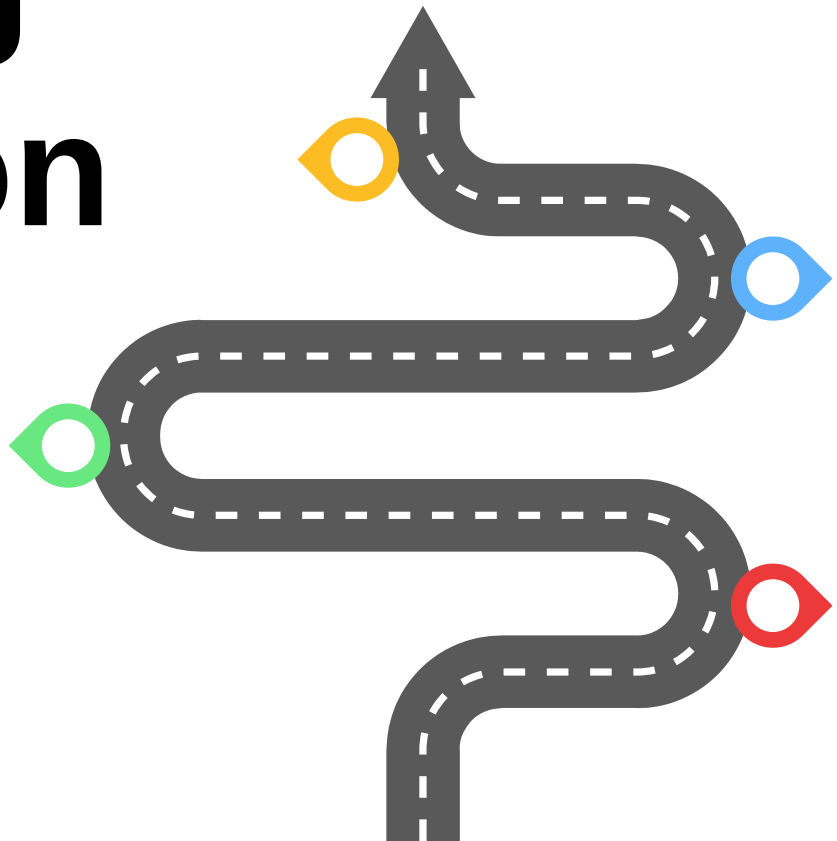
Optimizing Intersection Traffic



G. Alessio
D. Capone
S.J. Della Rovere

Our tasks

- Reduce the **waiting time** of cars
- Reduce size and amount of **queues**

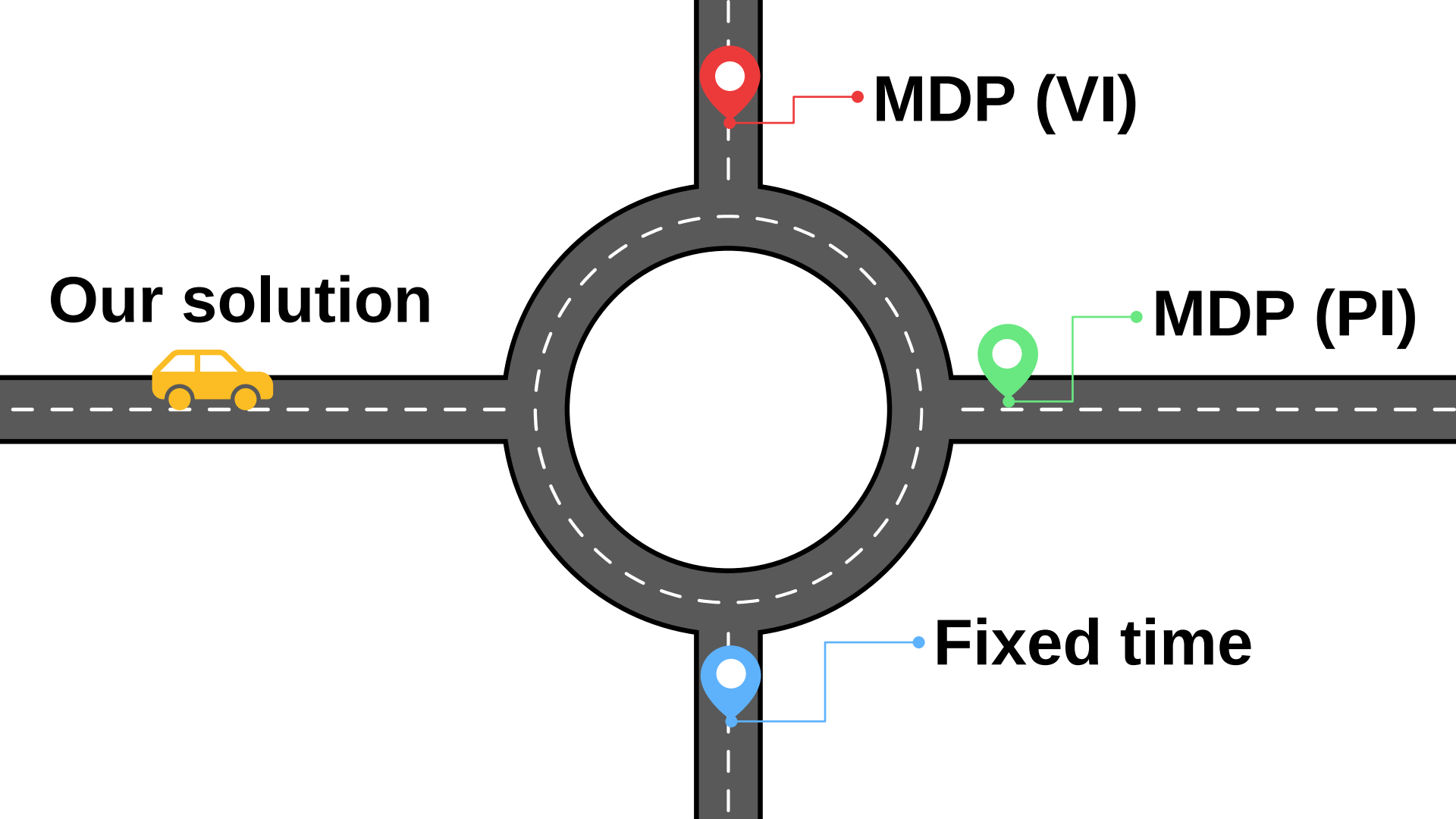
Environment

- pygame
- 2-way intersection
- User controls the frequency of car generation
- Cars can go *north*, *south*, *east* or *west*

Environment - Assumptions

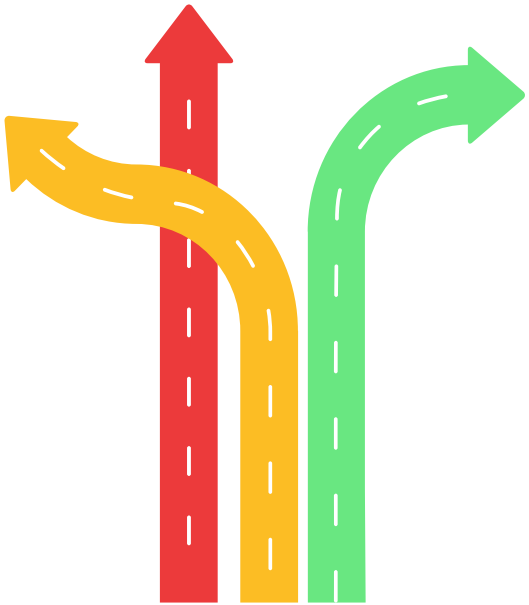
- Cars have identical *size* and *speed*

- At the intersection, cars can either go *straight* or *turn right*


MDP (VI)
Our solution
MDP (PI)
Fixed time

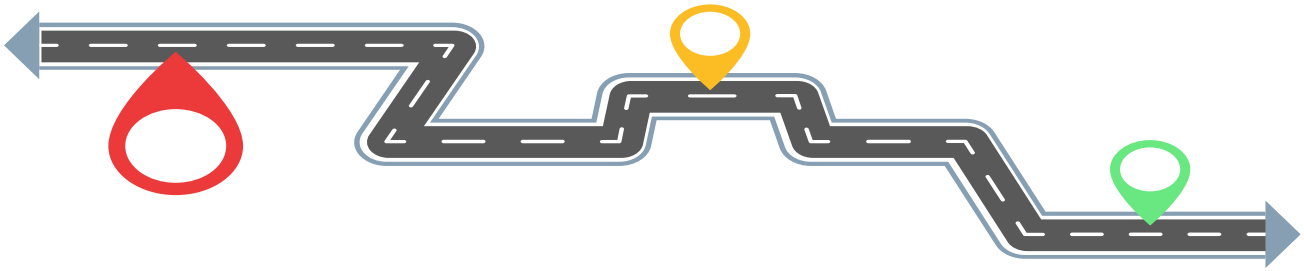
Stoplight agent - Fixed time

- Green light lasts 20 seconds
- Yellow light lasts 3 seconds
- Stoplight keeps switching with these fixed time intervals



MDP

- States: «EW», «NS»
- Actions: «Maintain», «Change»



MDP – Reward function

$$r(s,a) = \begin{cases} \text{If a} = \text{"change"}: \begin{cases} \frac{\mu_t}{n_c} & \text{if } n_c > 0 \\ \mu_t & \text{otherwise} \end{cases} \\ \\ \text{If a} = \text{"maintain"}: \begin{cases} \frac{n_c}{\mu_t} & \text{if } \mu_t > 0 \\ n_c & \text{otherwise} \end{cases} \end{cases}$$

- $\boldsymbol{\mu_t}$ is the average waiting time of stopped cars
- $\boldsymbol{n_c}$ is the number of cars transiting where the stoplight is green

