sciendo

Transport and Telecommunication, 2023, volume 24, no. 1, 1–12
Transport and Telecommunication Institute, Lomonosova 1, Riga, LV-1019, Latvia
DOI 10.2478/ttj-2023-0001

USING MACHINE LEARNING TECHNIQUES TO INCORPORATE SOCIAL PRIORITIES IN TRAFFIC MONITORING IN A JUNCTION WITH A FAST LANE

Orly Barzilai[1], Havana Rika[1], Nadav Voloch[2], Maor Meir Hajaj[3], Orna Lavi Steiner[4], Niv Ahituv[5]

[1]Information Systems Department, The Academic College of Tel Aviv-Yaffo, Yaffo, Israel
orlyba@mta.ac.il
[2]Computer Science Department, Ben -Gurion University of the Negev
Beer-Sheva, Israel
[3]Information Systems Department, University of Haifa
Haifa, Israel
[4]Industrial Engineering and Management Department, Ruppin Academic Center
Emek Hefer, Israel
[5]Coller School of Management, Tel Aviv University, Tel Aviv, Israel and Peres Academic Center
Rehovot, Israel

Traffic lights monitoring that considers only traffic volumes is not necessarily the optimal way to time the green/red allocation in a junction. A "smart" allocation should also consider the necessities of the vehicle's passengers and the needs of the people those passengers ought to serve.

This paper deals with a "smart" junction, where several cars approach the intersection from different directions and a traffic light is set to comply to a sequence of time intervals of red and green lights in each direction. The novel approach presented here is based not only on traffic congestion parameters, but also on the social and economic characteristics of the passengers (e.g. a handicapped person, a medical doctor, an employee who is extremely required in a certain organization due to an emergency situation). This paper proposes to enhance the smart junction with a fast lane, which has a flexible entry permit based on social and economic criteria. Machine learning (specifically, Reinforcement Learning (RL)) is added to the junction's algorithm with the aim of optimizing the social utility of the junction. For the purposes of this study, the utility of the junction is defined by the total social and economic potential benefits given a certain red/green time allocation is set. This is defined as the measure of the reward function which contains positive factors for vehicles which crossed the junction or advanced their position and a negative factor for vehicles which remains in their positions. In addition, a weight value for the vehicles with high priority is also part of the equation.

A simplified version of the smart junction has been used, serving as a model for incorporating RL into the "smart' junction with Fast Lane (FL). Specifically, the Q-Learning algorithm is used to maximize the reward function. Simulation results show that prioritizing high priority vehicles via FL is influenced by the weights and factors given to the reward components. Farther research should enhance the "Smart" junction with FL to a more complex and realistic one using a varying amount of vehicles crossing the junction.

Keywords: smart junction, Internet of Things, real-time algorithms, social dilemmas in traffic control, machine learning, Reinforcement Learning

1. Introduction and background

The dramatic growth of the population and the number of vehicles requires efficient design and sustainability of traffic systems that take full advantage of modern-day technology (Ata *et al.*, 2019; Patan *et al.*, 2016; Van Arem *et al.*, 1997; Abbas *et al.*, 2022; Bokaba *et al.*, 2022). Population increase, immense scale of vehicle trade and lack of efficient public transportation systems are the foremost reasons for traffic saturation (Patan *et al.*, 2016). Efforts devoted to mitigating the negative effects of traffic congestion have been invested in three directions: increasing infrastructure, promoting transport alternatives and managing traffic flows (Van Arem *et al.*, 1997).

The first of the three directions is limited by topographical, budgetary and social factors (Van Arem *et al.*, 1997). The second direction is based on public transportation and ride-sharing services. Ride-sharing services are already increasing congestion in cities (Schaller, 2017), and the expected autonomous vehicle (AV) on-demand services require large fleet sizes to satisfy the total travel demand – in the city's core as well as outskirts – further contributing to congestion (Ben-Dor *et al.*, 2019).

One of the transport alternatives is a Dedicated Bus Lanes (DBL) which are dedicated sections, sometimes short, implemented on loaded parts (Russo *et al.*, 2022), that aims to significantly improve the performance of bus services, while encouraging a mode switch from private cars to public transport (Ben-Dor *et al.*, 2018). The DBL provide a low cost and easily implementable strategy to improve transit services by Russo *et al.* (2022) analysed traffic data from 27 roads sections and showed that DBL reduces bus travel time by about 18 percent and waiting times at stops by about 12 percent.

Another transport alternative is the fast lanes for high-occupancy vehicles (HOV), free of charge (DeCorla-Souza, 2000; Boysen *et al.*, 2021). In most applications, at least one or two passengers must accompany the driver. Often, additional vehicles can use these lanes, such as emergency and law enforcement vehicles, public buses, and electric vehicles. The basic objective is to reduce the traffic load by increasing the average vehicle occupancy. In order to encourage carpool cash rewards for car-poolers and transit riders is suggested (DeCorla-Souza *et al.*, 2022). The fast lane can be used by single-occupant vehicles, at the cost of a toll. The authors in (DeCorla-Souza, 2000) suggest that fast lanes would be electronically tolled express lanes, where tolls are set in real time to limit traffic to the free-flowing maximum. Tolls are structured to vary according to the level of traffic in the fast lanes. As traffic volume approaches capacity, the tolls are increased to deter entry by additional vehicles into the lane.

