# Implementation of

# IPDALight: Intensity- and phase duration-aware traffic signal control based on Reinforcement Learning

*Journal of Systems Architecture, 2022*

***Wupan Zhao, Yutong Ye, Jiepin Ding, Ting Wang, Tongquan Wei, Mingsong Chen***

**발표자: 여지호**

# Introduction

# Previous RL-based traffic signal control

Table 1: Representative deep RL-based traffic signal control methods.

| Citation | Method | Cooperation | Simulator | Road net (# signals) | Traffic flow* |
|---|---|---|---|---|---|
| [2] | Value-based | With communication | Matlab | Synthetic (5) | 2,4 |
| [5] | Policy-based | Without communication | Aimsun | Real (50) | 5 |
| [10] | Policy-based | Without communication | Aimsun | Real (43) | 5 |
| [11] | Value-based | Without communication | CityFlow | Real (2510) | 5 |
| [12] | Policy-based | Joint action | SUMO | Real (30) | 4 |
| [22] | Value-based | - | SUMO | Synthetic (1) | 2 |
| [30] | Value-based | - | Paramics | Synthetic (1) | 4 |
| [39] | Value-based | Without communication | SUMO | Synthetic (9) | 2 |
| [42] | Both studied | - | SUMO | Synthetic (1) | 1 |
| [44] | Value-based | With communication | SUMO | Synthetic (6) | 2 |
| [48] | Value-based | Without communication | AIM | Synthetic (4) | 1 |
| [49] | Both studied | Single global | GLD | Sythetic (5) | 3 |
| [51] | Policy-based | - | SUMO | Real (1) | 5 |
| [58] | Value-based | Joint action | SUMO | Synthetic (4) | 2 |
| [60] | Value-based | With communication | SUMO | Real (4) | 5 |
| [65] | Value-based | - | SUMO | Synthetic (1) | 1,3,4,5 |
| [62] | Value-based | Without communication | CityFlow | Real (16) | 2,5 |
| [63] | Value-based | With communication | CityFlow | Real (196) | 2,5 |
| [74] | Value-based | Joint action | SUMO | Synthetic (36) | 1,2,3,4 |
| [78] | Value-based | Without communication | CityFlow | Real (16) | 3,5 |
| [77] | Value-based | Without communication | CityFlow | Real (5) | 4,5 |

**MPLight**

**PressLight**



**Source: Zheng et al., 2021**

# Limitations

- MP(Max-Pressure) only considers **the number of vehicles on the lanes**, while the **vehicle speed** and **position information** is neglected
- **No coordination** among intersections
- There is no selection of variable **phase duration(신호 길이)**
  : 신호가 빠르게 계속 바뀌는 문제
  : 현실 적용성이 떨어짐

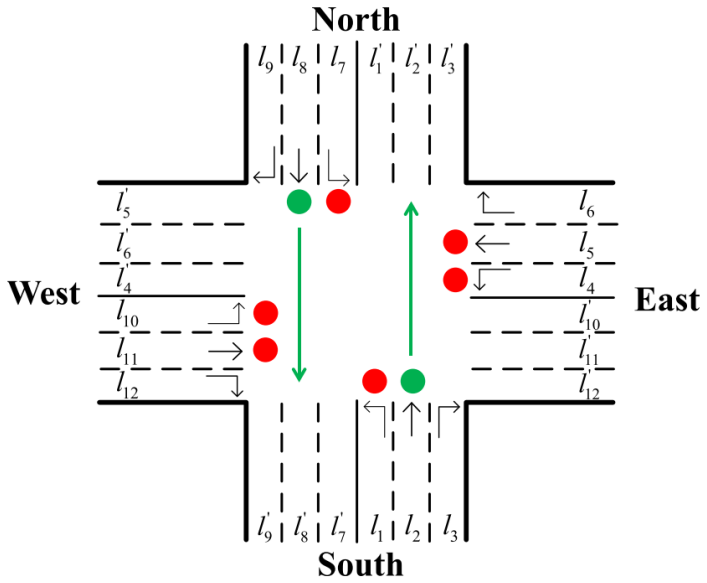
# IPDALight

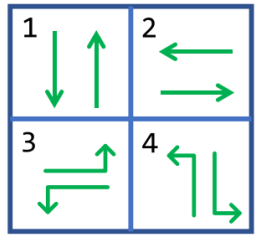
- **Propose concept of intensity**
  : considers detailed vehicle dynamics (i.e., speed and position)
  : coordination between neighboring intersections
- **Propose a heuristic algorithm**
  : support the fine-tuning of phase duration

# IPDALight approach

# Intersection modeling

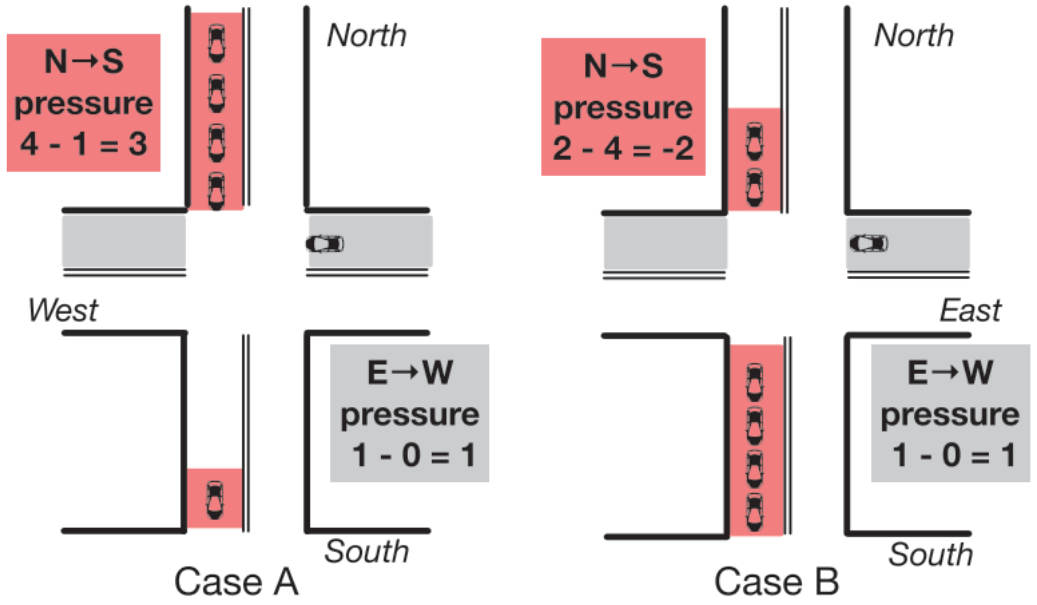
- **Definition 1 (Incoming Lanes and Outgoing Lanes of an Intersection)**
  - An incoming lane of some intersection is a lane on which vehicles can enter the intersection.
  - An outgoing lane of some intersection is a lane on which vehicles can leave the intersection.




**12 incoming and 12 outgoing lanes**

# The concept of pressure




- **Pressure**
  : # of vehicles on incoming lanes - # vehicles on outgoing lanes
- **Design an RL agent, PressLight**
  : using the pressure-based reward for long-term optimization.

# Intensity modeling

- **Unlike the pressure concept**, the definitions of intensity presented as follows consider more vehicle dynamics.
- **Definition 3 (Intensity of Vehicles)**

$$\mathcal{T}_{veh} = log\left(\frac{L-x}{L} \times \frac{\delta \times (v_{max} - v)}{v+1} + 1\right)$$

where $x$ (unit: meters) indicates the distance between the vehicle and the intersection, $L$ (unit: meters) denotes the lane length, $v$ (unit: meters/second) represents the current vehicle speed, $v_{max}$ (unit: meters/second) denotes the maximum allowed speed of the lane, and $\delta$ is a weight factor to adjust the influence of speed on intensity.