MDP – Transition probability

$$p(s', r(s,a)|s,a) = \begin{cases} \text{If } a = \text{"change"}: \begin{cases} \text{If } r(s,\text{"change"}) > r(s,\text{"maintain"}) : \begin{cases} 1 \text{ if s} \neq s' \\ 0 \text{ otherwise} \end{cases} \\ 0 \text{ otherwise} \end{cases} \\ \\ \text{If } a = \text{"maintain"}: \begin{cases} \text{If } r(s,\text{"maintain"}) > r(s,\text{"change"}) : \begin{cases} 1 \text{ if s} = \text{s'} \\ 0 \text{ otherwise} \end{cases} \\ 0 \text{ otherwise} \end{cases} \end{cases}$$

MDP – Initialization

$$V(s) = 0 \ \forall \ s \in S$$

$$\pi(a|s) = 0.5 \ \forall \ a \in A, s \in S$$

MDP – Policy evaluation

Input: π policy to be evaluated,
V value function,
θ threshold,
A actions,
S states

Loop:

$\Delta \leftarrow 0$

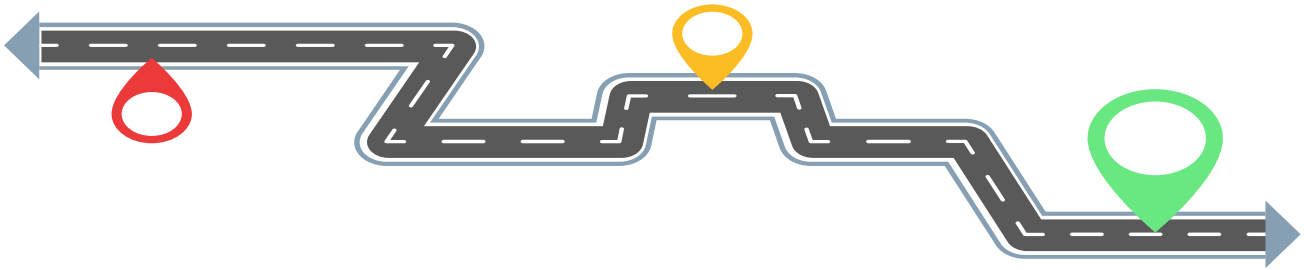
For each $s \in S$

$v \leftarrow V(s)$

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} p(s', r(s,a)|s,a)\,[r(s,a) + \gamma V(s')]$$

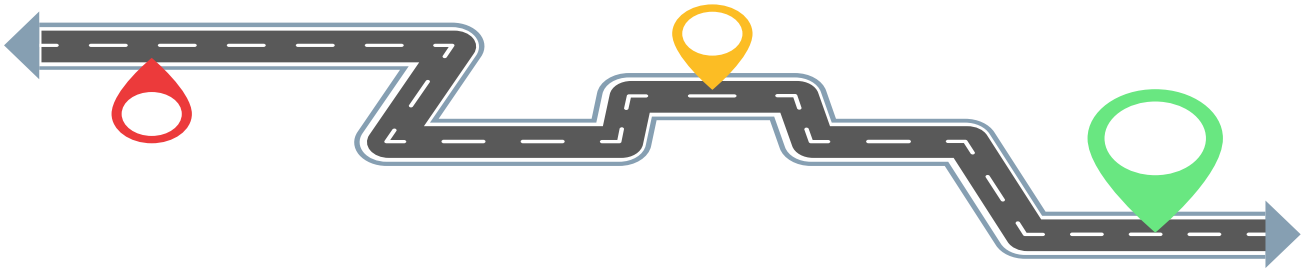
$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$



MDP – Policy improvement

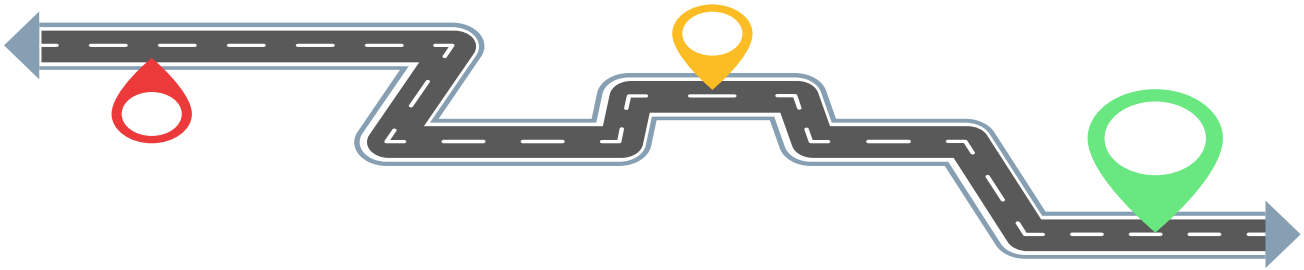
Input: π policy to be improved,
A actions,
S states
policy_stable $\leftarrow$ *true*
For each $s \in S$:
old_action $\leftarrow \pi(s)$
$\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s', r|s, a)\,[r + \gamma V(s')]$
If old_action $\neq \pi(s)$ then *policy_stable* $\leftarrow$ *false*
Return *policy_stable*