The third direction, managing traffic flows, has been continuously improving over the last decades with the expansion of data provided by sensors in roads and vehicles, as well as with the advancement of technology that can exploit those data (Chong and Ng, 2016; Rizwan and Babu, 2016; Mannion *et al.*, 2015). This enables measuring, modelling and interpreting traffic features such as flow, occupancy or travel times, which are useful for developing Intelligent Transportation Systems (ITS) (Mannion *et al.*, 2015), including both advanced traffic management systems and advanced traveller information systems (Lana, *et al.*, 2018).

Optimal Traffic Signal Control (TSC) using dynamic cycle of traffic lights instead of a fixed cycle, in which red and green lights are adjusted according to the density of the traffic has been suggested (Firdous and Niranjan, 2020). Different algorithms for smart transportation and smart junctions using traffic congestion estimation and monitoring have been developed with the help of sensors and cameras in the framework of the Internet of Things (IoT), alongside with the implementation of various technologies such as Artificial Intelligence (AI) and image processing. All these enable the traffic light controller to make time setting decisions based on real-time data (Ata *et al.*, 2019; Chong and Ng, 2016; Lana, *et al.*, 2018; Firdous and Niranjan, 2020; Patil *et al.*, 2020; Hartanti *et al.*, 2019). Finding the best approach for optimizing traffic lights is still an open question for researchers. However, one promising approach for optimum TSC is to use machine learning paradigms based on AI techniques (Haydari and Yilmaz, 2020). Reinforcement Learning (RL) is one of these paradigms (Miletić *et al.*, 2022; Noaeen *et al.*, 2022).

RL is the problem faced by an agent that must learn behaviour through trial-and-error interactions with a dynamic environment (Kaelbling *et al.*, 1996). As a learning problem, it refers to learning to control a system so as to optimize some numerical values which represent long-term objectives (Szepesvári, 2010). The numerical values are the results of a reward function (Sutton and Barto, 1999). The reward function defines what is objectively good and bad for the agent. The value function is the agent's current mapping from the set of possible states (or state-action pairs) to its estimates of the net long-term reward to be expected after visiting a state (or a state-action pair), after which the agent continues to act according to its current policy (Sutton and Barto, 1999). Moreover, the learner is not told which actions should be taken, as in many forms of machine learning, but instead must discover which actions yield the highest reward by trying them out (Sutton and Barto, 2018). In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. RL can be classified into two categories: model-free (e.g. Q-Learning, SARSA) where the reward function is unknown and explored by interacting with the environment, and model-based (e.g. Dyna, Prioritised Sweeping) where the reward function is known (Mannion *et al.*, 2016).

In RL-TSC, each smart junction is typically controlled by a single agent (Mannion *et al.*, 2016). Each agent has the responsibility for determining the light switching sequence at its assigned intersection. Many different objectives have been considered by authors when defining the reward function used by RL-TSC agents (Mannion *et al.*, 2016). These may include average trip waiting time, trip delay, average trip time, average junction waiting time, junction throughput/flow rate, achieving green waves, accident avoidance, speed restriction, fuel conservation and average number of trip stops. It has been shown that RL is a promising approach when applied to urban traffic signal control, and it offers many benefits (Mannion *et al.*, 2015). RL agents can learn continuously how to improve their performance, as well as how to adapt readily to changes in traffic demand.

Previous studies (Barzilay *et al.*, 2018a; Barzilay *et al.*, 2018b; Fine *et al.*, 2019; Barzilay *et al.*, 2020) have expanded the concept of a smart junction by introducing a social preference and a benefit factor for junction traffic management, combined with traffic load. Social preference is based on the social

characteristics of the vehicle passengers (e.g. a handicapped person, a driver with no previous traffic violations), while a benefit factor is based on the passenger utility of promoting the vehicle priority at the intersection.

The innovative idea behind adding social factors to a smart junction or to a preferred lane is based on the argument that increasing the social fairness of the junction or a lane will result in more tolerant driving and a reduction of road accidents. It may urge drivers who do not carry priority to switch to public transportation or to delay their travel to a less congested time. Results showed that people accept moral norms as a criterion for traffic load management. When prioritizing vehicles, people gave higher preference to passengers with special needs and difficulties (good virtues), passengers using car sharing and public transportation (benefit of all), and passengers who serve the community by providing services within different domains such as medical or security. A smart junction with social preference streamlines the traffic faster than a regular junction (a junction where a green light is allocated in a predefined order, where the time slot does not depend on the traffic load).

From an economic perspective, in terms of benefit-to-cost ratio (BCR), smart congestion solutions yield higher returns compared to just expanding road capacity (McKinsey, 2013). In recent years, new transportation forms have emerged, especially Mobility as a Service (MaaS) (Schaller, 2017). In addition, the Covid-19 pandemic has changed the way people travel by reducing the use of public transportation. A pluralistic approach towards the different road users, and their needs and constraints, will make it possible to prioritize travel in a more elaborate manner. Smart integration of public and private transportation is required as part of a pluralistic approach to implementing smart transportation.

The current study aims to expand upon our previous works on smart junctions with social priorities by adding smart fast lanes with economic and social priorities for different road users, coordinated with public transportation. Smart traffic lanes should, therefore, enable a heterogeneous mix of transport modes. The rules permitting vehicles to use fast lanes are usually related to the type of the vehicle or the number of passengers in the vehicle. For example: public transportation such as buses and taxis can almost always exploit the fast lane; private cars can do that when the number of passengers exceeds a certain threshold (2 or 3, usually).

