Markov on the Move: Traffic Simulation as an MDP

Ben Wurster
Aerospace Engineering
University of Colorado Boulder

Michael Buchanan
Computer Science
University of Colorado Boulder

Josh Myers-Dean
Computer Science
University of Colorado Boulder

***Abstract*—Efficient traffic signal control is vital in reducing congestion, emissions, and travel delays. Traditional traffic systems rely on static, rule-based schedules for switching traffic light signals, which often fails to adapt to dynamic traffic conditions and may compromise both efficiency and safety. In this paper, we present a custom traffic simulation framework that supports realistic traffic flow and collision detection to evaluate adaptive signal control strategies. We formulate the signal control task as a Markov Decision Process (MDP) and use both Monte Carlo Tree Search (MCTS) and a neural network to create policies that are effective at managing the traffic of a five-lane intersection. Experimental results show that the neural network achieves a satisfactory throughput while maintaining safety, underscoring the importance of safety-aware learning in traffic control and demonstrating the promise of neural networks as adaptable traffic signal controllers. All the code used in the project can be found on our GitHub.**

***Index Terms*—Sequential decision making, reinforcement learning, traffic control**

I. Introduction

Efficient traffic signal control is essential for managing urban mobility, reducing congestion, commuter cost and minimizing the environmental and economic costs associated with delays, such as idling which causes emissions and costs commuters money for gas. Despite advances in sensor deployment and intelligent transportation systems, many cities continue to rely on static, rule-based stoplight policies that fail to adapt to fluctuating traffic conditions [1]. This often results in suboptimal throughput, increased travel time, and localized bottlenecks, especially during peak hours (e.g., "rush hour") or in complex road networks with multiple interacting intersections.

Traditional approaches to traffic signal optimization, including heuristic scheduling [2], [3], rule-based controllers [4], and even some data-driven techniques [5], struggle with credit assignment for actions or to generalize to novel traffic conditions. Existing methods typically optimize for short-term metrics, struggle to scale across multiple intersections, or rely heavily on hand-engineered features and assumptions. Furthermore, few approaches allow for explicit comparison of different optimization objectives: such as minimizing the average travel time versus the time experienced by the slowest vehicle.

To address the limitations of existing methods, we formulate traffic signal control as a Markov Decision Process (MDP) over a discretized state space that captures both the spatial layout of intersections and the dynamic traffic conditions that exist. In our simplified setting, shown in Figure 1, each intersection has one lane per direction, resulting in a 12-dimensional state vector that represents the number of cars at each entry and exit position, respectively. To support future scalability (e.g., to multiple intersections), we encode directional flow using structured identifiers (e.g., south, north) that we believe can generalize to more complex lane configurations. We also represent the current traffic light configuration, either categorically (e.g., going north to south) or as a boolean vector over directional flows, enabling efficient generative transitions. The action space is intentionally constrained to either maintaining or switching the current light state, which we hypothesize helps reduce the branching factor for tree-based planning. Transitions are generated stochastically based on traffic rules and the chosen light configuration, simulating car movement under each action. We explore sparse sampling tree search [6], [7] for policy optimization, a method suitable for large, structured MDPs. Our reward function is based on throughput (i.e., the number of cars passed through the light) but is extensible to incorporate decaying rewards or penalties for starvation (e.g., if there are a lot of cars in one queue, they may be passed and cause long delays for cars in conflicting queues), allowing us to balance fairness and temporal efficiency. These extensions are akin to operating system scheduling heuristics [8].

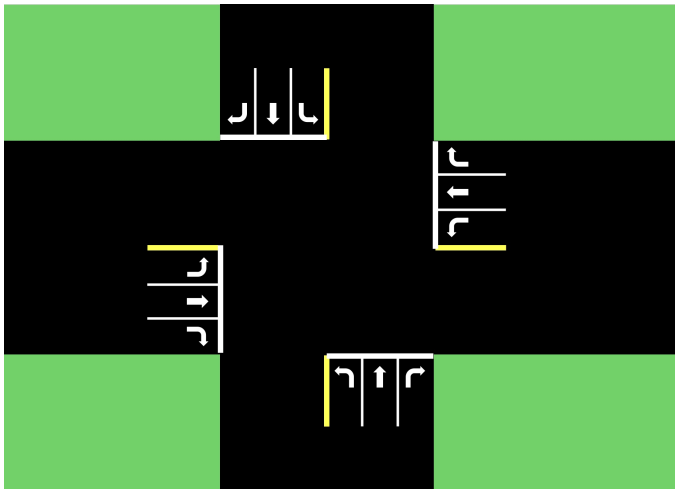

Fig. 1. Example of the environment in our MDP formulation. There is one lane to go each direction in the intersection, therefore the state space would be a vector of length 12, representing the number of cars that would be waiting at each position.