MDP – Policy iteration

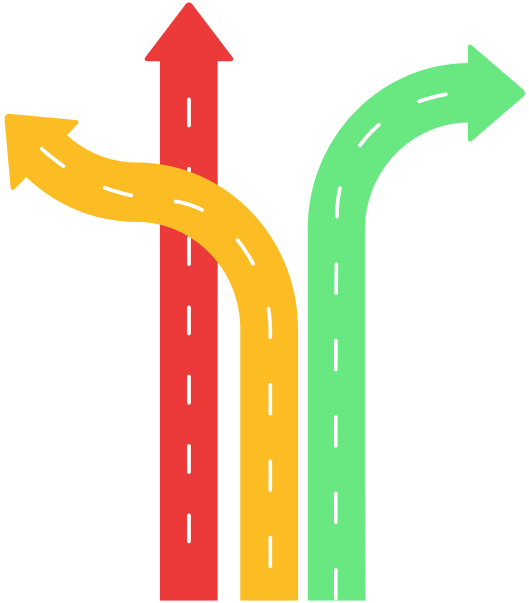
Input: π policy,
V value function,
θ threshold,
A actions,
S states

Loop:
Policy evaluation (π, V, θ, A, S)
until Policy improvement (π, A, S)



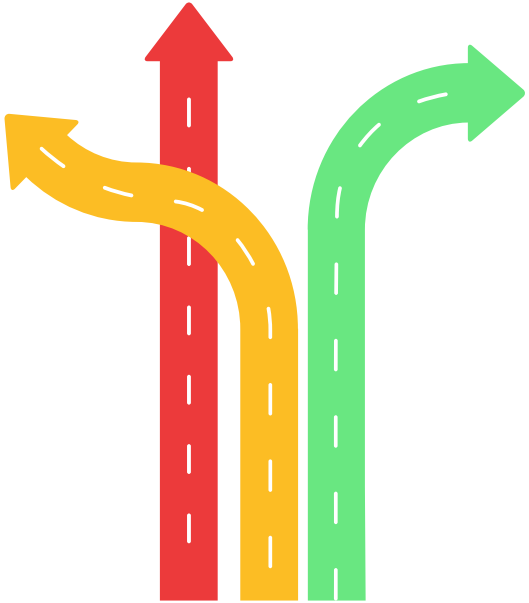
Stoplight agent – MDP (PI)

- Green light lasts at least 15 seconds
- Then, *Policy iteration* is executed
- Action $\pi(s)$ is taken



Stoplight agent – MDP (PI)

- If $\pi(s)$ = "maintain", repeat *PI* every second
- If $\pi(s)$ = "change", switch to Yellow (lasts 3 seconds)



MDP – Value iteration

Input: π policy,
V value function,
θ threshold,
A actions,
S states

Loop:

$\Delta \leftarrow 0$

For each $s \in S$

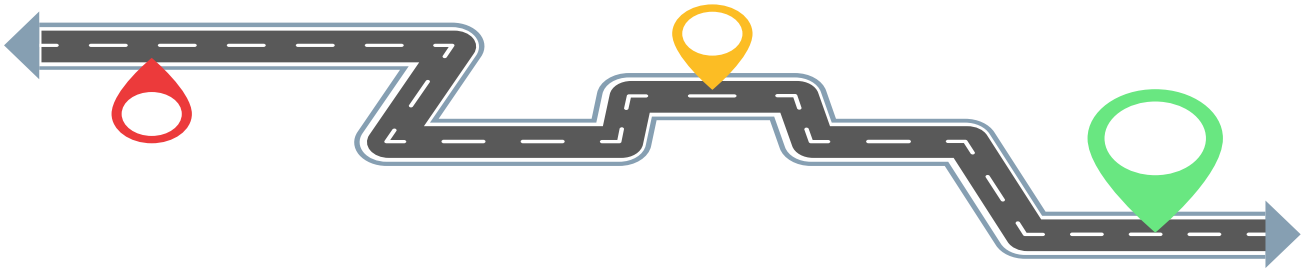
$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} p(s', r(s,a)|s,a)\,[r(s,a) + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

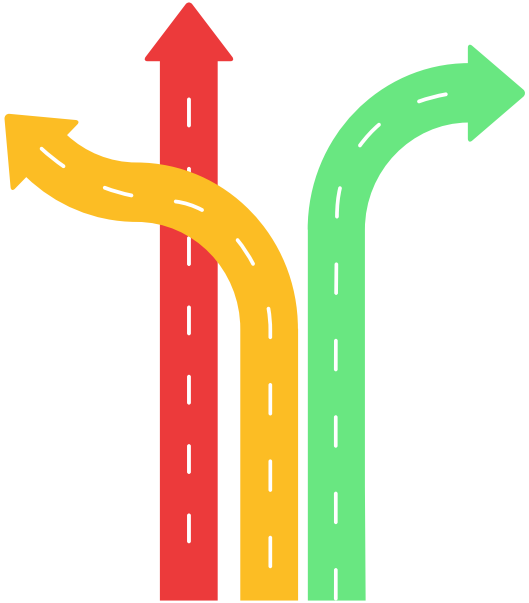
until $\Delta < \theta$

Return $\pi(s) = \text{argmax}_a \sum_{s'} p(s', r(s,a)|s,a)\,[r(s,a) + \gamma V(s')]$

