Multiagent Reinforcement Learning Applied to Traffic Light Signal Control

Carolina Higuera [1] Fernando Lozano [2] Camilo Camacho [1]
Carlos Higuera [3]

[1]Universidad Santo Tomás

[2]Universidad de los Andes

[3]Universidad Pedagógica y Tecnológica de Colombia

Conference on Practical Applications of Agents and multi-agents systems (PAAMS'19)

June 27th, 2019

Motivation and problem definition

Would you like to spend 272 hours/year in a traffic jam?

Motivation and problem definition

"According to the report made by the specialized firm INRIX to 38 countries and 1,360 cities, which evaluated the impact of mobility of these cities, Bogota is among the ten territories with more traffic jams in the world and the second in Latin America"

Source: Instituto de Estudios Urbanos - UNAL

Motivation and problem definition

Cities with the World's traffic congestion

Ciudades con el peor tráfico vehicular

Lugar	Ciudad
1	Bogotá
2	Moscú
3	Estambul
4	Ciudad de México
5	Sao Paulo
6	Londres
7	Río de Janeiro
8	Boston
9	San Petersburgo
10	Roma

Source: INRIX

BBC

Source: BBC MUNDO

Traffic Congestion in Bogotá, Colombia

Causes	Consequences
• Increase of vehicle fleet • Road infrastructure backwardness • Badly programmed traffic lights	• High travel times • Financial problems • Environmental problems

Proposed Approach

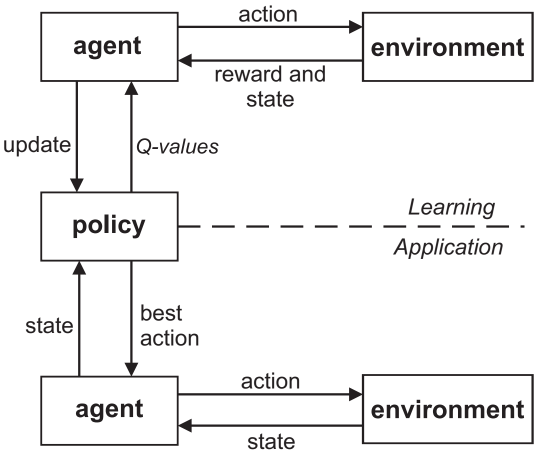
Generate a traffic light signal control strategy with the following features:

- Sensitive to traffic
- Independent of the mathematical model of the system
- Seeking to minimize specific goals

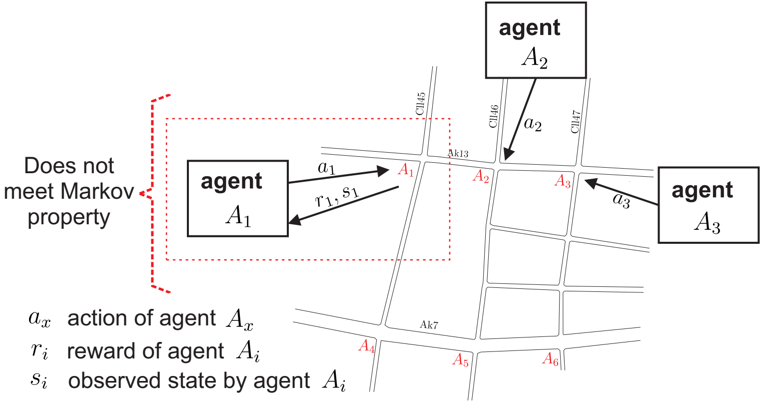
} Learn a policy based on the experience with the system → Multiagent Reinforcement Learing (MARL)

Reinforcement Learning - RL

One single agent:



Multiagent Reinforcement Learning - MARL



- The entire system can be described as a collaborative multiagent MDP model.

