based on the current specification

$$S_e^0 = [\quad] \rightarrow (add, 1, \text{linear}, 0, 0)$$

$$S_e^1 = [(1, \text{linear}, 0, 0)] \rightarrow (change, 1, \text{linear}, 1, 0)$$

$$S_e^2 = [(1, \text{linear}, 1, 0)] \cancel{\rightarrow (add, 1, \text{linear}, 0, 0)}\ (add, 3, \text{linear}, 0, 0)$$

$$S_e^3 = [(1, \text{linear}, 1, 0), (3, \text{linear}, 0, 0)] \rightarrow (change, 3, \log, 0, 0)$$

$$S_e^4 = [(1, \text{linear}, 1, 0), (3, \log, 0, 0)] \rightarrow (end)$$

Reward function

Reinforcement Learning

- Like human modellers, the agent receives feedback only after model estimation. Thus, the episodic final reward is distributed across all the actions taken during the episode:

$$R_e^l = R_e \cdot \gamma^{L-l}, \qquad R_e \equiv \widetilde{M_m} = \frac{M_{\max_e} - M_m}{M_{\max_e} - M_{\min_e}}$$

Where $l = 1, \ldots, L$ number of actions at episode e

- What if there are multiple modelling outcomes?

$$\mathrm{R}_e = \left[\sum_m^M \omega_m \widetilde{M_m}\right] \cdot \mathrm{I}_{\text{converged}}$$

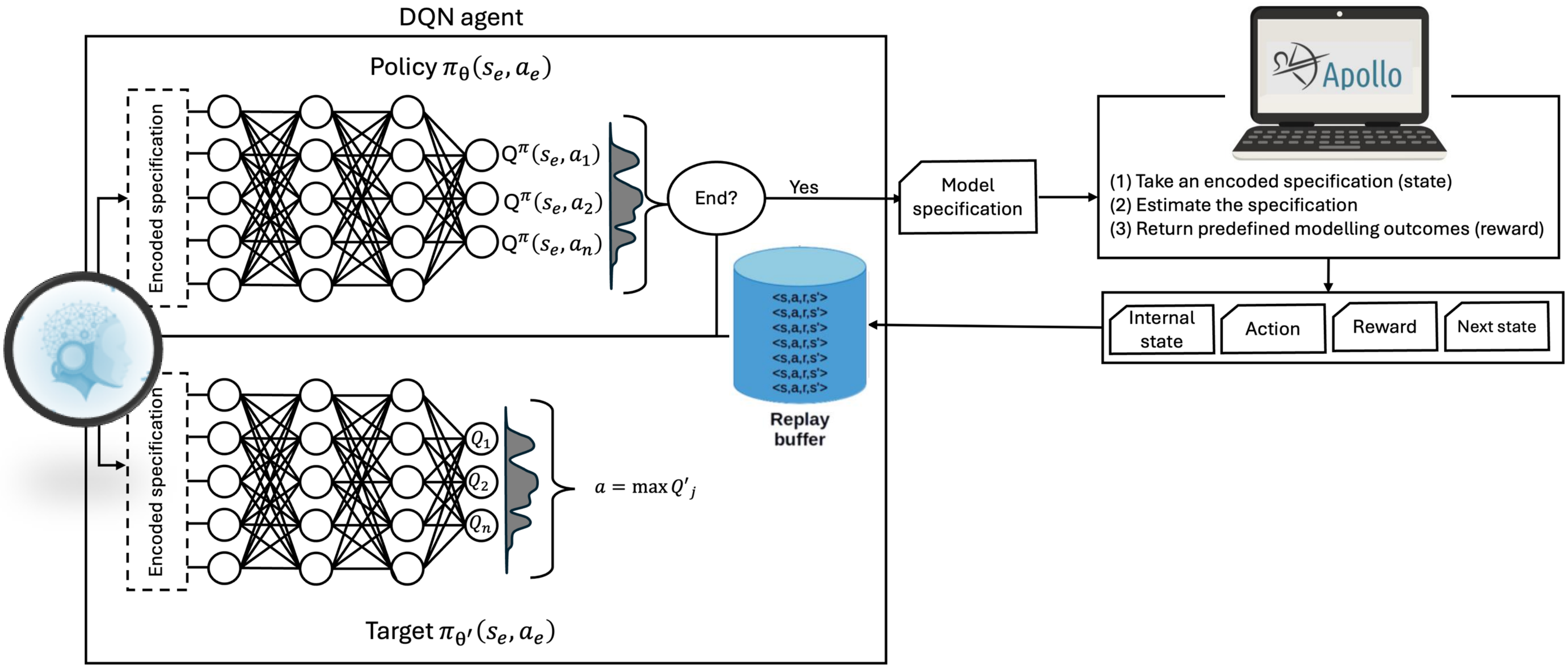
- How to incorporate behavioural expectations?

