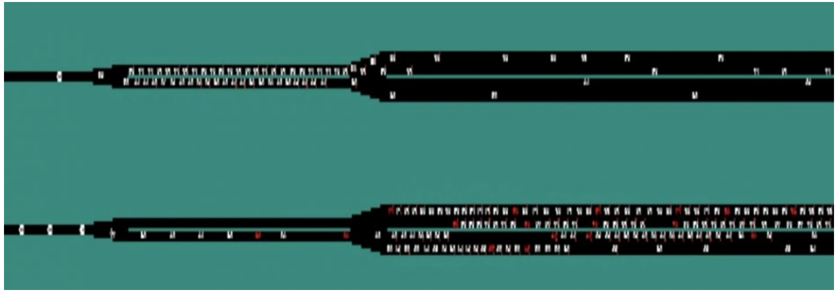
Traffic Redirection

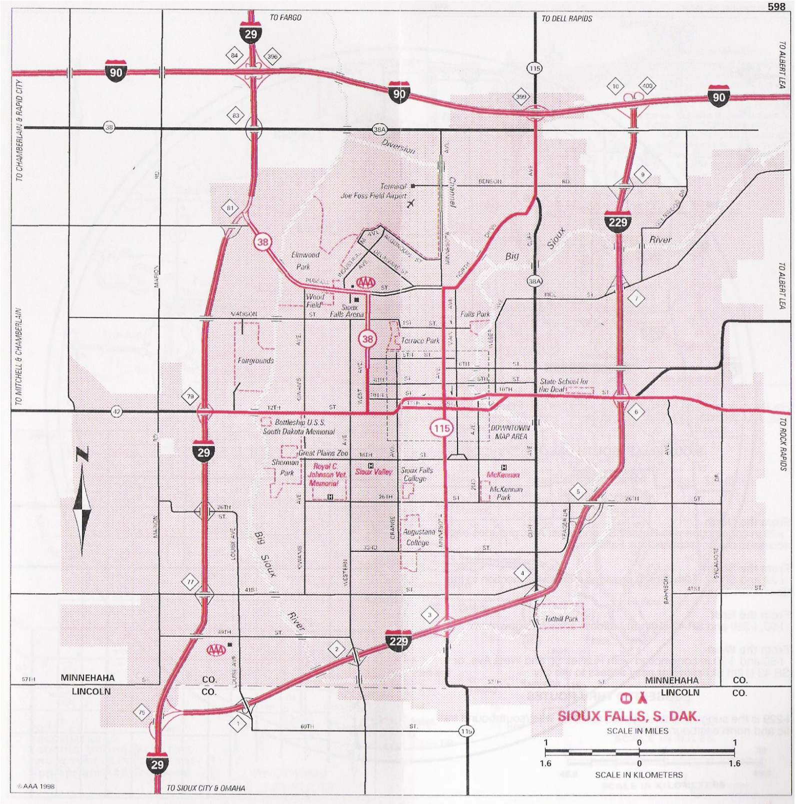
Marsalis, Theophile, Daniel, Vikrant

Outline

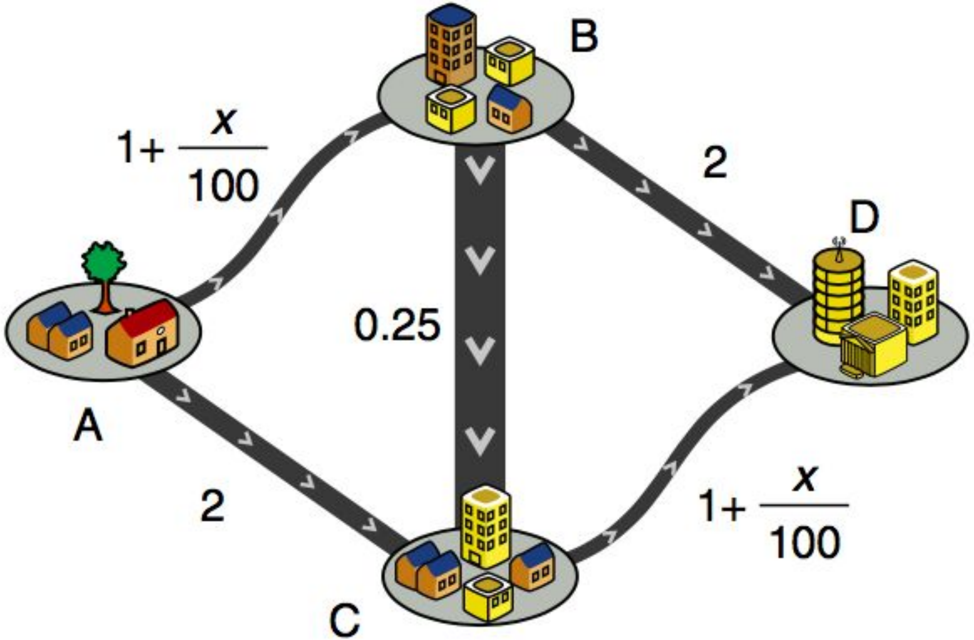
- Recap of the project
- Implementation issues
- Results
- Next steps

How to optimize traffic at the macroscopic scale?





Routing choices have the biggest impact on traffic



Can we learn to optimize the routing game?

- *Multiagent reinforcement learning in the Iterated Prisoner's Dilemma,* Tuamos Sandholm, 1996
- *Learning dynamics in social dilemmas,* Michal W. Macy, 2002
- *Iterated Prisoners Dilemma with Reinforcement Learning*, Keven Wang, 2018
- *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*, Ryan Lowe, 2018
- *Towards Cooperation in Sequential Prisoner's Dilemmas: a Deep Multiagent Reinforcement Learning Approach,* Weixun Wang, 2018

Challenges / considerations / issues

- Defining the action and state spaces
- Using different observation spaces. Initially a box(3) where every car knows time taken in every path. Then we decided to change it to box(n x 2) where n is the number of cars, each car observing the path it chose and the time it took in that path.