By grounding traffic optimization in a formal decision-theoretic framework, our approach offers a path toward more adaptable traffic control systems. Beyond efficiency gains, success in our proposed task could yield numerous benefits to society. First, by reducing time spent idling in traffic, carbon emissions could be reduced and the rate at which global warming is occurring may slow. Second, by reducing the time spent in traffic, commuters would have more time to spend with their families, leading to increased family happiness [9]. Third, by increasing throughput, bus systems, a critical piece of infrastructure for people with disabilities [10], could become more reliable, empowering disabled riders to improve their mobility and quality of life. Collectively, these improvements highlight the transformative potential of intelligent traffic signal control to enhance environmental sustainability, social well-being, and transportation equity.

II. Related Works

a) Decision-Theoretic Traffic Control: Formulating traffic signal optimization as a Markov Decision Process (MDP) enables structured reasoning over sequential decisions and long-term system performance. Early work in this space focused on single-intersection control using tabular methods or value iteration to optimize simple objectives like queue length or delay [11], [12]. More recent efforts have extended this approach to network-level control, employing deep reinforcement learning (e.g., Deep Q Networks) to scale to larger state spaces [13], [14]. Relatedly, other works have posed traffic control in a game-theoretic manner. For example, [15] develops a toll-based approach for MDP congestion games with unknown costs, using inexact gradient methods to enforce population constraints while [16] uses routing games to analyze the impact on parking on overall traffic patterns. However, many of these methods optimize only for expected throughput or average delay and do not explicitly reason over alternate reward formulations that balance equity (e.g., starvation) and efficiency (e.g., throughput). In contrast, our work explores multiple optimization objectives: minimizing both average and worst-case vehicle delay, and investigates how different formulations influence policy behavior in centralized planning while using an efficient, sparse tree search solution.

III. Problem Formulation

Our problem focuses on a four way intersection as shown in figure 1. We model this scenario as an MDP with a 12-dimensional state space, corresponding to the queues for each of the traffic lanes in the intersection. Each direction includes three lanes: one for left turns, one for right turns, and one for through traffic. The action space consists of four actions, each representing the activation of one of four traffic signals: a left turn light or the main light for each pair of opposing directions. The transition function is the probability of switching the traffic signal based on the state and is determined by a feedforward neural network. Finally, the reward function is a composite function that emphasizes the throughput of cars and fairness between directions.

For the heuristic and MCTS-based policies, we cast the problem of operating a four-way traffic light as a finite-horizon Markov-decision process (MDP). Formally, our MDP is defined as a tuple $(\mathcal{A}, \mathcal{S}, T, \mathcal{R}, \gamma)$. We define the entities in our formulation below and use $\gamma = 0.95$ to encourage the agent to empty the queues quickly but still value longer-term improvements. The MDP terminates when all 12 queues are empty.

a) $\mathcal{S}$: States: Our environment consists of twelve directed lanes: South-to-North (SN), South-to-West (SW), South-to-East (SE), North-to-South (NS), and so on, covering every straight, left-turn, and right-turn movement at the intersection. We keep an integer queue length for each lane, stored in a fixed order inside a 12-element vector. In addition, we store the signal phase as well in our states. A state is therefore a pair [queue vector, current phase]. The initial state has one vehicle waiting in every lane and the light turned off (i.e., no phase selected).

b) $\mathcal{A}$: Actions: At each decision step the controller chooses the next signal phase from exactly the same six options listed above. If the chosen phase differs from the current one, the light simply changes and no vehicles move during that time step. If the chosen phase equals the current one, vehicles proceed through the intersection subject to basic right-of-way rules encoded in the implementation:

- For straight phases, the two opposing straight-through lanes have priority; right turns always proceed; left turns proceed only if there is no conflicting opposing traffic.
- For a protected-left phase, all lanes from that approach move: the protected left lane, the adjacent straight lane, and its right-turn lane. Opposing right turns may also move provided they do not conflict.