$$\mathrm{R}_e = \left[\sum_m^M \omega_m \widetilde{M_m}\right] \cdot \mathrm{I}_{\text{converged}} \cdot \mathrm{I}_{\text{behavioural expectation}}$$

Delphos: A DQN agent that automate the utility specification process

Delphos framework overview

DQN - ENV



Hess et al., (2019). Apollo: A flexible, powerful and customisable freeware package for choice model estimation and application.

Experimental cases

Simulated experiment

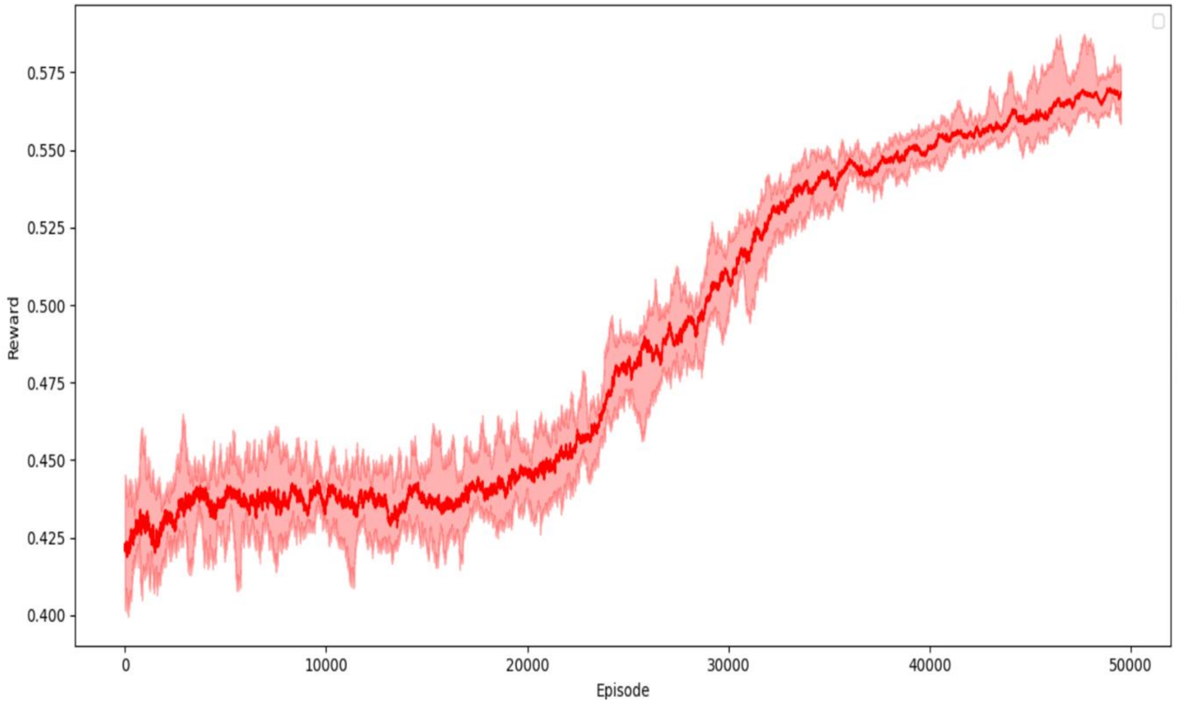
$$\text{modelling space} = \begin{cases} \text{variables} = [\text{ASC}, \text{X}_1, \text{X}_2, \text{X}_3, \text{X}_4, \text{X}_5, \text{X}_6]. \\ \text{transformations} = [\text{linear}, \text{logarithm}, \text{box} - \text{cox}] \\ \text{taste} = [\text{generic}, \text{specific}] \\ \text{covariates} = [\quad] \end{cases}$$

$$\text{agent} = \begin{cases} r\text{eward weights} = [\text{AIC: 1}] \\ \text{episodes} = [50000] \end{cases}$$