Software Implementation

- **Network** - Created as a python class
- **Renderer** - Created in JavaScript to visualize paths taken by vehicles
- **Environment** - Multi agent Environment created on OpenAI Gym
- **Training** - Using Ray and RLLib

RLlib: Scalable Reinforcement Learning

Implementing the game using Ray

States: {Your previous action and reward, best previous action and reward}

Actions: {Choice of path when entering the system, (broadcast message to others)}

Rewards: 1. Minimise self travel time
2. Minimize self marginal cost
3. Minimize overall travel time

Discount factor: 0, 0.5, 1

The parameters that we can change

The discount factor gamma

The social factor lambda: Reward =
$$c_e^{\lambda}(x_e) = (1-\lambda)t_e(x_e) + \lambda \frac{d[x_e t_e(x_e)]}{dx_e}$$
$$= t_e(x_e) + \lambda x_e \frac{dt_e(x_e)}{dx_e}$$

The state observation (full observed, partially observed)

The learning algorithm (DQN, PPO)

The type of communication allowed

A simple model without RL

Transition function: p chances to stay on previous path, (1-p) chances to go on the action path

p = (marginal regret) / cost = (tt_p - tt_p*) / (tt_p), $0<p<1$

Gamma = 0

A simple model without RL

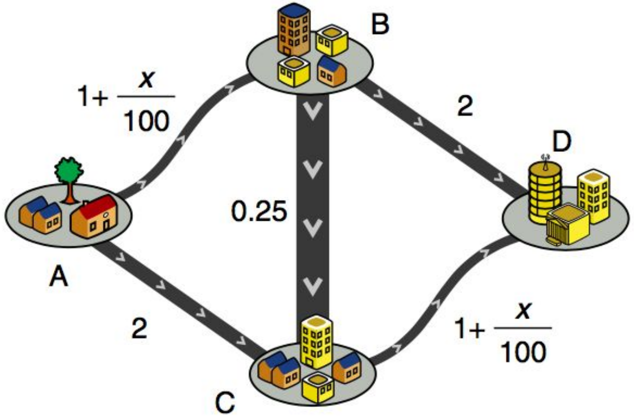
It converges:

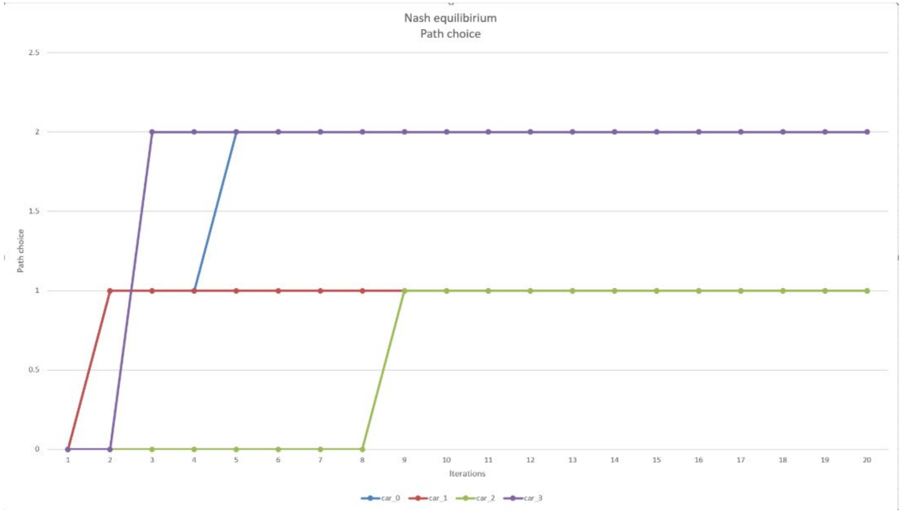
In theory: Nash equilibrium will not change, if state is not Nash, there is a probability to get Nash for the next state

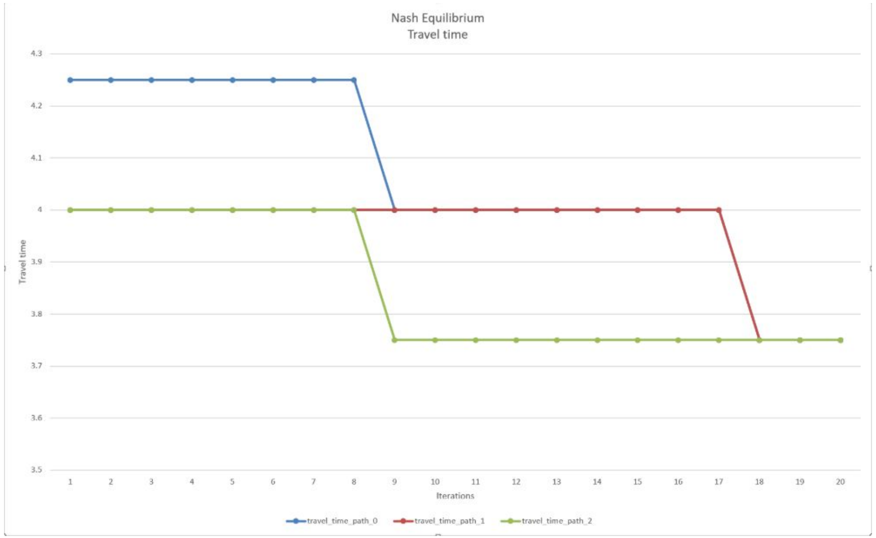
In practice: it converges after 5 to 10 iterations!