Multiagent Reinforcement Learning - MARL

A collaborative multiagent MDP model is described by:

- Discrete time k
- A set of n agents $A_1, A_2, \cdots, A_n$
- A finite set of states $\mathbf{s}^k \in \mathcal{S}$
- A finite set of joint actions $\mathbf{a}^k \in \mathcal{A}$
- A reward function $R_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that gives to agent i a real reward r_i^k

Where $\mathbf{R}(\mathbf{s}^\mathbf{k}, \mathbf{a}^\mathbf{k}) = \sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} \mathbf{R_i}(\mathbf{s}^\mathbf{k}, \mathbf{a}^\mathbf{k})$

Motivation

Decisions made at the individual level should result in decisions close to the optimal for the group.

Multiagent Reinforcement Learning - MARL

In a coordinated RL:

The global function Q can be split into a linear combination of the Q functions for each agent:

$$Q(\mathbf{s},\mathbf{a}) = \sum_{i=1}^{|\mathcal{N}|} Q_i(s_i, a_i)$$

General update rule for the multiagent case:

$$Q_i(s_i^k, a_i^k) := Q_i(s_i^k, a_i^k) + \alpha \left[R(\mathbf{s}^k, \mathbf{a}^k) + \gamma \max_{a' \in \mathcal{A}} Q(\mathbf{s}^{k+1}, \mathbf{a}') - Q_i(s_i^k, a_i^k) \right]$$

Multiagent Reinforcement Learning - MARL

In a coordinated RL:
The global function Q can be split into a linear combination of the Q functions for each agent:

$$Q(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^{|\mathcal{N}|} Q_i(s_i, a_i)$$

General update rule for the multiagent case:

$$Q_i(s_i^k, a_i^k) := Q_i(s_i^k, a_i^k) + \alpha \left[R(\mathbf{s}^k, \mathbf{a}^k) + \gamma \max_{a' \in \mathcal{A}} Q(\mathbf{s}^{k+1}, \mathbf{a}') - Q_i(s_i^k, a_i^k) \right]$$

Coordination in MARL

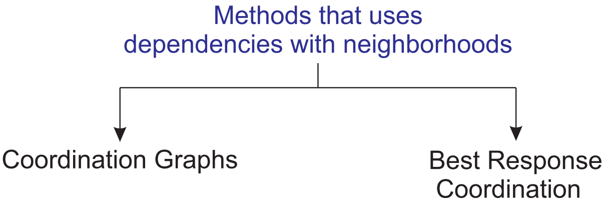
The problem of coordination is to find at each step the joint action:

Coordination Problem

$$\mathbf{a}^* = \mathrm{argmax}_{\mathbf{a}' \in \mathcal{A}} \, Q\left(\mathbf{s}^{k+1}, \mathbf{a}'\right)$$

Approaches to establish coordination

For the transit system: the action of each agent affects mostly the state around his neighborhood than away from it.



Method 1: Q-Learning and coordination graphs

- The graph $G = (\mathcal{V}, \mathcal{E})$ represents problems where agent i needs to coordinate actions with its neighbors $\Gamma(i)$.
- Allows to discompose the global Q function by edges.

$$Q(\mathbf{s}, \mathbf{a}) = \sum_{(i,j)\in\mathcal{E}} Q_{ij}(s_{ij}, a_i, a_j)$$

- Multiagent version of Q-Learning:[1]:

$$Q_{ij}^{k+1}(\mathbf{s}_{ij}^k, a_i^k, a_j^k) = (1-\alpha)Q_{ij}^k(\mathbf{s}_{ij}^k, a_i^k, a_j^k) + \alpha\left[\frac{r_i^{k+1}}{|\Gamma(i)|} + \frac{r_j^{k+1}}{|\Gamma(j)|} + \gamma Q_{ij}^k(\mathbf{s}_{ij}^{k+1}, a_i^*, a_j^*)\right]$$

[1] Proposed by: J. Kok in *Cooperation and Learning in Cooperative Multiagent Systems.* Ph.D thesis, University of Amsterdam, 2006.