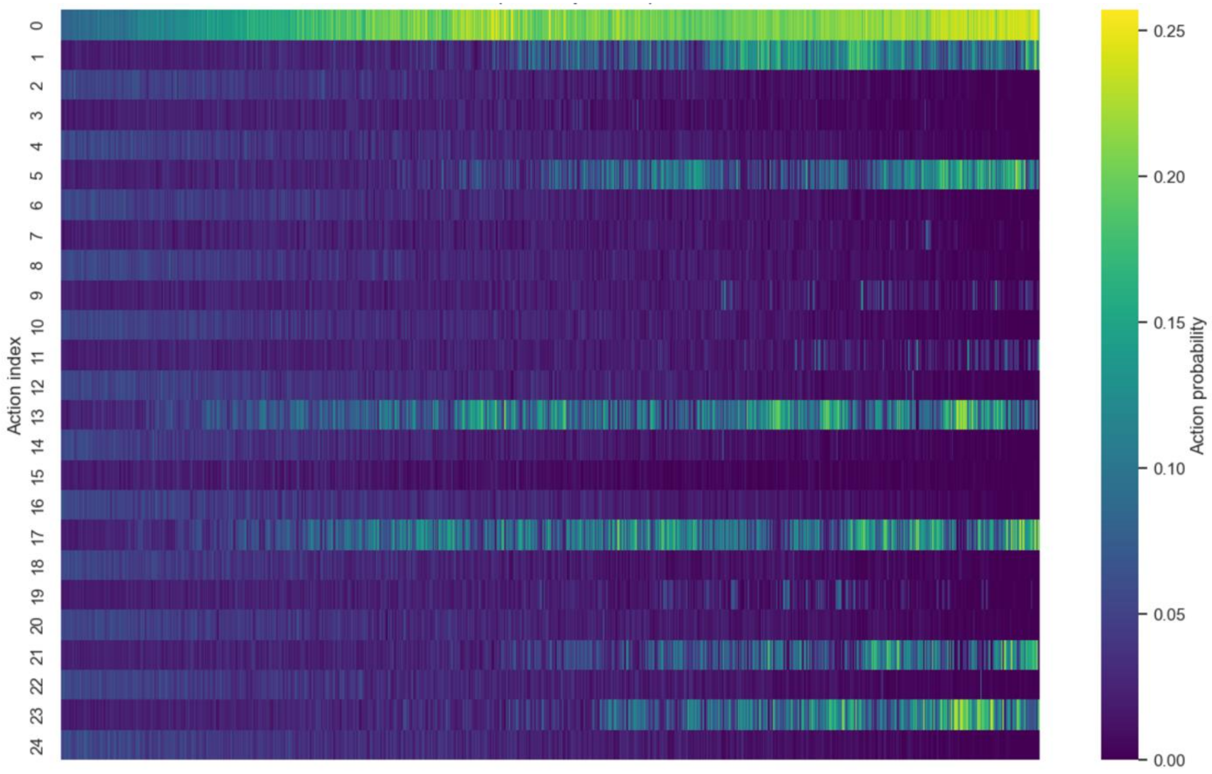
Results

Experiments

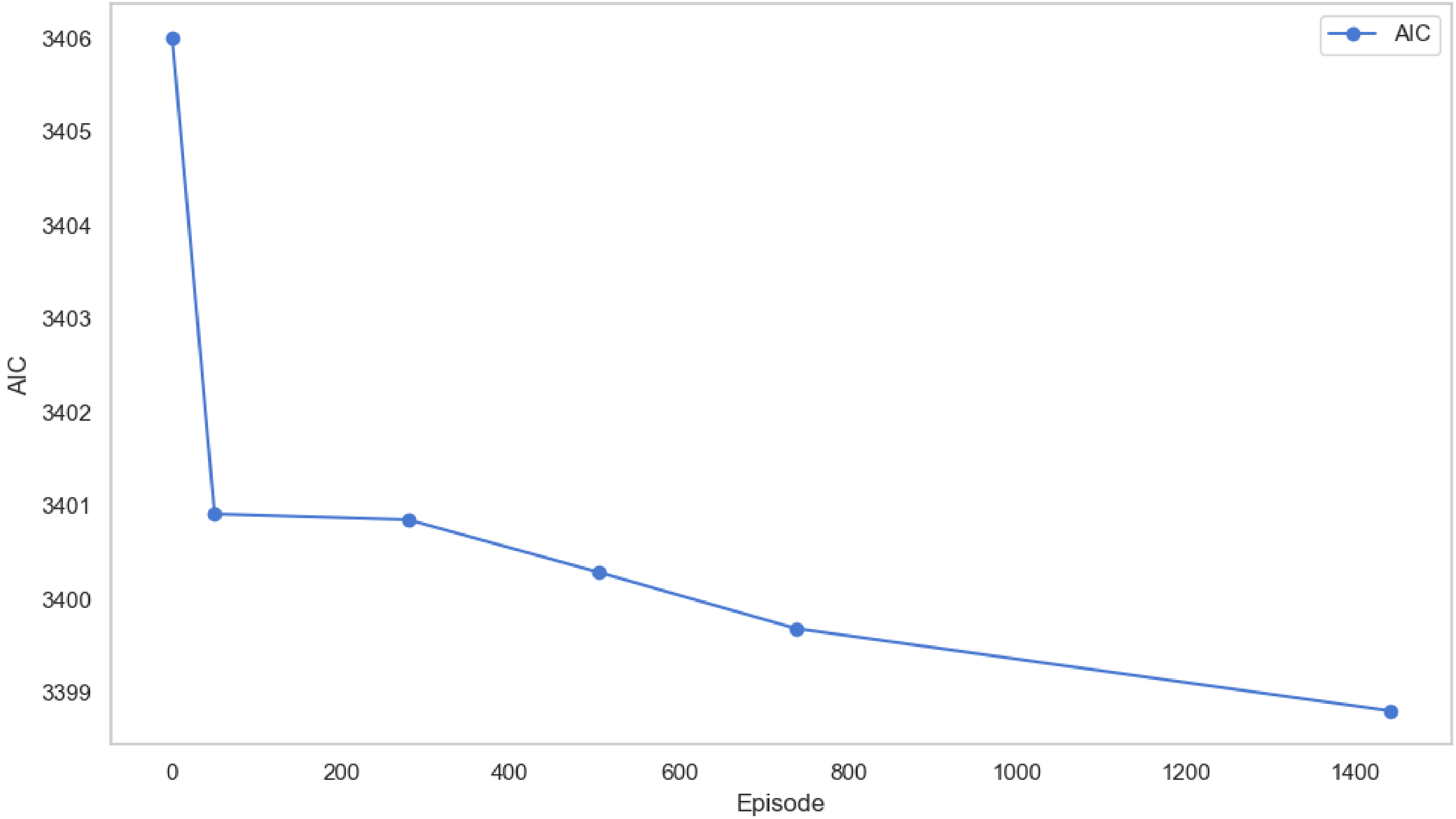
Learning curve (10 runs)



Q-value distribution



Results



[INFO] New best candidate for AIC at 0: 3406.01 (000_300_300_100_300_300_100)
[INFO] New best candidate for AIC at 50: 3400.92 (000_100_200_300_300_100_100)
[INFO] New best candidate for AIC at 279: 3400.85 (000_100_200_300_300_100_200)
[INFO] New best candidate for AIC at 505: 3400.29 (000_100_300_100_300_000_000)
[INFO] New best candidate for AIC at 738: 3399.69 (000_100_200_100_300_200_000)
[INFO] New best candidate for AIC at 1442: 3398.81 (000_100_200_100_300_100_000)