Assuming that the loss of working hours due to road congestion inhibits economic growth, and that public transport infrastructure is still deficient, preference should be given to extending the criteria for using existing infrastructure (such as fast lanes) by private vehicles. For example:

1. Cars carrying "very valuable" employees (such as employees who make a high contribution in an important organization or employees whose absence impairs the productivity of others) should have priority in using the "privileged" lane.
2. Cars carrying handicapped persons should get privilege of using those lanes.
3. Taxis that do not carry passengers should not have that permit, depending on the type of the passenger waiting for the ride.

This study intends to "think out of the box" and try to broaden the spectrum of considerations related to traffic control by providing additional criteria for fast lane users. The study intends to develop the research model, which will be used in farther researches, for applying RL method called Q-learning, for traffic allocation. For that, a basic and simplified junction is used.

2. Method

Our recent study models a complex smart junction that simulates a real-world junction, with all of the necessary and significant parameters that must be addressed in order to control traffic ideally. The suggested smart junction incorporates our innovative approach, which adds social priorities and FL. FL has flexible entrance criteria for high social priority vehicles. Junction green light allocation is optimized by RL algorithm, with its hyper parameters. In the future, we plan to implement our full model, based on the results and insights gained from the simplified simulation version, presented in section 2.2. Section 2.1 contains description of our fully smart junction simulation system. Section 2.2 contains description of the simplified smart junction simulation, presented in this research and the RL algorithm implementation using the Q-Learning algorithm.

2.1. Full smart junction model with social priorities, fast lane and prioritization by machine learning techniques

The current study proposes a smart junction simulation description which involves social priorities based on previous researches (Barzilay *et al.*, 2018a; Barzilay *et al.*, 2018b; Fine *et al.*, 2019; Barzilay *et al.*, 2020). FLs are added to the simulation. To simulate a junction in the most realistic and general way possible, the junction includes several lanes in different directions, covering all possibilities to pass the junction (e.g.

driving straight, turning left or right etc.) to make the junction as modular as possible. Furthermore, fast lanes are added to the junction with, with a flexible entering criterion that is set using ML algorithm. The vehicles are constantly arriving in every lane and waiting to pass the junction.

We proposed a unique algorithm attribute - a social priority for every vehicle (such as the priority evaluation of a person late for work, or for a person late for a doctor's appointment, etc.), where a special interrupting priority is given to various emergency vehicles. The social priority is simulated by assigning a number chosen at random from 1 to max to each of the vehicles, where max is the highest priority value. The uniform random distribution can be changed to reflect the real-life distribution, which can vary depending on rush hours, junction geographical location, and so on. A more complex option is to associate a social priority with each one of the vehicle's passengers (including the driver). In that case, the vehicle priority can be calculated as a weighted sum of these passengers' priorities or by a max function on all of them.

This novel approach was refined and adapted to new real-life vehicle and junction parameters. The parameters taken into consideration were the vehicle's length, average speed, type (public transport with multiple priorities or private), and for the junction – its size (crossing length), the number of lanes in every direction and the maximal green-time per inflow lane.

After inputting these different attributes, the goal of our algorithm is to give an efficient and fair traffic light timing-schedule that reflecting the most socially and conditionally appropriate traffic scheme in a given amount of time. For example, the algorithm needs to give an emergency vehicle, if exists, the first time slice, until it passes the junction, and then the rest of the queues get their time slices. The time parameter can be calculated as a function of the vehicle lengths and speeds, as well as the junction dimensions. In the algorithm, we defined the term "conflict side" as follows: a conflict side is all the lanes in all the directions of a traffic junction that can be opened simultaneously. A general scheme of the smart junction simulation is presented in Figure 1.

The current research adds three important aspects that improve the method substantially, as described below. The first major advantage and novelty of this algorithm is that it takes into consideration both current traffic loads and current social priorities when deciding which conflict side to open in the next step and for how long.

The second one is to use a machine learning algorithm, specifically Reinforcement Learning agent that will control and manage the traffic lights of the junction. The RL algorithm contains hyper-parameters that need to be set in order to achieve an optimal result, thus creating an efficient method of optimizing the utility of the junction.

The third aspect is the addition of smart fast lanes, with economic and social priorities for different road users and coordinated with public transportation. The rules permitting vehicles to use fast lanes are usually related to the type of the vehicle or the number of passengers in the vehicle. For example: public transportation such as buses and taxis can almost always exploit the fast lane. In our model we determine a certain social priority threshold that is dynamic by nature and can be manage and varies dynamically as a function of the load and current priorities in the junction for example.

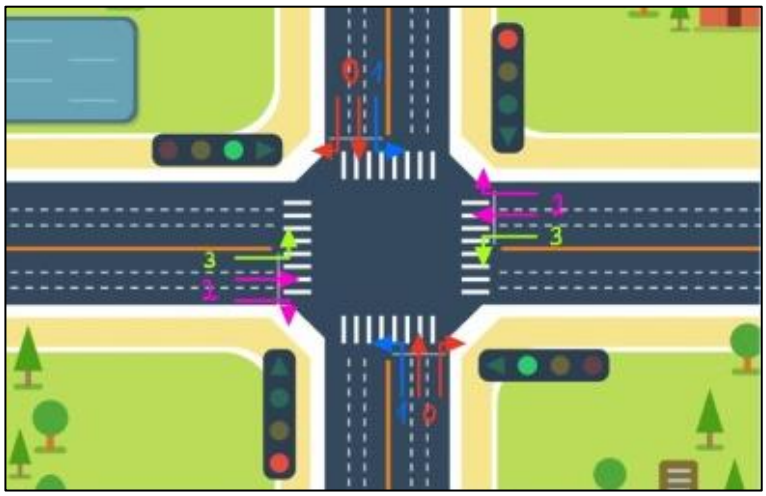


Figure 1. The smart junction simulation's scheme with 4 conflict sides coloured red, blue, pink and green

