



The

Partially Observable Travelling Officer

Problem



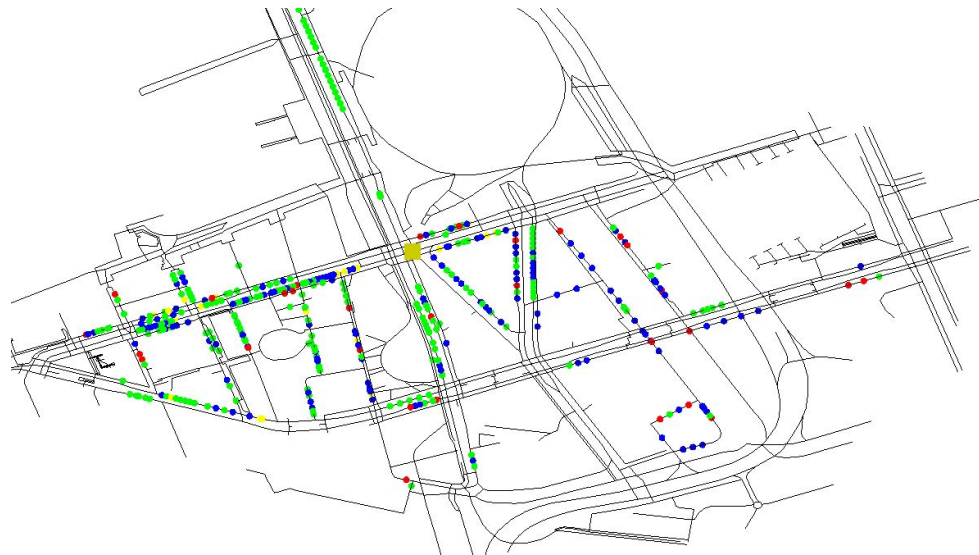
Hasan Turalic
Nikolas Gritsch
Oliver Schrüfer
Philipp Althaus
Selen Erkan

Practical Big Data Science
Final presentation, 2021-07-21

Supervisor: Niklas Strauss

Outline

1. Problem
2. Environment
3. Baseline agents
 - a. Random
 - b. Greedy
 - c. ACO
4. Advanced agents
 - a. PPO
 - b. DDQN (independent and shared policy)
5. Results
6. Lessons learned
7. Future work



Partially Observable Travelling Officer Problem

- What is POTOP?
- Research questions:
 - Single vs multi agent
 - Full vs partial observability
- Challenges
 - Dynamic environment
 - Semi MDP
 - POMDP



The Simulation Environment

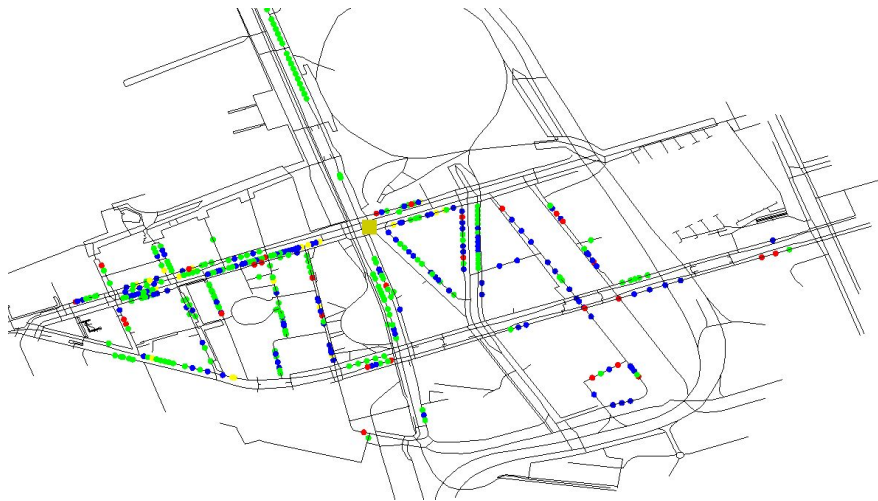


Real-World Data

- Melbourne 2017 Dataset
 - Event based architecture
 - Train - Validation - Test split
- Open Street Maps Graph

OpenAI Gym Compatible Environment:

- State space
 - Parking spots and their information
- Action space
 - All edges that contain parking spots



The Simulation Environment: States



- State is passed from environment to agents as a numpy array:

One hot encoding of the spot				Walking time	current time	Time of arrival of agent	Indicator for violation
Free	Occupied	In Violation	Fined	in h	[0;1]	[0;1]	[-1;2]
1	0	0	0	0.21	0.6	0.7	-1
0	0	1	0	0.16	0.6	0.65	1.2
...

