Strategies for simulating pedestrian navigation with multiple reinforcement learning agents

Francisco Martinez-Gil, Miguel Lozano, Fernando Fernández

Presented by: Daniel Geschwender

Overview

- Use MAS techniques to learn realistic pedestrian behavior
  - Control velocity
  - Avoid obstacles/other agents
  - Reach goal location
- Two proposed algorithms combining vector quantization and Q-learning
- Evaluated in room and corridor scenarios

Outline

Day 1:

- Introduction and related work
- Modeling pedestrian navigation
- State space generalization

Day 2:

- Algorithms
- Experimental set-up
- Learning results
- Simulation results
- Conclusion

Outline

Day 1:

- Introduction and related work
- Modeling pedestrian navigation
- State space generalization

Introduction and related work

- Application of pedestrian simulation:
  - Architecture
  - Civil engineering
  - Game development

Introduction and related work

- Two perspectives of pedestrian dynamics:
  - Microscopic: individual features, local perceptions, local interactions
  - Macroscopic: global functions, flows, densities
- Recent focus on microscopic perspective:
  - Collective effects are direct consequence of microscopic dynamics
  - Allows high-level decision-making without major changes to behavior model

Introduction and related work

- Benefits of a Multi-Agent Reinforcement Learning (MARL) approach:
  - Low computational cost associated to the agents' behavior
  - The richness of the group behavior
  - Model-free design of the problem
  - Emergent collective behaviors

Introduction and related work

- Previous non-RL approaches:
  - Cellular automata models
  - Force-based models
  - Rule-based models
  - Queueing models
  - Psychological and cognitive models
  - Crowd simulation
  - Models calibrated from pedestrian video data

Introduction and related work

- Previous RL approaches:
  - Single agent navigation using learning
  - Multi-agent navigation with learning for a centralized control
  - Basic microscopic learning framework developed for small number of agents

Introduction and related work

- Contributions of the paper:
  - Analysis of knowledge transfer approaches
  - Extensive performance analysis
  - New learning and simulation scenario
  - New micro and macro simulation metrics
  - Comparison to Helbing model

Modeling pedestrian navigation

- Markov decision process (MDP)
  - State space
    $$S$$
  - Action space
    $$A$$
  - Probabilistic transition function
    $$P : S \times A \times S \rightarrow [0, 1]$$
  - Reward function
    $$R : S \times A \rightarrow \mathbb{R}$$

Modeling pedestrian navigation

- Find a policy to maximize discounted expected reward $V(s) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\}$
- Q-learning to estimate the expected rewards of each state-action pair
- Q table is updated using

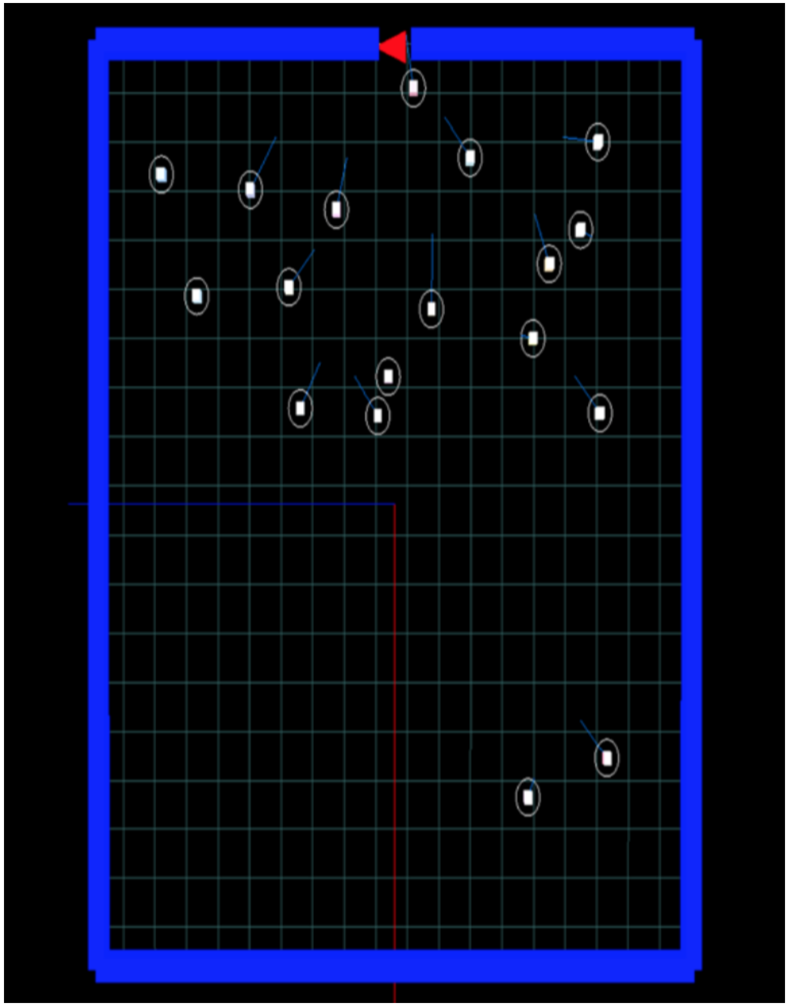
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a \{Q(s_{t+1}, a)\} - Q(s_t, a_t)]$$

Modeling pedestrian navigation

- Natural extension to multi-agent systems are Markov games
- Actions become joint actions of all agents
- All agents receive their own rewards
- Difficult to use in practice due to explosion of action space
- Instead, each agent learns independently

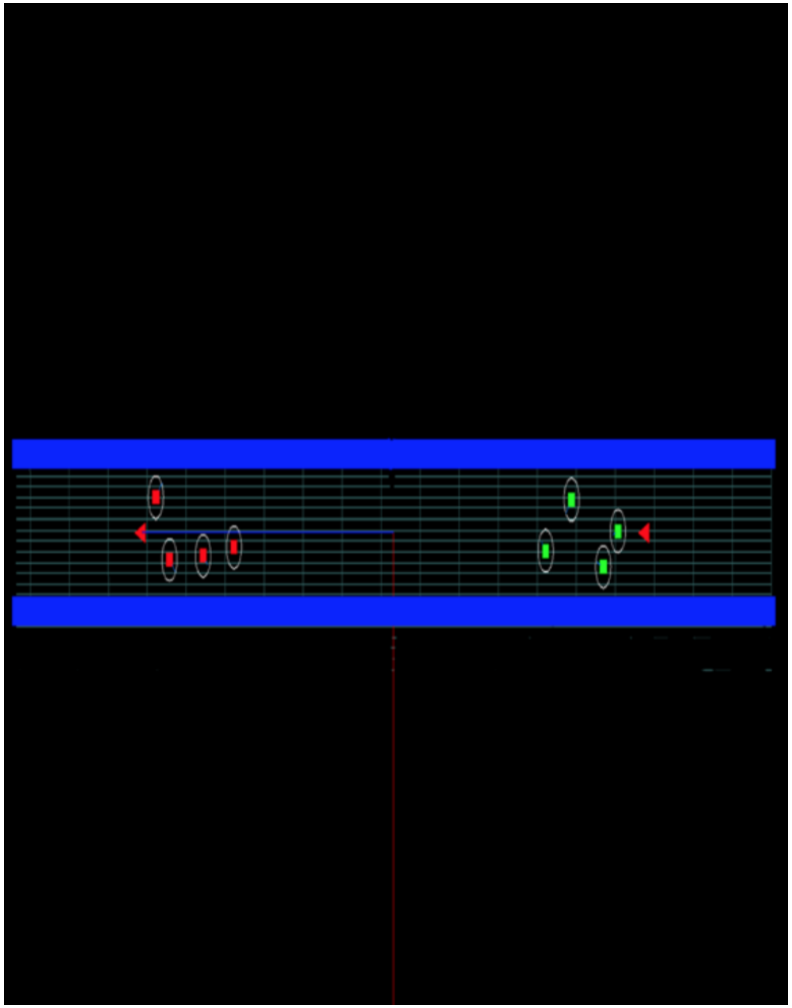
Modeling pedestrian navigation

- Scenario 1:
  - Square room
  - Filled with multiple agents
  - Single exit
  - Exit becomes a bottleneck



Modeling pedestrian navigation

- Scenario 2:
  - Narrow corridor
  - Two groups of agents at opposing ends
  - Must cross to opposite to reach the goal
  - Must form lanes to avoid collision



Modeling pedestrian navigation

- Environment is two dimensional continuous plane
- Room: 15m x 15m with 0.8m wide door
- Corridor: 15m x 2m
- Pedestrians:
  - Bounding circumference with radius 0.3m
  - Maximum speed of 1.8m/s

Modeling pedestrian navigation

- State Space
  - Deictic representation: local or relative to agent (e.g. nearest neighbors)
  - Global state is too extensive to track
  - Always track state of two nearest walls
  - Vary the number of nearest neighbors to track depending on scenario

Modeling pedestrian navigation

- State Space

Sag	Module of the velocity of the agent.
Av	Angle of the velocity vector relative to the reference line.
$Dgoal$	Distance to the goal.
$Srel_i$	Relative scalar velocity of the ith nearest neighbor.
Dag_i	Distance to the ith nearest neighbor.
Aag_i	Angle of the position of the ith nearest neighbor relative to the reference line.
Lag_i^{a}	Label to identify the group that the neighbor belongs to.
Dob_j	Distance to the jth nearest static object (walls).
Aob_j	Angle of the position of the jth nearest static object relative to the reference line.