Method 1: Q-Learning and coordination graphs

- The graph $G = (\mathcal{V}, \mathcal{E})$ represents problems where agent i needs to coordinate actions with its neighbors $\Gamma(i)$.
- Allows to discompose the global Q function by edges.

$$Q(\mathbf{s}, \mathbf{a}) = \sum_{(i,j)\in\mathcal{E}} Q_{ij}(s_{ij}, a_i, a_j)$$

- Multiagent version of Q-Learning:[1]:

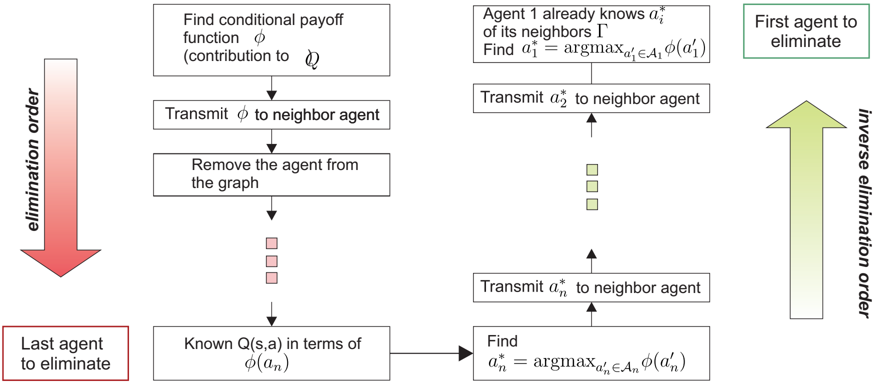
$$Q_{ij}^{k+1}(\mathbf{s}_{ij}^k, a_i^k, a_j^k) = (1-\alpha)Q_{ij}^k(\mathbf{s}_{ij}^k, a_i^k, a_j^k) + \alpha\left[\frac{r_i^{k+1}}{|\Gamma(i)|} + \frac{r_j^{k+1}}{|\Gamma(j)|} + \gamma Q_{ij}^k(\mathbf{s}_{ij}^{k+1}, a_i^*, a_j^*)\right]$$

$$a_i^*, a_j^* \in \underset{\mathbf{a}'\in\mathcal{A}}{\operatorname{argmax}}\, Q(\mathbf{s}, \mathbf{a}')$$

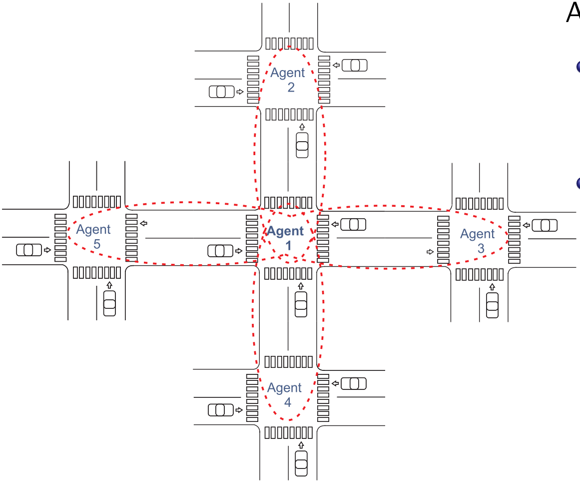
[1] Proposed by: J. Kok in *Cooperation and Learning in Cooperative Multiagent Systems.* Ph.D thesis, University of Amsterdam, 2006.

Method 1: Q-Learning and coordination graphs

Variable Elimination Algorithm (VE): solves the coordination problem, finding $\mathbf{a}^* = \text{argmax}_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$



Method 2: Q-Learning and Best Response



Agent i:

- Needs to coordinate actions with neighborhood $\Gamma(i)$
- Plays in a two-player game with each neighbor $j \in \Gamma(i)$

Method 2: Q-Learning and Best Response

Agent i for each time step k:

1. Estimate the likelihood of action selection for each neighbor:

$$\theta_{ij}\left(s_{ij}^{k-1}, a_j^{k-1}\right) = \frac{v(s_{ij}^{k-1}, a_j^{k-1})}{\sum\limits_{a_j \in \mathcal{A}_j} v(s_{ij}^{k-1}, a_j)}$$

