Smart TASEP

Deep Q-Learning in Intelligent Matter Simulations

Jonas Märtens

February 10, 2024

LMU Munich

Introduction

Smart TASEP: Deep Q-Learning in Intelligent Matter Simulations

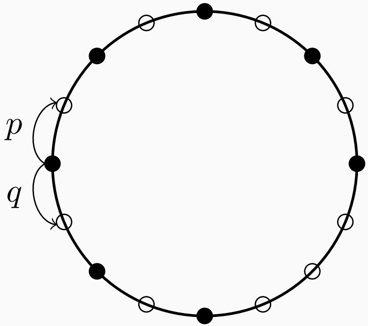
- TASEP: Totally Asymmetric Simple Exclusion Process
- Smart TASEP: TASEP with intelligent agents
  $\rightarrow$ Intelligent Matter Simulations
- Deep Q-Learning: Reinforcement Learning with Neural Networks

TASEP: Totally Asymmetric Simple Exclusion Process

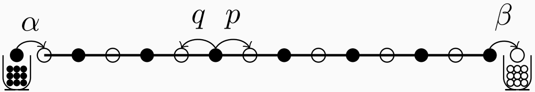
- Stochastic Process on a grid lattice
- Exclusion: Only one particle per site
- Simple: Jumps only one cell far
- Asymmetric: Only to the right

Possible with either

- Periodic Boundary Conditions
- Insertion and Extraction rates at the boundaries



ASEP with periodic boundary conditions



ASEP with insertion and extraction rates

Modifications

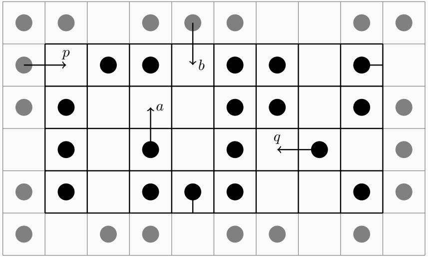
Two modifications (before smart)

(T)ASEP

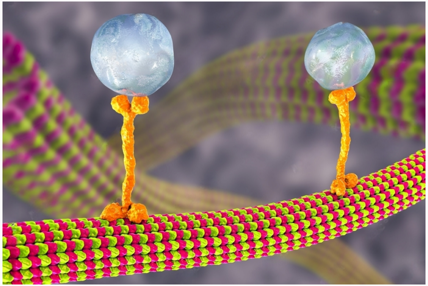
- Symmetric in one direction and totally asymmetric in the other
- Directed flow with multiple lanes
- compare traffic flow, kinesin transport
- motivates transport optimization

Speeds

- Each particle has a speed
- Speed is probability to jump
- Drawn from a distribution
- If different, jams



2D ASEP

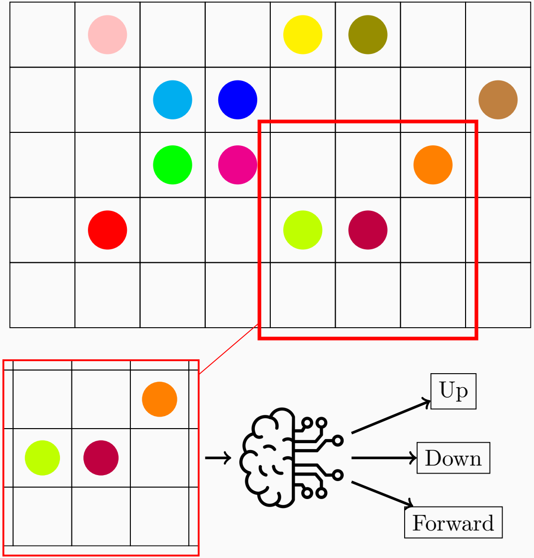

Intracellular transport

Make it Smart!