The reference line joins the agent's position with its goal position
[a] This feature is used in the crossing experiment only

Modeling pedestrian navigation

- Action Space
  - Modify the agent's velocity vector
  - Increase/reduce speed

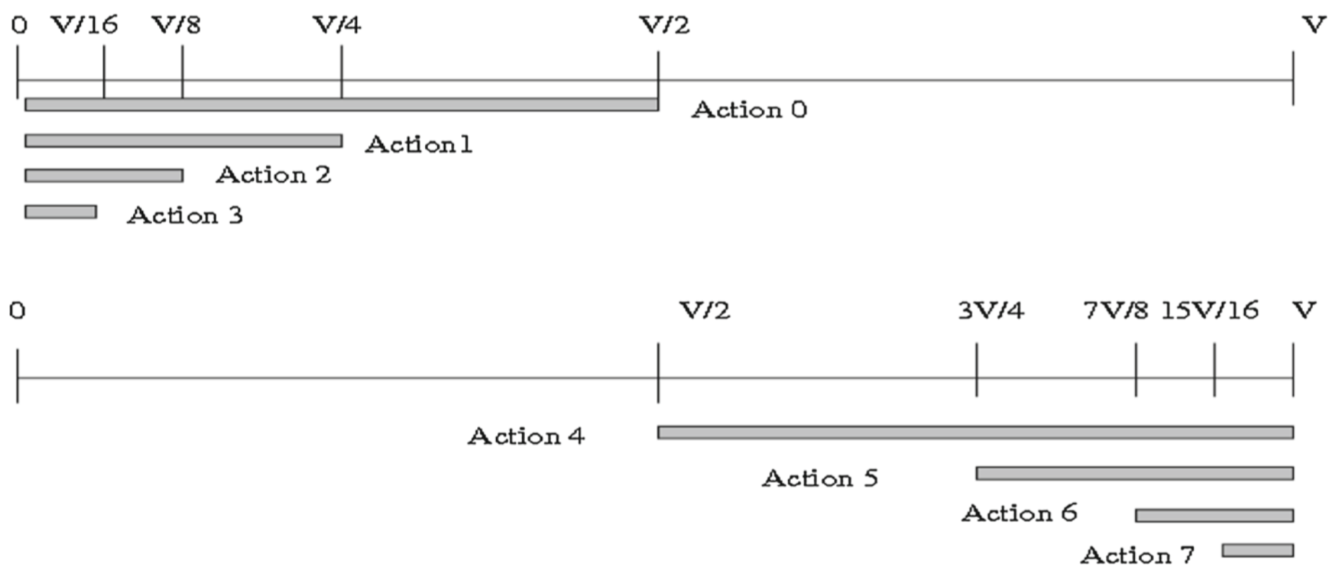
$$\{-1, -\tfrac{1}{2}, -\tfrac{1}{4}, -\tfrac{1}{8}, +0, +\tfrac{1}{8}, +\tfrac{1}{4}, +\tfrac{1}{2}, +1\} \cdot 0.9\,\frac{m}{s}$$

  - Adjust the orientation clockwise/counterclockwise

$$\{-1, -\tfrac{1}{2}, -\tfrac{1}{4}, -\tfrac{1}{8}, +0, +\tfrac{1}{8}, +\tfrac{1}{4}, +\tfrac{1}{2}, +1\} \cdot 45^\circ$$

Modeling pedestrian navigation

- Action Space
  - Minimum speed: 0 m/s
  - Maximum speed: 1.8 m/s
  - Some overlap in velocity actions due to limits



Modeling pedestrian navigation

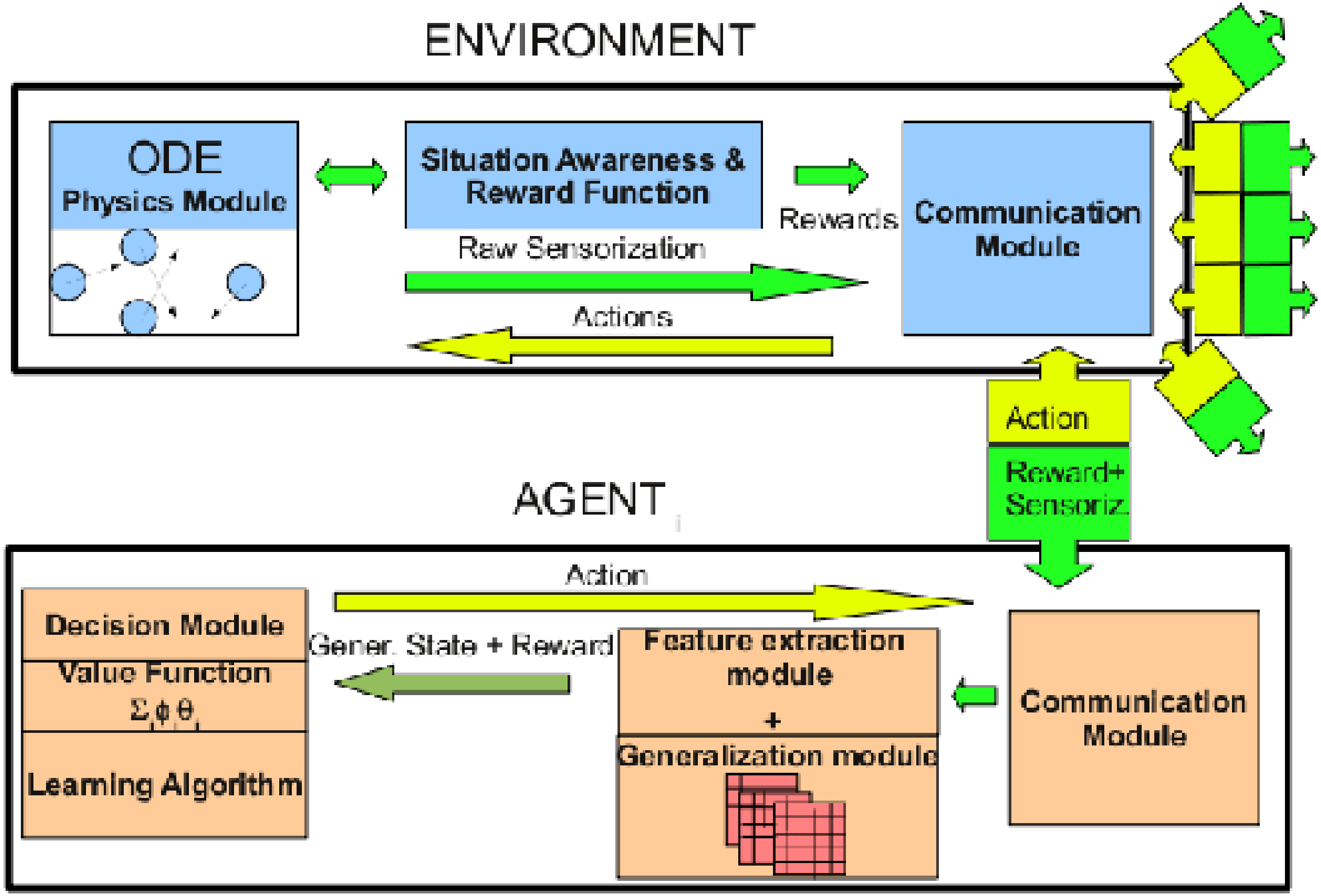
- Rewards

First scenario (agents in a room)	
Crash against other agent	−0.1
Crash against a wall	−2.0
Reach the goal	+100.0
Default	0.0
Second scenario (crossing)	
Reach the goal	+100.0
Default	0.0

Modeling pedestrian navigation

- Kinematic model
  - Discretized time steps of configurable size
  - At each step:
    - Provide state for each agent
    - Accept action from each agent
    - Simulate movements using constant velocities unless there is a crash

Modeling pedestrian navigation



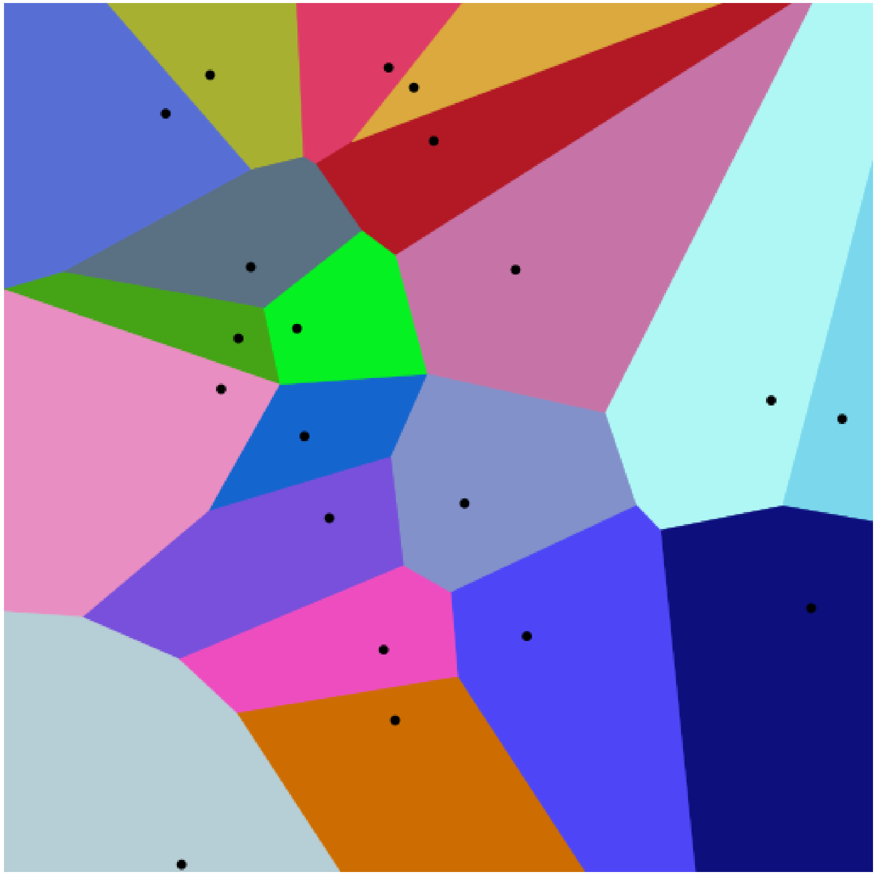
State space generalization

- The state space is too large as-is
- Vector Quantization (VQ) can be used to discretize the state space to any desired resolution
- Map states from k-dimensional Euclidean space to a finite set of states
  - Sensorized state: $x \in \mathbb{R}^k$
  - Prototypes states: C