Method 2: Q-Learning and Best Response

2. Update Q values with each neighbor:

$$Q_{ij}^{k}\left(s_{ij}^{k-1}, a_{ij}^{k-1}\right) = (1-\alpha)\, Q_{ij}^{k-1}\left(s_{ij}^{k-1}, a_{ij}^{k-1}\right) + \alpha\left[r_i^k + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q(\mathbf{s}^k, \mathbf{a}')\right]$$

Method 2: Q-Learning and Best Response

2. Update Q values with each neighbor:

$$Q_{ij}^{k}\left(s_{ij}^{k-1}, a_{ij}^{k-1}\right) = (1-\alpha)\, Q_{ij}^{k-1}\left(s_{ij}^{k-1}, a_{ij}^{k-1}\right) + \alpha\left[r_i^k + \gamma \mathrm{br}_i^k\right]$$

$$br_i^k = \max_{a_i \in \mathcal{A}_i}\left[\sum_{a_j \in \mathcal{A}_j} Q_{ij}\left(s_{ij}^k, a_{ij}\right) \times \theta_{ij}\left(s_{ij}^k, a_j\right)\right]$$

Best response

- Payoff $Q_i()$
- Likelihood θ_{-i} over the neighbor's strategy

Strategy $a_i \in \mathcal{A}_i$ for player i is a *best response* if for all $a_i{}'$ satisfies:

$$Q_i(a_i, \theta_{-i}) \geq Q_i(a_i{}', \theta_{-i})$$

Method 2: Q-Learning and Best Response

3. Select best response action at the neighborhood level:

$$a_i^* = \underset{a_i \in \mathcal{A}_i}{\operatorname{argmax}} \left[\sum_{j \in \Gamma(i)} \sum_{a_j \in \mathcal{A}_j} Q_{ij}\left(s_{ij}^k, a_{ij}\right) \times \theta_{ij}\left(s_{ij}^k, a_j\right) \right]$$

[2] Proposed by: El-Tantawy *et al.* en *Multiagent Reinforcement Learning for MARLIN-ATSC.* IEEE Transactions on Intelligent Transportation Systems, 2013.

States and Actions

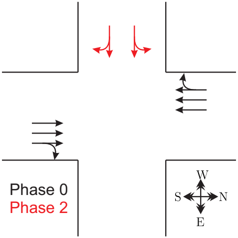
State

For an agent with i edges, the state vector has the following items:

- Hour (h)
- Maximum queue length (in vehicles) in edge i (q_i)
- Queuing delay (in minutes) of stopped vehicles in edge i (w_i)

Actions

Phase to apply (right of way to one or more nonconflicting movements). For example:



Reward Function

$$r_i = -\sum_{k=1}^{edges} \beta_q (q_k)^{\theta_q} + \beta_w (w_k)^{\theta_w} \quad \forall i \in \mathcal{N}$$

$$\beta_q, \beta_w, \theta_q, \theta_w \in [0, 1]$$

$$\beta_q + \beta_w = 1$$

Where:

- *edges:* number of approaches of agent i
- q_k and w_k : maximum queue length and queuing delay in edge k
- β_q and β_w: coefficients to set priority
- θ_q and θ_w: to balance queue lengths and waiting times across approaches

Reward Function

$$r_i = -\sum_{k=1}^{edges} 0.3(q_k)^{1.75} + 0.7(w_k)^{1.75} \quad \forall i \in \mathcal{N}$$

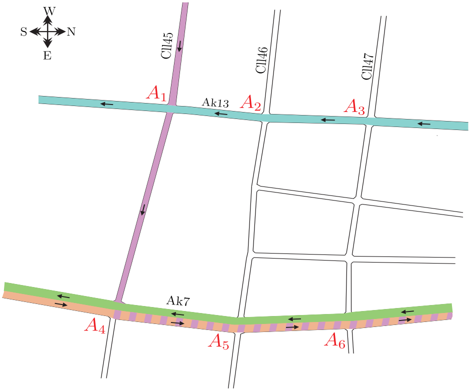
$$\beta_q, \beta_w, \theta_q, \theta_w \in [0, 1]$$

$$\beta_q + \beta_w = 1$$

Where:

- *edges:* number of approaches of agent i
- q_k and w_k : maximum queue length and queuing delay in edge k
- β_q and β_w: coefficients to set priority
- θ_q and θ_w: to balance queue lengths and waiting times across approaches

Test Framework



Route 1: Ak7 north to south
Route 2: Ak7 south to north
Route 3: Ak13 north to south
Route 4: Ak13-Cl45 to Ak7-Cl47

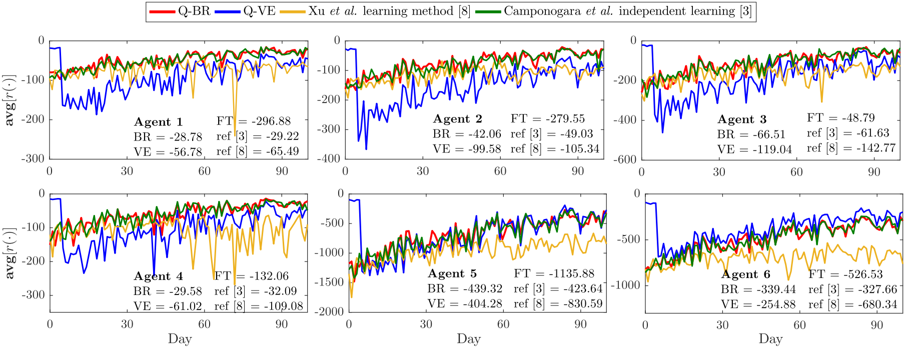
Data of vehicular flow and fixed-time control provided by the District Mobility Office

Simulation Setup:

- SUMO as traffic simulator
- Agent control through TraCI environment
- Training using Amazon Elastic Compute Cloud (Amazon EC2)
- Duration: 36 hours aprox.

Experimental results

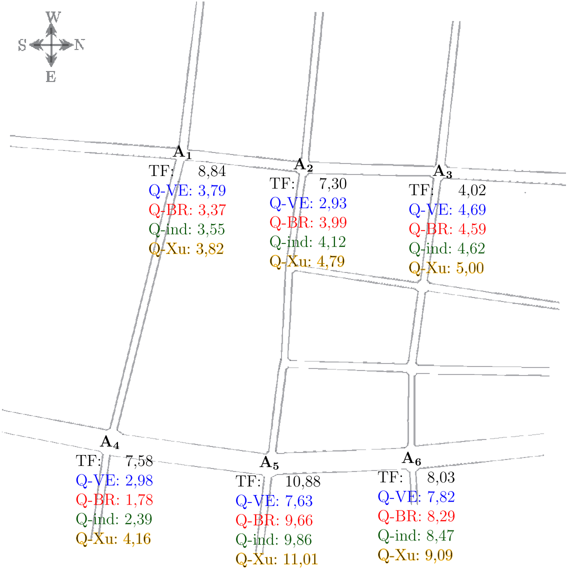
Learning curves



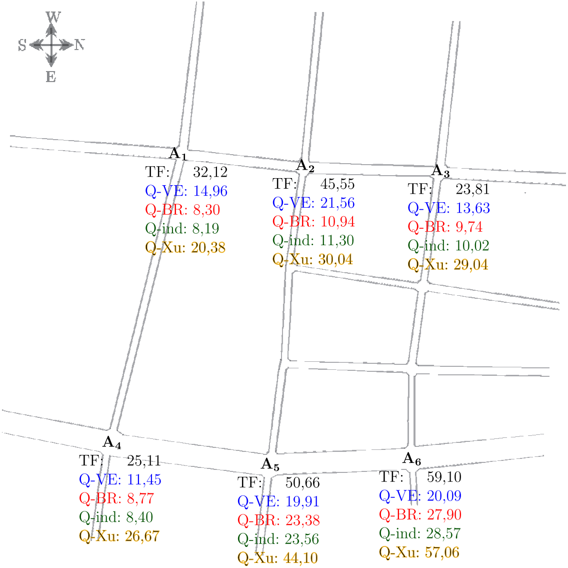
- Maximum average queue length per intersection (veh)
- Average queuing delay per vehicle (s/veh)
- Average speed (m/s)
- Travel time for selected routes