2.2. The simulation and the RL algorithm

Following is a detailed description of our simplified smart junction simulation. The junction contains two lanes, one in a horizontal direction, and the second one in a vertical direction. Each lane is implemented by two queues, one for vehicles that wait to pass the junction and a second queue for those that already had passed. The junction does not allow turns, therefore every vehicle can only pass it by continuing to drive in the same direction, for example, a car on the horizontal lane can only drive horizontally. As a result, there are exactly two conflict sides in that junction, the first opening the green light to the horizontal lane and the second opening the green light to the vertical lane.

Two types of priorities are been used, regular (lowest social priority) and important (highest social priority). Let x be the number of regular vehicles and ley y be the number of important ones. The regular vehicle priority is set to 1, and the important vehicle priority is determined by a function of x and y. In particular, assuming that $x > y$ we set the highest priority to be $c \cdot x/y$, where c stands for some positive constant. All the vehicles in our simplified simulation have equal size and speed, in other words, it takes the same time to each of them to pass the junction. Moreover, every time step of a green light allows to only one car at a time to pass the junction from one side to the other, which causes to all the vehicles in that direction (those who are waiting to pass and those who already passed the junction) to drive one step forward in the queues' lanes.

The junction initialize state determined by setting x regular vehicles in the vertical lanes and y important cars in the horizontal lanes that wait to pass the junction, while a final state is determined when all the vehicles had passed the junction. Noting that the horizontal lane in our simulation represents the FL, where the vertical lane represents the regular one, as can be seen in Figure 2. Thus, the goal of our RL algorithm is to alternatingly open the green light of both directions, but with some priority to the FL (horizontal direction) without starvation of the other lanes.

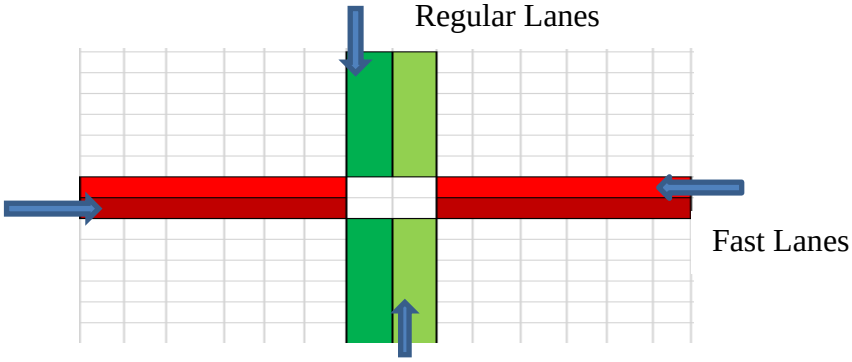


Figure 2. Schema of SJ simple version

Before describing the Q-learning algorithm, we define a few required notions as follows. A state is a snapshot of the junction, which is defined by the position of all the vehicles in the junction. Note that all the states contain the same amount, priority and order of vehicles in the lanes, but they differ in their vehicle's positions in the junction, i.e. the ones who passed the junction and the ones who doesn't. Each state has a unique identity number. Our simulation contains exactly two possible actions – opening green light horizontally or vertically. Note that operating an action on a current state transfer us to a different state. The set of actions that took us from initialize junction state until we get to its final state is called an episode. An agent decides in each time step which action to execute on its current state according to the policy of the Q-learning algorithm describe next. An iteration run many numbers of episodes until the agent policy is stabilized. The policy is the element that needs to be learned, which tries to predict the action that will drive us to a new state with high reward.

Every state has a reward value which is calculated as a function of all the vehicles who passed the junction and the ones who still wait to pass it, taking into consideration their priorities too. In our simplified version, the reward function of a given state equals to a weighted sum of priorities as follows. Let $0 < \omega_a < \omega_p < 1$ be two positive constants and let $-1 < \omega_n < 0$ be a negative constant. Every vehicle in a current state with priority p, contributes $\omega_p \cdot p$ to the reward if it is the last vehicle that passed the junction among all the one who did, otherwise it contributes $\omega_a \cdot p$ to the reward if that vehicle got advanced in the queue (i.e. it got a step forward towards passing the junction), and it contributes $\omega_n \cdot p$ to the reward if that vehicle still waits to pass the junction.

The Q-Learning algorithm calculates a q-value for every pair of state and action, and keeps that value in a predefined table initialized to zero, that contains row for each state and column for every possible action. The Q-Learning algorithm fill the table in a lazy fashion, i.e. when the agent need to know which action to execute in the current state s, it calculate the q-values for all the possible actions (opening the green light horizontally or vertically) on that current state, i.e. $Q(s, horizontal)$ and $Q(s, vertical)$ and choose to execute the action with the maximum value among them. That maximum q-value defines the policy of the algorithm, as it determines which action to choose and execute for every given state. These q-values are calculated by the following formula (see Figure 3)

$$Q^{new}(s,a) = Q^{old}(s,a) + \alpha(r + \gamma \cdot \max_a Q^{old}(s',a) - Q^{old}(s,a)),$$

where s stands for the current state, a stands for the action, s' stands for the next state that we will get after executing the action a on the state s, r stands for the reward of state s, the constant $0 \leq \gamma \leq 1$ stands for the discount factor which determines how much our agent cares about rewards in the distant future relative to those in the immediate future, and the constant $0 \leq \alpha \leq 1$ stands for the learning rate (a small number that controls how large are the updates to the old values). As can be seen from the above formula, the policy wishes to choose the action that will make us closer to the state with the maximize reward, which in our case is equal to the final state.