$$V_Q(x) = \arg\min_{y \in C}\{dist(x, y)\}$$

State space generalization

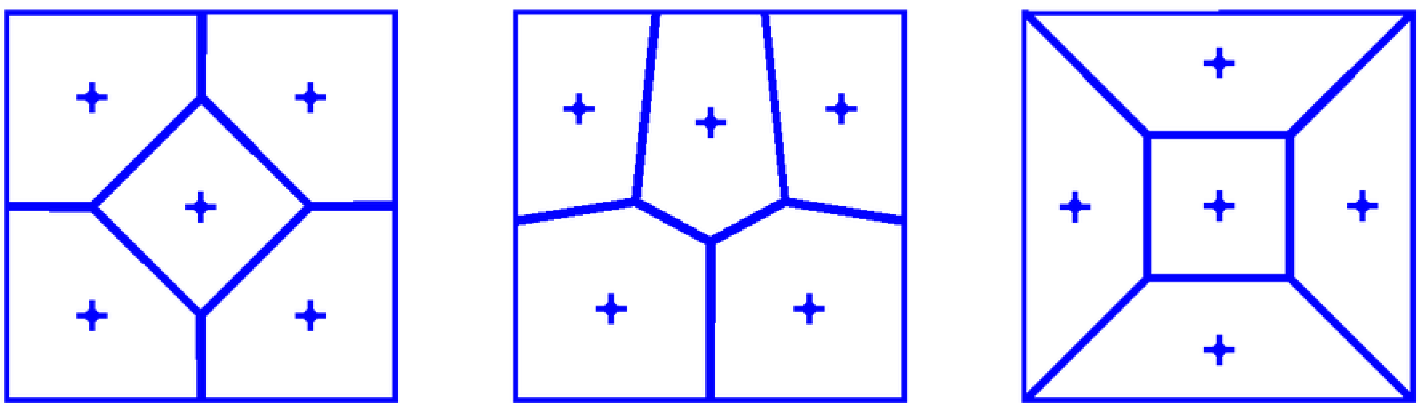
- Generalized Lloyd Algorithm (GLA)
  - Used to generate the prototype states
  - Operates on Voronoi regions of the state space



By Balu Ertl - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=38534275

State space generalization

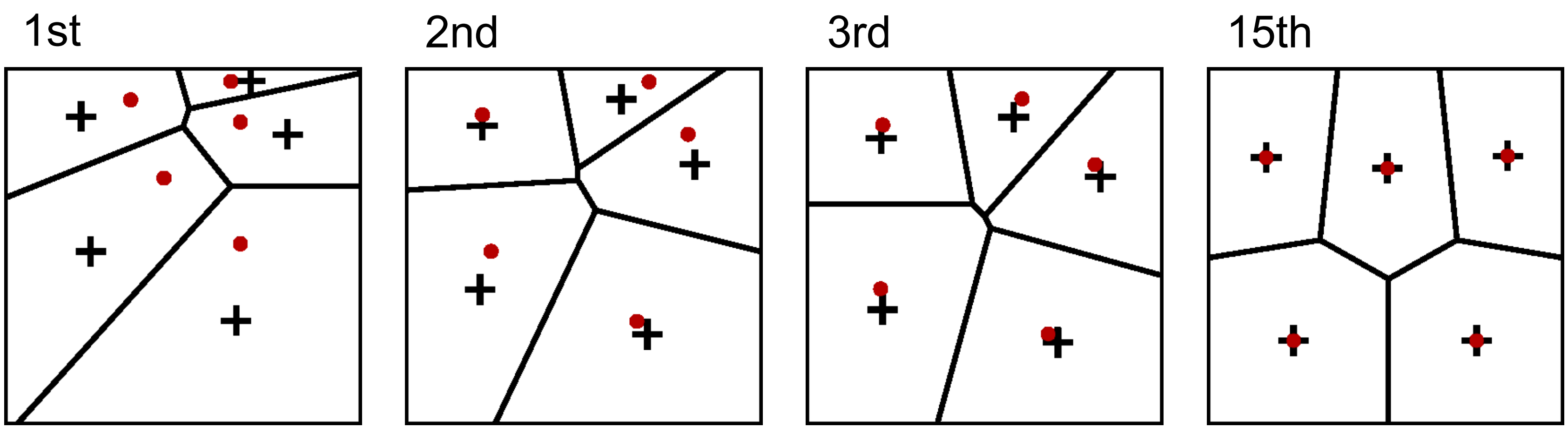
- Generalized Lloyd Algorithm (GLA)
  - Input: a set of points in the space
  - Output: a set of points defining a Centroidal Voronoi Tessellation



By RuppertsAlgorithm - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=9867802

State space generalization

- Generalized Lloyd Algorithm (GLA)
  - Iteratively perturbs points towards the centroids of the Voronoi regions



By Dominik Moritz - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=26443219

State space generalization

- Vector quantization for Q-learning (VQQL)

Single-agent VQQL algorithm

1. Generate the set T of samples of the state space S interacting with the environment using an exploratory policy.
2. Discretize the state space
   (a) Use GLA to obtain a state space discretization $C \in S$ from the sample set T.
   (b) Let $VQ : S \rightarrow C$ be the function that, given any state in S, returns the discretized value in C.
3. Learn the Q-table

   While the final condition is not reached

   i. Get an experience tuple $< s_1, a, s_2, r >$ by interacting with the environment.

   ii. Map the states of the experience tuple using VQ. Each adquired tuple of experience $< s_1, a, s_2, r >$ is mapped to $< VQ(s_1), a, VQ(s_2), r >$

   iii. Apply the Q-learning update function defined in Eq. 1 to learn a tabular value function Q: $C \times A \rightarrow \Re$, using the mapped experience tuple.
4. Return Q and VQ

State space generalization

- Vector quantization for Q-learning (VQQL)
  - Requires two elements
    - An input dataset, T, collected to represent the whole state space
    - A desired resolution (number of prototype points)
  - Random walks performed to obtain initial sensorized data
  - Resolution tested in 6 experiments using:

$$k = \{512, 1024, 2048, 4096, 8192, 16384\}.$$

State space generalization

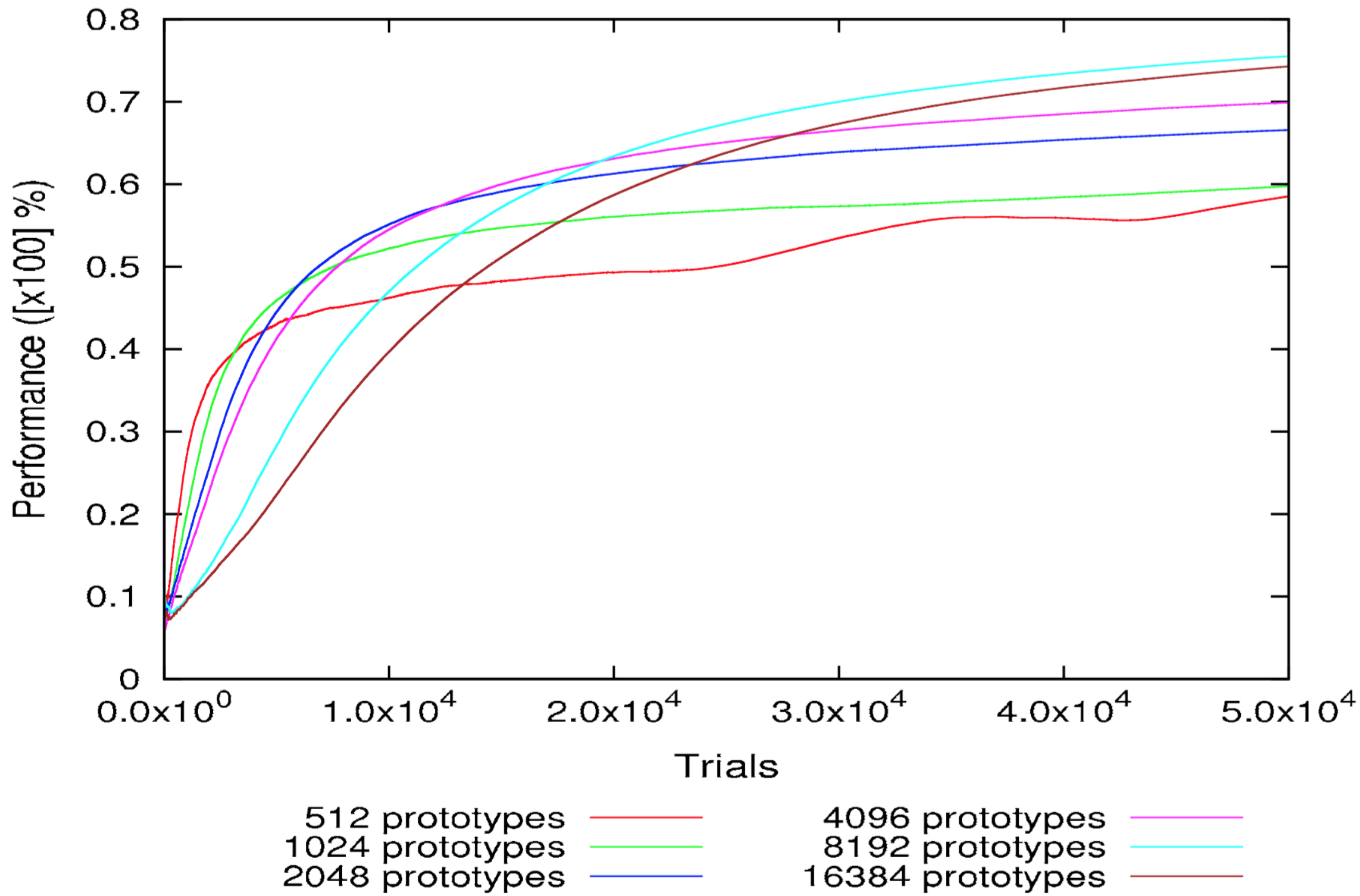


Fig. 3 Learning *curves* for the first scenario using the VQQL algorithm and vector quantizers with different numbers of prototypes. The *curves* are the means of 18 learning processes (18 agents)

State space generalization

- Vector quantization for Q-learning (VQQL)
  - Doubling number of prototypes also doubles the computation time for GLA
  - Best trade-off between performance and computational cost:
    - Room scenario: 4096 prototypes
    - Corridor scenario: 8192 prototypes

Outline

Day 1:

- Introduction and related work
- Modeling pedestrian navigation
- State space generalization

Day 2:

- Algorithms
- Experimental set-up
- Learning results
- Simulation results
- Conclusion

Outline

Day 2:

- Algorithms
- Experimental set-up
- Learning results
- Simulation results
- Conclusion

Algorithms

- Three challenges:
  1. How many agents should learn from scratch and should they be introduced gradually?
  2. How to handle different state spaces with different numbers of agents present?
  3. How to generate a representative dataset for use in generating prototype states?
- Propose two iterative learning schemes to address these challenges

Algorithms

- Iterative vector quantization for Q-learning (**ITVQQL**)
- Incremental vector quantization for Q-learning (**INVQQL**)
- Both
  - perform iterations of VQQL
  - conduct a policy and learning transfer
- Differ in how they handle number and placement of agents

Algorithms

Multi-agent ITVQQL/INVQQL schemas

Entry: the number of iterations N

Return: the sets Q_N and V_N (the value table of Q-learning and the vector quantizer respectively).