The result of these rules is a deterministic vector state_delta that indicates how many vehicles depart from each queue during that step.

c) T: Transition Dynamics: The next queue vector is obtained by subtracting state_delta from the current vector, and the light phase remains unchanged (unless the agent explicitly switched it). Because the rules are deterministic and we do not model random arrivals, the transition function is deterministic.

d) $\mathcal{R}$: Reward Function: The controller receives a base penalty of -0.1 each step, reflecting the cost of time passing. It then receives +1 for every vehicle that actually clears the intersection in that step. Thus, holding a phase that efficiently services traffic yields higher returns, while phase changes accumulate only the –0.1 penalty.

A. Simulation

We use a custom simulation to replicate traffic conditions in the intersection in accordance to our MDP as shown in figure 2.

For the neural network-based policy, we use a simulation to imitate different traffic patterns to mimic real-world conditions, and a pattern that randomly increases the amount of cars in one direction of travel by 75% every 20 time steps. This is to replicate how traffic often gets more congested in

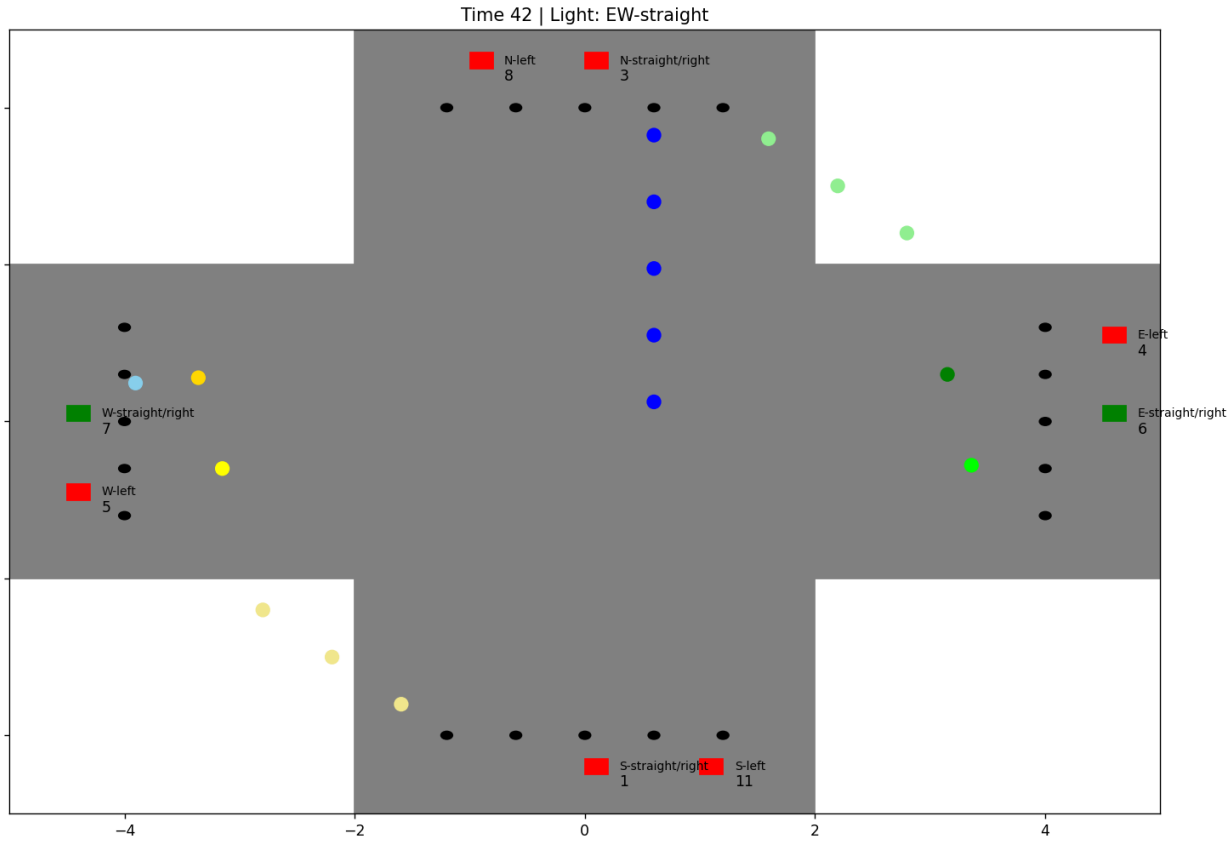

Fig. 2. An image of the traffic simulation where the traffic lights are controlled on a fixed time interval. At the top, the time indicates the time step within the environment. The light text indicates which light is currently active, as shown by the colored squares at each intersection. Currently, the light for the EW-straight direction is active, so the corresponding squares are green while all others are red. The numbers beside each traffic signal display the number of cars waiting in that lane. Cars are represented by colored dots, with primary colors (red, blue, green, yellow) designating the direction of travel. Variations in the color shade indicate whether a car is turning left, right, or moving straight.

a specific direction of travel during different times of the day due to commuting schedules, though further traffic patterns can be tested within the simulation. Additionally, the speed of the cars can be adjusted to mimic intersections with different speed limits. On top of this, the simulation detects collisions if two cars intersect at opposing angles as safety is a vital factor in the design of traffic systems. Notably, the car's do not avoid collisions in the simulation, an intentional choice to ensure the model does not rely on car behavior for controlling traffic signals. This collision detection can be used to prevent dangerous light switches and better understand the optimal duration of a yellow light.

We implement slight differences in the reward functions to encourage more stable learning. Our states and other aspects stay the same as the MDP.

B. Reward Function for the Neural Network

The reward function aims to minimize the time cars spend in a queue and consists of: (1) a collision penalty, (2) throughput bonus, (3) duration reward, and (4) imbalance penalty.

$$\begin{aligned}\text{reward} = \ &\text{collision\_penalty} \\ &+ \text{throughput\_bonus} \\ &+ \text{duration\_reward} \\ &+ \text{imbalance\_penalty}\end{aligned}$$