# Intensity modeling

$$\mathcal{T}_{veh} = log\left(\frac{L-x}{L} \times \frac{\delta \times (v_{max} - v)}{v+1} + 1\right)$$

- assume that the intensity of vehicle increases when it approaches some intersection or the speed of vehicle decreases due to traffic congestion
- the vehicles crossing an intersection with higher speed will pose lower intensity on the intersection

# Agent design - Observation (State)

- **Information around the intersection and its immediate neighbors**
  : the intensity of each phase
  : the impacts derived from each immediate neighbor in four directions
  : the current phase

Under the four-phase setting, a state of the intersection can be encoded as $(\mathscr{T}_{phase_1}, \mathscr{T}_{phase_2}, \mathscr{T}_{phase_3}, \mathscr{T}_{phase_4}, \mathscr{P}_{I,I'_1}, \mathscr{P}_{I,I'_2}, \mathscr{P}_{I,I'_3}, \mathscr{P}_{I,I'_4}, phase)$, assuming that $I'_1$, $I'_2$, $I'_3$ and $I'_4$ are the four neighboring intersections of the current intersection $I$.

# Agent design - Action

- The main task of an RL agent is to select the best phase to minimize **the intensity of the intersection**.
- When the time of the current phase is exhausted, the agent needs to select a new feasible phase

# Agent design - Reward

- The **intensity of intersections** can reflect the **average travel time of vehicles** crossing the intersection more accurately

$$r = -\mathscr{T}_I, \text{where } \mathscr{T}_I \text{ is the intensity of } I$$

# Phase duration design

- our approach needs to figure out **a set of phase durations** based on the given $t_{min}$ and $t_{max}$ , which represent the minimum and maximum allowable duration, respectively.
- When M=1, the phase duration is fixed and equals to $t_{max}$. When M>1, the phase duration set *D* can be constructed as follows

$$D = \left\{ t_{min} + i \times \Delta t : i \in \mathbb{N} \quad \& \quad i < M \quad \& \quad \Delta t = \frac{t_{max} - t_{min}}{M-1} \right\}$$
$$where\ M \in \mathbb{N}\ and\ M > 1.$$

# Phase duration design

- **After selecting a phase**, the agent will **determine a proper duration for the phase** from the set *D* based on the observed number of vehicles on the incoming lanes under the restriction of the chosen phase.

$$duration = \left[ \underset{t}{argmin}\left(t - \left\lceil \frac{N}{n} \right\rceil\right) \right]_{t_{min}}^{t_{max}}$$
$$where\ t \in D,\ and\ t \geq \left\lceil \frac{N}{n} \right\rceil.$$

Here, $n$ is the number of vehicles passing through the intersection per unit time, and $N$ represents the sum of total number of vehicles on all the incoming lanes. The operator $y = [x]_a^b$ means that: (i) if $x \geqslant a$ and $x \leqslant b$ then $y = x$; (ii) if $x \leqslant a$ then $y = a$; or (iii) if $x \geqslant b$ then $y = b$.

# Experiment

# Datasets

- **Synthetic datasets**: We considered four synthetic traffic datasets with different scales (i.e., $1 \times 3$, $2 \times 2$, $3 \times 3$, $4 \times 4$). The simulator generated 500 vehicles/h/lane on average following a Gaussian distribution. All the vehicles entered and left the network from rim edges (see details in [36]). The ratios of vehicles turning left, going straight, and turning right are 10%, 60% and 30%, respectively.
- **Real-world datasets**: We used two datasets collected from the real-world traffic of two cities (i.e., Hangzhou and Jinan) in China via roadside surveillance cameras. To enable the simulation of these two datasets on Cityflow, we adopted the traffic networks exported from GoogleMap as shown in Fig. 4. Fig. 4(a) shows the traffic network of Dongfeng sub-district used for the Jinan dataset, which contains 12 intersections in the form of a $3 \times 4$ grid. Fig. 4(b) shows the traffic network of Gudang sub-district for the Hangzhou dataset, which contains 16 intersections in the form of a $4 \times 4$ grid.


(a) Dongfeng, Jinan, China (b) Gudang, Hangzhou, China

# Implementation

- **https://github.com/Dokyyy/IPDALight**

# 강화학습을 이용한 교통신호 제어 논문 리뷰

**Toward A Thousand Lights:**
**Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control**

: 맨하탄을 대상으로 2500개 이상의 신호를 제어

: MPlight

**IPDALight: Intensity- and phase duration-aware traffic signal control based on Reinforcement Learning**

: Phase duration에 대한 제시

$$D = \left\{ t_{min} + i \times \Delta t : i \in \mathbb{N} \quad \& \quad i < M \quad \& \quad \Delta t = \frac{t_{max} - t_{min}}{M-1} \right\}$$
$$where\ M \in \mathbb{N}\ and\ M > 1.$$

# 두 논문을 리뷰해 보며...

- **현실 적용 가능한 신호제어 알고리즘**
  : Yellow light를 환경에서 고려하지 않고 있음(반영함)
  : 보행자에 대한 고려가 전혀 되고 있지 않음(개선중)
- **다양한 강화학습 알고리즘 Test**
  : 대부분의 모델이 DQN을 사용함
  : PPO, DDQN과 같은 알고리즘 적용하여 Test (진행중)
- **강화학습 환경에 대한 개선**
  : 모든 Test site의 차선이 3차선으로 고정되어 있음
  : 다양한 기하구조, 도로차선수를 반영할 필요가 있음
  : 대전시를 대상으로 새로운 환경을 구축 (반영함)

# Intensity에 대한 수정

- 기존의 Intensity 식은 차량의 속도를 이용함
- 시뮬레이션이 아닌 실제 환경에서 차량의 속도를 매 순간 측정하는 것은 불가능
- 속력 대신 교차로에서 차량이 기다린 시간과 차량이 교차로에 머문 전체 시간을 사용

### IPDALight

- **start lane**

$$\mathcal{T}_{veh} = log(\frac{X}{L} * \frac{\sigma * (v_{max} - v)}{v + 1} + 1)$$

- **end lane**

$$\mathcal{T}_{veh} = log(\frac{(L - X)}{L} * \frac{\sigma * (v_{max} - v)}{v + 1} + 1)$$

$X$ : 차량이 해당 위치해 있는 도로에서 이동한 거리

$L$ : 차량이 위치해 있는 도로의 길이

$\sigma$ : 1.5

$v_{max}$ : 차량의 최대 속력 ⇒ 11.111 m/s

$v$ : 차량의 현재 속력

### ETB

- start lane

$$\mathcal{T}_{veh} = \frac{X}{L} * \frac{waiting\_time}{total\_time}$$

- end lane

$$\mathcal{T}_{veh} = \frac{(L - X)}{L} * \frac{waiting\_time}{total\_time}$$

$X$ : 차량이 해당 위치해 있는 도로에서 이동한 거리

$L$ : 차량이 위치해 있는 도로의 길이

waiting_time : 차량이 해당 위치해 있는 도로에 들어와서 멈춘 시간의 합

total_time : 차량이 해당 위치해 있는 도로에 있는 총 시간

# Intensity에 대한 수정

- 기존의 Intensity 식은 차량의 속도를 이용함
- 시뮬레이션이 아닌 실제 환경에서 차량의 속도를 매 순간 측정하는 것은 불가능
- 속력 대신 교차로에서 차량이 기다린 시간과 차량이 교차로에 머문 전체 시간을 사용

### IPDALight

- **start lane**

$$\mathcal{T}_{veh} = log(\frac{X}{L} * \frac{\sigma * (v_{max} - v)}{v + 1} + 1)$$

- **end lane**

$$\mathcal{T}_{veh} = log(\frac{(L - X)}{L} * \frac{\sigma * (v_{max} - v)}{v + 1} + 1)$$