1. $i \leftarrow 1$
2. **Set p to the initial number of agents in the environment**
3. For each agent, k $(1 \leq k \leq p)$ set:
   - its initial vector quantizer, $V_0^k(s) = \emptyset$
   - its initial policy $\pi_0^k = random$
4. Repeat:
   (a) **Decide whether or not to include new agents.** Set p consequently.
   (b) For each agent, k $(1 \leq k \leq p)$ do:
      i. Collect a dataset T_i^k for agent k using the policies π_{i-1}^k with V_{i-1}^k
      ii. **Build V_i^k using T_i^k for agent k following a transfer learning strategy**
   (c) Learn Q_i^k $\forall k, 1 \leq k \leq p$ and hence the policies π_i^k using Q-Learning (with the option of using transfer of value functions).
   (d) $i \leftarrow i + 1$

   Until $i = N$
5. Return Q_N and V_N

The bold statements mean different options in each schema as explained in the text

Algorithms

Feature	ITVQQL	INVQQL
Number of prototypes	Fixed	Variable
Number of features per prototype	Fixed	Variable
Number of agents per iteration	Fixed	Variable
Inter-iteration policy transfer	Yes	Yes
Inter-iteration prototype transfer	No	Yes
Inter-iteration value function transfer	Yes	Yes

Algorithms

- Policy Transfer
  - Uses a policy learned in previous iteration to generate the initial dataset of current iteration
  - Dataset become progressively more reflective of actual use
  - States of differing dimensionality handled by projection

Algorithms

- Prototype Transfer
  - INVQQL prototypes have higher dimensionality through the first 8 iteration
  - Lower dimensional prototypes are preserved and passed on through each iteration

Algorithms

- Value Function Transfer
  - Initialize the Q-table with values learned in previous iterations
  - Prototypes change between iterations so map to the nearest prototype from prior iteration
  - Only transfer between tables of the same dimension

Algorithms

- Performance Metrics
  - Jumpstart: improvement in initial performance of an agent
  - Asymptotic performance: improvement in the final performance of an agent
  - Time to threshold: learning steps needed to reach a pre-specified performance threshold

Experimental set-up

- Perform four categories of experiments:
  - Room scenario, corridor scenario
  - Evaluating learning performance, evaluating final performance
- Comparing several algorithm variants:
  - ITVQQL
  - ITVQQL w/ transfer
  - INVQQL
  - INVQQL w/ transfer
  - VQQL

Experimental set-up

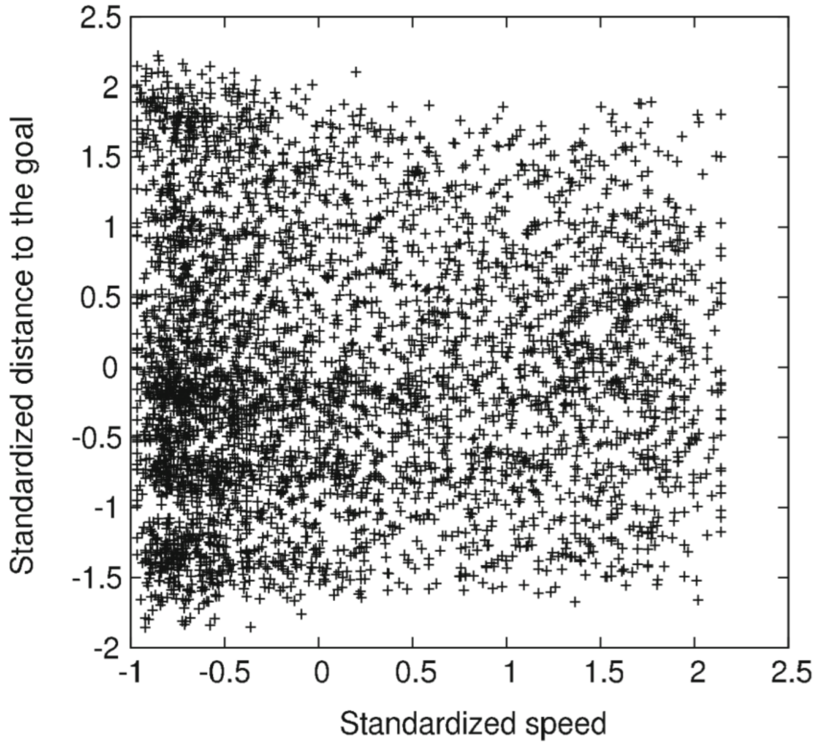
- Experiments conducted through a series of ‘episodes’
  - 1 episode = 150 actions
    (or all agents reach goal)
  - Environment resets for each episode
  - Learning is preserved across episodes
- Learning algorithms use multiple iterations
  - 1 iteration ≈ thousands of episodes
  - Learning resets across iterations
    (but may be transferred)

Learning results – room

- Settings for learning in room scenario:

Key	ITVQQL	INVQQL	VQQL
Episodes	50000	50000	900000
Iterations	18	18	1
Prototypes	4096	From 4096 to 32768	4096
Features per prototype	28	From 7 to 28	28
Agents per iteration	18	From 1 to 18	18
Inter-iteration prototype transfer	0	4096	0

Learning results – room



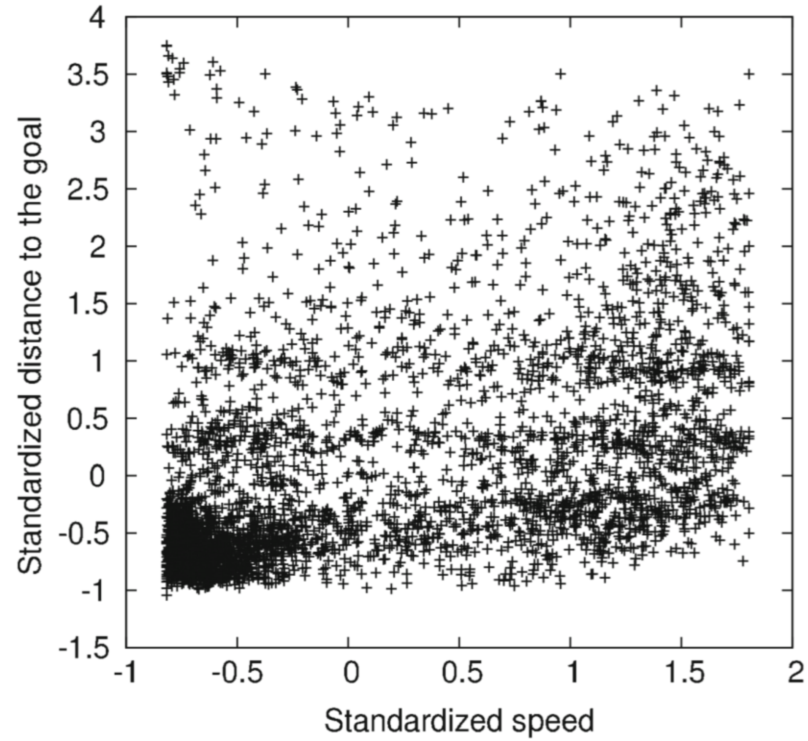
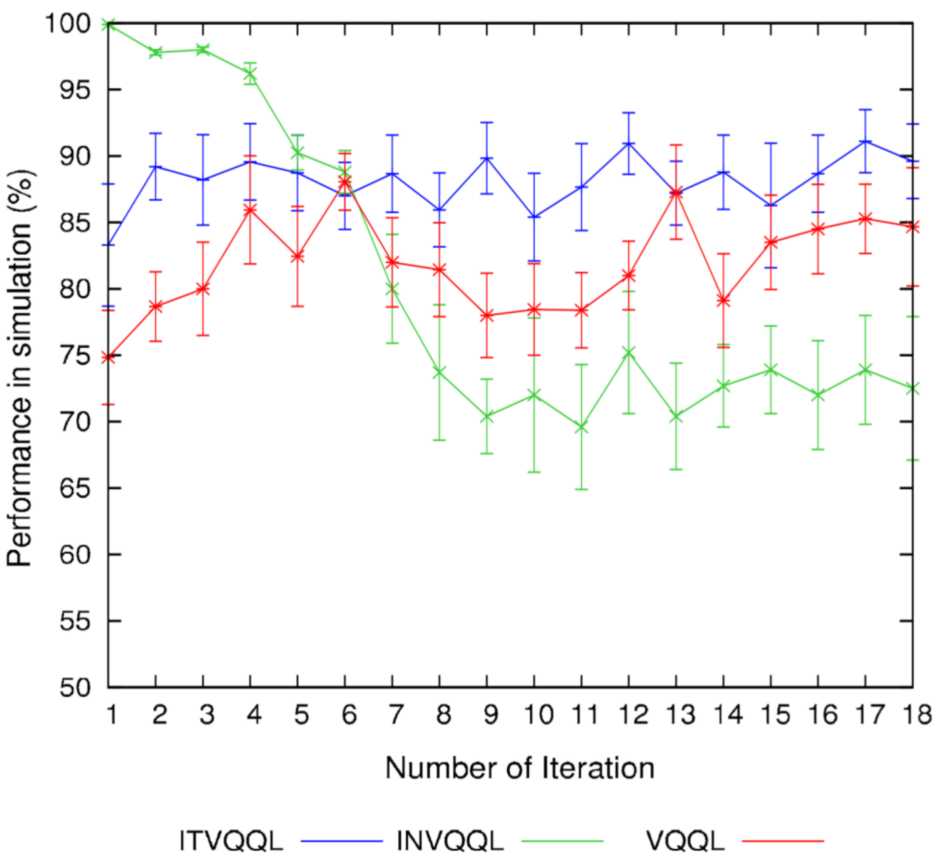
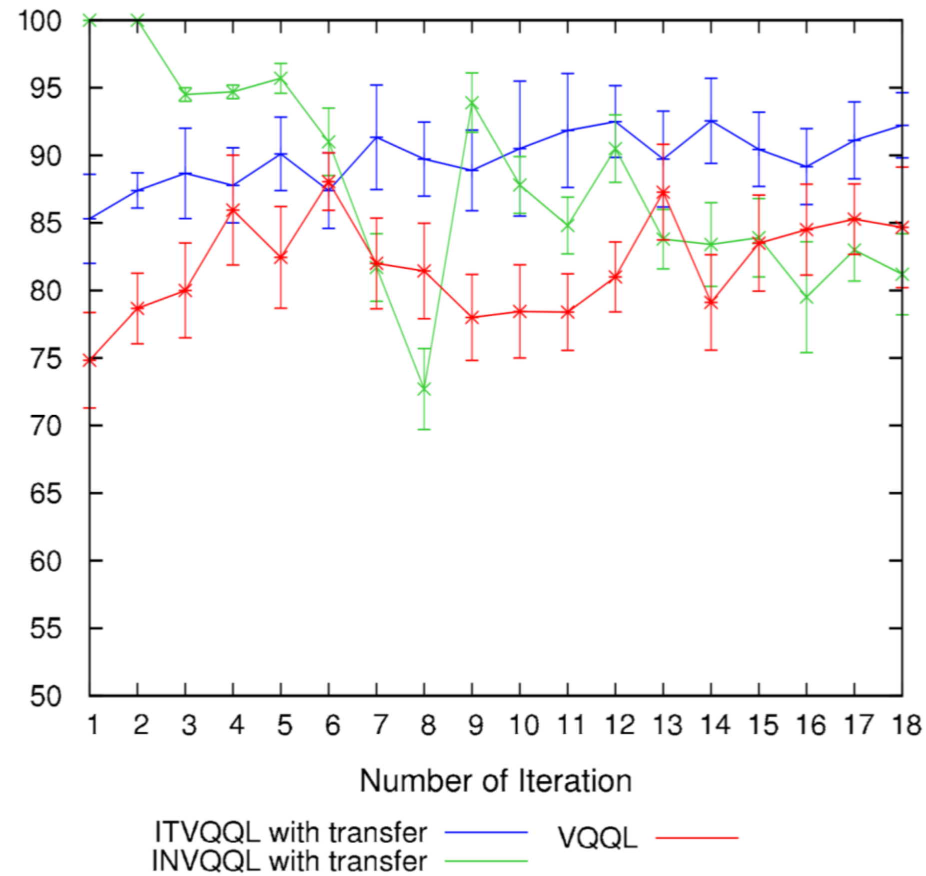