Smart TASEP: TASEP with intelligent agents

- have a goal (e.g. go forward)
- sense the environment
- act according to the goal and the environment
- learn from the environment themselves

→ Intelligent Matter Simulation
→ Reinforcement Learning - reward based

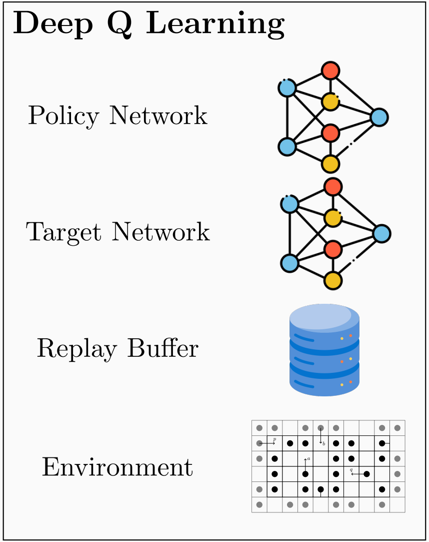


Deep Q-Learning

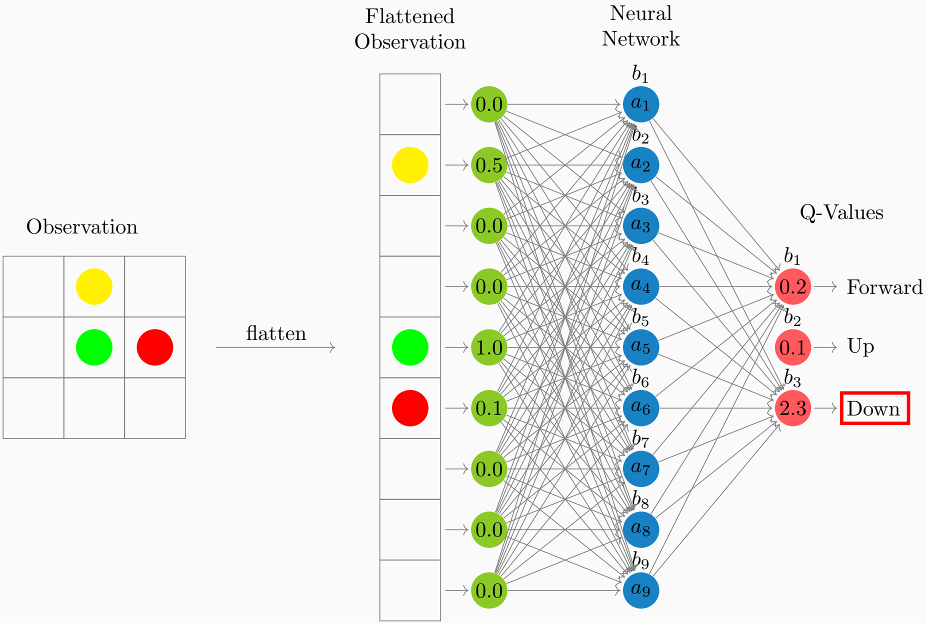
Deep Q-Learning: Overview

Deep Q-Learning Algorithm

- Reinforcement learning with deep neural networks
- Model-free: No knowledge of the environment
- Q-Learning: Value-based method
- Off-policy: Learns from old experiences

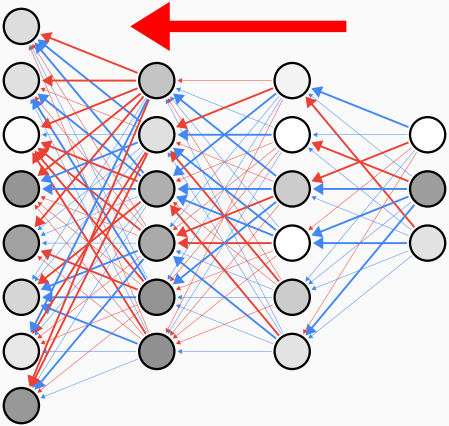


Deep Q-Learning: The policy



How does the network learn?