$X$ : 차량이 해당 위치해 있는 도로에서 이동한 거리

$L$ : 차량이 위치해 있는 도로의 길이

$\sigma$ : 1.5

$v_{max}$ : 차량의 최대 속력 ⇒ 11.111 m/s

$v$ : 차량의 현재 속력

### ETB

- start lane

$$\mathcal{T}_{veh} = \frac{X}{L} * \frac{waiting\_time}{total\_time}$$

- end lane

$$\mathcal{T}_{veh} = \frac{(L - X)}{L} * \frac{waiting\_time}{total\_time}$$

$X$ : 차량이 해당 위치해 있는 도로에서 이동한 거리

$L$ : 차량이 위치해 있는 도로의 길이

waiting_time : 차량이 해당 위치해 있는 도로에 들어와서 멈춘 시간의 합

total_time : 차량이 해당 위치해 있는 도로에 있는 총 시간

# State, Action, Reward 정의

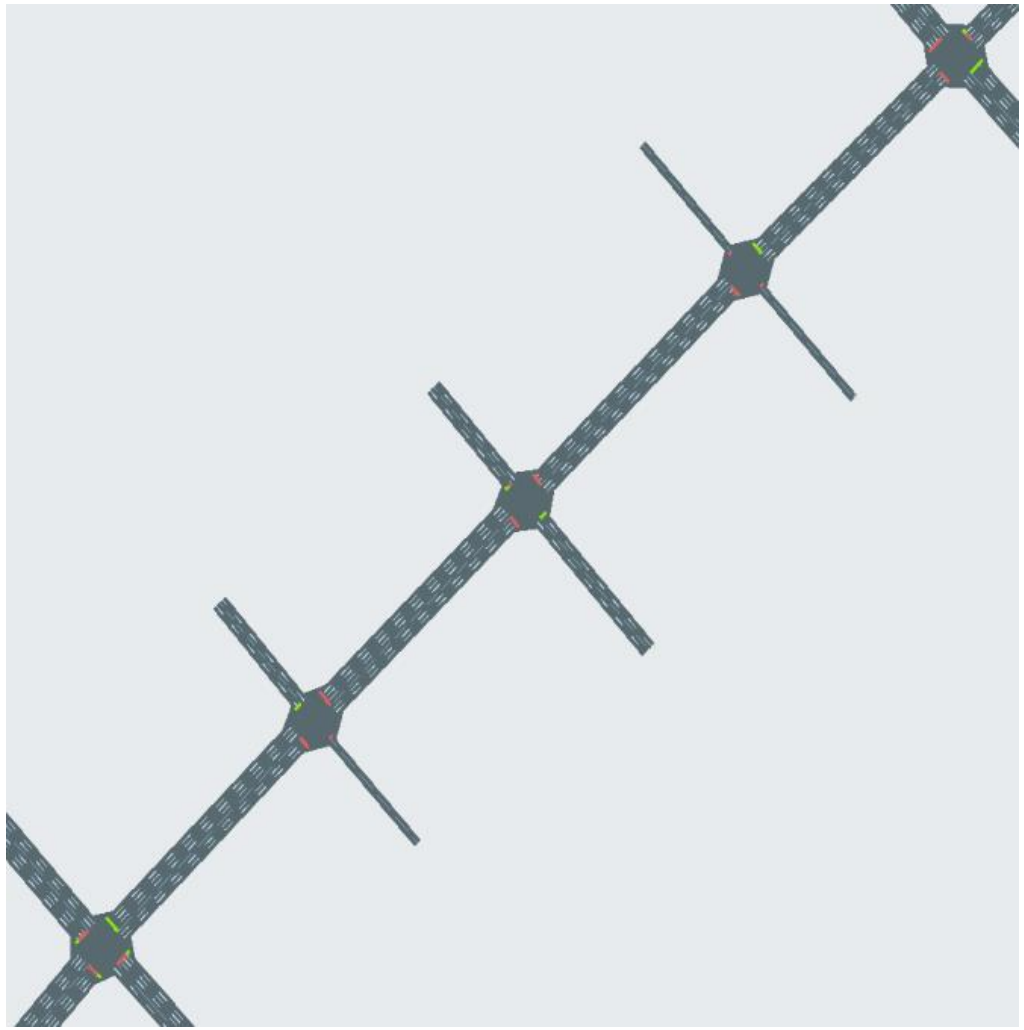
- **State**
  - 분할 구간(5) X 한 교차로의 Start Lane 차선 수
  - 각 방향에 맞는 Start Lane과 End Lane의 Intensity 차이
  - 주변 교차로의 State

- **Action**
  - Phase Set

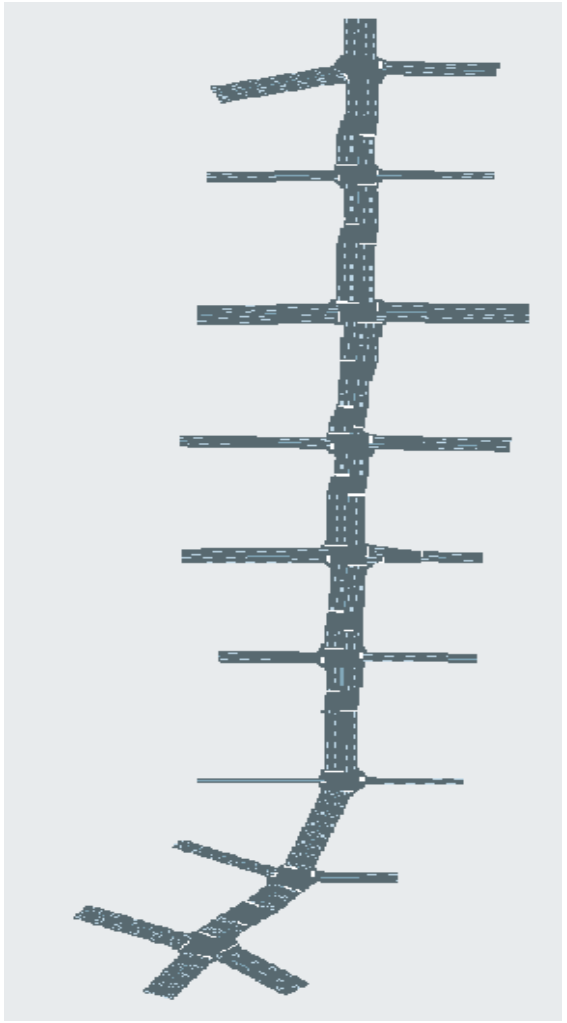
- **Reward**
  - End Lane의 모든 차량 수 - Start Lane의 모든 차량 수

# 대전시 환경 구축

● Real - 대전광역시 서구 (5)