Empirical experiments

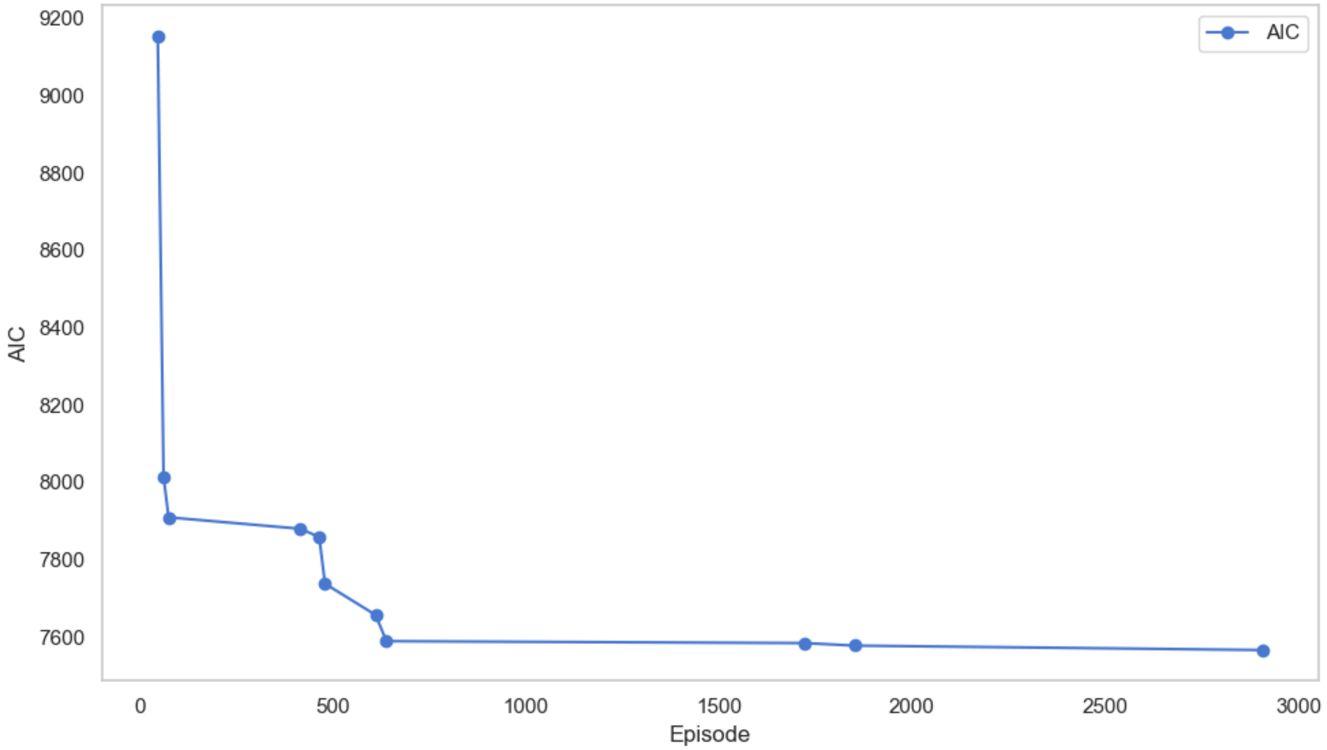
a. Swissmetro (Bierlaire et al., 2001)

$$\text{modelling space} = \begin{cases} \text{variables} = [\text{ASC}, \text{TT}, \text{TC}, \text{HE}, \text{SE}] \\ \text{transformations} = [\text{linear}, \text{logarithm}, \text{box} - \text{cox}] \\ \text{taste} = [\text{generic}, \text{specific}] \\ \text{covariates} = [\text{age, income, class, ga, gender}] \end{cases}$$

$$\text{agent} = \begin{cases} r\text{eward weights} = [\text{AIC: 1}] \\ \text{episodes} = [50000] \\ \text{early stopping} = [0.05] \end{cases}$$

ASC: Alternative-specific constant; TT: Travel time; TC: travel cost HE: Headway SE: Seat conf.