- Neural Network $\rightarrow$ backpropagation on loss
- Problem: no target value for Q-Learning
- Solution: Use current best guess as target: $y = r + \gamma \max_{a'} \hat{Q}(s', a')$
- Target network with soft updates to stabilize learning
- Use gradients for AdamW optimizer
- Replay buffer
  - Sample efficiency
  - Break correlation



Implementation

The `smarttasep` package

Features

- Easy installation with `pip`
- Fully Customizable
  - Environment size
  - Reward function
  - Network architecture
  - Hyperparameters
  - Training parameters
  - Algorithm choices
- Real-time visualization
- Interactive evaluation
- Management of experiments

`smarttasep` components

- `smarttasep.GridEnv`: The 2D (T)ASEP environment
- `smarttasep.DQN`: The neural network
- `torchrl.ReplayBuffer`: Efficient, tensor-based replay buffer
- `smarttasep.Trainer`: Wrapper class for training, simulation and experiment management
- `smarttasep.Playground`: Real-time interactive evaluation of trained agents
- `smarttasep.EnvParams`, `smarttasep.Hyperparams`: Configuration classes

The `smarttasep` package: Examples

Let's have a look at some examples! (show usage videos)

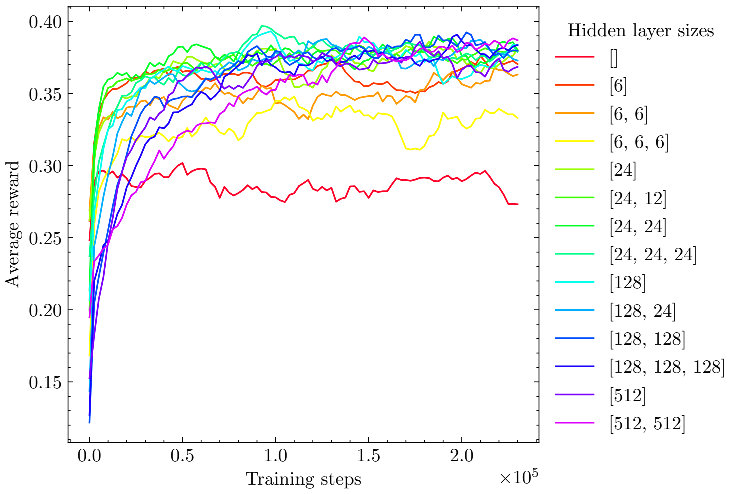
Hyperparameter Optimization

For each experiment, the Hyperparameters should be optimized.

Parameters

- **Learning rate** of the AdamW optimizer.
- **Discount factor** γ.
- **Replay buffer size**.
- **Batch size** for training.
- **Target network update rate** τ.
- **Exploration-exploitation tradeoff** via the decay constant η of the exploration rate $\epsilon(t) = \epsilon_{\text{end}} + (\epsilon_{\text{start}} - \epsilon_{\text{end}})e^{-n_{\text{steps}}/\eta}$.
- **Neural network architecture** (number of hidden layers and neurons per layer).
- **Activation function** of the neural network.

Hyperparameter Optimization: Example



Example of a hyperparameter optimization for the hidden layer sizes.

Results - Baseline

Overview of Experiments

Goals

- Current optimization
- General insights into intelligent matter simulations

Experiments

1. **Baseline**: Classical 2D TASEP with speeds
2. **Naive current optimization**: Hard-coded agents
3. **Smart TASEP**: Deep Q-Learning agents with slightly different goals
    - **Simple reward**: Go forward
    - **Complex reward**: Go forward while forming lanes

Results: Baseline

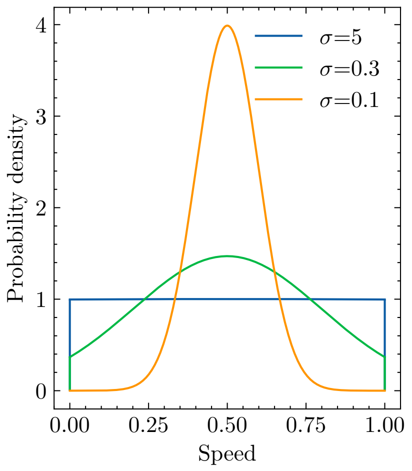
Setup

- Classical 2D TASEP
- Periodic boundary conditions
- Checkerboard configuration

Results: Baseline

Setup

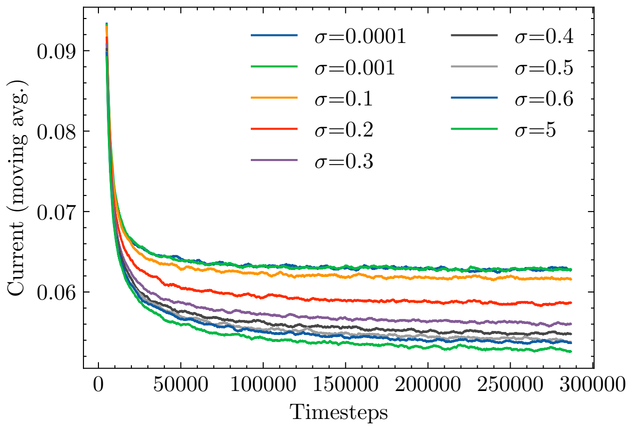
- Classical 2D TASEP
- Periodic boundary conditions
- Checkerboard configuration
- Normally distributed speeds



