KDD cup 2021: How many vehicles can you serve at most with a city-scale road network

Kan Wu, Tu Xu

Research Center for Intelligent Network, Zhejiang Lab

Hua Wei, Tencent AI Lab

Webinar for Committee on Traffic Flow Theory and Characteristics (TRB ACP50)

04/21/2021

Overview

- Background
- The competition
- Benchmark solutions
- Get started
- Q&A

Background

KDD cup

- KDD cup is the most prestigious Data Science competition that's been run by SIGKDD (Special Interest Group on Knowledge Discovery and Data Mining) since 1997.
- In KDD Cup 2019, it had more than **2,800** registered teams from over **39** countries and **230** institutions. Among the **1,200** most actively participating teams, over **5,000** individuals participated, and more than **17,000** submissions were made.
- Recently, data driven intelligent transportation has attracted a surge of interest from machine learning and data mining researchers. There are transportation-related competitions in KDD Cup 2017 and 2020.
- This year, we are hosting "**The City Brain Challenge**", aiming for an efficient traffic coordination strategy to **serve more vehicles** in a city-scale road network.

KDD

https://www.kdd.org/kdd2021/

Urban traffic congestion

Background

	Tokyo	New York City
Number of registered vehicles	3.13 million **(+43%)**	2.19 million
Road network mileage (km)	24,650 **(+32%)**	18,684
Number of traffic signals	15,000 **(+15%)**	13,000

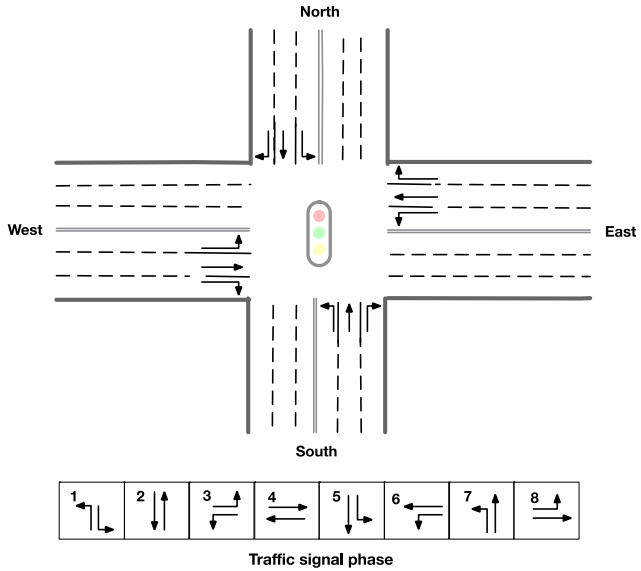
- **Question**: How many **vehicles can be served** with a city-scale road network (meanwhile, maintaining an acceptable delay)?
  - Road network size (usually remain unchanged)
  - Traffic demands
  - Traffic management strategy

The competition

Problem statement

- **Scenario:**
  - City-scale road network (more than **1000** intersections)
  - Traffic signal timing can be optimized
- **Objective:** maximizing **total number of vehicles served** via optimizing traffic signal setting:
  - For each intersection, select a **signal phase** to be actuated for each time step.
- **Given:**
  - Road network (including traffic signal installation);
  - Sample OD demand and fixed route (mitigate the impact of randomness)

City road network



Traffic signal

Evaluation/scoring metrics

- **Cumulative number of vehicles served**
  - The vehicles that are in the network
  - The vehicles that have left the network

The trip delay D_i of vehicle i is defined as $D_i = \frac{TT_i + TT_i^r}{TT_i^f}$, where,

- TT_i: travel time of vehicle i;
- TT_i^r: rest of trip travel time, estimated with free-flow speed;
- TT_i^f: full trip travel time at free-flow speed

- **Delay index (>1.0)**
  - For a trip, the delay is computed as actual travel time divided by free-flow travel time;
  - For an uncompleted trip, the rest of trip travel time is estimated using free-flow speed;
  - The overall delay index is computed as average over all vehicles served.

- **For scoring:**
  - We will keep monitoring the **delay index** computed over all vehicles;
  - Once the delay index reaches the predefined threshold (say 2.0), the test process will be terminated and the solution is scored as **cumulative number of vehicles served**
  - *Note: For the first/warmup round, we did not set the delay index threshold*.