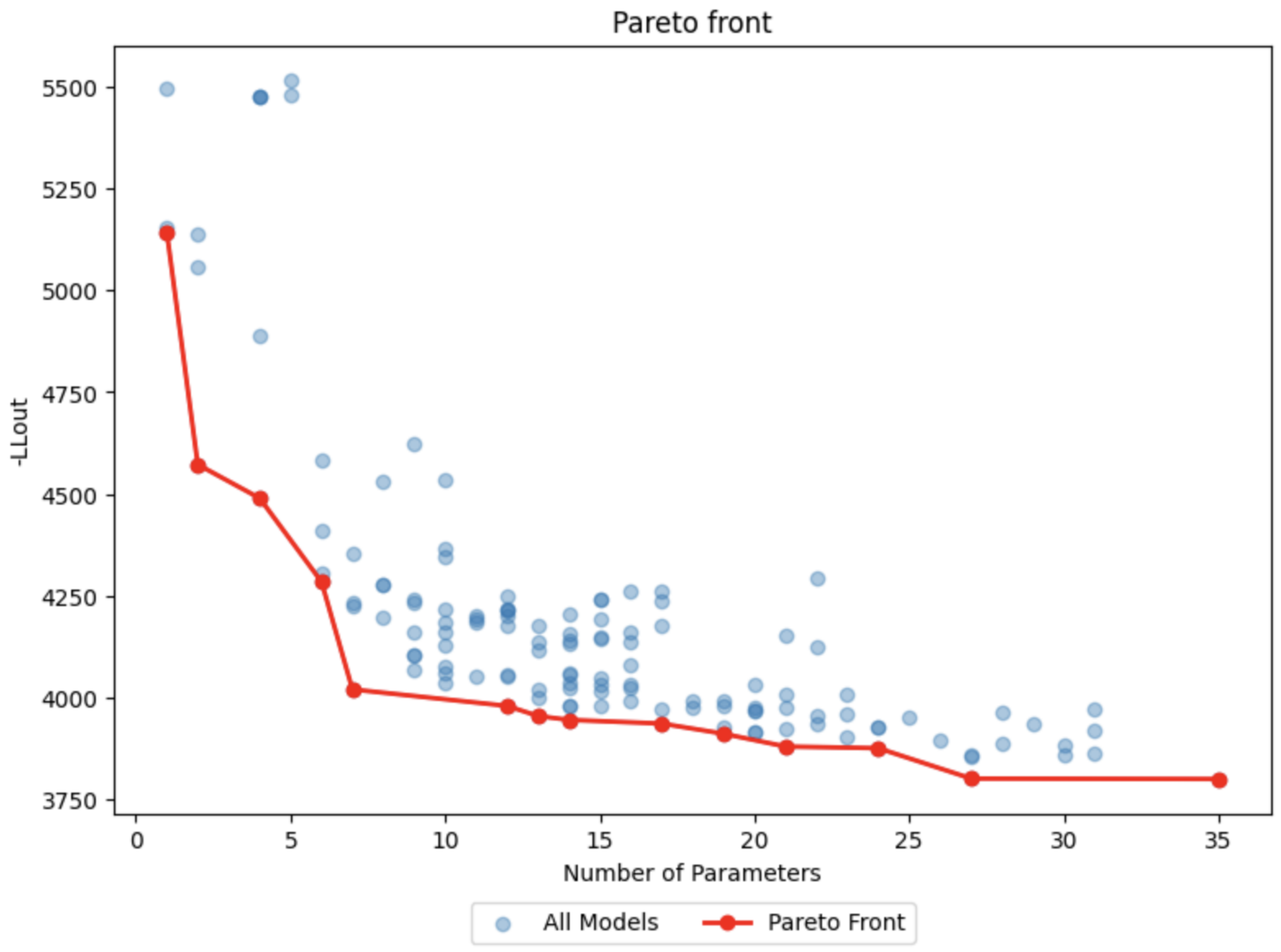
Results



2025-05-21 02:59:42,812 [INFO] Training started at Wed May 21 02:59:42 2025
2025-05-21 02:59:44,174 [INFO] New best candidate for AIC at episode 47: 9153.1186 (000_200_000_000_000)
2025-05-21 02:59:44,963 [INFO] New best candidate for AIC at episode 62: 8014.5405 (100_110_101_111_101)
2025-05-21 02:59:45,232 [INFO] New best candidate for AIC at episode 75: 7909.2325 (000_101_113_113_100)
2025-05-21 03:00:36,634 [INFO] New best candidate for AIC at episode 416: 7879.7047 (000_210_111_100_111)
2025-05-21 03:00:47,277 [INFO] New best candidate for AIC at episode 465: 7858.4485 (100_101_113_110_111)
2025-05-21 03:00:50,213 [INFO] New best candidate for AIC at episode 480: 7737.6426 (000_200_111_111_100)
2025-05-21 03:01:18,928 [INFO] New best candidate for AIC at episode 611: 7656.8514 (100_101_112_101_103)
2025-05-21 03:01:25,465 [INFO] New best candidate for AIC at episode 639: 7589.1305 (100_111_112_101_112)
2025-05-21 03:08:02,441 [INFO] New best candidate for AIC at episode 1720: 7584.1841 (000_111_112_111_110)
2025-05-21 03:08:59,581 [INFO] New best candidate for AIC at episode 1851: 7577.6789 (100_111_112_111_112)
2025-05-21 03:18:20,737 [INFO] New best candidate for AIC at episode 2908: 7566.1644 (100_300_112_111_112)