Speeds of particles in the baseline experiment.

Setup

- Classical 2D TASEP
- Periodic boundary conditions
- Checkerboard configuration
- Normally distributed speeds
- Waiting for steady state

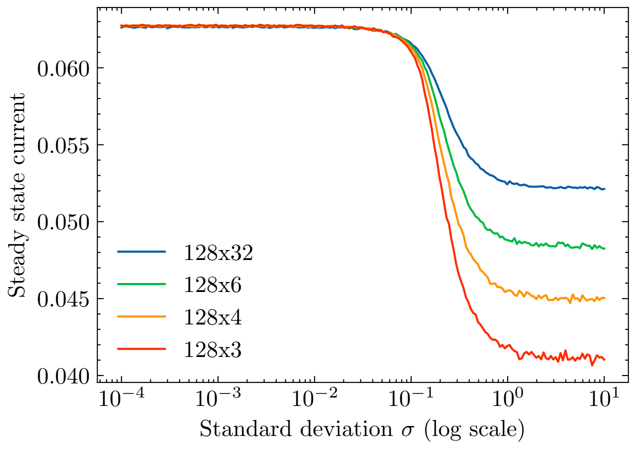


Equilibration of the current

Results: Baseline

Setup

- Classical 2D TASEP
- Periodic boundary conditions
- Checkerboard configuration
- Normally distributed speeds
- Waiting for steady state
- Measuring average current



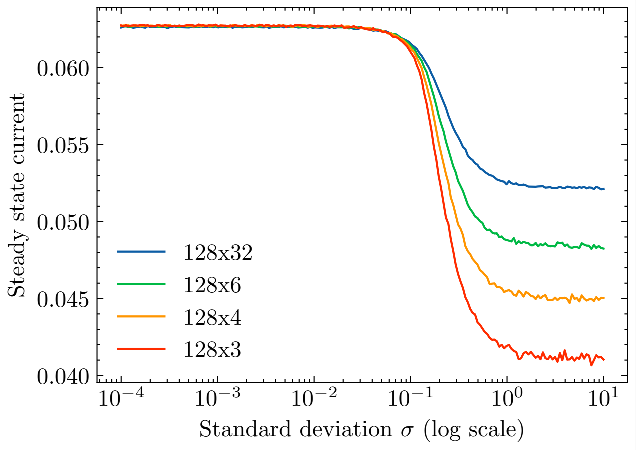
Steady state current as a function of σ for different system sizes. 800 runs and 150k steps averaged.

Results: Theory

Conditions for a forward move

1. randomly picked cell is occupied ($p_{\mathrm{occ}} = \rho$)
2. speed allows move ($p_{\mathrm{spd}} = \bar{v} = \mu$)
3. forward direction is picked ($p = 0.5$)
4. next cell is empty ($p_{\mathrm{emp}} = 1 - \rho$)*

$$\implies \langle J \rangle = p \cdot \mu \cdot \rho \cdot (1 - \rho) = 0.0625$$



Steady state current as a function of σ for different system sizes.

Naive Policy

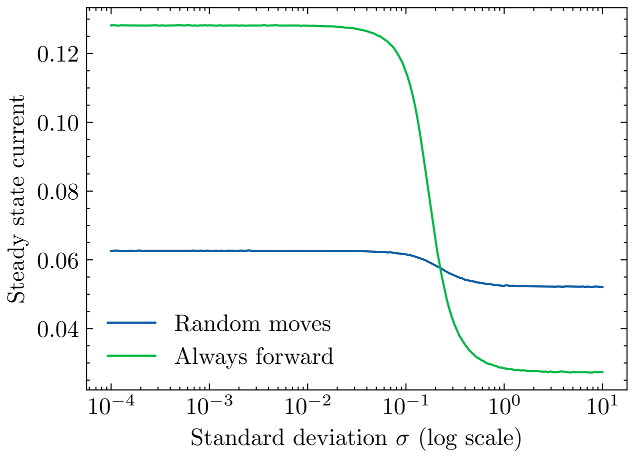
A Naive Policy

Just go forward?