Extensions				
Max parking time for spot	Parking spot booking			
[0;1]	Agent 1	Agent 2	Agent 3	Agent 4
0.166	1	0	0	0
0.083	0	1	0	0
...

Baseline Agents



1. Random
 - a. Random edge
 - action space: adjacent edges of the street network
 - b. Random route
 - action space: all edges in the graph
 - follows shortest path to chosen edge
2. Greedy
 - pick the node with the minimum total violation time
$$\min(\text{total violation time} + \text{walking time})$$
 - in no violation case pick the closest node
3. Ant colony optimization
 - every ant finds a path using probabilities assigned to the nodes
 - pick the best path
 - return the next node of the best path

Advanced Agents



- **Single agent**
 - PPO
 - DDQN
- **Multiple agents**
 - DDQN, PPO: Independent Q-Learning
 - DDQN: Shared policy
 - COMA
 - Q-MIX
- **Extensions**
 - Prioritized experience replay memory
 - Reward/gradient clipping
 - Using additional columns
 - Partial observability...



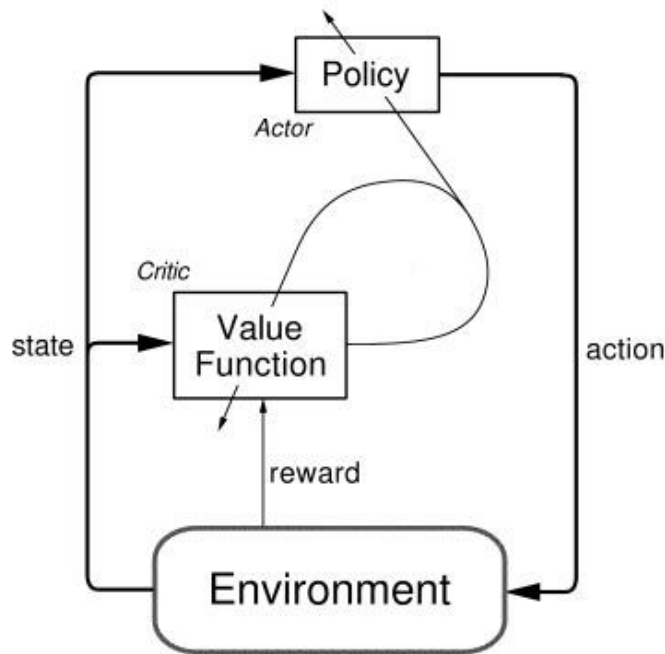
Advanced Agents: PPO



Proximal Policy Optimization:

Policy gradient method (online learning)

1. OpenAI (2017), balances:
 - a. sample efficiency,
 - b. ease of implementation
 - c. ease of tuning
2. Improves over costly trust region policy optimization
3. Advantage function predicts future reward in given state
4. Weight updates are clipped



Advanced Agents: DDQN



Double Deep-Q-Network:

Function approximation for Q-learning

1. **Distance module**

Input: precomputed distances

2. **Resource module**

Input: state

3. **Final layers**

Output: predicted Q-values

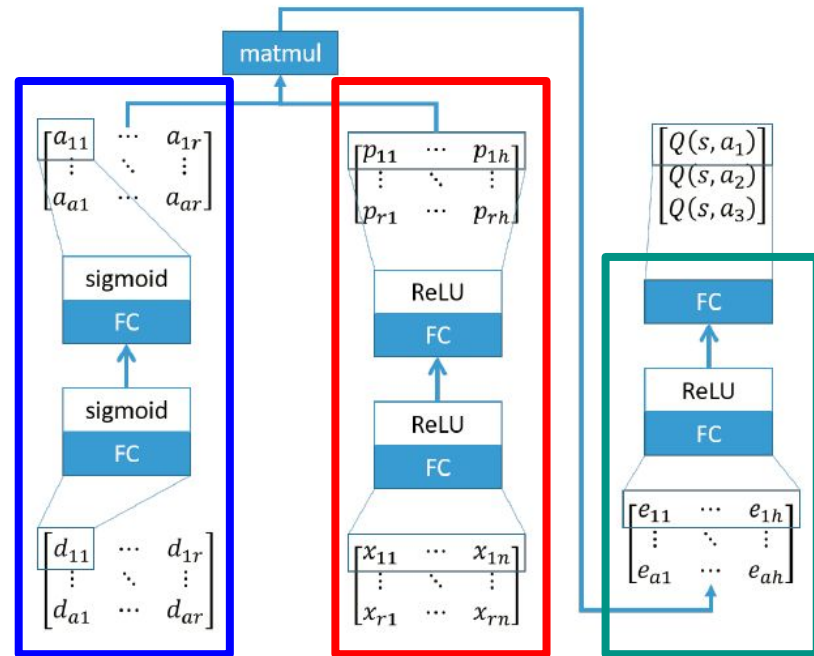


Fig. 1: Architecture by Schroll & Schubert (2020)