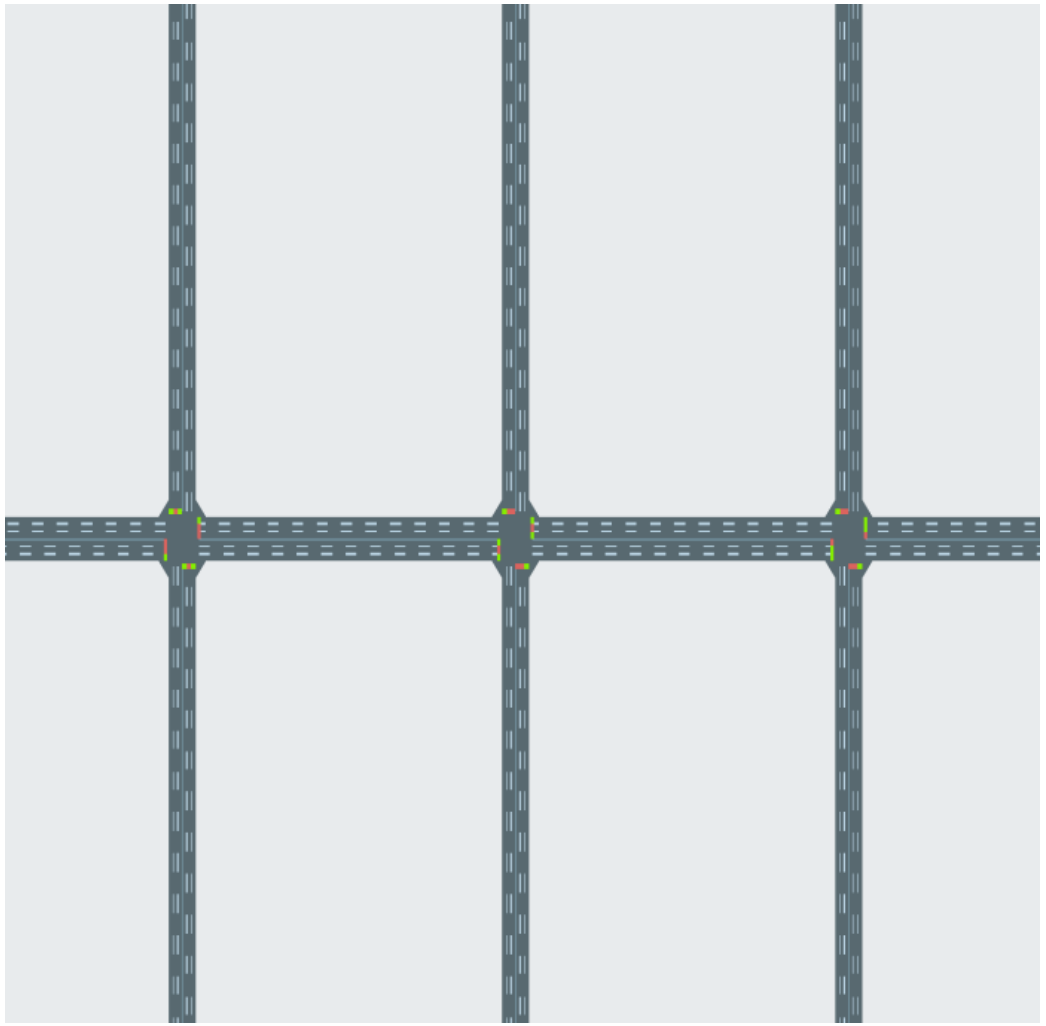


● Real - 대전광역시 대덕구 (9)

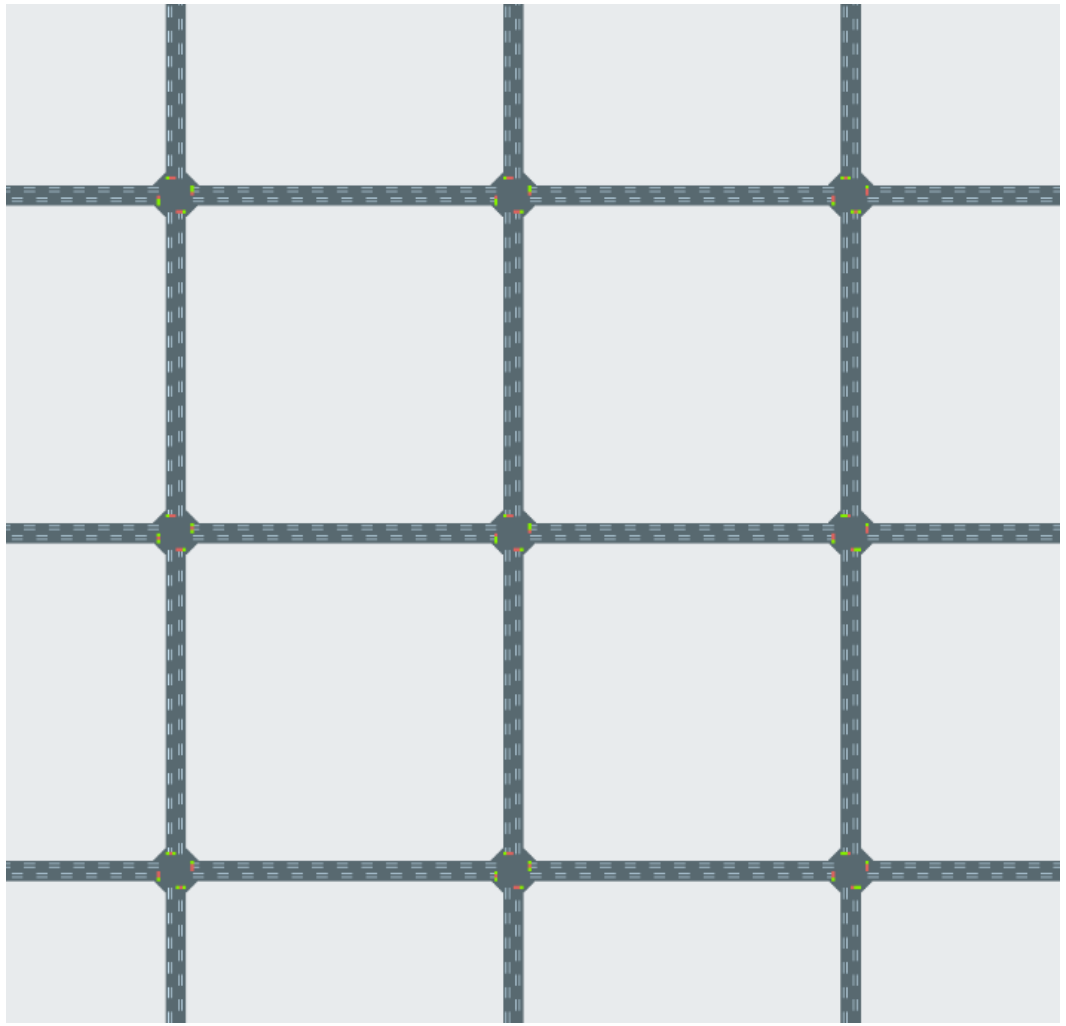


# 기존 논문에서 활용한 지역

- Synthetic - 1 X 3 (3)

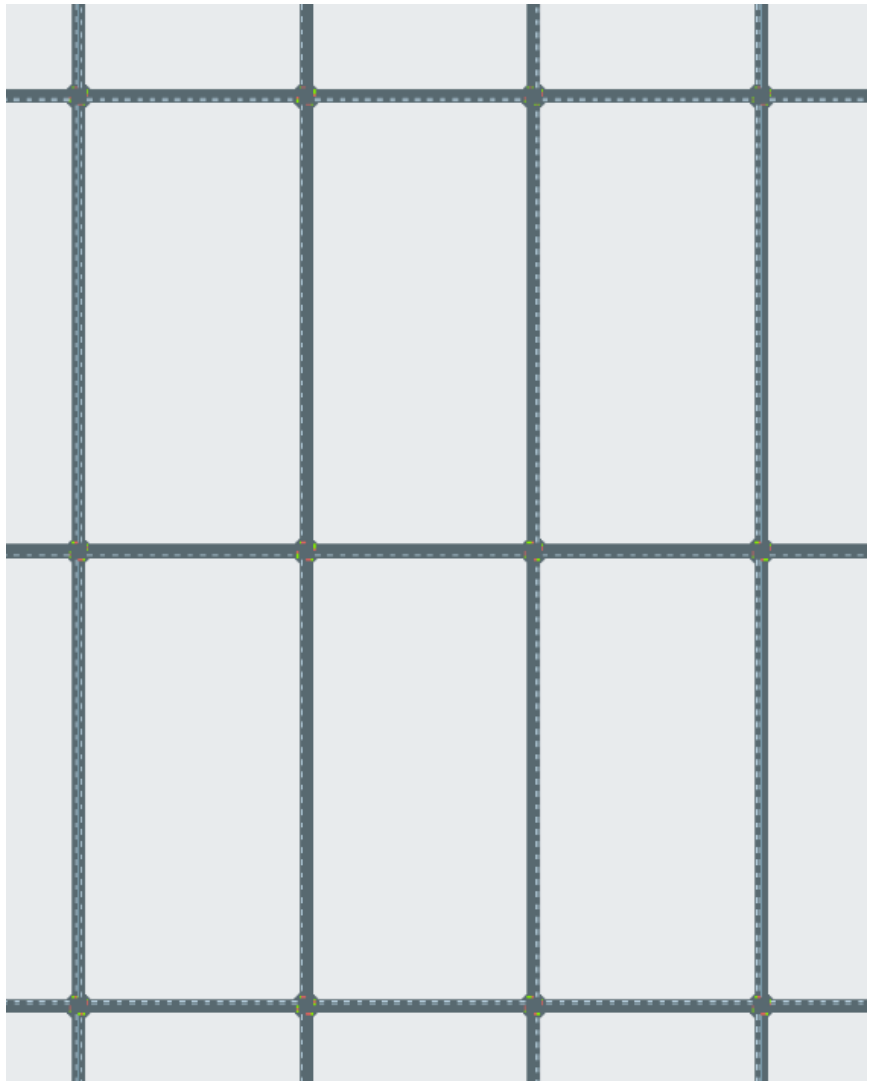


- Synthetic - 3 X 3 (9)