Fig. 4 Visualization of the prototypes calculated for the first iteration (*left*) and the last iteration (*right*) of the ITVQQL schema in the room scenario. The speed (in the x-axis) and the distance of the agent to the goal (in the y-axis) features are displayed. The prototypes of the *left* graphic are calculated using data collected from a random policy. The prototypes of the *right* graphic are calculated using data collected with a learned policy correspondent to the penultimate iteration

Learning results – room

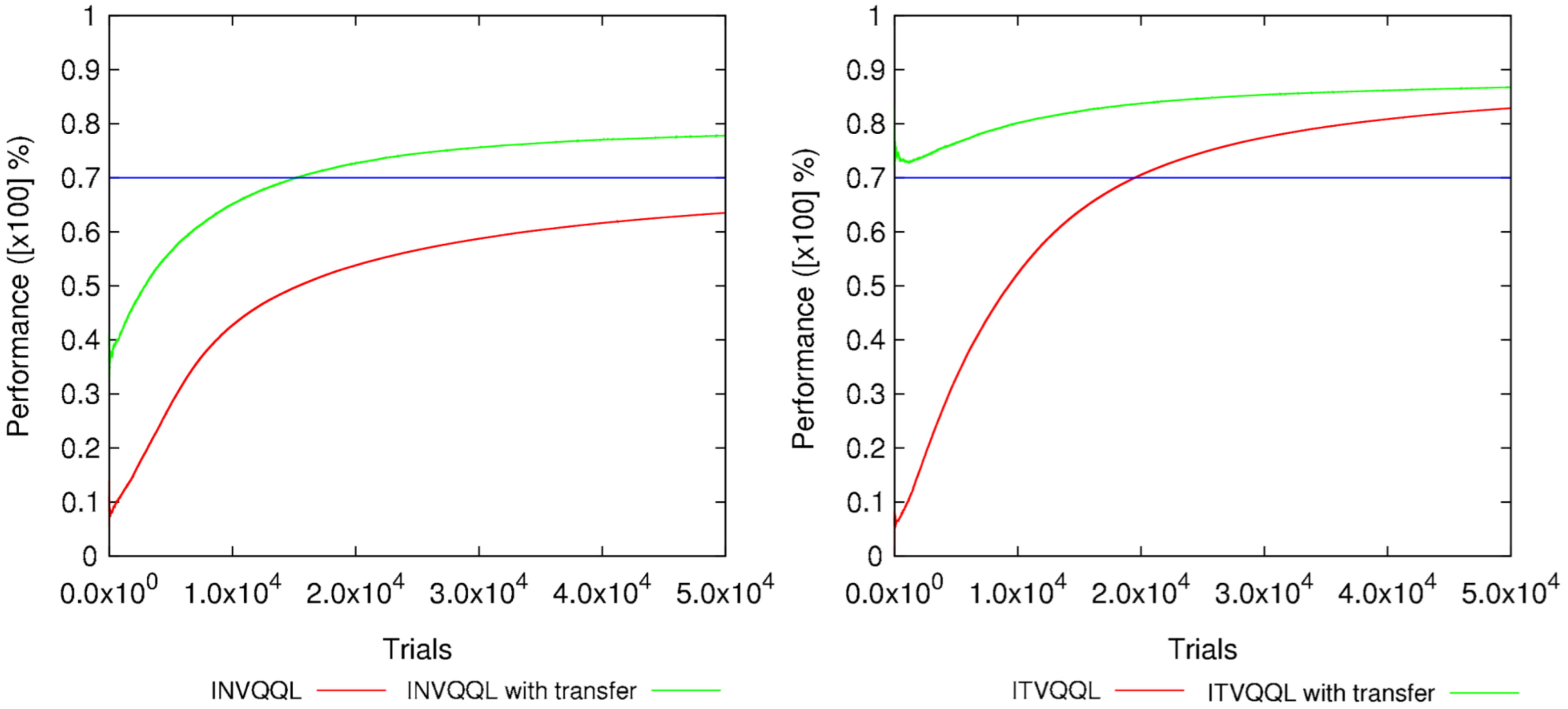
- Performance during learning:





Learning results – room

- Effect of knowledge transfer:



Learning results – corridor

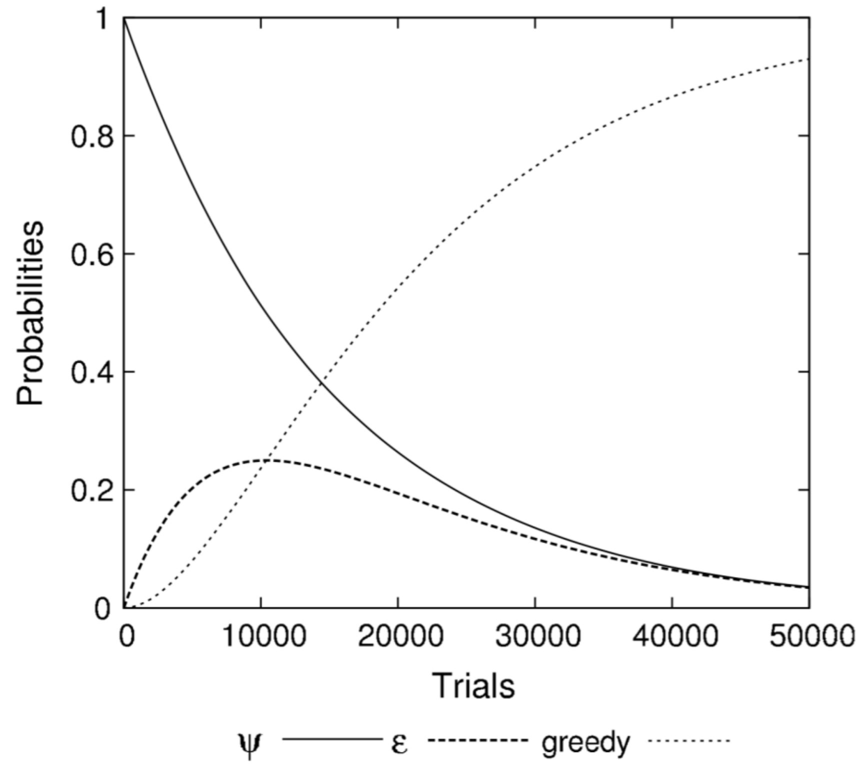
- Settings for learning in corridor scenario:

Key	ITVQQL	INVQQL	VQQL
Episodes	50, 000	50,000	400, 000
Iterations	8	8	1
Prototypes	8, 192	From 8,192 to 40,960	8, 192
Features per prototype	24	From 8 to 24	24
Agents per iteration	8	From 1 to 8	8
Inter-iteration prototype transfer	0	8182	0

Learning results – corridor

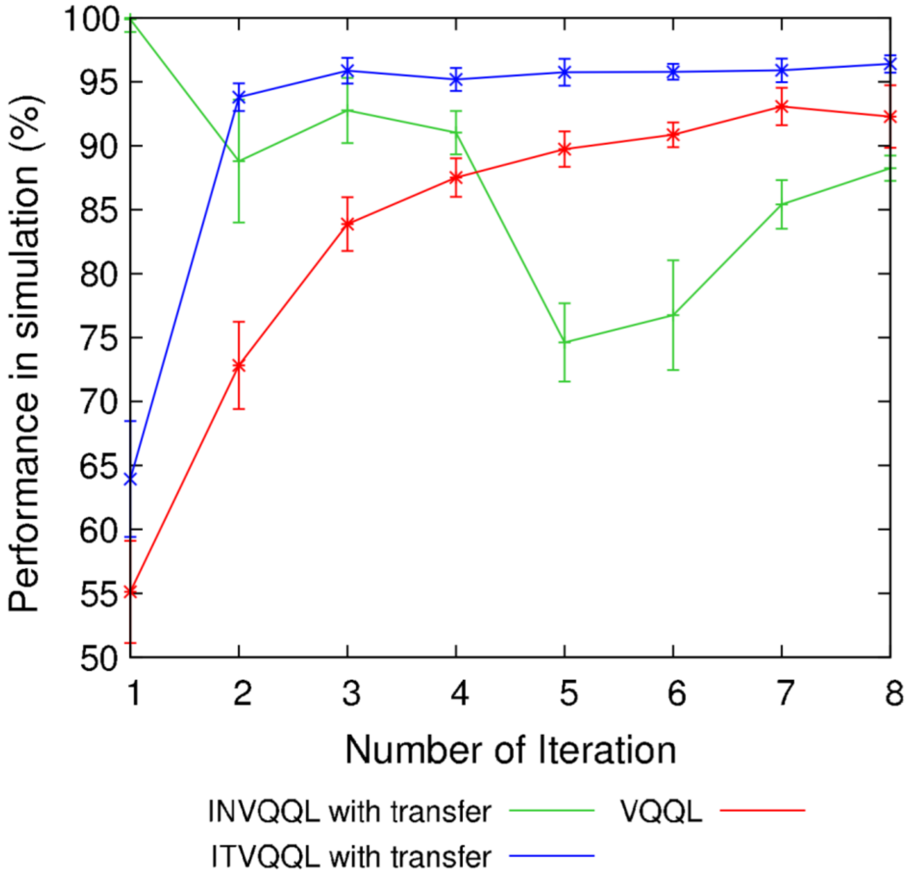
- Policy Reuse to provide bias (advice)
  - π_0 always drives agent to the opposite end of the corridor

$$\begin{cases} \psi & \text{choose the } \pi_0 \text{ policy} \\ (1-\psi)\epsilon & \text{choose an aleatory action} \\ (1-\psi)(1-\epsilon) & \text{choose the greedy policy} \end{cases}$$



Learning results – corridor

- Performance during learning:



Simulation results

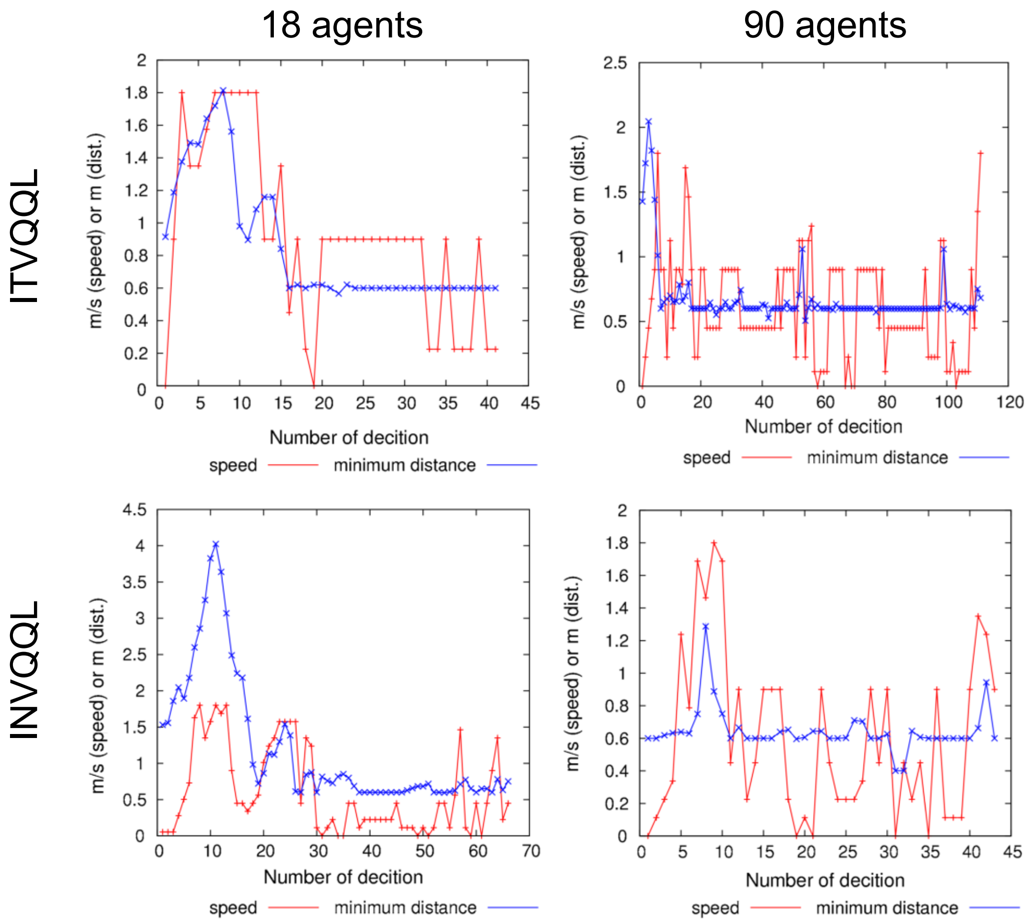
- Trained performance is evaluated in three categories:
  1. Local interactions: velocity vs. collision dist
  2. Macro-dynamics: fundamental diagram and density maps
  3. Performance: path length, # of failures
- Each experiment is conducted over 100 episodes of 700 decisions each

Simulation results - room

- Each experiment is conducted over 100 episodes of 700 decisions each
- Scalability is addressed by using 18, 36, 54, 72, 90 agents

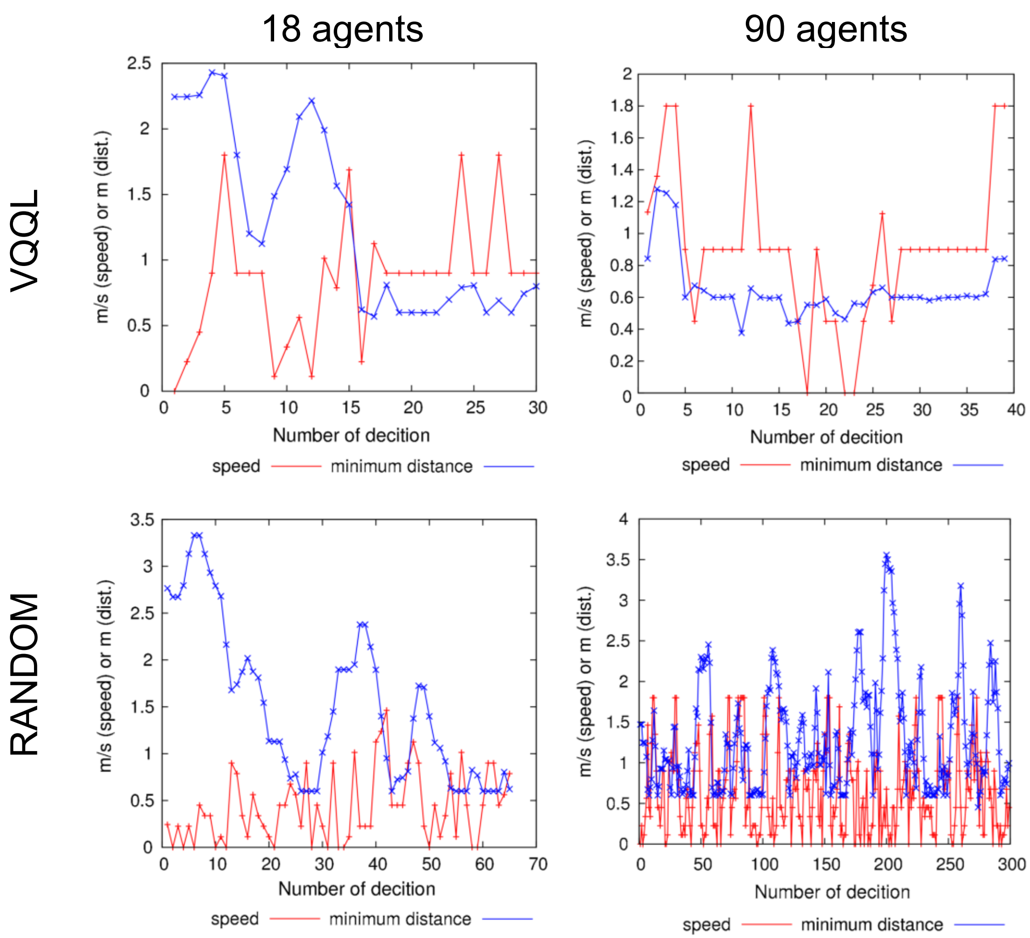
Simulation results - room

- Local interactions:



Simulation results - room

- Local interactions:



Simulation results - room

- Local interactions:

Table 10 Percentage of episodes in which the selected speed and distance to the nearest neighbor of an agent have a correlation coefficient greater than +0.5