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}}\Big)}^{\text{temporal difference}}$$

Figure 3. The Q-value formula

Since the q-value is calculated for every state that we arrived to and the action is chosen by the calculated policy, there is a chance that the algorithm will not be able to get to all the possible states and actions. Thus, in every step, with ϵ probability we select to execute an action uniformly at random, and in $1 - \epsilon$ probability we select an action according to the policy. The constant $0 \leq \epsilon \leq 1$ is known as the exploration probability, and its value can be decreased as the number of iterations increases and the q-values stabilized, see the pseudo code of the Q-Learning Algorithm in Figure 4.

Pseudo code of the Q-Learning Algorithm:

1. Create a table Q of size $\#states \times \#actions$ and set all its entries to 0.
2. Create a vector of size $\#states$ and keep the reward of each state.
3. Runs N episodes (an iteration):
    a. For each episode:
        i. Given a current state s calculate and update the new values $Q(s, a_1)$ and $Q(s, a_2)$ in table Q, where a_1 and a_2 stand for the 2 possible actions (horizontal and vertical).
        ii. with ϵ probability, choose randomly the next action.
        iii. Otherwise, i.e. with $1 - \epsilon$ probability, choose the action a_i which maximize the Q-value, i.e. $argmax_{i=1,2}\{Q(s, a_1), Q(s, a_2)\}$
        iv. Moving to the next state s' by executing the selected action on the current state s.

Figure 4. Pseudo code of the Q-Learning Algorithm

The high-level code design is presented in Appendix A. The Python-based simulation built for this research implemented the simplified simulation described above together with the Q-learning-algorithm to manage the junction green light allocation. A visualization of the junction according to our simplified simulation system is presented in Figure 2, and the simulation conclusions and results are presented in the next section.

3. Results

Tables 1 and 2 presents the parameters values of the smart junction simulation. Table 1 presents the Q learning parameter's values and table 2 presents the reward function parameter's values. The simulation has been run for 35 episodes with the value of the epsilon parameter decreasing over time to lower the exploration component (starting from a value of 0.5 and ending with a value of 0.01). The epsilon component value is reflected by the number of random decisions made to select which conflict side (Horizontal or Vertical) will gain the green light. Table 3 presents the distribution of light allocation decisions by the randomization parameter. It can be seen that the number of randomized decisions is decreased to zero during the last episode, reflecting the probability for randomized decision of 0.01 (value of epsilon in the last episode). The last selections are signed as no selections because all vehicles in the Horizontal conflict side had crossed the junction and the remained selected activity is to give a green light to the Vertical conflict side.

Table 1. Q-learning parameters in smart junction simulation

Parameter	**Values**
Number of episodes	35
Epsilon	Starting value=0.5. Episode 07: value=0.4. Episode 14: value=0.3 Episode 21: value =0.2 Episode 28: value=0.01
Gamma	0.5
Learning Rate	0.5
Discount Factor	0.5

Table 2. Reward parameters in smart junction simulation

Parameter	**Values**
Number of regular cars (Reg)	10
Number of priority cars (Prio)	10
Priority Weight (PW)	3
Priority equation	PW* (Reg/Prio)
Passed car reward factor	1
Advanced car reward factor	0.5
Stayed car reward factor	-0.1

Table 3. Light allocation decisions by randomization for first and last episodes

Episode	**Random/Non-Random**
1 (First)	R, NR, NR, R, R, R, R, R, NR, NR, NR, R, R, NR, NR, NR, NR (No selection), NR (No selection), NR (No selection)
35 (Last)	NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR, NR (No selection), NR (No selection), NR (No selection)

Table 4 presents the Q values calculated for the initial position of the junction, which is the initial distribution of the vehicles at the beginning of each episode (presented in figure 2). Q values are presented for the first two episodes and the last 4 episodes. It can be seen that Q-values converge to 27.9880 for the horizontal lane and to 13.2941 for the vertical one.

Table 4. Q values calculated for the initial junction position by initial and final episodes

Episode	Q-Value (Horizontal)	Q-Value (Vertical)
1 (First)	7.75	1.25
2 (Second)	11.625	3.825
…		
32	27.9678	13.2941
33	27.9764	13.29414
34	27.9829	13.2941
35 (Last)	27.9880	13.2941

To examine the effect of the different parameters of the reinforcement function, we changed one parameter at a time and examined the effect of this change on the allocation of the green light between the

Horizontal (FL used by high priority vehicles) and Vertical (used by low priority vehicles) conflict sides. Table 5 presents the route selection, for three levels of priority weights, for the first and last episodes. In addition, the average delay per each route is calculated. Each route contains a series of H (horizontal) and V (vertical) conflict sides which represents the order in which the green light was given to one of the two conflict sides. Per each green light opened for a fixed interval of 1 millisecond, one vehicle crosses the junction. In order to calculate the average delay of the vehicles waiting in the junction, each conflict side in a series received numerical number based on its delay time. The first vehicle crossing the junction received a number of 1 representing 1 millisecond delay waiting for the green light. The second vehicle crossing the junction received a number of 2 representing 2 millisecond delay, ext. The average delay per each type of conflict side (H or V) is calculated based on the numerical numbers. It can be said that a lower value of the average delay, indicates less time to cross the junction for the matched conflict side. Results shows that the smallest average delay for the H lane (FL-high priority) was gained for priority weight value equals to 3 in addition to the biggest reduction of delay value from the first iteration to the last one.