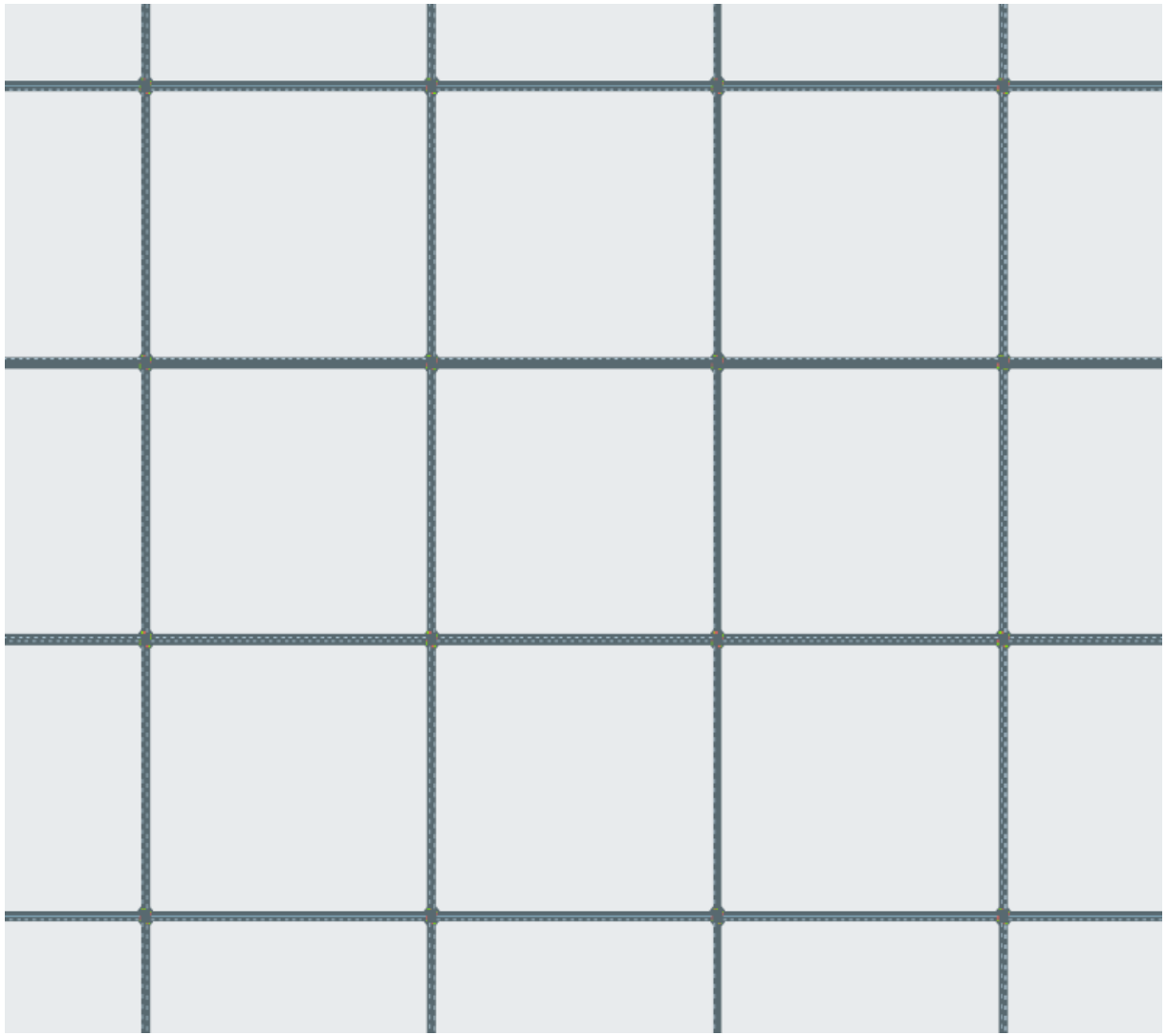


# 기존 논문에서 활용한 지역

● Real - Jinan (12)



● Real - Hangzhou (16)



# 기존 논문에서 활용한 지역


- Upper East Side, Manhattan, New York, USA
- 교차로 196개

# 성능

- Average Vehicl Travel Time (s)

| Type | Method | 3X1 | 2X2 | 3X3 | 4X4 | Jinan | Hangzhou | NewYork | daejeon(서구) | daejeon(대덕구) |
|---|---|---|---|---|---|---|---|---|---|---|
| Non-RL | Fixed-Time | 384.47 | 454.01 | 508.87 | 565.99 | 405.91 | 488.51 | | | |
| Non-RL | SOTL | 247.07 | 331.64 | 424.66 | 474.32 | 410.65 | 505.53 | | | |
| RL | GRL | 208.21 | 239.13 | 431.43 | 523.01 | 562.91 | 598.17 | | | |
| RL | Colight | 210.01 | 312.29 | 328.7 | 397.07 | 327.62 | 337.45 | 1459.28 | | |
| RL | PressLight | 98.74 | 123.9 | 166.28 | 215.32 | 285.65 | 341.99 | | | |
| RL | IPDALight | 88.01 | 109.66 | 146.92 | 184.54 | 255.35 | 298.99 | | | |
| Ours | ETB | 85.34 | 105.47 | 139.57 | 177.71 | 254.32 | 297.92 | 890.7 | 97.4 | 97.06 |
| | | | | | | | | | | |
| difference | | 2.67 | 4.19 | 7.35 | 6.83 | 1.03 | 1.07 | 568.58 | 97.4 | 97.06 |

# 향후 개선점

- **Action space에 Phase duration까지 반영할 수 있도록 설계**
  : 강화학습 Agent가 신호 Phase뿐 아니라, Duration까지 선택할 수 있도록
- **교차로간 신호 연동이 잘 일어날 수 있도록**
  : 단순히 인접교차로의 State를 현재 교차로의 State에 더해주는 것은 비효율적인 것 같음
  : 방법론에 대해 고민 중

# Additional information

- **Source code**
  : https://github.com/Dokyyy/IPDALight
- **Reinforcement learning for traffic signal control**
  : https://traffic-signal-control.github.io/
- **Cityflow**
  : https://arxiv.org/abs/1905.05217
  : Twenty times faster than SUMO
- **Tutorial**
  **:**https://docs.google.com/presentation/d/12cqabQ_V5Q9Y2DpQOdpsHyrR6MIxy1CJlPmUE3Ojr8o/edit