Experimental results

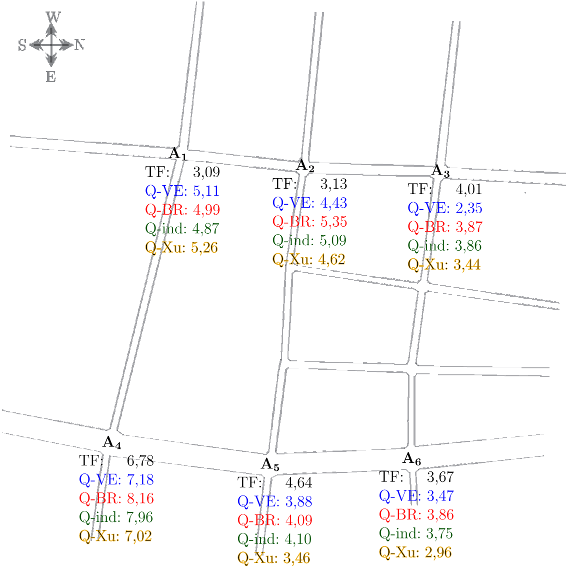
Maximum average queue length per intersection (veh)



Average queuing delay per vehicle (s/veh)



Average speed (m/s)



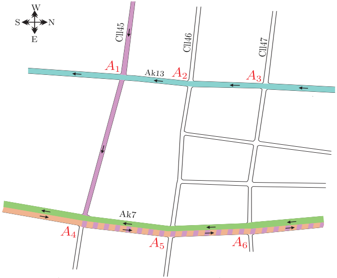
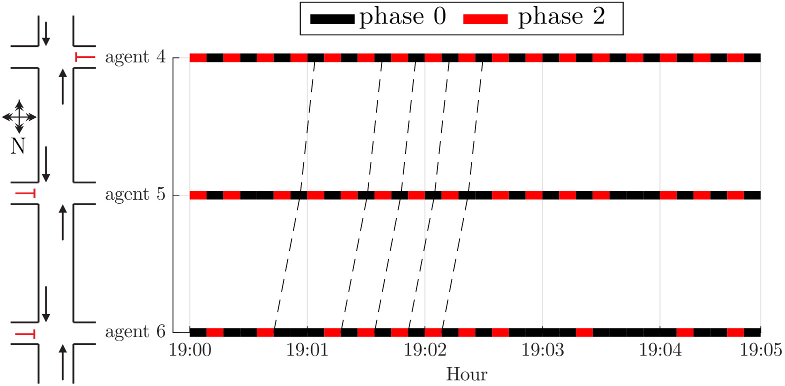


Table: Average travel time (in minutes) for selected routes using fixed time control and the policies learned

Method	*Route 1*	*Route 2*	*Route 3*	*Route 4*
FT	2.41	4.17	1.65	5.58
Q-VE	1.74	**2.17**	1.41	2.90
Q-BR	**1.52**	2.33	1.04	**2.75**
Q-ind [?]	2.44	3.26	**0.93**	3.72
Q-Xu [?]	4.20	5.33	1.02	5.67


phase 0
phase 2
N
agent 4
agent 5
agent 6
19:00
19:01
19:02
19:03
19:04
19:05
Hour

Conclusions

- Q-VE and Q-BR reduces average waiting time per vehicle for more than 55%, and average queue length by intersection by more than 30%.
- The policies obtained prioritize green waves along routes where the major demand is.
- Distributing the reward function into contribution per agent simplifies the problem.

Conclusions

- Q-VE and Q-BR reduces average waiting time per vehicle for more than 55%, and average queue length by intersection by more than 30%.
- The policies obtained prioritize green waves along routes where the major demand is.
- Distributing the reward function into contribution per agent simplifies the problem.

	Coordination graph	*Best response*
Determination of $\mathbf{a}^*$	Exact, running VE	An approx. at neighborhood level
Scalability	Not easily	Completely
Communications between agents	Subject to change	Defined *a priori*

Questions?