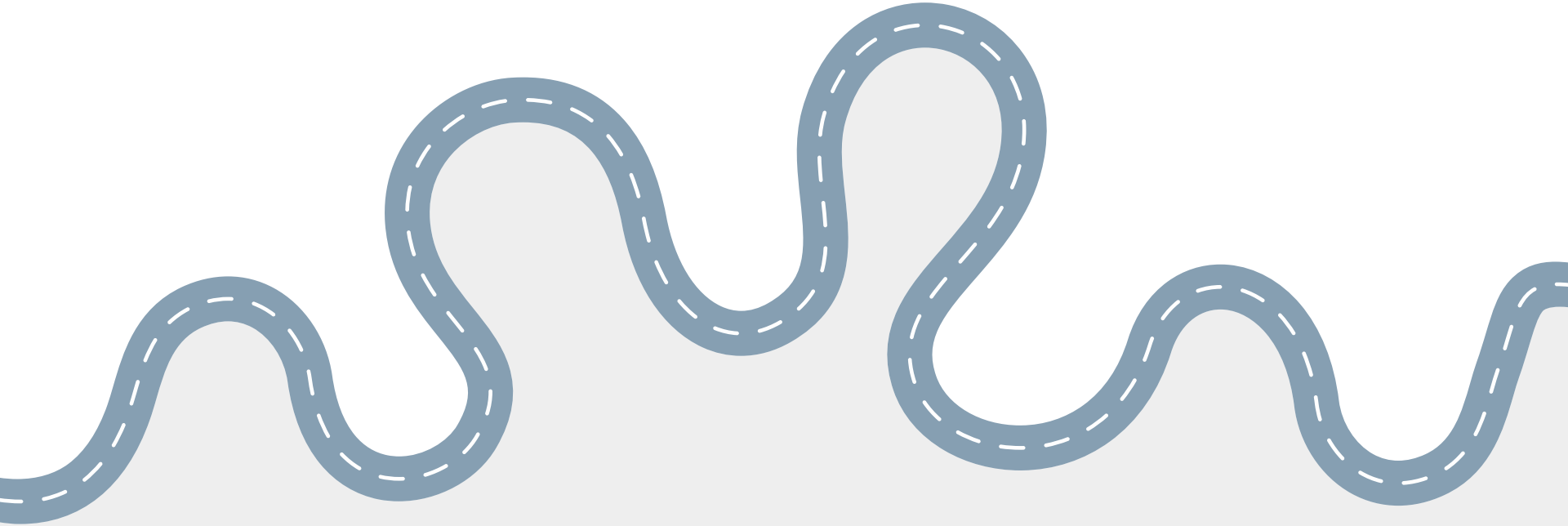


Stoplight agent – MDP (VI)

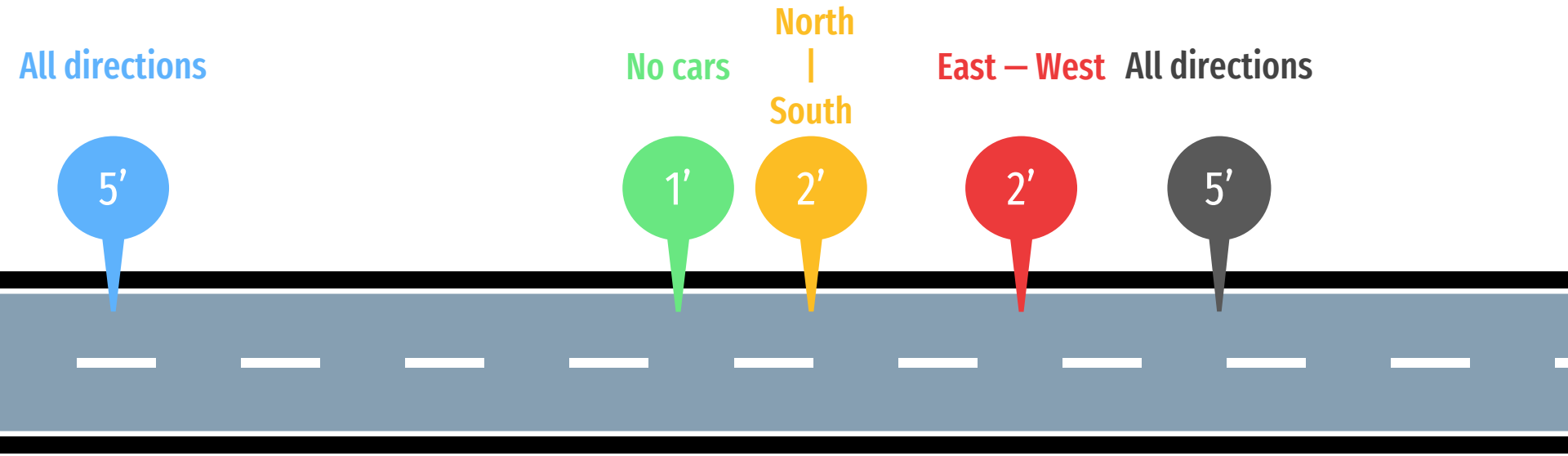
- The same as before but *Value iteration* is executed