# Implementation of

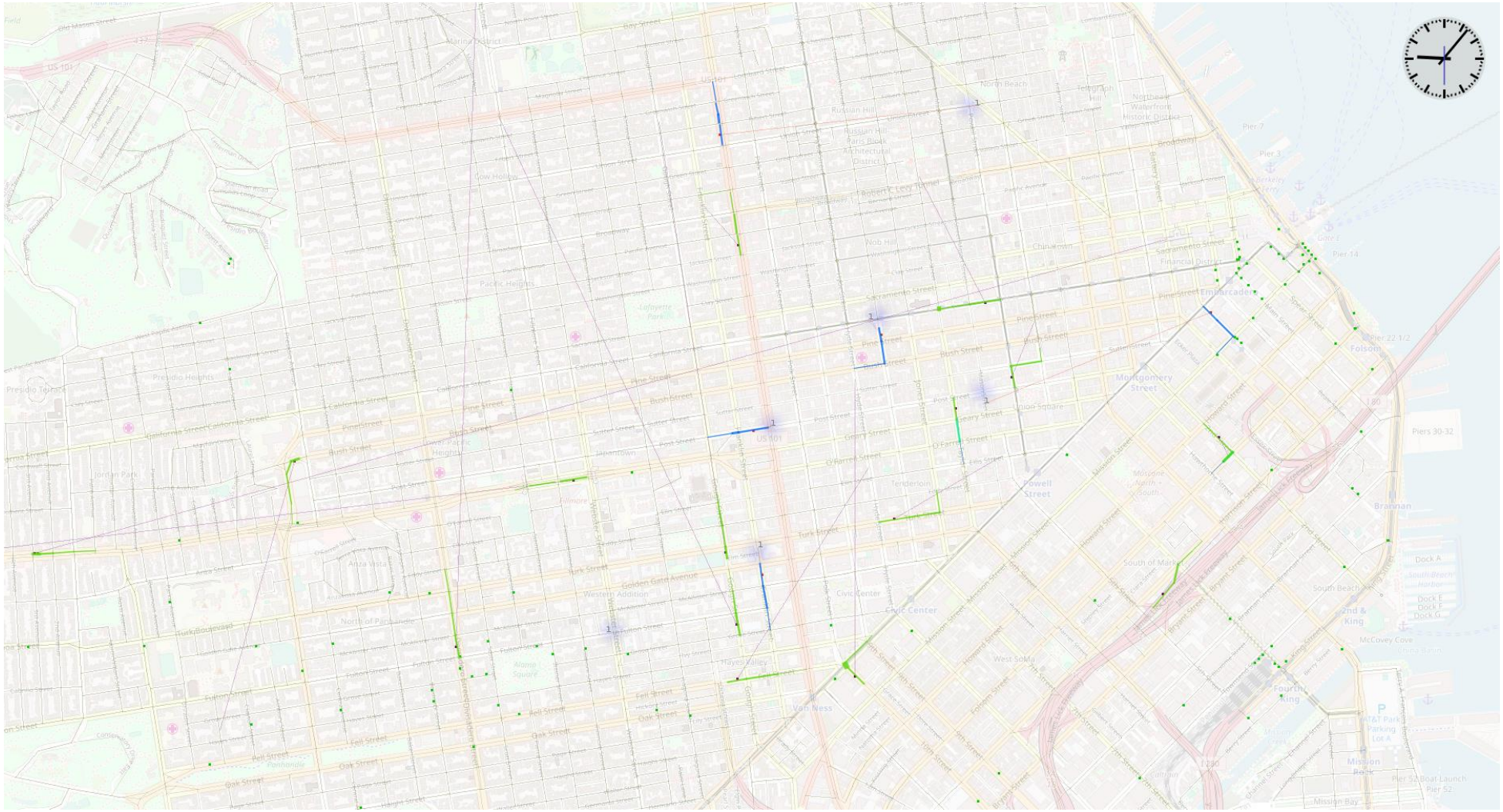
# Mobility-on-Demand simulation

# Current Mobility-on-Demand simulation - 1

https://github.com/wenjian0202/mod-abm-2.0

# Current Mobility-on-Demand simulation - 2

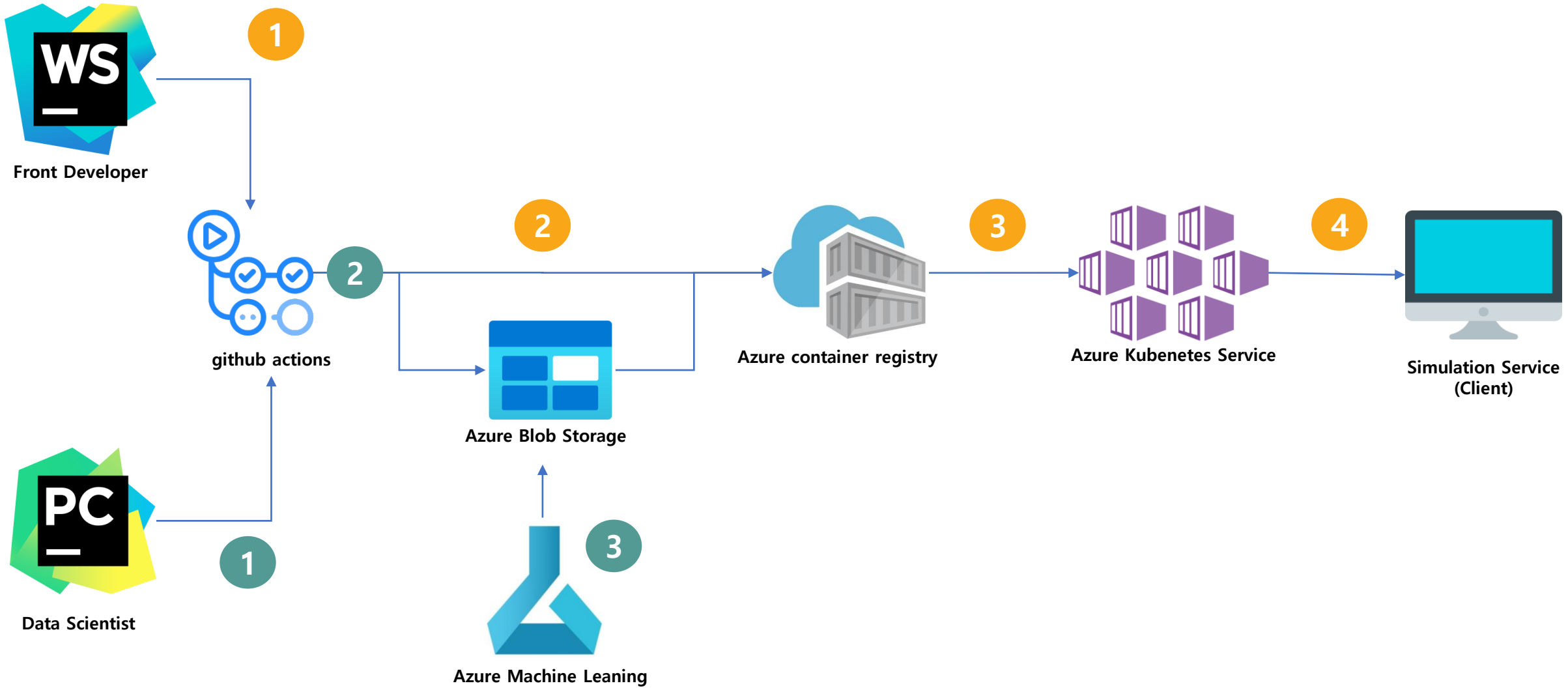
**Amodeus: https://www.amodeus.science/**