Table 5. Green Light allocations by Priority weight and episodes (First and Last)

Priority Weight	Episode	Routes	H (FL) Average Delay	V Average Delay
3	1 (First)	V, H, H, H, V, H, V, V, H, H, H, H, H, V, V, H, V, V, V, V	8.6	12.4
	35 (Last)	H, H, H, H, H, H, H, H, V, V, H, V, V, V, V, V, H, V, V, V	6.4	14.6
2	1 (First)	H, H, H, V, V, H, H, H, H, H, V, V, V, H, V, V, H, V, V , V	7.7	13.3
	35 (Last)	H, H, H, H, H, H , V, V, H, V, V, H, V, V, H, V, V, H, V, V	7.5	13.5
1	1 (First)	H, H, H, V, V, V, H, V, H, H, V, V, H, V, H, V, H, V, H, V	9.6	11.4
	35 (Last)	H, V, H, V, H, V, H, V, H, H, V, V, V, H, H, V, H, V, H, V	10	11

In order to examine the impact of the three reward factors (passed, advanced and stayed), per each simulation (containing 35 episodes) only one reward factor, per simulation, was updated by multiplying the base factor values by 4. Table 6 presents the routes series during the last episode and the average delay of H (FL) lane, per base line values and for each change in one reward factor. Results showed that the crossed reward factor decreased the H average delay as compared to the base line by 1 point. The advanced reward factor decreased the H average delay by 0.2 and the stayed reward factor increased the average delay by 0.7.

Table 6. Green Light allocations for last episode by reward factors

Reword Parameter	Parameter Value	Last Episode - Routes	H (FL) Average Delay
Base Line: Passed Reward Factor Advanced Reward Factor Stayed Reward Factor	1 0.5 -0.1	H, H, H, H, H, H, H, V, H, V, V, H, V, V, V, H, V, V, V, V	6.5
Crossed Car Reward Factor	4	H, H, H, H, H, H, H, H, H, H, V, V, V, V, V, V, V, V, V, V	5.5
Advanced Car Reward Factor	2	H, H, H, H, H, H, H, V, V, V, H, V, V, V, H, V, V, H, V, V	7.2
Stayed Car Reward Factor	-0.4	H, H, H, H, H, H, H, H, V, V, H, V, V, V, V, H, V, V, V, V	6.3

4. Discussion and conclusion

This study elaborates the smart junction with social priority presented in previous research (Barzilay *et al.*, 2018a; Barzilay *et al.*, 2018b; Fine *et al.*, 2019; Barzilay *et al.*, 2020), by adding a fast lane dedicated to high priority vehicles. The traffic allocation is managed by RL algorithm, specifically the Q-Learning algorithm. RL appears to be a promising approach when applied to urban traffic signal control, and it offers many benefits (Mannion *et al.*, 2016; Mannion *et al.*, 2015).

The main idea in this study is to prioritize vehicles based on social and economic priorities, while maintaining efficient management of the junction traffic load. The novel approach is to achieve social and economic prioritization using FL combined with the RL method. Usually, FLs are used for high-occupancy vehicles, free of charge, or for all other vehicles, collecting some toll (DeCorla-Souza, 2000). The idea presented here is to use a flexible threshold as an entrance criterion to the fast lane based on social and economic priority values. The RL method is used to optimize the junction traffic flow in order to gain maximal reward.

The current study is aimed to define a base model that uses reinforcement learning for transportation management. For that purpose, a simple version of the smart junction was defined, contained two conflict sides, one for the horizontal direction, used as FL for high priority vehicles and another one for the vertical direction used as a regular lane for low priority vehicles. At every opening of a green traffic light, one vehicle is crossing the junction. The decision to which lane to give the green light is based on a Q-learning algorithm, encouraging exploration during the first episode and then gradually raises the share of the exploitation part, as reflected by gradually lowering the value of the epsilon component (see Tables 1 and 3). Per each episode, the amount of vehicles per each conflict side, and their positioning is fixed and predetermined. Each episode ends when all vehicles crosses the junction. The reward function is composed of three components: a positive reward for crossing the junction, a positive reward for promoting the position in the lane and a negative reward for staying in the same position. Each component has a weight factor value which influence the simulation traffic light scheduling. It should be specified that the smart junction target is to prioritize the FL over the regular lane while preventing a starvation of the regular lane. In addition, the reward function is influenced by the priority weight component given to the high priority vehicles, located in the FL.

The expression of the success implementation of the Q-Learning algorithm for the smart junction described above, can be seen from table 4, which shows that Q values are converged to certain values. In addition, Table 5 demonstrates the learning that took place throughout the episodes via the changes made in the selections of conflict side (H or V) between the first episode and the last one. The learning is expressed by the promotion of vehicles located in the FL (H) over the vehicles located in the regular lane (V).