- $p_{\text{fwd}} = 1, p_{\text{up/down}} = 0$
- starting in checkerboard configuration
- no messy mixing
- bad for high σ
- optimum for low σ?

$$\max \langle J \rangle = \max \left(p_{\text{occ}} \cdot p_{\text{spd}} \cdot p_{\text{fwd}} \cdot p_{\text{emp}} \right)$$
$$\leq \rho \cdot \mu \cdot 1 \cdot \max(p_{\text{emp}})$$

$$\text{and} \quad 0.5 \leq \max(p_{\text{emp}}) < 1$$

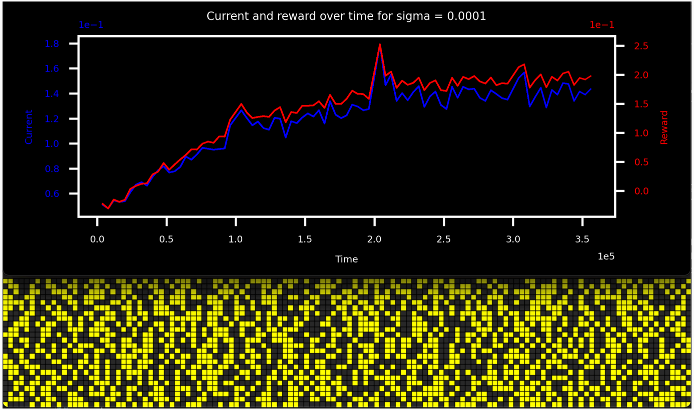


Naive policy

Smarticles

Simple reward

- Positive reward for going forward
- Negative reward for occupied destination
- Zero reward for going up or down or staying
- Small negative reward for blocking others

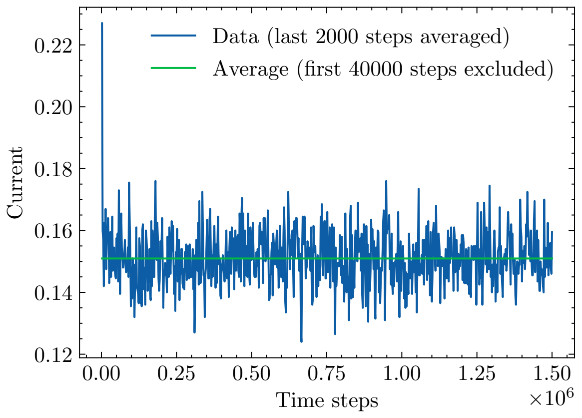


Screenshot of the first smarticle training

Smarticles: Equal speeds

Resulting current

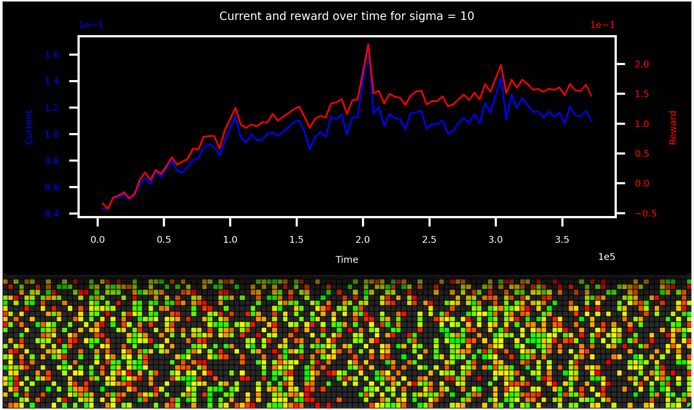
- $\langle J \rangle \approx 0.152$
- $+143\%$ compared to baseline (0.0625)
- $+22\%$ compared to naive policy (0.125)



Steady state current over time for the trained smarticles.