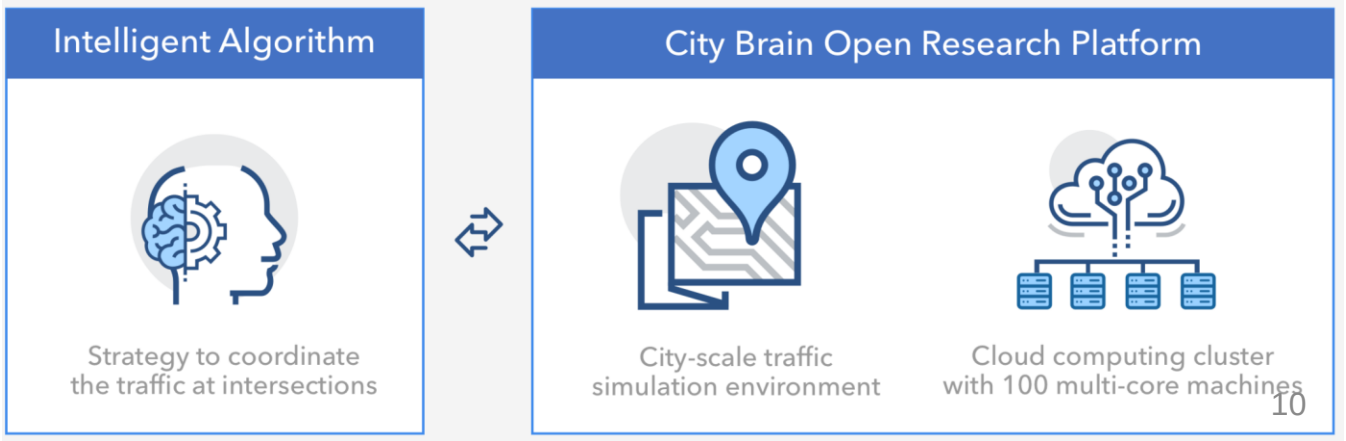
https://kddcup2021-citybrainchallenge.readthedocs.io/en/latest/city-brain-challenge.html#evaluation

Competition phases

- **Warmup** (4/1/2021-4/30/2021)
- Participants will practice with regional road network with light traffic to get familiar with the simulation environment**. All participants (team members) need to sign-up by 4/30/2021.**
- **Qualification** (5/1/2021-5/31/2021)
- Participants will deal with city-scale road network **(about 1000 intersections)** traffic. Teams that can serve more vehicles will enter the final round (**test traffic will be provided**).
  - The test traffic data are given, but you are encouraged to create your own dataset for training if use data-driven method
- **Final** (6/1/2021-6/30/2021)
- Large-scale **cloud computing platform** is provided. Teams will develop methods to handle diverse unknown traffic flows at the city-scale **(about 1000 intersections)**.
  - The sample traffic data are given, however, the traffic data for scoring **differs from the sample data**;
  - Hence, the solution need to adapt to changes in demands of OD pairs.

City Brain open research platform - CBEngine

- **High-efficiency engine**
  - Support experiments with more than 1000 intersections and 100,000 vehicles
  - Large-scale parallel computing
  - Cloud computing cluster with 100 multi-core machines
- **Microscopic traffic simulation environment**
  - Fixed route (mitigate the impact of randomness)
  - Safety distance car-following models
  - Lane change models
    - Mandatory (for turn movement) and discretionary (deprecated to minimize randomness) lane change
    - Safety distance gap acceptance



Benchmark solutions

Benchmark solutions

Rule based method (transportation engineering)

- Fixed time (pre-timed, e.g. Webster);
- Green wave (Maxband);
- Max-pressure (Decentralized);
- Perimeter metering control (MFD-based) …

Reinforcement learning based method

- IntelliLight
- PressLight …

For more information, please refer to:

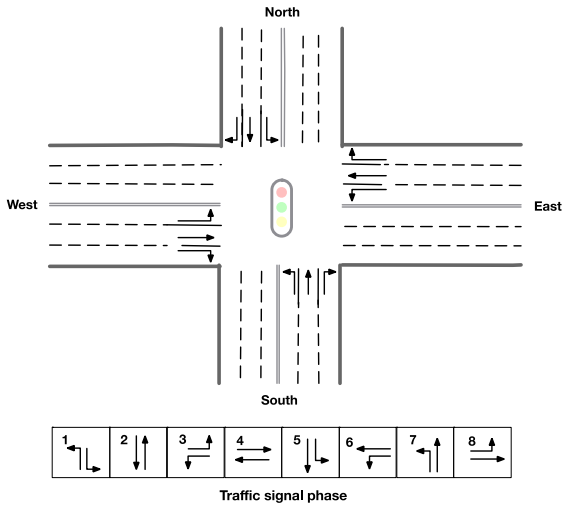
- Reinforcement learning for traffic signal control: https://traffic-signal-control.github.io/
- A Survey on Traffic Signal Control Methods: https://arxiv.org/pdf/1904.08117.pdf