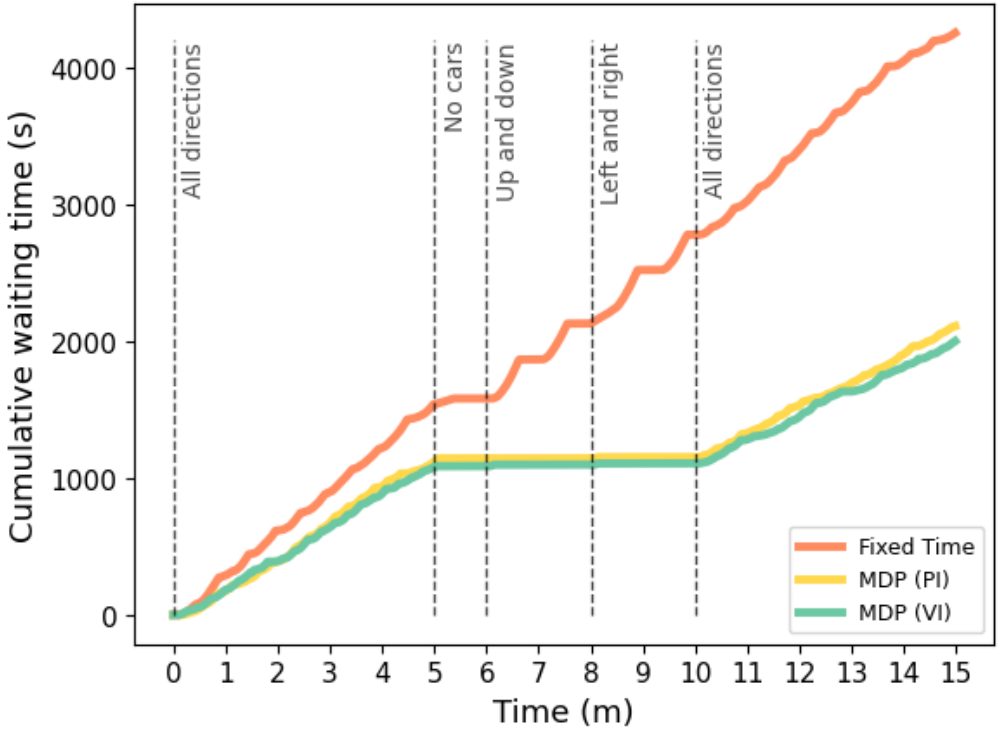
Demo

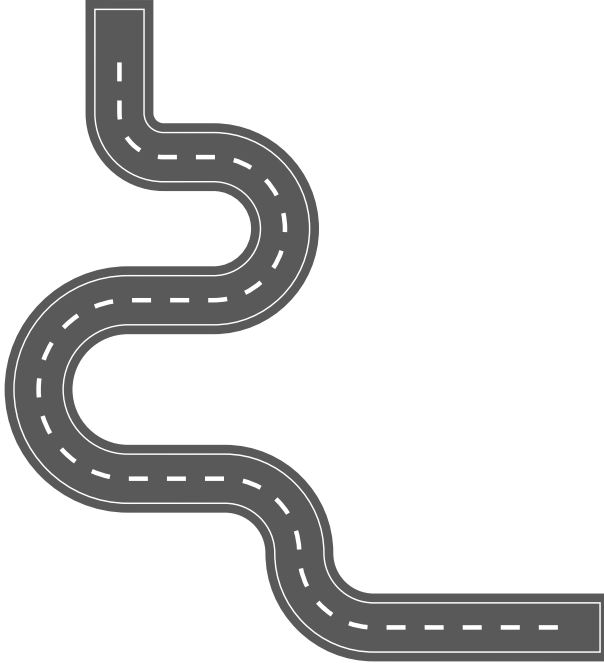


Simulation

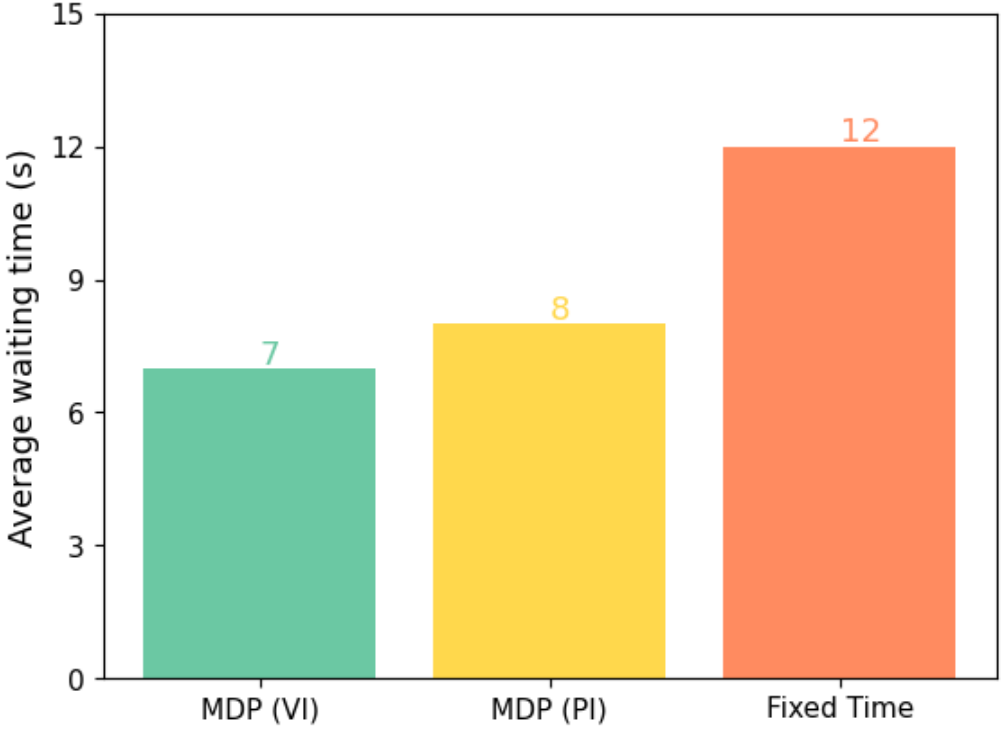

All directions
5’
No cars
1’
North
|
South
2’
East — West
2’
All directions
5’