Results

Experiments



Empirical experiments

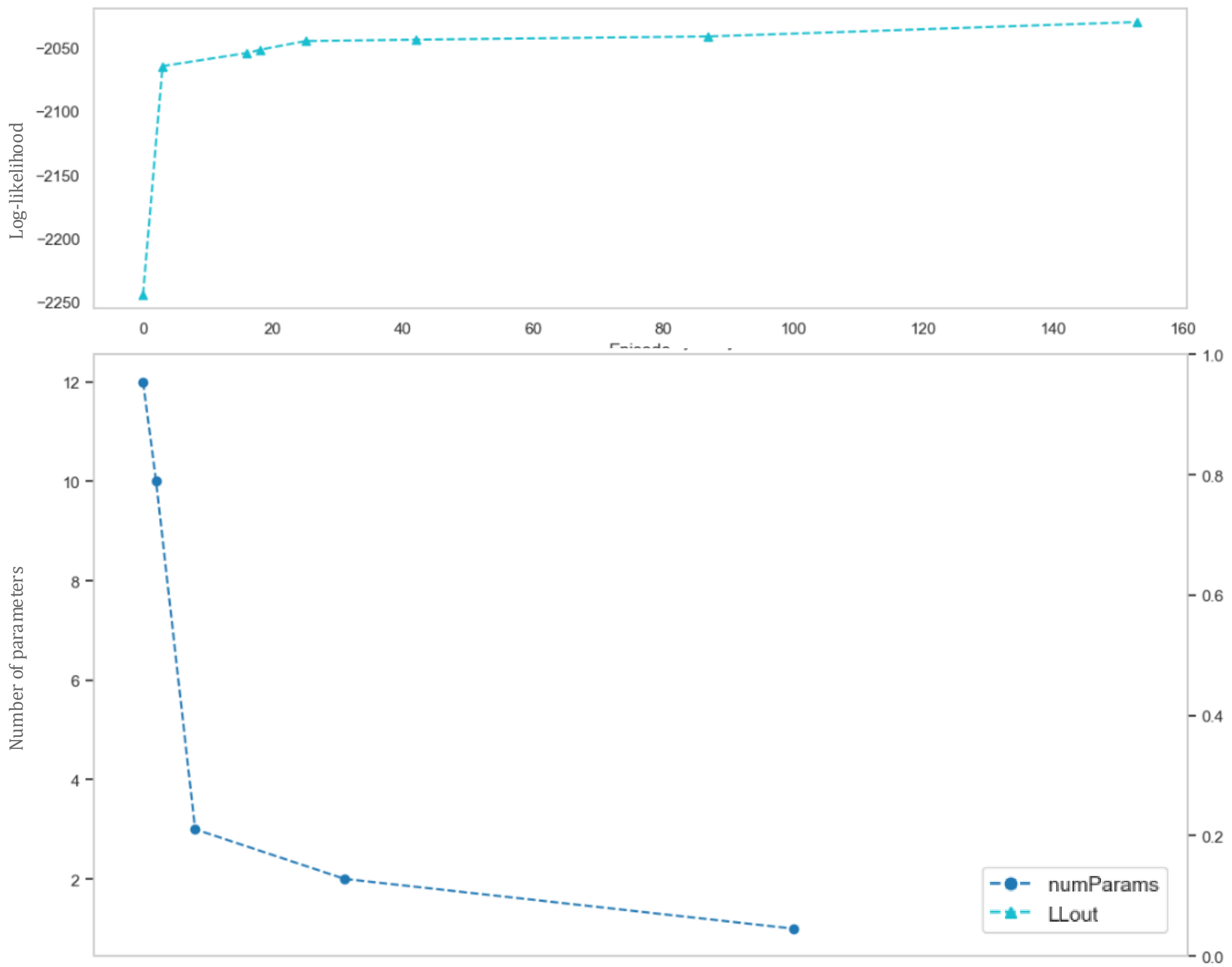
b.1. Decisions (Calastri et al., 2020)

$$\text{modelling space} = \begin{cases} \text{variables} = [\text{ASC}, \text{ITT}, \text{OTT}, \text{TTC}\,] \\ \text{transformations} = [\text{linear}, \text{logarithm}, \text{box} - \text{cox}] \\ \text{taste} = [\text{generic}, \text{specific}] \\ \text{covariates} = [\quad] \end{cases}$$

$$\text{agent} = \begin{cases} r\text{eward weights} = [\text{LL: 0.7, Params:0.3}] \\ \text{episodes} = [50000] \\ \text{early stopping} = [0.01] \\ \text{b}_{\text{expectations}} = [1: " - ", 3: " - "] \end{cases}$$