Rule based method (transportation engineering)

Fixed time (pre-timed)

- Signal phase setting (phase set and phase sequence)
- Cycle length
- Phase split

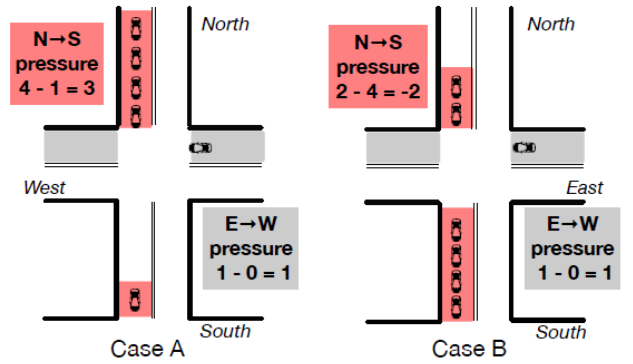
Example: Phase-1: 30s, Phase-2: 20s, Phase-3: 40s, Phase-4: 30s, cycle length = 120s.



Max pressure (decentralized controller)

- A simple MP controller: Each time step (10s), actuate the signal phase with maximum difference in upstream and downstream queue lengths

Example: In Case-A, Phase-2 (southbound and northbound through movement) is actuated.



Algorithm 1: Max-pressure Control

Input: Current phase time t, minimum phase duration length t_{min}

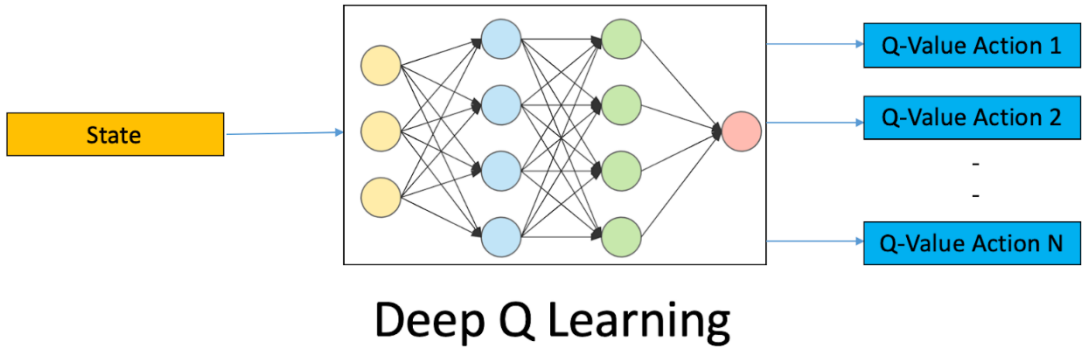
forall *timestamp* **do**
  $t = t + 1$
  if $t >= t_{min}$ **then**
    Calculate the pressure P_i for each phase i;
    Set the next phase as $\arg\max_i\{P_i\}$;
    $t = 0$;
  end
end

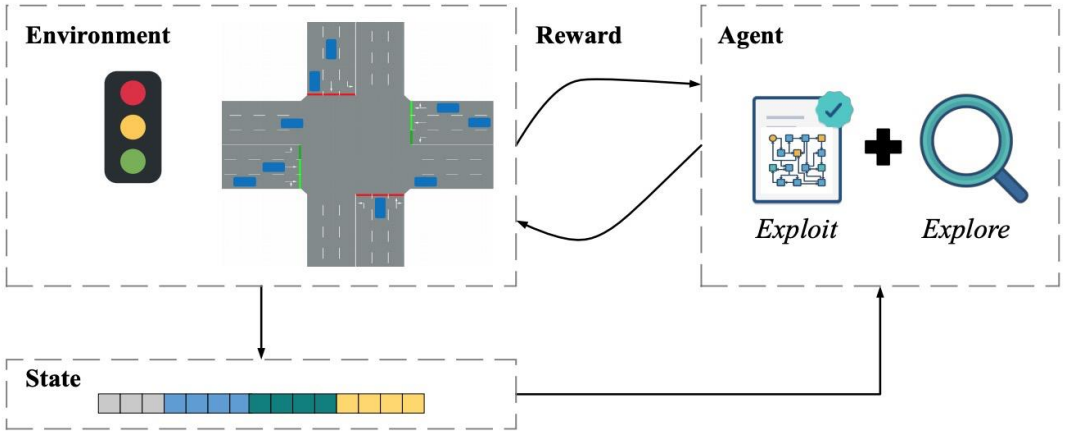
Reference:
Wei et al., 2019, PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network

Reinforcement learning based method

A RL agent learns about the sequences of decision to inform future decisions.