A simple model without RL

Learning the Nash equilibrium

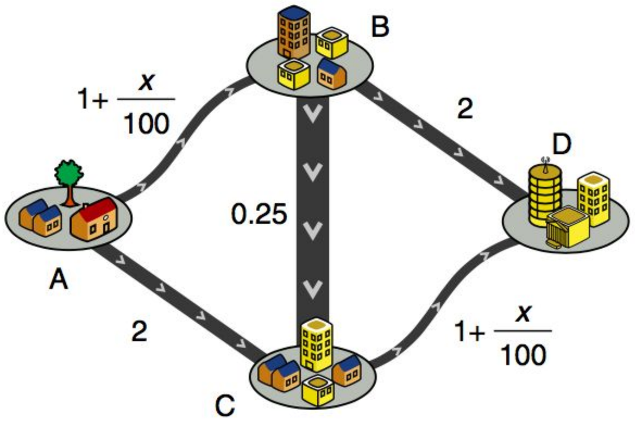


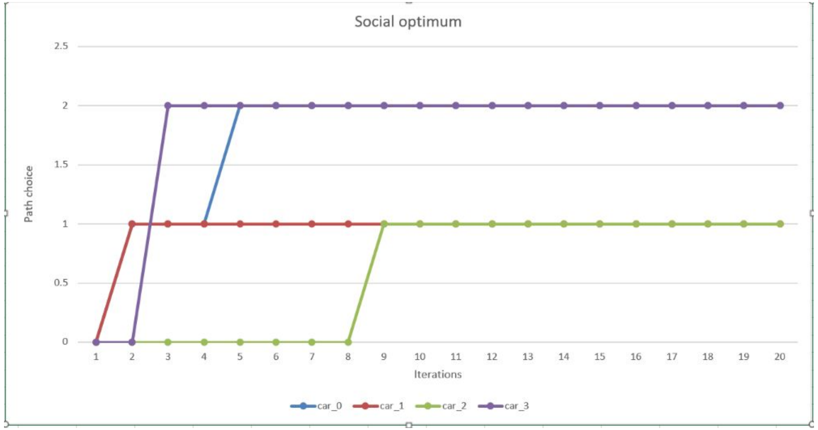


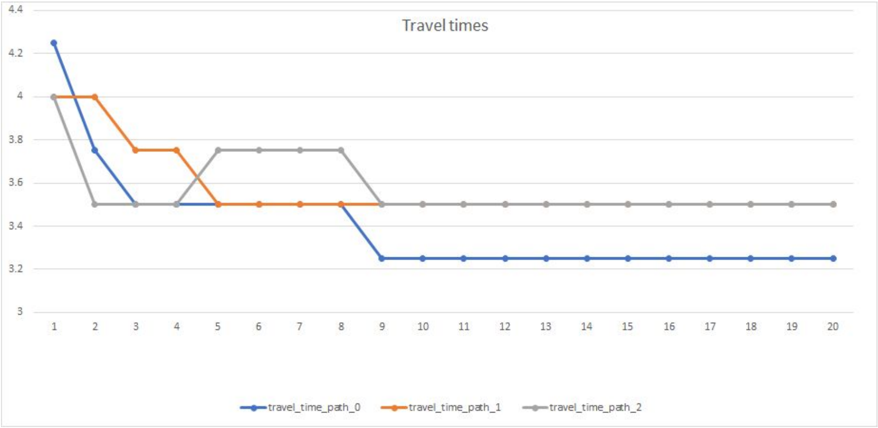


A simple model without RL

Learning the social optimum







A more complicated model without RL

Online learning (cf slides of the class lecture):

Chose the path which would have minimize your regret

$$R_k^{(t)} = \sup_{x_k \in \Delta^{\mathcal{A}_k}} \sum_{\tau \leq t} \left\langle x_k^{(t)} - x_k, \ell_k(x^{(t)}) \right\rangle$$

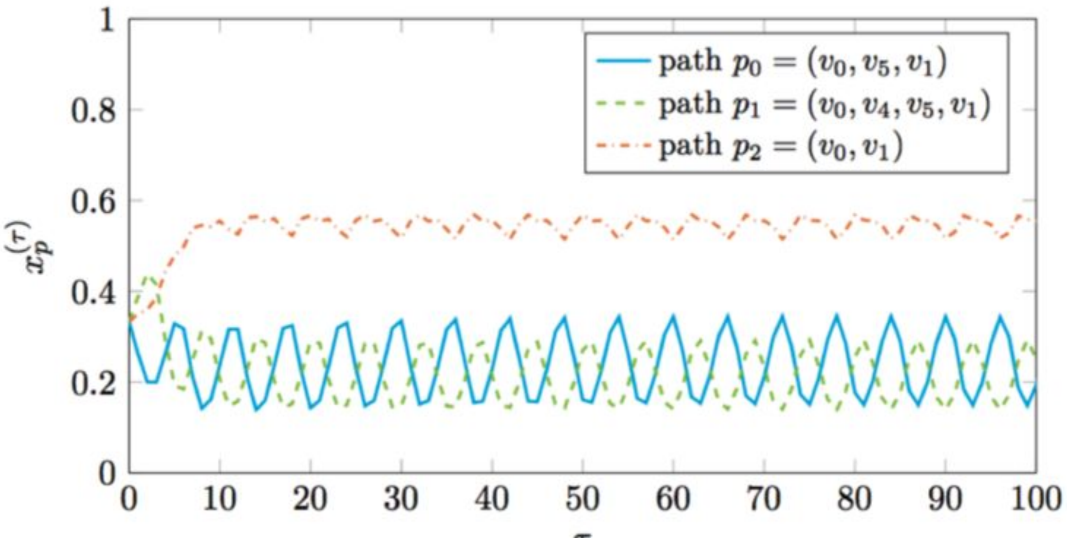
It converges:

In theory, in average

A more complicated model without RL

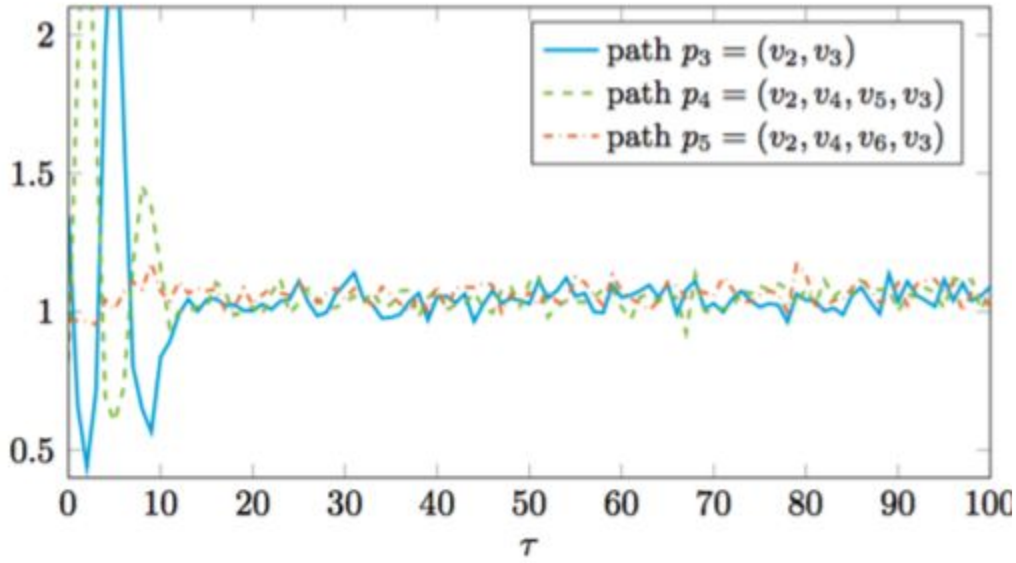
It converges:

In practice, in average



A more complicated model without RL

If we penalize changing to often your path (adding a Bremian divergence), it will converge to Nash



What we can expect from RL ?

Convergence in average when gamma = 1 (online learning)

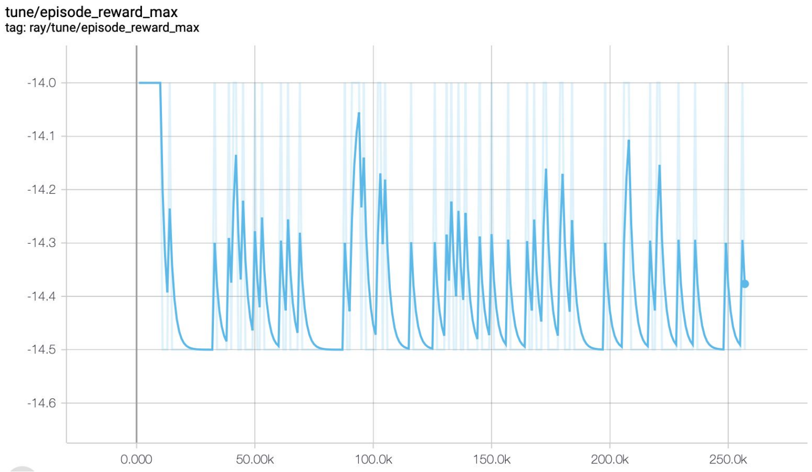
Convergence when gamma = 0

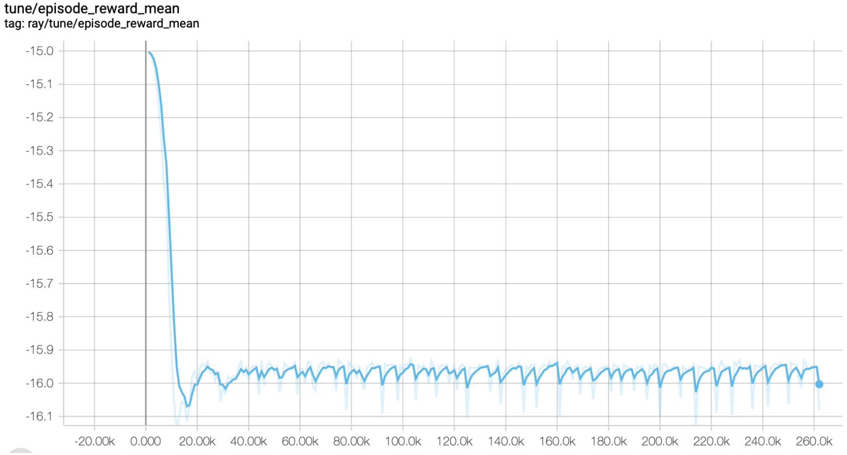
It is highly depend on the learning algorithm used :-(

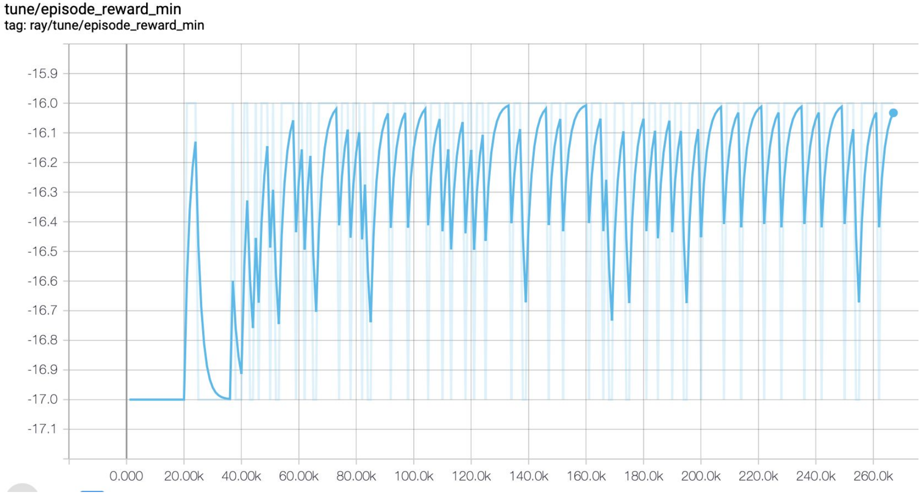
DQN = What does it learn ?