The simulation findings, presented in tables 5-6, show that the prioritizing of the FL (Horizontal lane) over the regular ones (Vertical lane) is influenced by the weights and factors given to the reward components. Increasing the priority weight or the crossing reward factor reduce the average delay of the vehicles in the horizontal conflict side (FL), which means that the vehicles in the FL cross the junction faster. Changing these weights effects the balance between the travel time of the FL vehicles and the regular lane vehicles.

The novel idea of this study is to create a flexible fast lane which will ensure a speedy ride by adjusting its transport load using social and economic priorities. Future work should enhance the simple RL model presented here to a more complicated and realistic junction, receiving varying volumes of traffic changed over time

Acknowledgments

The authors would like to thank Gadi Pinkas and Daniel Rika for fruitful discussions on the subject of reinforcement learning. A great thanks is granted also for the Academic college of Tell-Aviv Jaffa for funding this research.

References

1. Ata, A., Khan, M. A., Abbas, S., Ahmad, G., & Fatima, A. (2019) Modelling smart road traffic congestion control system using machine learning techniques. *Neural Network World*, 29(2), 99-110.
2. Patan, R., Suresh, K., Babu, M.R. (2016) Real-time smart traffic management system for smart cities by using the Internet of Things and big data. In: *Emerging Technological Trends (ICETT), International Conference, IEEE*, 1-7.
3. Van Arem, B., Kirby, H. R., Van Der Vlist, M. J. M. and Whittaker, J. C. (1997) Recent advances and applications in the field of short-term traffic forecasting. *Int. J. Forecasting*, 13(1), 1–12.
4. Schaller, B. (2017) UNSUSTAINABLE? The Growth of App-Based Ride Services and Traffic, Travel and the Future of New York City. http://schallerconsult.com/rideservices/unsustainable.pdf.
5. Ben-Dor, G., Ben-Elia, E., & Benenson, I. (2019) Determining an Optimal Fleet Size for a Reliable Shared Automated Vehicle Ride-Sharing Service. *Procedia Computer Science*, 151, 878-883. https://doi.org/10.1016/j.procs.2019.04.121
6. Ben-Dor, G., Ben-Elia, E. and Benenson, I. (2018) Assessing the impacts of dedicated bus lanes on urban traffic congestion and modal split with an agent-based model. *Procedia computer science*, 130, 824-829.
7. Tsitsokas, D., Kouvelas, A. and Geroliminis, N. (2021) Modelling and optimization of dedicated bus lanes space allocation in large networks with dynamic congestion. T*ransportation Research Part C: Emerging Technologies*, 127, 103082.
8. DeCorla-Souza, P. (2000) Making the Pricing of Currently Free Highway Lanes Acceptable to the Public. *Transportation Quarterly*, 54(3), 17-20.

9. Chong, H. F. and Ng, Danny Wee Kiat (2016) Development of IoT device for traffic management system. In: *2016 IEEE Student Conference on Research and Development (SCOReD)*. IEEE, 2016.
10. Rizwan, P., Suresh, K. and Babu, M. R. (2016) Real-time smart traffic management system for smart cities by using Internet of Things and big data. In: *2016 international conference on emerging technological trends (ICETT)*. IEEE, 2016.
11. Mannion, P., Duggan, J. and Howley, E. (2015) Parallel reinforcement learning for traffic signal control. In: *Procedia Computer Science*, 52, 956-961.
12. Lana, I. *et al.* (2018) Road traffic forecasting: Recent advances and new challenges. *IEEE Intelligent Transportation Systems Magazine*, 10(2), 93-109.
13. Firdous, A, and Niranjan, V. (2020) Smart Density Based Traffic Light System. In: *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO).* IEEE, 2020.
14. Patil, P. G. *et al.* (2020) Real Time Smart Traffic Control System. *International Journal of Research in Engineering, Science and Management*, 3(2), www.ijresm.com | ISSN (Online): 2581-5792.
15. Hartanti, D., Rosida, N. A., and Siswipraptini, P. C. (2019) Optimization of smart traffic lights to prevent traffic congestion using fuzzy logic. *TELKOMNIKA Telecommunication Computing Electronics and Control*, 17(1), 320-327.
16. Al-Abaid, S. A. F. (2020) A Smart Traffic Control System Using Image Processing: A Review. *Journal of Southwest Jiaotong University*, 55(1).
17. Haydari, A. and Yilmaz, Y. (2020) Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*.
18. Kaelbling, L. P., Littman, M. L. and Moore, A. W. (1996) Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4, 237-285.
19. Szepesvári, C. (2010) Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1), 1-103.
20. Sutton, R. S., and Barto, A. (1999) Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1), 126-134.
21. Sutton, R. S., and Barto, A. (2018) *Reinforcement learning: An introduction*. MIT press, 2018.
22. Mannion, P., Duggan, J. and Howley, E. (2016) An experimental review of reinforcement learning algorithms for adaptive traffic signal control. *Autonomic road transport support systems*, 47-66.
23. Mannion, P., Duggan, J. and Howley, E. (2015) Parallel reinforcement learning for traffic signal control. In: *Procedia Computer Science*, 52, 956-961.
24. Barzilay, O. Voloch, N. Hasgall, A. Lavi Steiner, O. Ahituv, N. (2018a) Traffic Control in a Smart Intersection by an Algorithm with Social Priorities. *Contemporary Engineering Sciences*, 11(31), 1499–1511.
25. Barzilay, O. Voloch, N. Hasgall, A. Lavi Steiner, O. (2018b) Real life applicative timing algorithm for a smart junction with social priorities and multiple parameters. In: *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, Eilat, Israel, 2018, 1-5. doi: 10.1109/ICSEE.2018.8646018
26. Fine, Z., Brayer, E., Proshtisky, I., Barzilai, O., Voloch, N., & Steiner, O. L. (2019). Handling traffic loads in a smart junction by social priorities. In: *2019 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, IEEE, 1-5.
27. Barzilay, O., Voloch, N., Giloni, A., Lavi Steiner, O. (2020) Auction based algorithm for a smart junction with social priorities. *Transport and Telecommunication Journal*, 21(2), 110-118. DOI: 10.2478/ttj-2020-0008.
28. McKinsey (2013) *Global Institute Infrastructure productivity: How to save $1 trillion a year*. January 2013. Report: https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/infrastructure-productivity.
29. Schaller, B. (2017) UNSUSTAINABLE? The Growth of App-Based Ride Services and Traffic, Travel and the Future of New York City. http://schallerconsult.com/rideservices/unsustainable.pdf
30. Abbas, S., Khan, M. A., Athar, A., Shan, S. A., Saeed, A., & Alyas, T. (2022) Enabling smart city with intelligent congestion control using hops with a hybrid computational approach. *The Computer Journal*, 65(3), 484-494.
31. Bokaba, T., Doorsamy, W., & Paul, B. S. (2022) A Comparative Study of Ensemble Models for Predicting Road Traffic Congestion. *Applied Sciences*, 12(3), 1337.
32. Russo, A., Adler, M. W., & van Ommeren, J. N. (2022) Dedicated bus lanes, bus speed and traffic congestion in Rome. *Transportation Research Part A: Policy and Practice*, 160, 298-310.