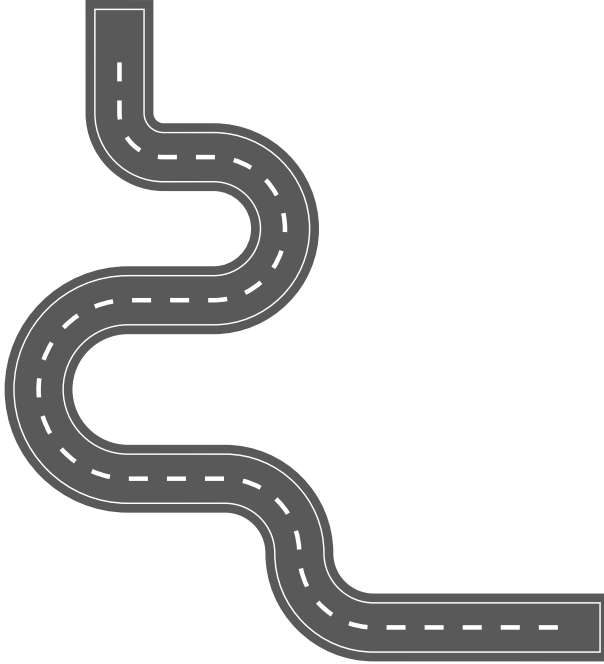
The fixed time agent makes people arrive late!



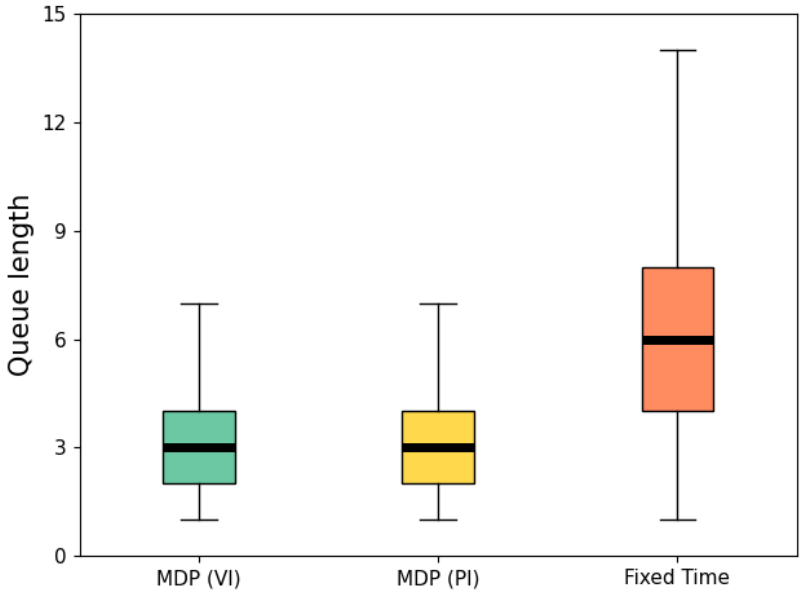


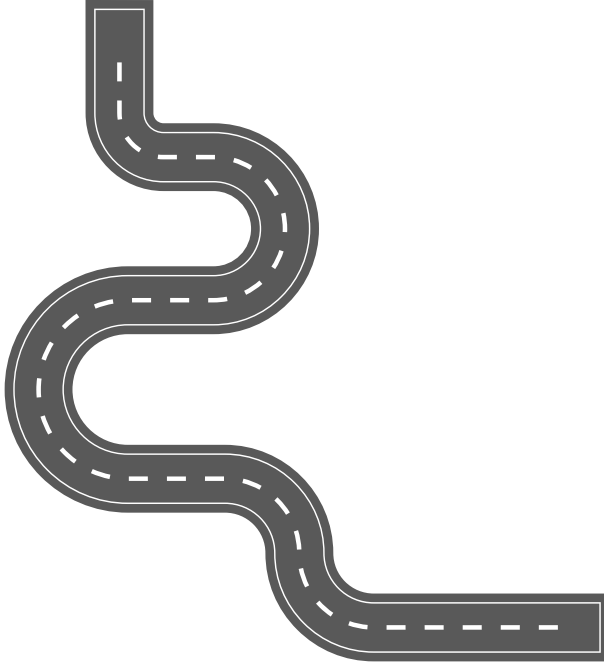
The fixed time agent makes people arrive late!