It only learns to avoid the worst case.

Reward max, mean and min

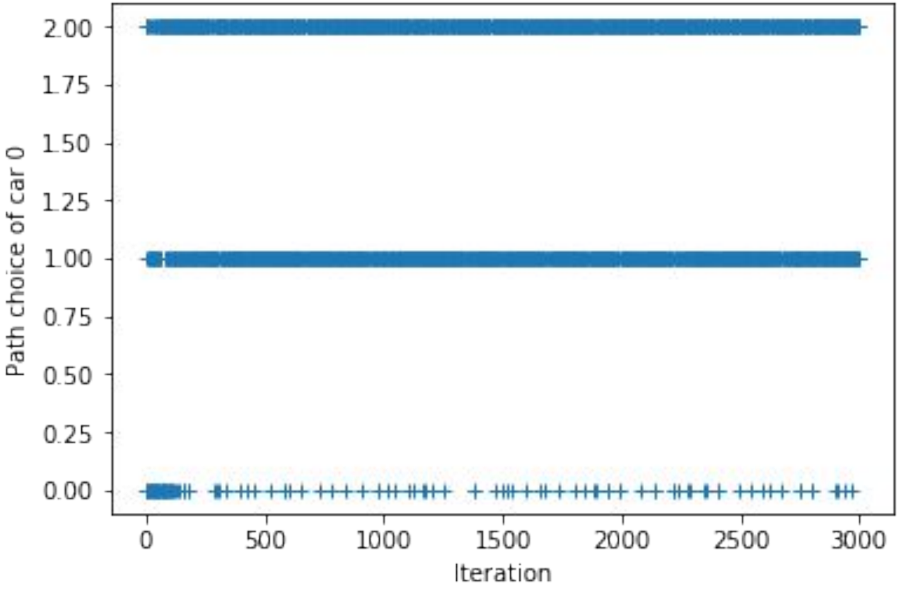


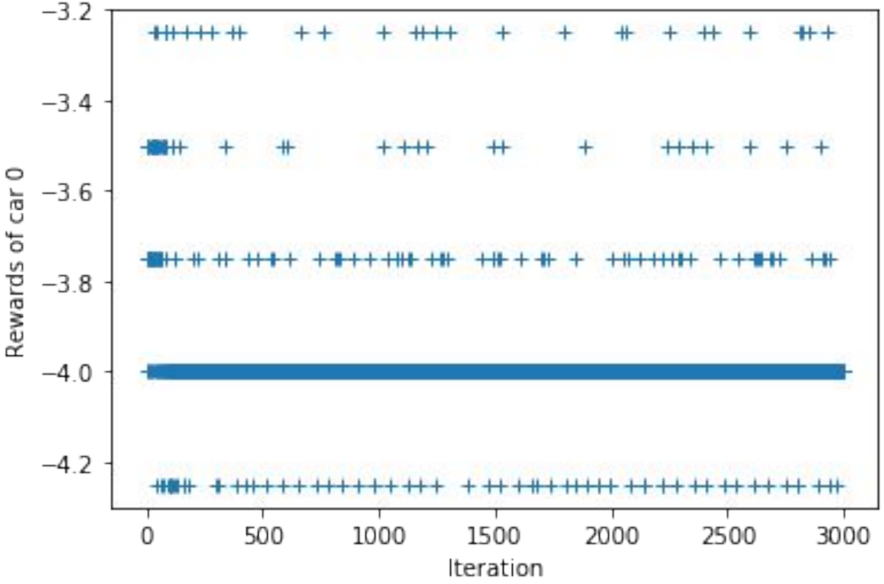




DQN = Avoiding the worst case

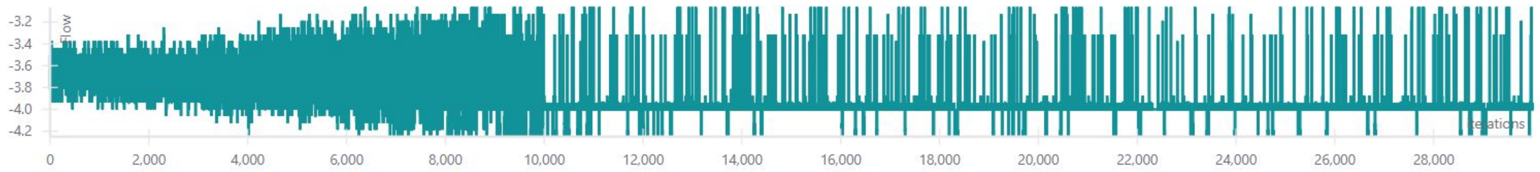
Does not depend on gamma when 300,000 repeated iteration (100 learning iteration)



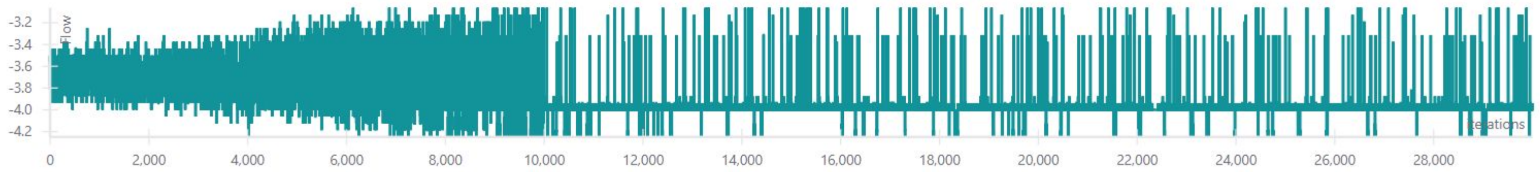


DQN: In average the reward is not Nash (it is worth!)