Advanced Agents: Multi-Agent Setting



1. **Multi-agent approaches for DDQN**
 - a. **Independent Q-learning**
separate network for each agent
 - b. **Shared policy**
Shared weights + memory
-> converges faster
2. **Coma**: multi-agent policy gradient method with centralized critic, but decentralized actors
3. **Q-mix**: extension for DDQN



Training Process

Single-Agent

- Easiest task
 - ➡ most time spent
- Finding bugs
- Finding optimal hyperparameters

Multi-Agent

- Easy transition after few adjustments

Multi-Agent with Partial Observability

- Huge jump in complexity
- Experiments started

Training Process

Single-Agent

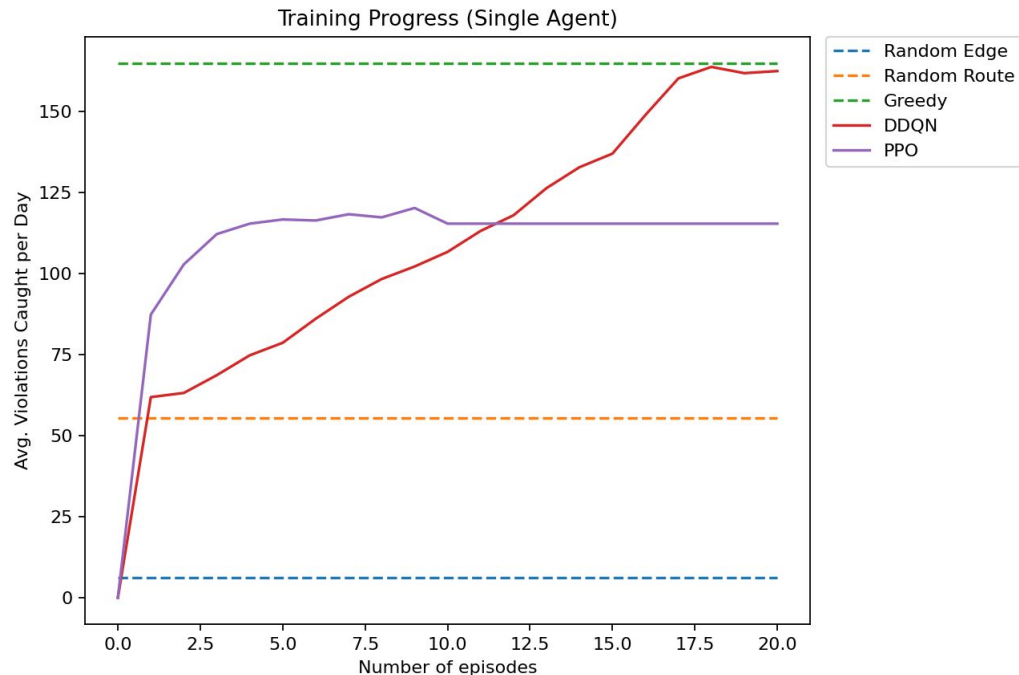
- Easiest task
 - ➡ most time spent
- Finding bugs
- Finding optimal hyperparameters