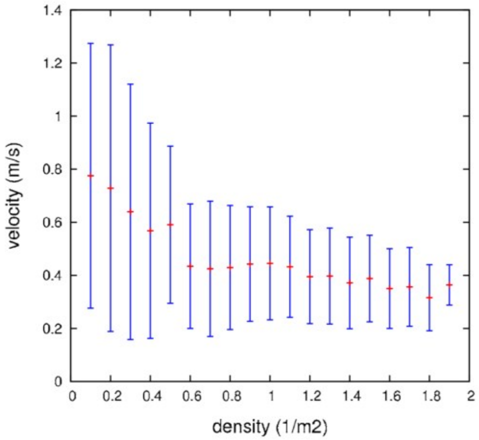
# Ag.	IT (%)	IN (%)	TF_IT (%)	TF_IN (%)	VQQL (%)	RANDOM (%)
18	77.3	37.1	79	46.5	45.7	1.72
90	48.8	12.4	48.6	12.1	41.8	1.1

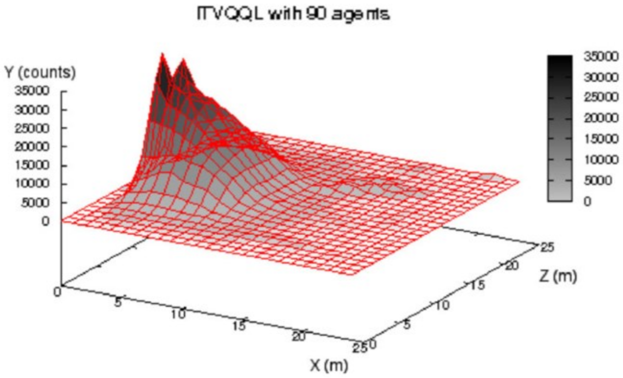
The data comes from the simulation of 100 episodes with 18 agents each (a total of 1800 episodes) in the first row and with 90 agents (a total of 9000 episodes) in the second row

Simulation results - room

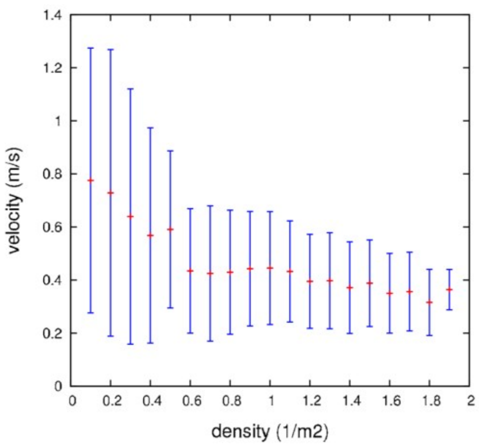
- Macro-dynamics:

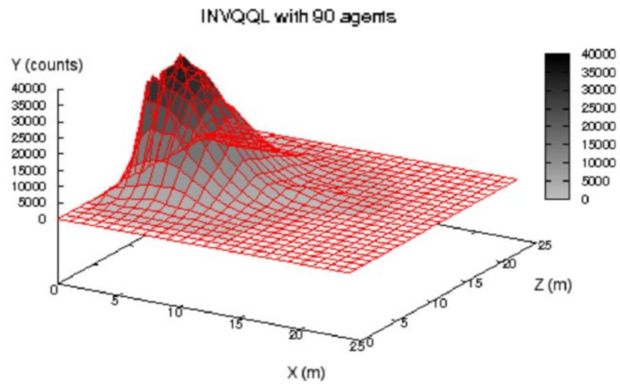
ITVQQL





INVQQL

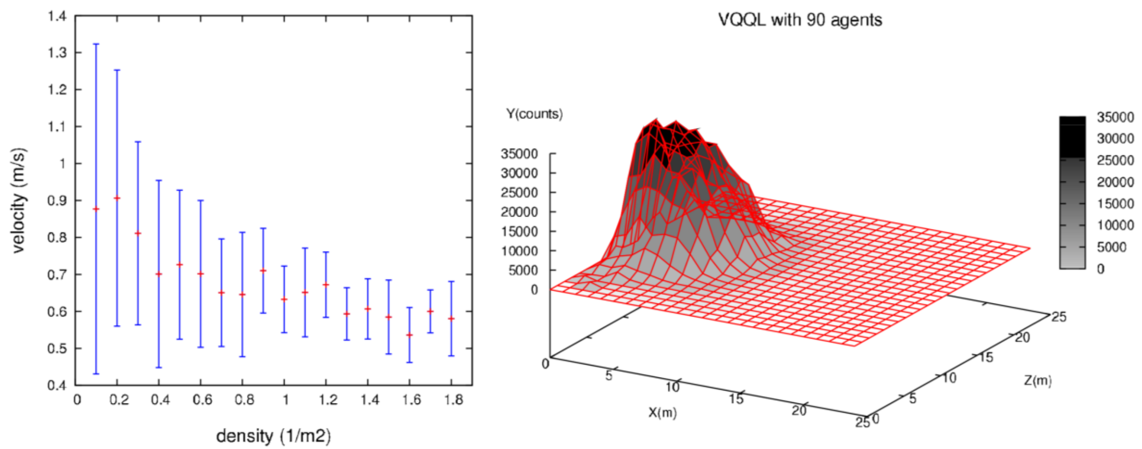




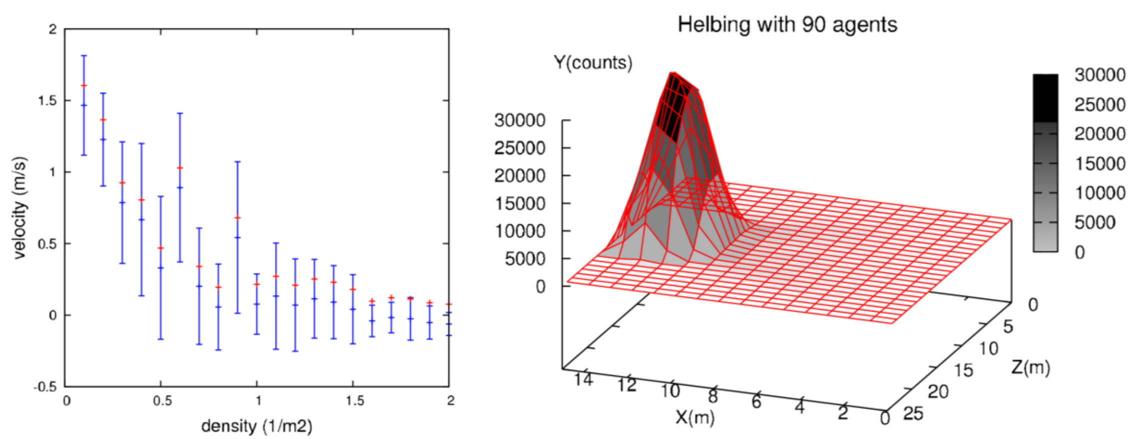
Simulation results - room

- Macro-dynamics:

VQQL



Helbing



Simulation results - room

- Performance:

Table 11 Averaged lengths and standar deviation for the paths in meters

# Ag.	IT	IN	TF_IT	TF_IN	VQQL
18	14 ± 7	20 ± 21	15 ± 7	18 ± 10	15 ± 11
36	13 ± 5	18 ± 15	15 ± 7	18 ± 8	15 ± 17
54	14 ± 6	18 ± 13	16 ± 8	19 ± 10	15 ± 11
72	15 ± 6	18 ± 12	16 ± 8	19 ± 11	15 ± 10
90	15 ± 7	18 ± 11	17 ± 9	20 ± 11	17 ± 10

The averages are over 100 episodes and for all the agents

Table 12 Average number and standar deviation of decisions per episode

# Ag.	IT	IN	TF_IT	TF_IN	VQQL
18	28 ± 56	41 ± 36	28 ± 56	63 ± 35	27 ± 21
36	32 ± 41	75 ± 52	67 ± 53	93 ± 58	43 ± 60
54	51 ± 55	118 ± 77	95 ± 74	135 ± 86	52 ± 60
72	68 ± 70	121 ± 78	120 ± 84	175 ± 112	69 ± 62
90	84 ± 85	144 ± 95	145 ± 96	211 ± 131	105 ± 135

The figures are averaged over 100 episodes and for all the agents

Simulation results - room

- Performance:

Table 13 Medians and means (in parenthesis) for the agents that do not reach the door (fails) when scaling up the number of agents

#Ag.	IT	IN	TF_IT	TF_IN	VQQL	P value
18	1 *b* (2.4)	4 *c* (4.2)	0 *a* (1.0)	0 *a* (2.9)	1 *b* (2.8)	0
36	1 *ab* (6.5)	4 *c* (4.8)	0 *a* (2.8)	3.5 *bc* (6.2)	0 *ab* (6.8)	4×10^{-9}
54	1 *a* (6.4)	4 *b* (6.0)	0 *a* (3.6)	4 *b* (6.4)	10 *ab* (15.7)	7×10^{-10}
72	1 *ab* (9.4)	4 *b* (5.6)	1 *a* (3.9)	4 *b* (6.6)	4 *b* (22.1)	4×10^{-8}
90	1 *ab* (10.3)	4 *b* (6.0)	1 *a* (4.1)	3 *b* (6.5)	18.5 *c* (25.0)	4×10^{-10}

The means are averaged over 100 episodes ($N = 100$) and are considered a measure of performance. Median values separated by different letters for the same number of agents (within a row), are significantly different ($P \leq 0.05$) according to Kruskal–Wallis test