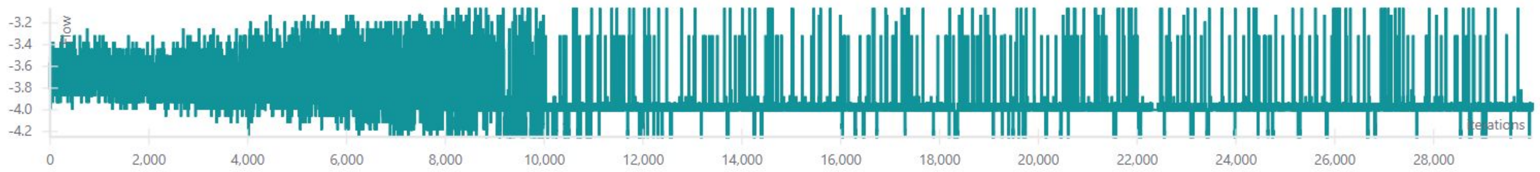
car_6



car_7

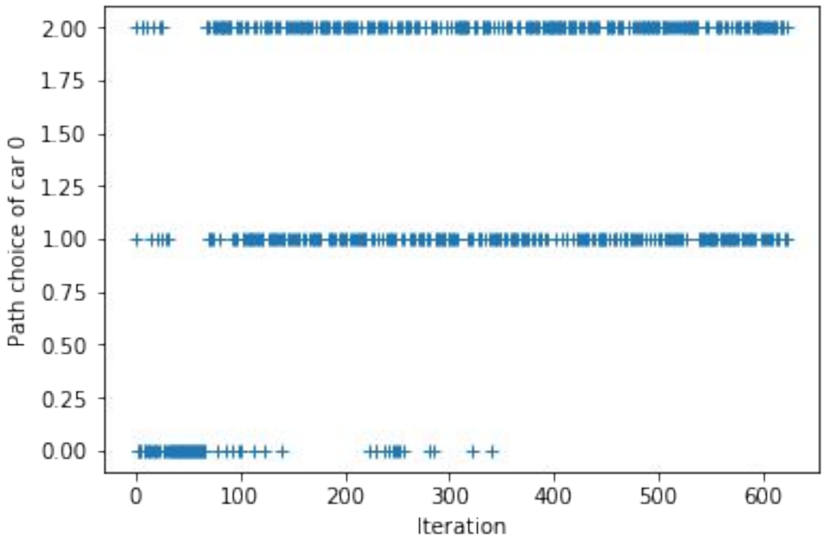


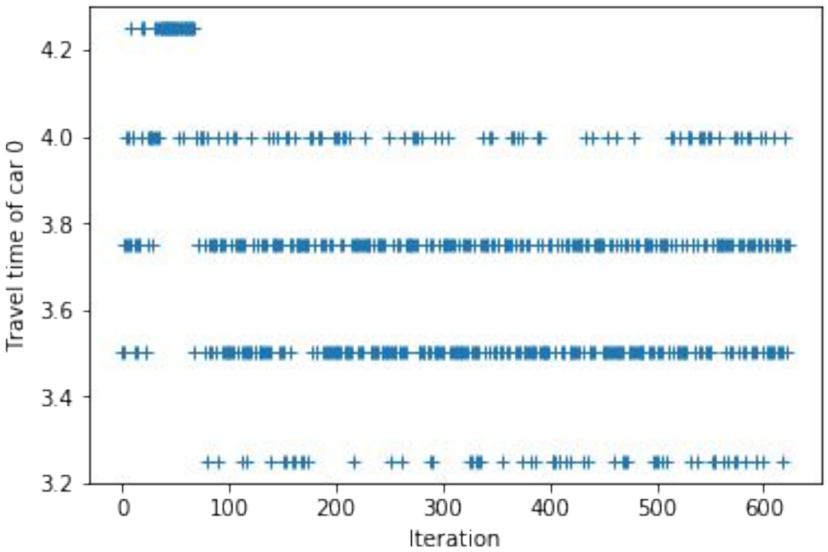
car_8



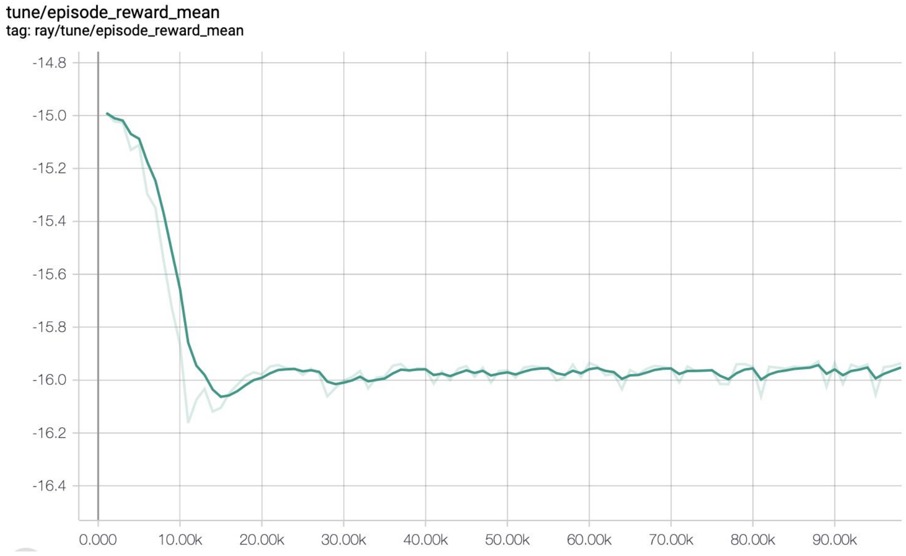
PPO = it converges!