- Environment
- Agent
- State
- Action
- Reward





State (observation)

- Queue length, volume, delay, speed
- Phase duration, position of vehicles, etc.

Reward

- Queue length, waiting time, change of delay, speed
- Number of stops, throughput, frequency of signal change, pressure, etc.

Action: signal phase actuation

Sources:

https://medium.com/@novacek_48158/connect-x-with-dqn-and-pbt-be11915dd860

Wei et al., 2018, IntelliLight: a Reinforcement Learning Approach for Intelligent Traffic Light Control

Get started

- Competition environment setup
- Code structure
- Visualization
- Demos

Get started: competition environment setup (CBEngine)

Step - 1 download docker image and the starter-kit

- docker pull citybrainchallenge/cbengine:0.1.1
- git clone https://github.com/CityBrainChallenge/KDDCup2021-CityBrainChallenge-starter-kit.git

Step - 2 run test file in the docker

- docker run -it -v /path/to/your/starter-kit:/starter-kit citybrainchallenge/cbengine:0.1.1 bash
- cd starter-kit
- python3 evaluate.py --input_dir agent --output_dir out --sim_cfg cfg/simulator.cfg # run evaluate.py

Tutorial: https://kddcup2021-citybrainchallenge.readthedocs.io/en/latest/

Get started: code structure explained

docker image citybrainchallenge/cbengine:0.1.1: Simulator and OpenAI Gym APIs

starter-kit: agent.py to be implemented，evaluate.py for solution evaluation.

To be implemented:

agent/agent.py

To submit your results, you need to update:

agent/gym_cfg.py

cfg/simulator.cfg

To evaluate and score your results

evaluate.py

To train your model, you need modify

evaluate.py

A simple demo:

demo.py

Other files:

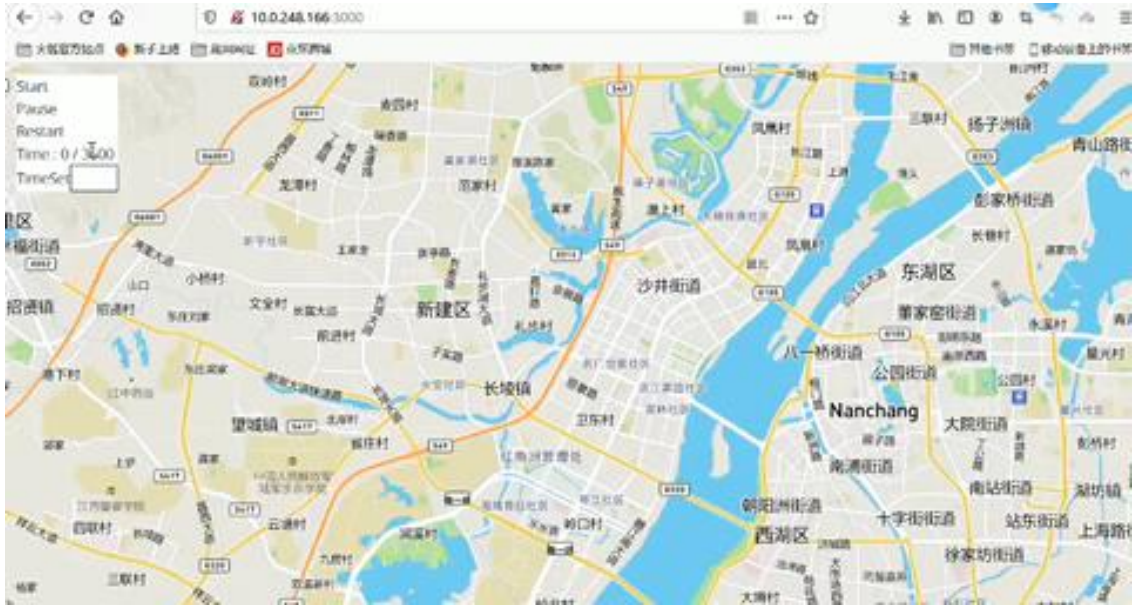
data/

log/

out/ # output

Tutorial: https://kddcup2021-citybrainchallenge.readthedocs.io/en/latest/

Get started: visualization



Prerequisites:

- You need install **yarn**
- You need apply a **mapbox** token via creating your own mapbox account

To visualize your results:

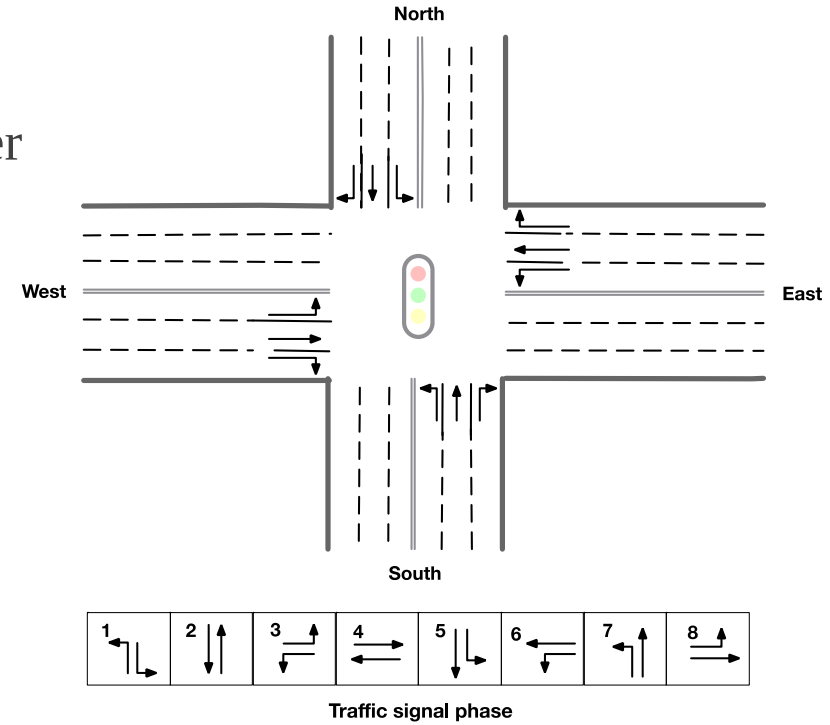
- **Move** log/lightinfo.json, log/roadinfo.json, log/time*.json files **into** ui/src/log
- **Modify** ui/src/index.js **and set** this.maxTime (in seconds) and mapboxgl.accessToken (Your mapbox token)
- cd (change disk) ui/, then: yarn start
- Open your browser at: localhost:3000

Tutorial: https://kddcup2021-citybrainchallenge.readthedocs.io/en/latest/

Get started: A demo of Fixed Time implementation

- All traffic signal phases are pre-timed;
- Traffic signal phases will be selected sequentially in a pre-defined order
- Select signal phase based on current time step.

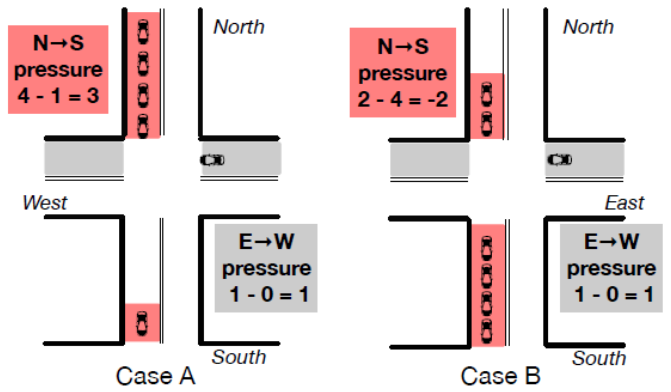
```
# get actions
for agent in self.agent_list:
    # select the now_step
    for k,v in observations_for_agent[agent].items():
        now_step = v[0]
        break
    step_diff = now_step - self.last_change_step[agent]
    if(step_diff >= self.green_sec):
        self.now_phase[agent] = self.now_phase[agent] % self.max_phase + 1
        self.last_change_step[agent] = now_step


    actions[agent] = self.now_phase[agent]
return actions
```