Results

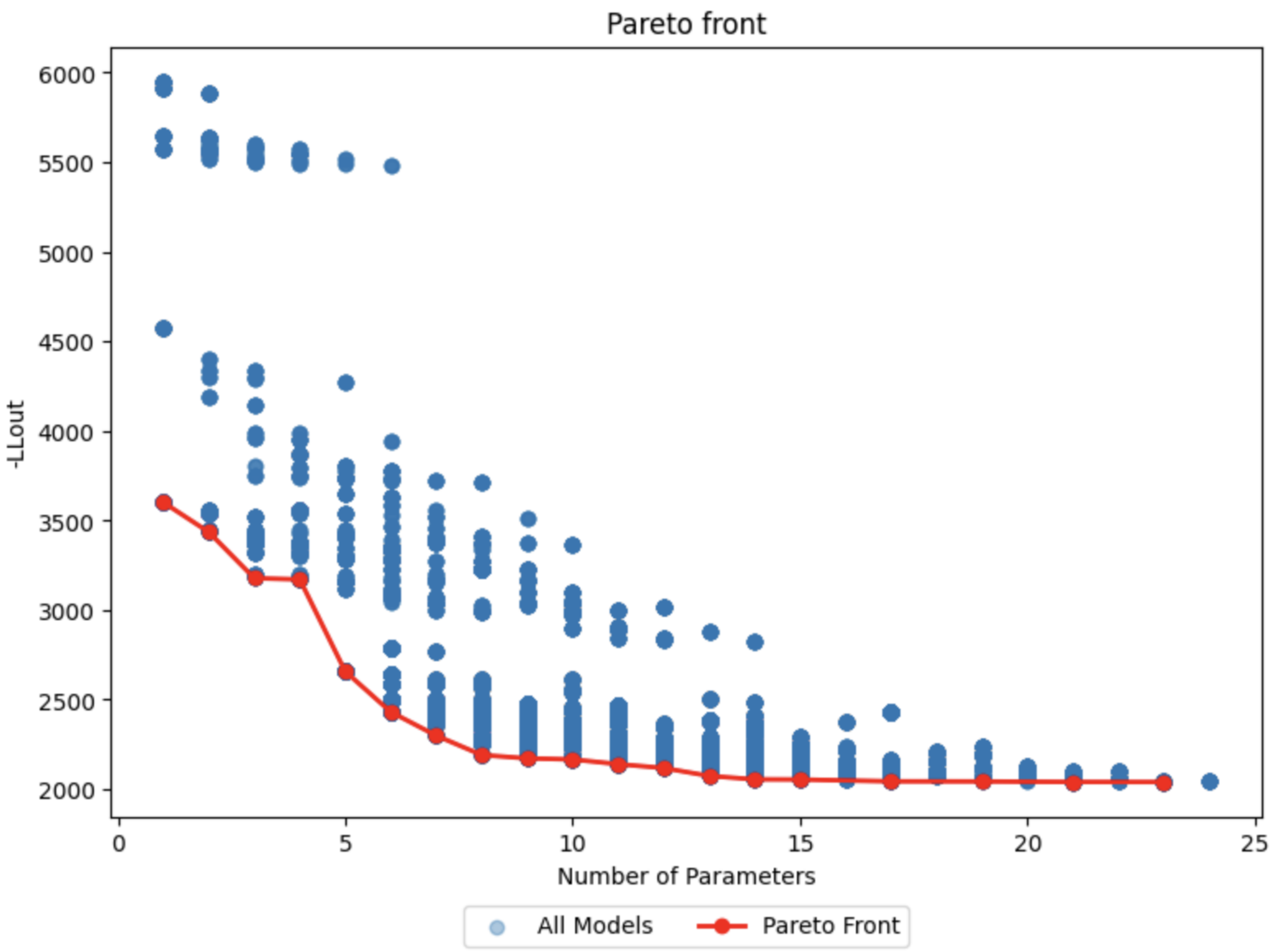
Experiments



Training time : 15 min
Unique models: 686

Results

Experiments



Training time : 15 min
Unique models: 686

Limitations and future research

Future research

The agent learns to take actions; though, they are randomly sampled from the memory buffer
→ prioritise them

Further refinement of reward function is possible

- behavioural realism
- Significance of parameters
- Willingness to pay

How to incorporate choice data knowledge to transfer knowledge?

How can multiple model family–specialised agents (MNL, LC, MXL) collaborate by sharing specifications and reward signals?

Comments, suggestions, questions?

Gabriel Nova*[1], Sander van Cranenburgh[1], Stephane Hess[2]

[1]CityAILab, Transport and Logistics group, Delft University of Technology

[2] Choice Modelling Centre - Institute for Transport Studies, University of Leeds