33. Boysen, N., Briskorn, D., Schwerdfeger, S., & Stephan, K. (2021) Optimizing carpool formation along high-occupancy vehicle lanes. *European Journal of Operational Research*, 293(3), 1097-1112.
34. DeCorla-Souza, P. T., Zahedian, S., Nohekhan, A., Nejad, M., & Erdogan, S. (2022) Exploratory Evaluation of Incentives to Increase High-Occupancy Vehicle Use on Priced Highway Facilities. *Transportation Research Record*, 03611981221076837.
35. Miletić, M., Ivanjko, E., Gregurić, M., & Kušić, K. (2022) A review of reinforcement learning applications in adaptive traffic signal control. *IET Intelligent Transport Systems*.
36. Noaeen, M., Naik, A., Goodman, L., Crebo, J., Abrar, T., Abad, Z. S. H., ... & Far, B. (2022) Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Systems with Applications*, 116830.

Appendix A: Design Description of the Current Smart Junction Simulation

The following is a design description of our current smart junction simulation, which implements the Q-reinforcement learning algorithm described above. The simulation's system is partitioned into 7 different classes: direction, vehicle, junction, state, q-algorithm and agent.

A.1: Direction

An enum class which contains all the conflict sides in the junction, also known as "actions"**.**

A.2: Vehicle

A class which contains all the vehicle parameters, such as unique number, priority, size, position in the junction etc.

A.3: Lane

A class which contains a queue for all the vehicles before the junction, and a queue for all the vehicles that pass the junction. It has 3 main functions as follows.

- move – the function move a single vehicle to pass the junction.
- try_move – the function creates a copy of the current lane and operates the move function on it.
- calc_reward – the function calculates the local reward according to the last vehicle that passes the lane (i.e. after operating the move function).

A.4: Junction

A class which contains all the lanes in the junction. It has 2 main functions as follows.

- open_direction – the function gets as an input a conflict side and opens the relevant lanes, i.e. operates the move function on these lanes.
- try_open_direction – the function creates a copy of the current junction and operates the open_direction function on it with the given conflict side.

A.5: State

A class which contains and represent a snapshot of a current state of the junction, and a corresponding unique identifier number. It has 2 main functions as follows.

- set_q_value – the function set a q-value that was calculated as a function of the current state and a given action.
- equals – the function gets as input another state object and returns "true" if it equals our current state; otherwise returns "false".

A.6: Q-Algorithm

A class which calculates the optimal policy of the junction, which determines the conflict side that is recommended to open in order to maximize the reward. It has 3 main functions as follows.

- calculate_q_value – the function calculates and returns a new q-value to a given state.
- activate_probability – the function calculates the exploration probability of the q-learning-algorithm, i.e. it returns true in epsilon probability, and it returns false in 1-epsilon probability.
- select_next_state – given a current state, the function returns a selected action, i.e. a selected conflict side to be opened, in accordance with the policy established by the q-learning-algorithm.

A.7: Agent

A class which represents the RL agent, that contains the junction, and in each step decided to open one of the conflict sides according to the policy of the Q-Algorithm. It has 3 main functions as follows.

- move_to_next_state – the function returns the chosen action to be operated on the current junction state. The action is chosen by the q-algorithm policy, and it refers to the conflict side that will be open in the junction.
- run_episode - the function executes a single episode of the junction, where an episode begins with a junction containing vehicles waiting to pass it and ends when all these vehicles have passed that junction. In each step, a vehicle passes the junction in accordance with the policy established by the q-learning-algorithm.
- run_iterations - the function operates the function run_episode for a given number of times in order to stable the q-values of each state, i.e. the policy, which are calculated by the q-learning-algorithm.

A.8: Simulation Measurements

A module which contains all the simulation hyper-parameters.