Similar reward, $\sigma = 10$

- Same setup as before
- Negative reward for blocking proportional to speed

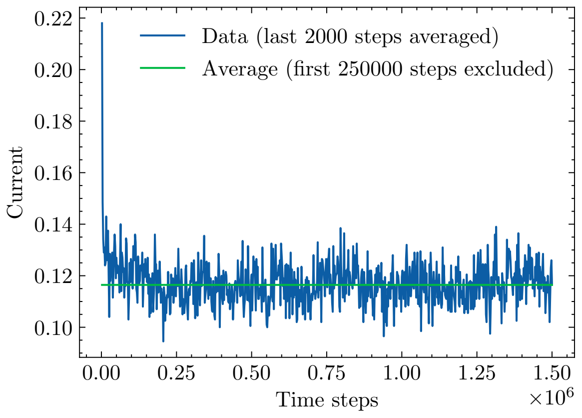


Screenshot of the second smarticle training

Smarticles: Uniform speed distribution

Resulting current

- $\langle J \rangle \approx 0.117$
- $+125\%$ compared to baseline (0.052)
- Lower than with equal speeds (we will see why)



Steady state current over time for the trained smarticles.

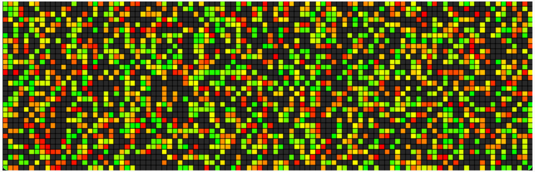
What did the smarticles learn? (Playground video)

Observation

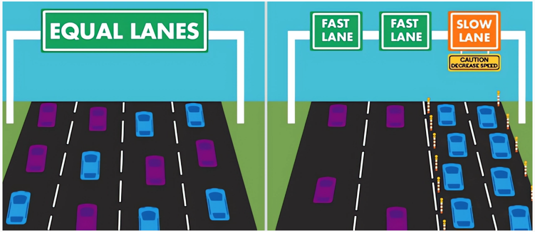
- So far, the system still looks messy
- Only individual behavior
- Compare highway: Fast lane and slow lane $\rightarrow$ Could be beneficial also in TASEP

Potential

- Less jamming
- Directed flow without dispersion
- Inhomogeneous density



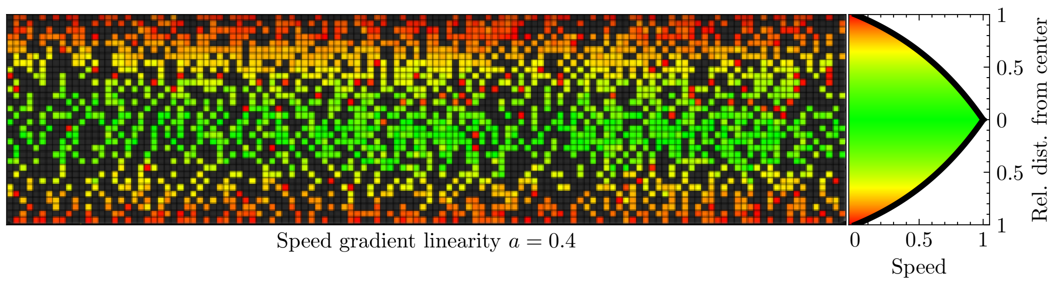
Screenshot of the simulation with the current policy



Separating fast and slow lanes increases the mean velocity

Approach 1: Hard code the lanes, supply smarticles with lane information

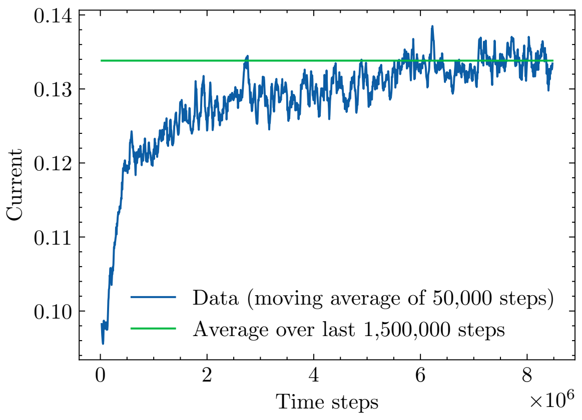
$$\Delta y = 1 - a \cdot \left(\frac{1+a}{a}\right)^{v} + a$$



Learned smarticle behavior with the hard-coded lanes reward function (left), and the speed gradient mapping (right).