a) Collision Penalty: As mentioned previously, the simulation can detect when cars collide, so a collision penalty is applied to the reward function whenever two cars collide. This is to primarily counteract the strategy of rapidly switching lights to clear traffic quickly. While it is optimal to rapidly switch traffic signals to quickly clear out queues, it is undesirable for traffic since it creates chaos and dangerous situations within the intersection.

b) Throughput Bonus: The throughput bonus is the main motivator for getting cars through the intersection, encouraging the model to clear out queues. The throughput bonus is calculated by the following equation:

$$\text{Throughput\_Bonus} = N_{\text{exited}} \cdot W_{\text{throughput}}$$

where N_{exited} refers to the number of cars that have exited a queue, and $W_{Throughput}$ being an adjustable weight to scale the bonus.

c) Duration Reward: The duration reward is meant to finely control the duration of a green light in the simulation. While queues can be cleared quickly through rapid light changes, this is not desirable as it can cause intersections to become dangerous. Additionally, drivers do not have perfect reactions or acceleration, so a traffic signal needs to be green for a short duration to give drivers the chance to cross the intersection. As a result, the duration reward function consists of two components, with the first being a static negative reward if the light is switched before a constant time t_{min} is exceeded. Once this time has been exceeded, the following exponential function determines the duration reward D_R:

$$D_R = N_A \times W_{Active\_Queue} \times (d)^{t_{signal}}$$

where N_A represents the number of cars in the queue where the green light is active, and W_{Active_Queue} being an adjustable weight to this function. Then, $d < 1$ is a constant number indicating the decay of the duration reward, and t_{signal} is the difference between how long the light has been green and t_{min}.

In short, the duration reward is a decaying exponential function meant to reward having green signals for a reasonable period of time. However, it is a decaying function as to not encourage keeping green signals on for too long and ignoring the other queues in the intersection.

d) Imbalance Penalty: Continuing with this idea, we introduce an imbalance penalty to ensure one queue is not being heavily favored over another. This is vital in ensuring equity across queue wait times. For example, the North/South direction of travel may have a larger flow of traffic than the East/West direction of travel, so an optimal strategy for maximizing overall traffic throughput would be to only keep the North/South flow of traffic moving. Of course, this means that the East/West direction of traffic could be stuck waiting for the traffic signal to change for a long period of time. As a result, we introduce a penalty for letting the inactive queue (N_I) grow.

$$\text{Imbalance\_Penalty} = \begin{cases} W_I \times (N_I - N_A), & \text{if } N_I > N_A \\ 0 & \text{otherwise} \end{cases}$$

Notably, W_I is an adjustable weight.