Tutorial: https://kddcup2021-citybrainchallenge.readthedocs.io/en/latest/

Get started: a demo of DQN implementation

- **State**: number of vehicles by lane
- **Action**: select signal phase each time step
- **Reward:** traffic pressure




```python
def get_action(self, ob):

    # The epsilon-greedy action selector.

    if np.random.rand() <= self.epsilon:
        return self.sample()
    ob = self._reshape_ob(ob)
    act_values = self.model.predict([ob])
    return np.argmax(act_values[0])
```

```python
# The main loop
for e in range(args.episodes):
    last_obs = env.reset()
    # Begins one simulation.
    i = 0
    while i < args.steps:
        if i % args.action_interval == 0:
            # Get the state.
            observations_for_agent = {}
            for key, val in observations.items():
            # Get the action, note that we use act_() for training.
            actions = agent.act_(observations_for_agent)
            # We keep the same action for a certain time
            for _ in range(args.action_interval):
                i += 1
                # Interacts with the environment and get the reward.
                observations, rewards, dones, infos = env.step(actions_)
            # Get next state.
            new_observations_for_agent = {}
            for key, val in observations.items():
            # Remember (state, action, reward, next_state) into memory buffer.
            for agent_id in agent_id_list:
                agent.remember(observations_for_agent[agent_id]['lane'], actions[agent_id], rewards[agent_id],
                               new_observations_for_agent[agent_id]['lane'])
                episodes_rewards[agent_id] += rewards[agent_id]
            total_decision_num += 1
            last_obs = observations
        # Update the network
        if total_decision_num > agent.learning_start and total_decision_num % agent.update_model_freq == 0:
            agent.replay()
        if total_decision_num > agent.learning_start and total_decision_num % agent.update_target_model_freq == 0:
            agent.update_target_network()
        if all(dones.values()):
            break
```

Reference: Wei et al., 2019, PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network

Tutorial: https://kddcup2021-citybrainchallenge.readthedocs.io/en/latest/

Timeline

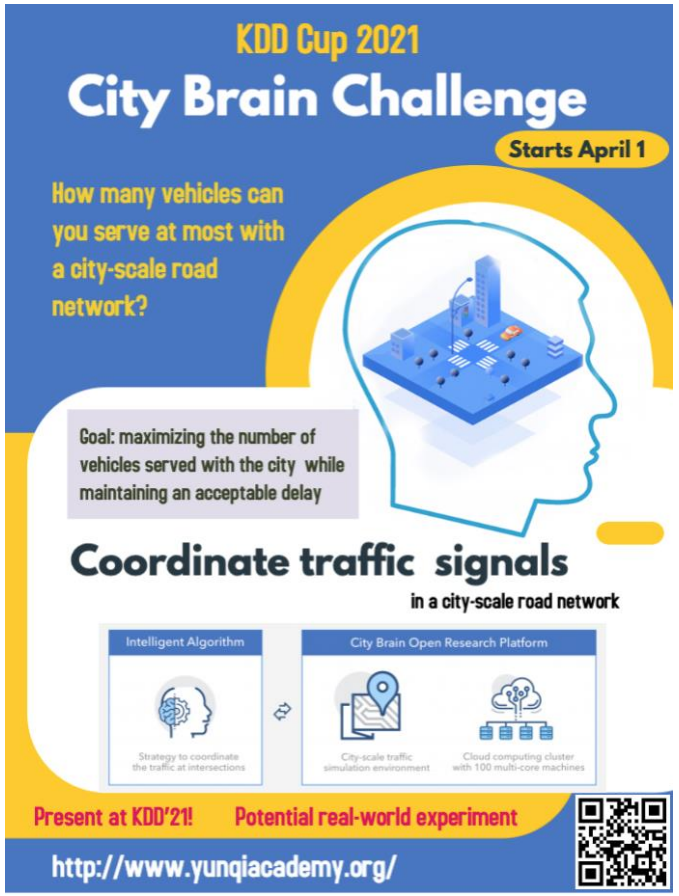
- **Important dates**
  - All participants need to sign-up by **4/30/2021 11:59 PM** (anywhere on earth).

- **Warmup** (4/1/2021-4/30/2021)
- **Qualification** (5/1/2021 - 5/31/2021)
- **Final** (6/1/2021 - 6/30/2021)

Note: all deadlines are on 23:59pm anywhere on earth

Q&A

Come and join the KDD Cup 2021 - City Brain Challenge!



Homepage: http://www.yunqiacademy.org/

Tutorial:
https://kddcup2021-citybrainchallenge.readthedocs.io/en/latest/

- Email: CityBrainChallenge@gmail.com
- Slack: citybrainchal-kwx6085.slack.com
- Google groups: https://groups.google.com/g/citybrainchallenge
- Github: https://github.com/CityBrainChallenge/KDDCup2021-CityBrainChallenge-starter-kit.git

https://www.kdd.org/kdd2021/