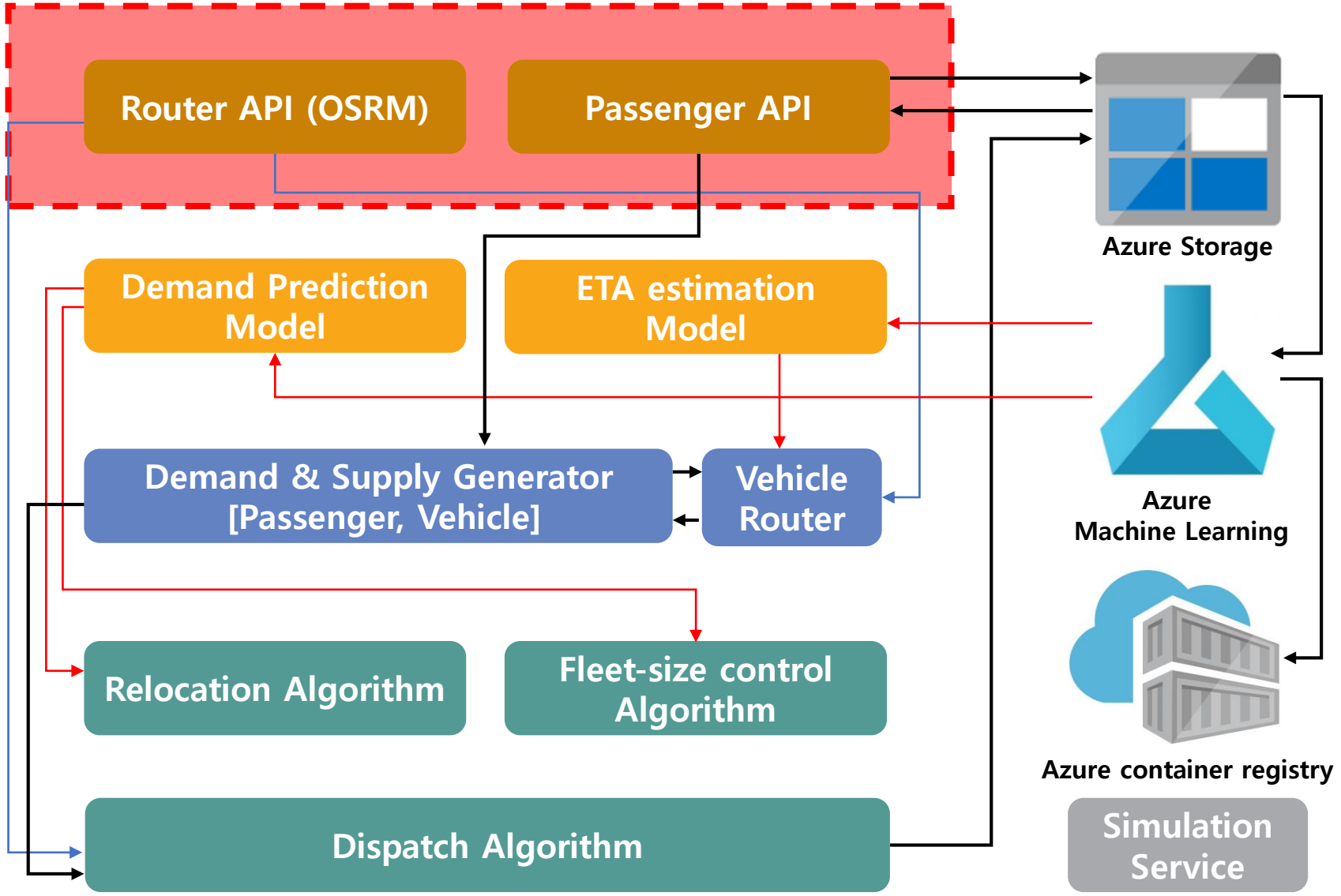
# 기존 시뮬레이션(본인)의 한계

- Mod-abm은 C++를 기반으로, Amodeus를 Matsim(Java 기반)으로 만들어짐
- Python 기반으로 직접 만들어보기로 결정

# Azure를 활용한 시스템 아키텍쳐



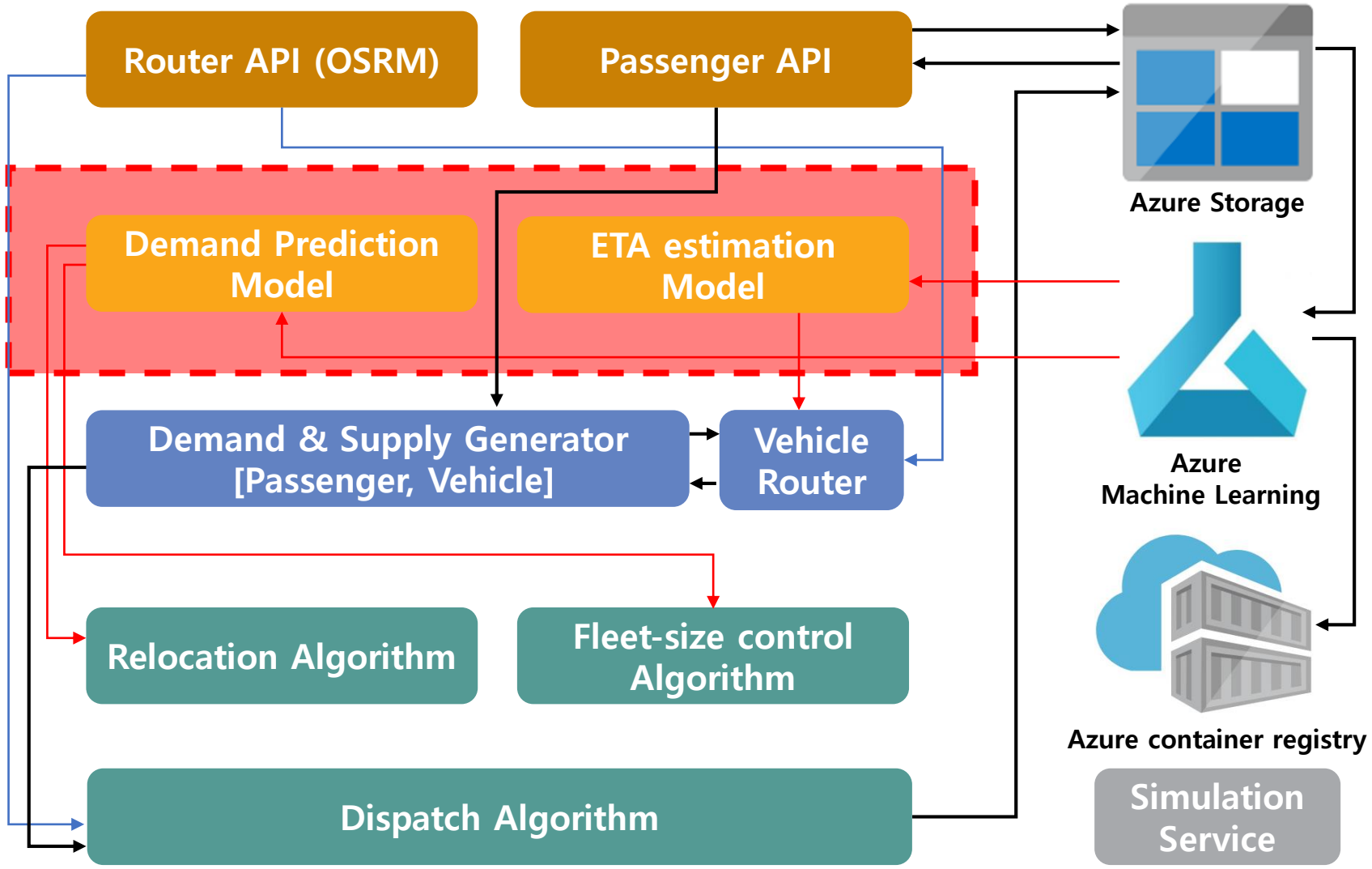
# 시스템 세부 내용




01 모델 구축을 위한 데이터

- Passenger Demand API
  - 승객 수요 예측
  - 장애인 콜택시 API 사용
- Router API
  - 차량 통행시간
  - 최적 경로
  - ETA 모델 학습에 사용