IV. Methods

The solve the MDP, we employ two methods. First, we employ standard Monte Carlo Tree Search (MCTS) in Julia[1]. We do offline planning for a max of 10,000 iterations with a depth of 40 nodes, and an exploration constant of 5 for the UCB. We train the planner using a queue size of 5. We also consider a baseline ϵ-greedy-like policy which keeps the light the same with $p(1-\epsilon)$ and changes the light with $p(\epsilon)$ and we use $\epsilon = 0.2$ in our experiments.

[1]https://github.com/JuliaPOMDP/MCTS.jl

Queue Size	Reward Bound	Reward ϵ	Reward $\mathcal{T}$
3	36	33.5	34.6
5	60	55.8	56.9
10	120	66.0	80.9
12	144	122.0	138.3

TABLE I
EXPERIMENTAL RESULTS ON 4 DIFFERENT QUEUE SIZES. ϵ REPRESENTS THE EPSILON-GREEDY POLICY, AND $\mathcal{T}$ REPRESENTS THE MCTS POLICY.

The second method is a feedforward neural network with two hidden layers using the PyTorch [17] package in Python. The input layer consists of the number of lanes in the intersection while the output is desired traffic control signal. Each hidden layer uses ReLU as the activation function, and the neural network is trained on 100 time steps within the environment to maximize the reward function.

A. Selecting Hyperparameters

In order to select the hyperparameters for the reward function used in the neural network, we ran 729 simulations testing different variations of the values of the reward function, and selected the model that maximized the throughput of cars in the intersection. For MCTS, early experiments showed little sensitivity to the choice of hyperparameters.

V. Results

Results for the MCTS are shown in Table I. As the number of cars in each of the queues increase, there is generally a larger gain in the reward from the epsilon-greedy action selection policy to the MCTS policy, suggesting that the learned policy has greater effect for larger problems. Furthermore, a review of the action history shows that the MCTS approach never encounters a situation where it is not claiming a reward by switching the light, whereas the epsilon-greedy approach can sometimes find itself in situations where it stalls and leaves some reward on the table.

For the neural network approach, the simulation was more intricate due to the inclusion of collision detection. As a result, the results are measured by the throughput and number of collisions averaged over 50 simulations of a 100 time steps. We compare the neural network against a baseline policy that changes the traffic light on a fixed time interval. To better reflect real-world disruptions caused by collisions, we also report an adjusted throughput metric, which penalizes flow based on traffic stoppages following a crash (with each collision causing a 4-time-step delay).

As shown in Table II, the neural network maintained a perfect safety record with 0.00 average collisions and an adjusted throughput of 57.96, whereas the fixed-cycle policy, despite a higher raw throughput of 230.54, averaged 22.86 collisions, reducing its adjusted throughput to just 17.25. These results demonstrate that raw throughput alone can be misleading, and that safety-aware policies offer significantly more reliable traffic performance.

Model	Throughput	Collisions	Adjusted Throughput
Neural Network	57.96	0.00	57.96
Fixed-Cycle Model	230.54	22.86	17.25

TABLE II
AVERAGE NUMBER OF CARS EXITING THE INTERSECTION OVER 50 RUNS

VI. DISCUSSION

Regarding the MDP-based approach, it was discovered that an optimal policy can be learned via a MTCS method where a set of queues associated with a light decision are depleted and then the policy switches to the next best option that yields high potential for future rewards. The learned decision policy wastes no time in switching to a light configuration that produces further throughput, although in all cases it executes until such queues are empty, which is likely a byproduct of the penalty, or lack of reward, that comes with switching. It is therefore reasonable to suspect that the MCTS method has discovered the optimal policy for maximizing the throughput of static queues despite this being an unrealistic representation of the equitable conditions necessary for an intersection in reality. Refining this must be taken up at the level of fundamental model formulation.

While the fixed policy had a higher raw throughput than the neural network, it did so at the cost of safety, exhibiting a higher number of collisions. In contrast, the neural network maintained a perfect safety record, resulting in a higher adjusted throughput.