Simulation results - room

https://www.uv.es/agentes/RL/itvqql.htm

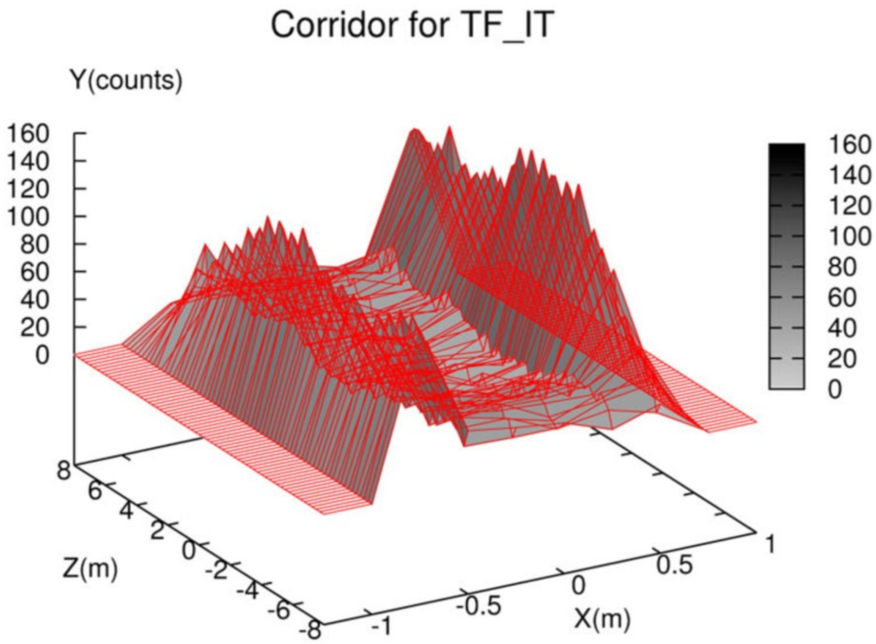
https://www.uv.es/agentes/RL/invqql.htm

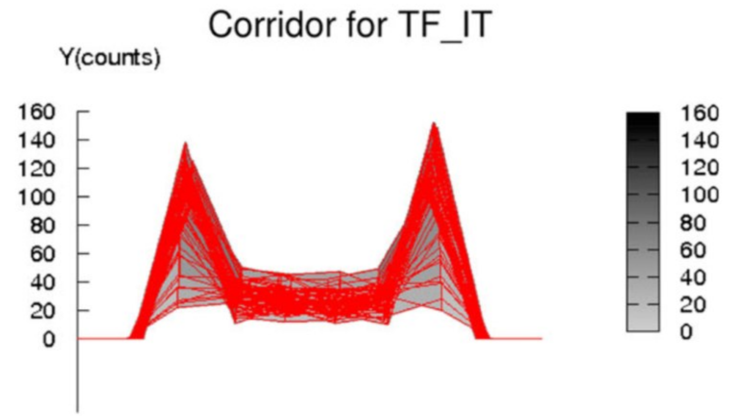
Simulation results - corridor

- Less detailed results than the room scenario (only density maps and # of successes)
- Runs 100 episodes of 80 decisions each
- Only counts as a success if all agents cross the hallway

Simulation results - corridor

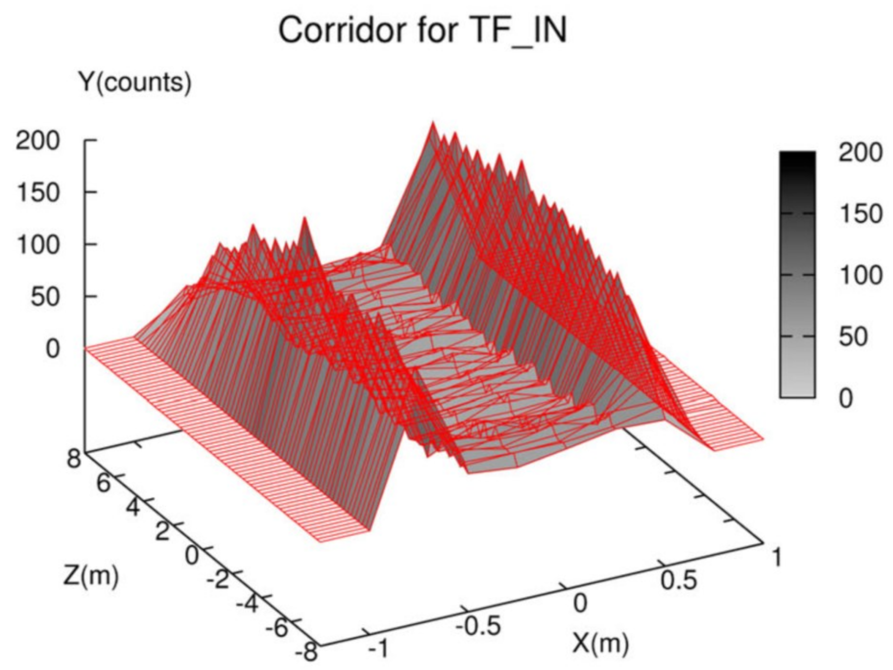
- Macro-dynamics:

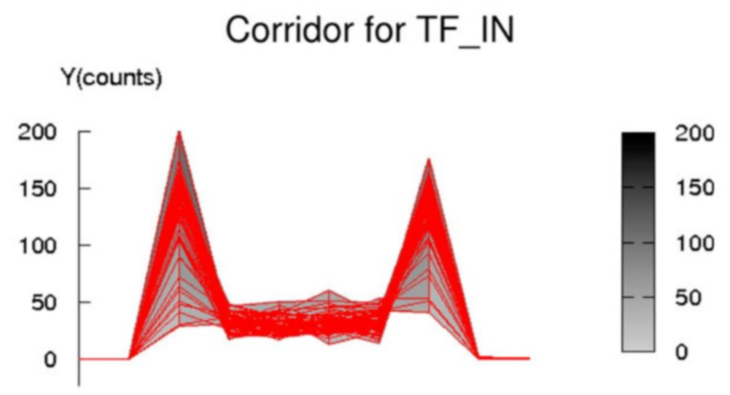




Simulation results - corridor

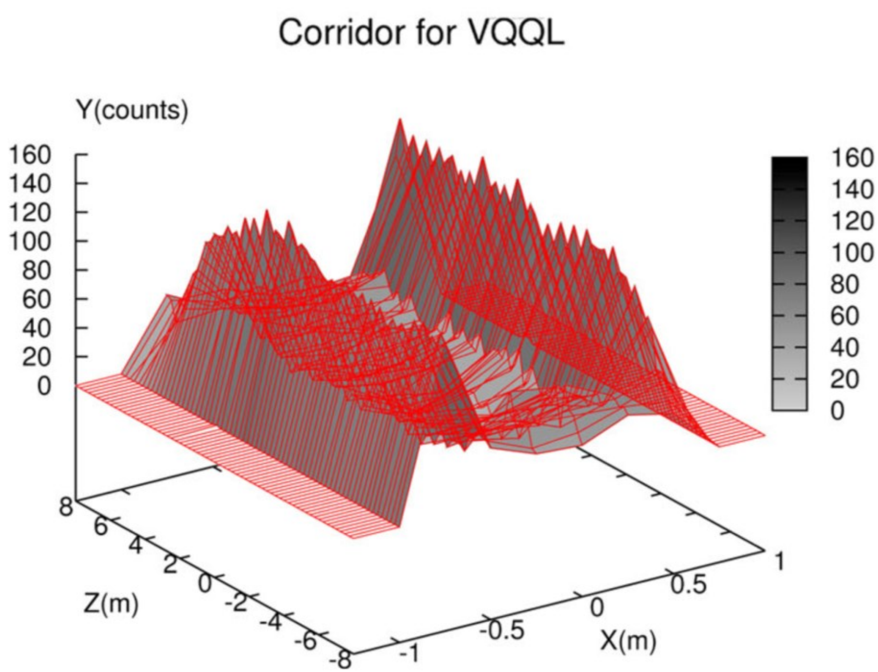
- Macro-dynamics:

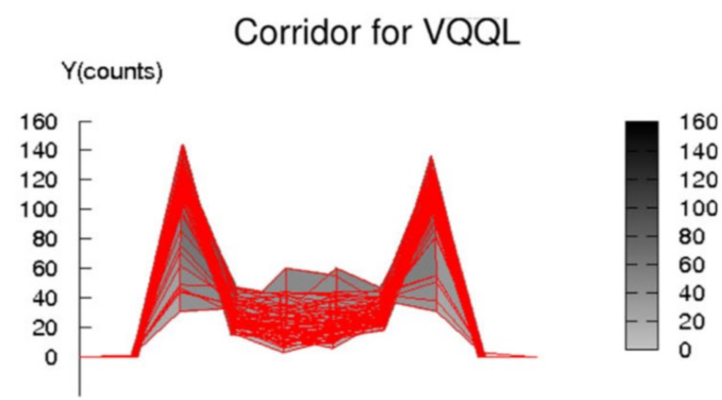




Simulation results - corridor

- Macro-dynamics:





Simulation results - corridor

- Performance:

Table 14 Mean of the number of episodes that end successfully from a series of 100 episodes

TF_IT	TF_IN	VQQL	*P* value
81 *a*	52 *b*	63 *c*	0.0000

A successful episode means that all the agents reach to the correspondent goal. Ten series have been carried out. The data was analyzed with an ANOVA test. It proved that the means are significantly different ($P \leq 0.05$). The letters classify the different groups according to the Duncan's multiple range test

Simulation results - corridor

https://www.uv.es/agentes/RL/crossingcorridor_iterative.htm

Simulation results - other

https://www.uv.es/agentes/RL/shortvsquick.htm
https://www.uv.es/agentes/RL/crossing_sarsa.htm
https://www.uv.es/agentes/RL/maze.htm

Conclusion

- MARL provides several advantages for pedestrian simulation:
  1. Independent learning with unique behaviors
  2. Offline learning and low computation execution
  3. Avoids hand-coded domain knowledge
  4. Allows for incorporating external knowledge through knowledge transfer techniques

Conclusion

- Successfully addressed goals:
  - Demonstrated that VQQL, ITVQQL, and INVQQL are convergent in two pedestrian scenarios
  - Learned basic rules of pedestrian dynamics (confirmed from micro and macro perspective)
  - Learned behaviors scale robustly in the first scenario
  - Similarity to Helbing model supports behavior plausibility

Conclusion

- Future work:
  - Complex environments
  - More realistic physics interactions (friction)
  - Increased detail in the agent physical representation
  - Additional state space generalization methods (tile coding)