When we want to minimize the overall travel time

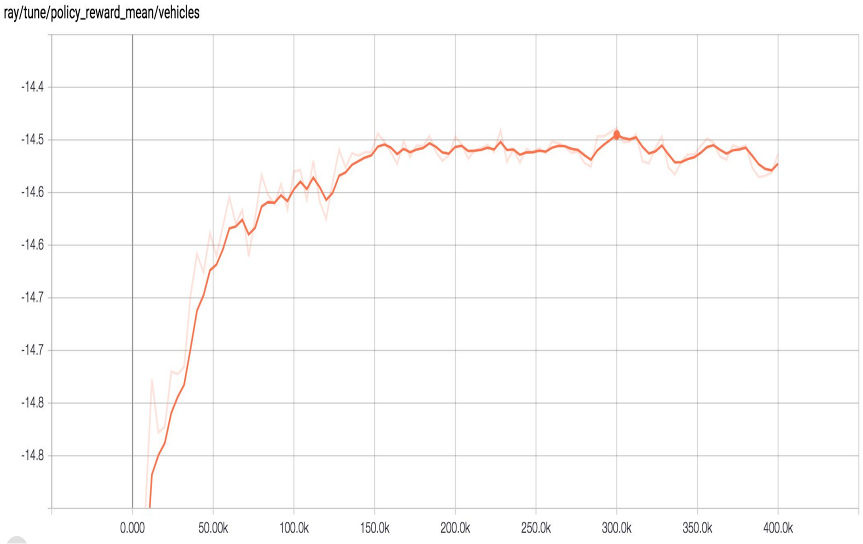




The differences between DQN and PPO

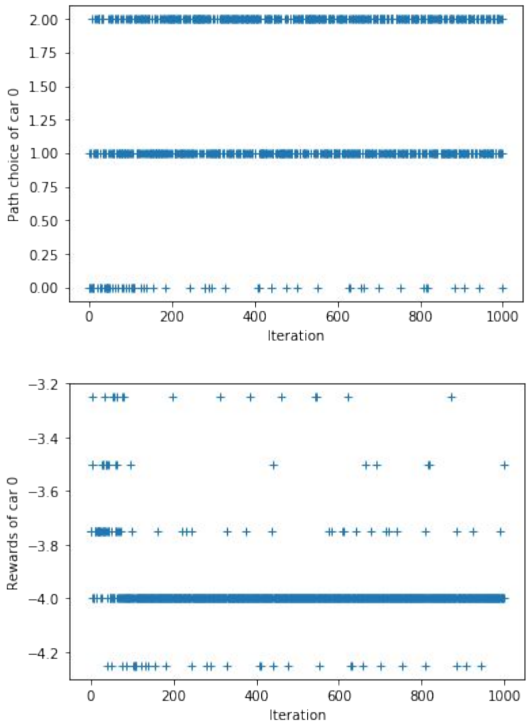


DQN: avoiding the worst case

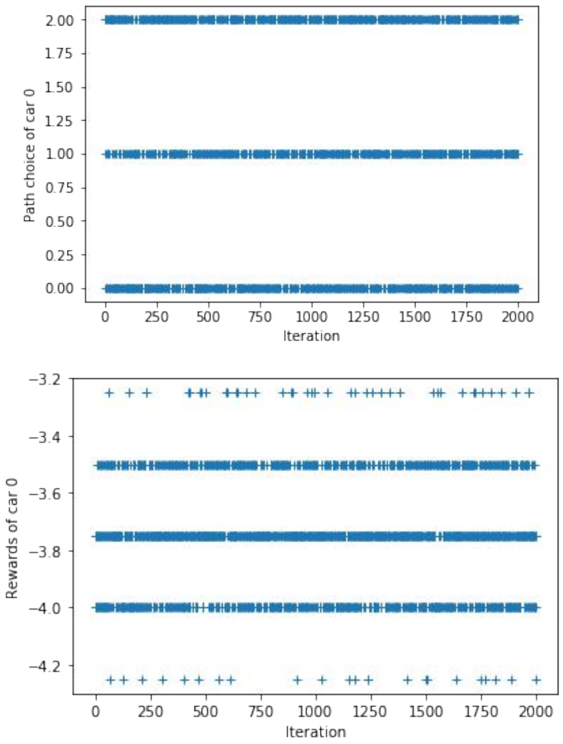


PPO: clever! Learn

PPO / DQN: gamma = 0.5, lambda = 0

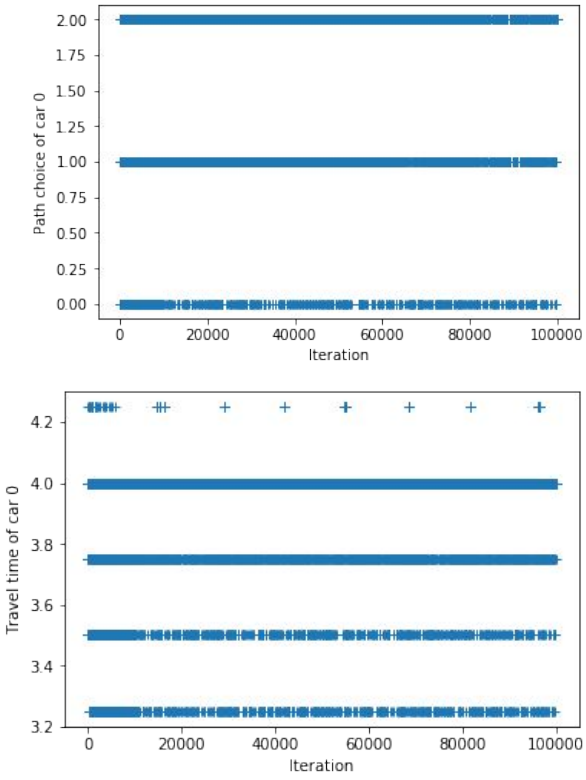


DQN: avoid the worst

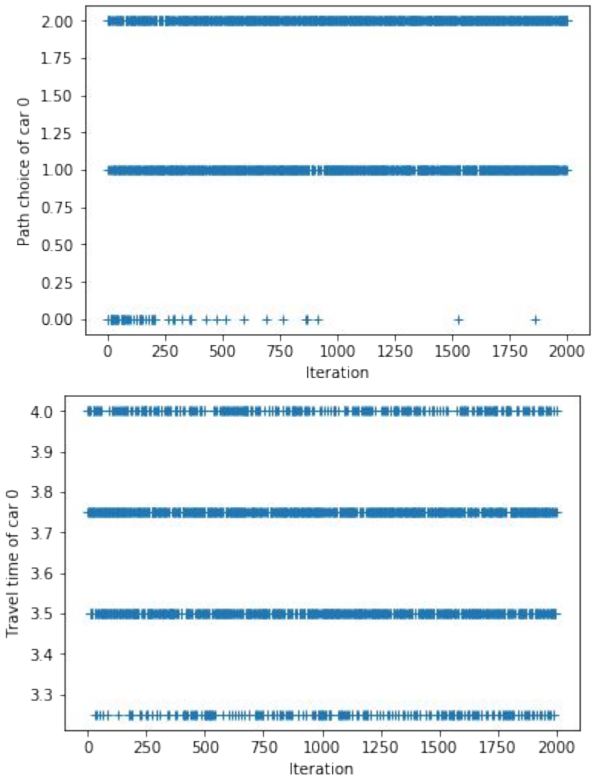


PPO: Nash in average

PPO / DQN: gamma = 0.5, lambda = 1