Multi-Agent

- Easy transition after few adjustments

Multi-Agent with Partial Observability

- Huge jump in complexity
- Experiments started





Results

Evaluation Metric: Average violations caught per day

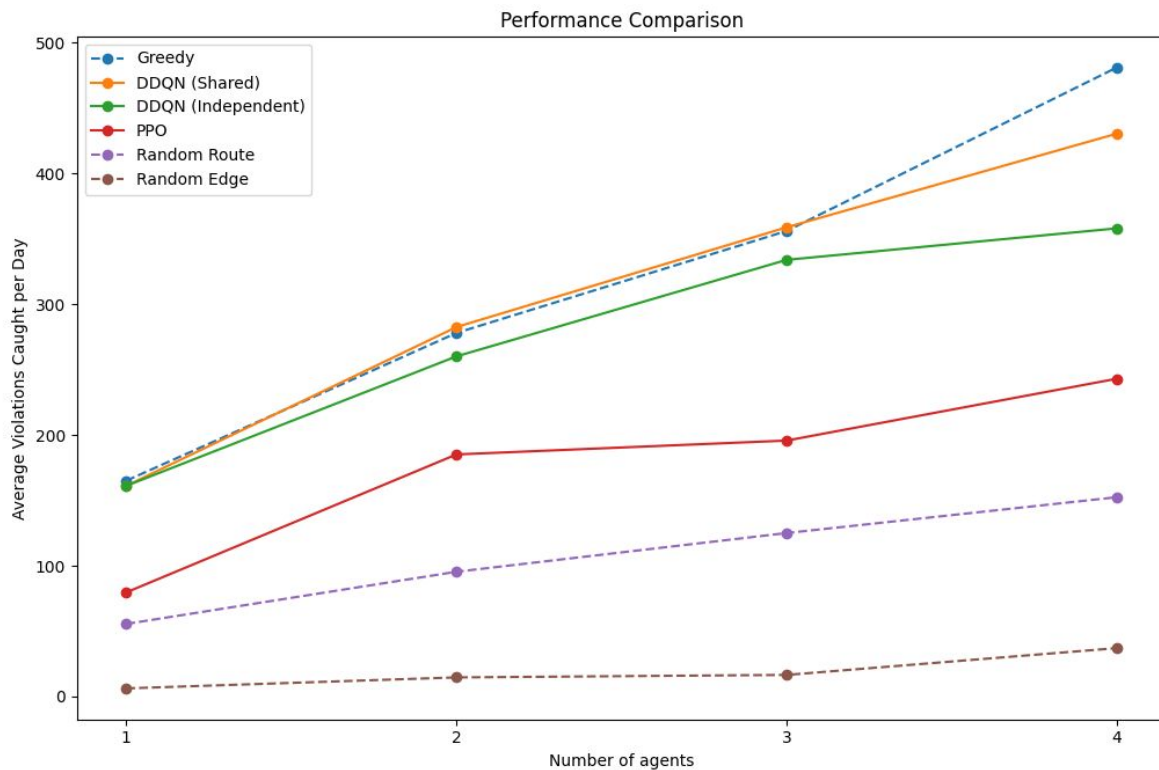
	Number of Agents			
Algorithm	1	2	3	4
Random Edge	6.2			
Random Route	55.5			
Greedy	165.9			
PPO (independent)	79.6			
DDQN (independent)	161.1			
DDQN (shared policy)				

Results

Evaluation Metric: Average violations caught per day

	Number of Agents			
Algorithm	1	2	3	4
Random Edge	6.2	14.6	16.5	37.0
Random Route	55.5	95.4	125.0	152.4
Greedy	165.9	278.4	356.1	481
PPO (independent)	79.6	187.8	195.7	 243.0
DDQN (independent)	161.1	260.1	333.9	358.0
DDQN (shared policy)		282.5	368.8	 430.4

Results



Lessons Learned



- The environment must be **reliable** and **efficient**
 - write unit test
 - use profilers
 - use numpy over plain python objects/pandas dataframes as much as possible
- The NNs are very sensitive to **input encoding**
 - Test out different encodings
 - Normalize inputs
 - Test on reduced input matrices and see if performance actually decreases
- Visualizations can help to debug your program
 - For remote runs: record frames and convert to video
- Great workflow with ray-tune and ml-flow

Future Work



- Improve existing agents
 - deeper evaluate the current agents
 - try different input matrices
 - try different pre processing
- Improve agents for multi agent setting
 - QMix and Coma
 - Learning Communication between agents
- Improve agents for partial observability
 - Recurrent neural network (LSTM)

Summary



- Created an environment based on real world data to train and test our agents
- Implemented 5 different agents to solve the POTO-Problem
 - 3 conventional (random, greedy, aco) 2 reinforcement learning (PPO, DDQN)
- Extended agents and environment
 - handle multiple agents
 - partial observability
- Improved agents
 - random edge -> random route
 - prioritized replay memory
 - epsilon greedy exploration (epsilon decay)
 - improved input matrix (normalized, extra columns)
 - independent multi agents -> single shared policy

Thank you!

Questions?



Hasan Turalic
Nikolas Gritsch
Oliver Schrüfer
Philipp Althaus
Selen Erkan
Supervisor: Niklas Strauss

Backup: Graph Size

Graph

- District: **Docklands**
- # nodes: 2911
- # edges: 6025
- # parking spots (state space): 531
- # edges with spots (action space): 188

Events

- # events (training): 4.6m
- # violations (training): 242.000
- avg. violations per day: 771