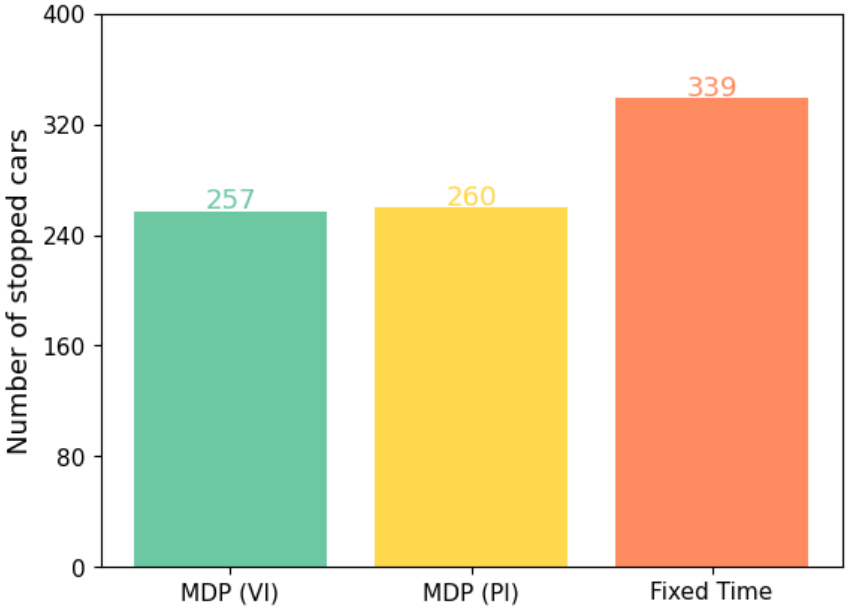


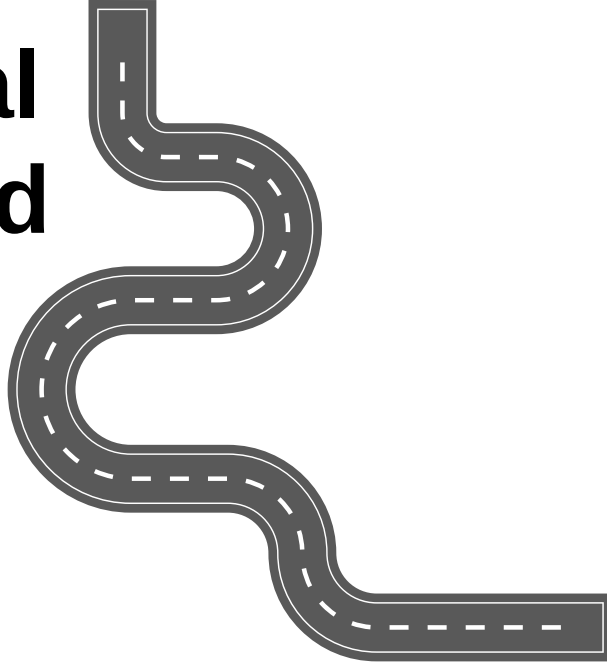
Queues might become annoying with the fixed time agent!



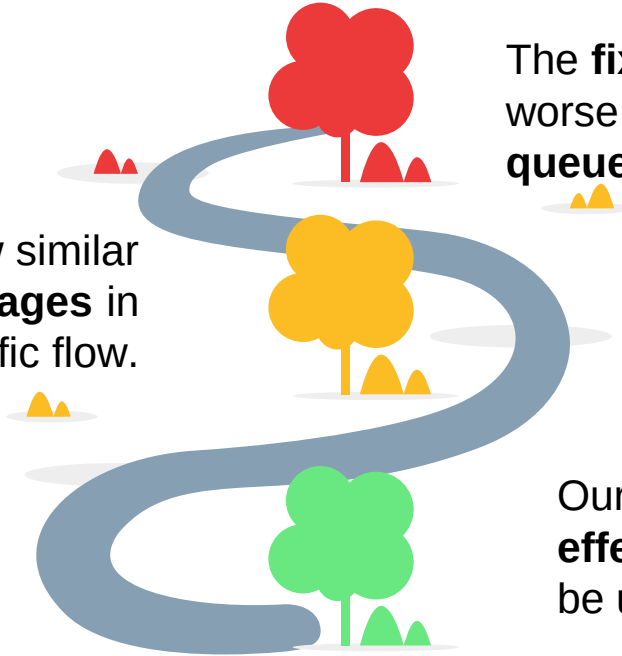


You press the break pedal less with an MDP powered stoplight!





Conclusions



The **fixed-time agent** performs worse, causing **delays** and **annoying queues**.

The other two methods show similar performance, offering **advantages** in terms of traffic flow.

Our solutions propose a **simple** yet **effective** implementation, that could be useful in a real world scenario.