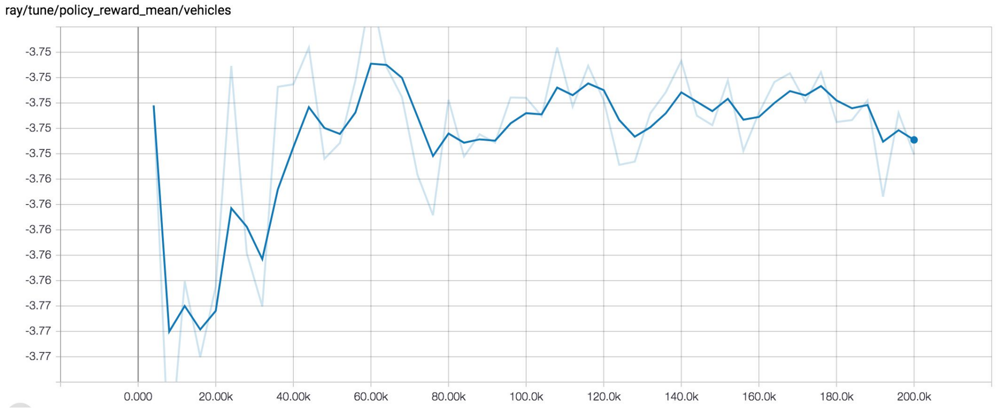
DQN: avoid the worst



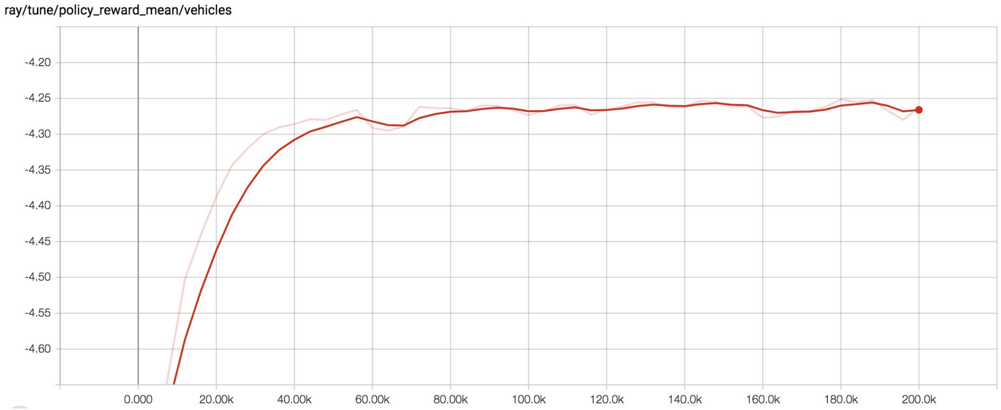
PPO: Nash in average

The convergence of PPO for lambda=0 and 1

Lambda = 0



Lambda = 1



We need to continue to deep inside

1.) State observation parameters
2.) Discount factor
3.) Social factor
4.) The learning algorithm (DQN, PPO)
5.) The communication

6.) Ray issues :-(

Conclusion

We have some really interesting results. We need to continue to work on it.