Resulting current

- $\langle J \rangle \approx 0.134$
- $+157\%$ compared to baseline (0.052)
- $+15\%$ compared to previous policy (0.117)
- long equilibration time



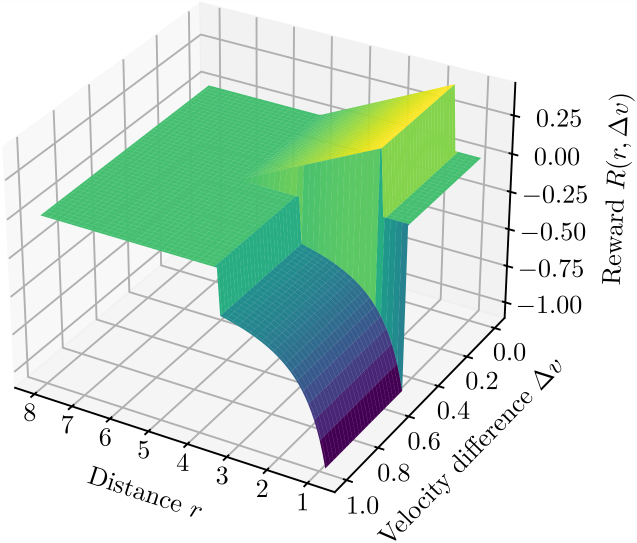
Current over time for the trained smarticles.

Approach 2: Let the smarticles learn the lanes themselves from local "interactions"

$$V(r, \Delta v) = \begin{cases} -0.125 \cdot r + 0.625 & \text{if } \Delta v < 0.5 \text{ and } 1.5 < r \leq 5 \\ -0.75 \cdot r^{-1.3} - 0.15 & \text{if } \Delta v \geq 0.5 \text{ and } r \leq 3.5 \\ 0 & \text{otherwise} \end{cases}$$

Modifications

1. Binary speed distribution
2. Different neural networks



Visual representation of the lane reward function / “potential”

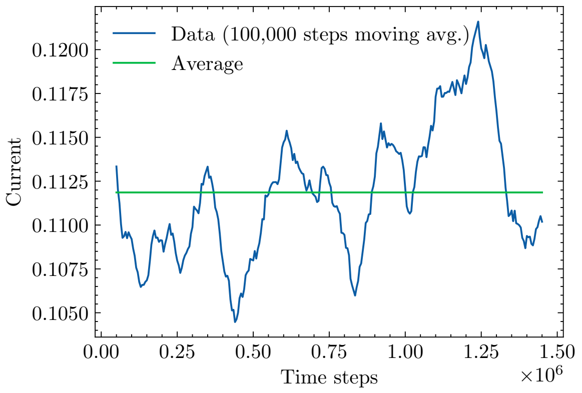
Smarticles: Self-organized Lanes - Results

Let's see it in action! (lane formation video)

Smarticles: Self-organized Lanes - Results

Resulting current

- $\langle J \rangle \approx 0.112$
- baseline
  $J = 0.4 \cdot (1 - 0.4) \cdot 0.6 \cdot 0.5 = 0.072$
- $+56\%$ compared to baseline
- lower than previous policies
- quicker equilibration
- fluctuations seem higher than they are
- proof of concept, room for improvement



Current over time for the trained smarticles.

Conclusion and Outlook

Conclusion and Outlook

Achievements and Insights

- Baseline: Theory and simulation match
- Established framework for intelligent matter simulations with interacting agents
- Intelligent matter in form of smarticles dramatically increases the current
- Large-scale structures and collective behavior emerge from local "interactions"

Future Research

- Modular `smarttasep` package for easy experimentation
- Generalize results:
  - Structure formation with more than two particle types
  - Different cluster densities for different speeds
  - Reduce the gap between the lanes
- Try different strategies other than lane formation
- Generalize results to more complex systems (e.g. higher dimensions, continuous space)