# 시스템 세부 내용

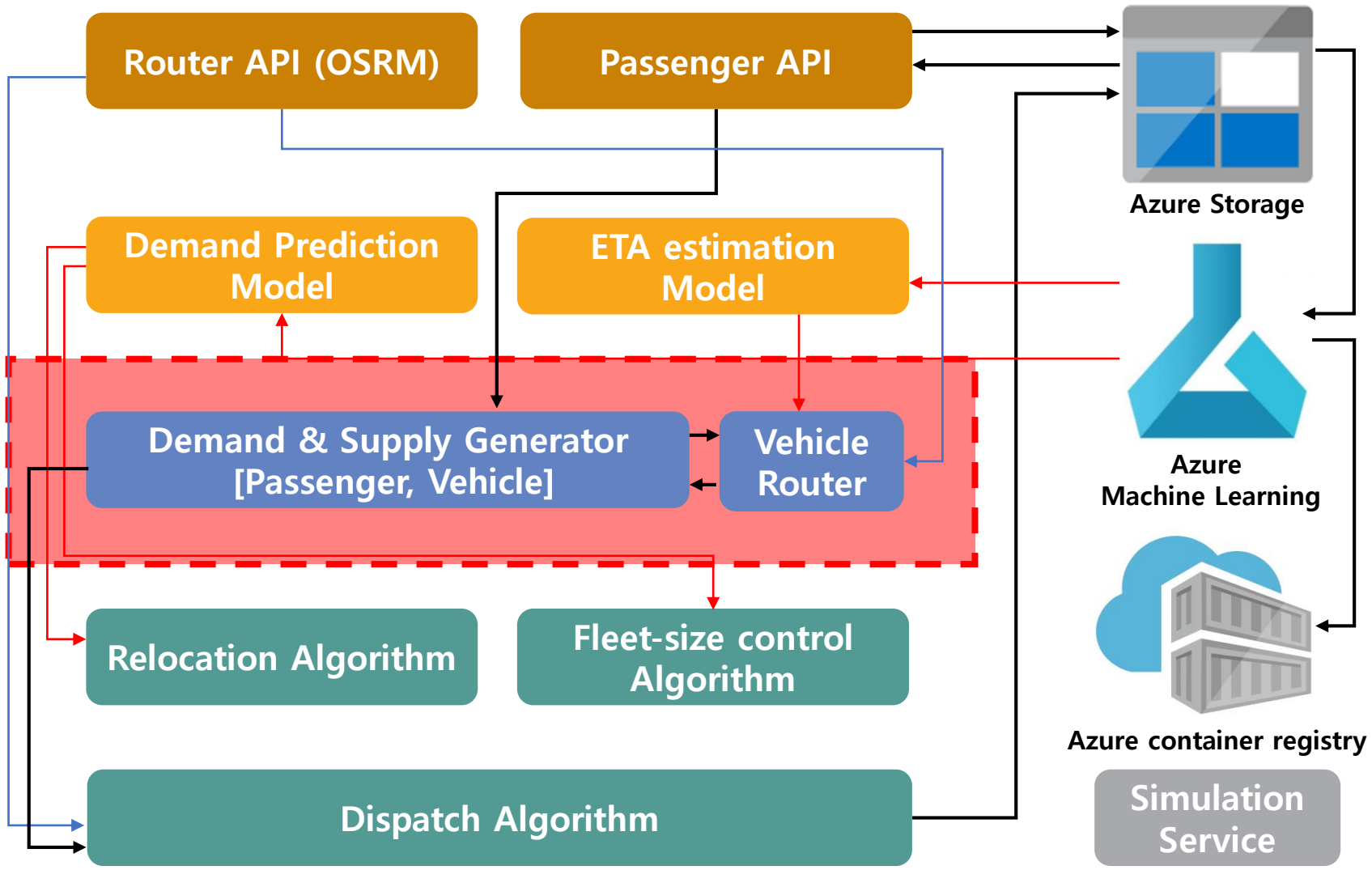



02

## AI/ML 모델

- ETA Model (Estimated Time of Arrival)
  - 현실과 유사한 디지털트윈 구축을 위한 핵심 알고리즘
  - 서울시 택시 승하차 데이터를 활용하여 학습
- Demand Prediction Mode
  - 지역/시간에 따라 승객의 단기 수요를 예측하는 알고리즘
  - 차량의 재배치, 차량운영 대수 조절에 사용

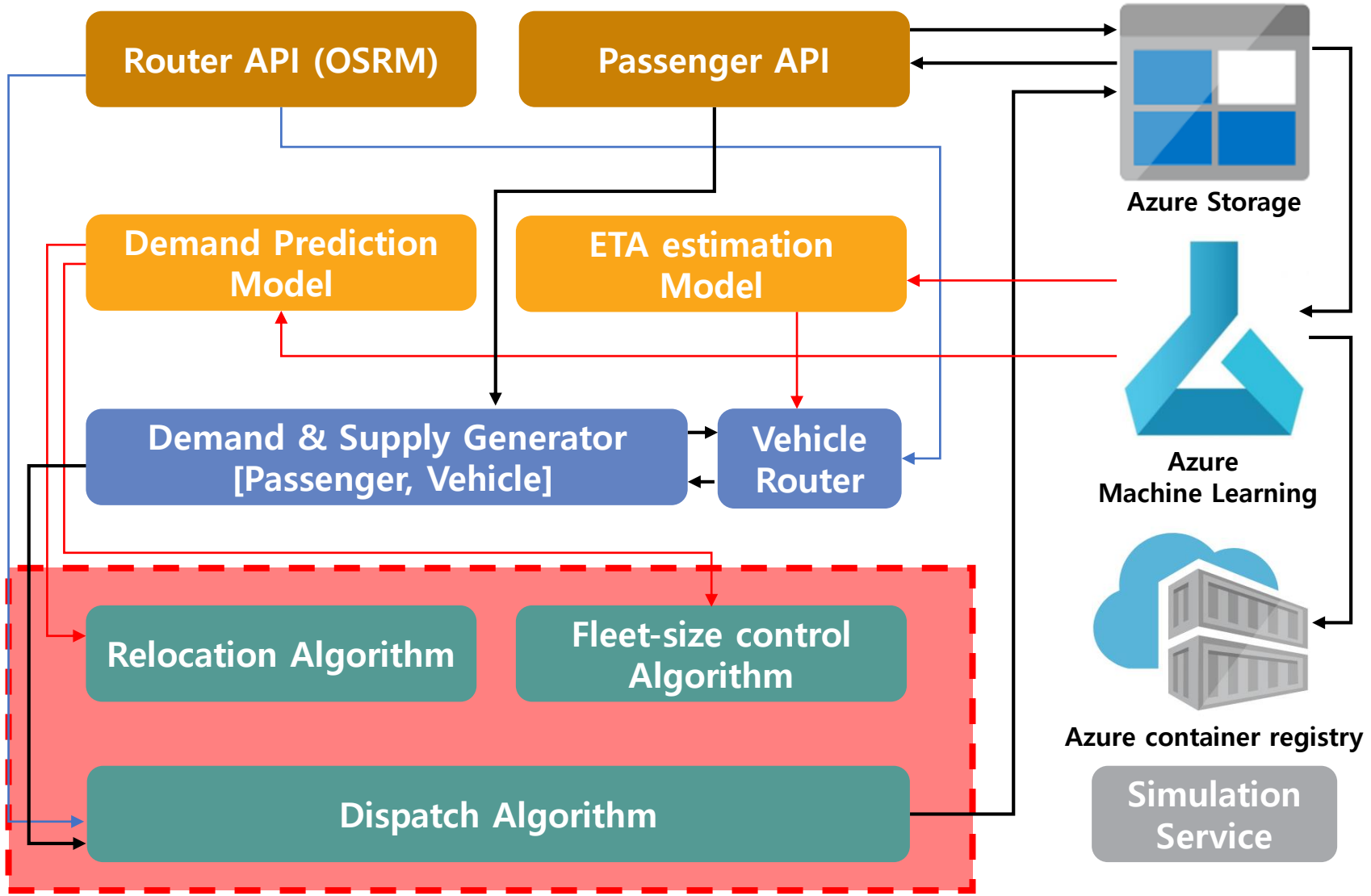
# 시스템 세부 내용




03 디지털트윈 구현 모듈

- 승객 및 차량 생성 모듈 (Demand & Supply Generator)
- 차량 경로 생성 모듈 (Vehicle Router)

# 시스템 세부 내용




04 모빌리티 시스템 운영 알고리즘

- 배차 알고리즘 (Dispatch)
- 차량 재배치 알고리즘 (Relocation)
- 차량 대수 컨트롤 알고리즘 (Fleet-size control)

# Uber의 Deck.gl 프레임워크 활용

- 차량의 움직임을 Point로 매핑하는 것 보다 훨씬 더 가벼움
- 도시단위 스케일에서 표현이 가능
- https://deck.gl/examples/trips-layer/

# 구축 결과

- 데이터:
  서울시 장애인 콜택시 API를 활용
  2022년 4월 21일 하루동안의 시뮬레이션을 수행
- Result Report : http://203.247.62.228:3388/
- Visualization : https://hnu209.github.io/UMOS/

# 향후 계획

- **구축한 환경을 기반으로 다양한 알고리즘 학습**
  : 강화학습 기반의 차량 배차
  : 강화학습 기반의 빈 차량 재배치
- **다양한 모빌리티 시스템 구현**
  : 장애인 콜택시
  : 수요응답형 버스
  : Urban Air Mobility

# 감사합니다

# Q & A