Collisions introduce major disruptions in a traffic system by stopping traffic and endangering lives, so an ideal traffic system should balance both safety and efficiency. Although the cars in our simulation did not avoid collisions, unlike humans in the real world, the sheer difference in collisions illustrate that the fixed policy model creates a more dangerous intersection. Our results indicate that collision-avoidance should be strongly taken into account when designing traffic systems, and future iterations of the model should continue to improve upon collision-avoidance by performing safer light switches, like prolonged yellow lights or delays between switching from a red to a green light.

VII. CONCLUSION

In this paper, we develop a traffic simulation to evaluate strategies for controlling traffic lights in a five-lane intersection. The simulation imitates real world traffic patterns, supports adjustable flow patterns, and detects collisions to account for safety. We use this simulation to solve a MDP that maximizes the amount the traffic flow in an intersection. Taking this a step further, we use a neural network to control the traffic signals for a more complex scenario that takes into account car collisions. We compare this against a fixed policy and found that the neural network is significantly safer and has a higher throughput due to the lack of collisions. As a whole, our results show that traffic flow can be improved through intelligent traffic signal control that encourages safety and adapts to the amount of traffic in the system.

VIII. FUTURE WORK

A next step in the development of the MDP model is further refinement of the reward function to prevent the starvation or long delay of queues in large problems. This would promote greater equity in the transitioning of the light, albeit at a cost. The reward can therefore be expanded on the reflect the hidden cost that having this equity would bear.

While the methods in this paper can be improved through the usage of different models such as Double Q Learning and through better hyperparameter tuning, the main focus of improving traffic signal control relies on extending the problem past a single intersection. Multiple intersections allow for complex traffic patterns, and further improvements can be made by incorporating traffic light control across a series of intersections. Also, incorporating different types of roads such as school zones or highways can help replicate a cities' transportation system. Finally, real world data can can be incorporated to improve the adaptability and relevance of our models.

IX. CONTRIBUTIONS AND RELEASE

Michael created the simulation depicted in figure 2 as well as the traffic pattern and neural network used in the paper. Michael also wrote the parts of the paper pertaining to the neural network (abstract, problem formulation, the reward function for the neural network, methods, results, conclusion, and future work).

Ben formulated the MDP problem and implemented the MDP in Julia. Additionally, Ben wrote the results and discussion section for the MDP.

Josh created the MCTS solution and wrote the introduction, problem formulation, and background section.

The authors grant permission for this report to be posted publicly.

REFERENCES

[1] J. Wu, "Revolutionizing urban mobility: Intelligent traffic light systems and smart traffic signals - leotek: Led streetlights & traffic signal solutions," Feb 2024.

[2] Q. He, K. L. Head, and J. Ding, "Heuristic algorithm for priority traffic signal control," *Transportation research record*, vol. 2259, no. 1, pp. 1–7, 2011.

[3] F. Percassi, S. Bhatnagar, R. Guo, K. McCabe, T. L. Mccluskey, and M. Vallati, "An efficient heuristic for ai-based urban traffic control," in *2023 8th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 1–6, IEEE, 2023.

[4] I. Marti, V. R. Tomas, A. Saez, and J. J. Martinez, "A rule-based multi-agent system for road traffic management," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3, pp. 595–598, IEEE, 2009.

[5] N. H. Gartner, S. F. Assman, F. Lasaga, and D. L. Hou, "A multi-band approach to arterial traffic signal optimization," *Transportation Research Part B: Methodological*, vol. 25, no. 1, pp. 55–74, 1991.

[6] T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans, "Bayesian sparse sampling for on-line reward optimization," in *Proceedings of the 22nd international conference on Machine learning*, pp. 956–963, 2005.

[7] M. Kearns, Y. Mansour, and A. Y. Ng, "A sparse sampling algorithm for near-optimal planning in large markov decision processes," *Machine learning*, vol. 49, pp. 193–208, 2002.

[8] T. Li, D. Baumberger, D. A. Koufaty, and S. Hahn, "Efficient operating system scheduling for performance-asymmetric multi-core architectures," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, pp. 1–11, 2007.

[9] A. Brömmelhaus, M. Feldhaus, and M. Schlegel, "Family, work, and spatial mobility: The influence of commuting on the subjective well-being of couples," *Applied Research in Quality of Life*, vol. 15, no. 3, pp. 865–891, 2020.

[10] J. Schlingensiepen, E. Naroska, T. Bolten, O. Christen, S. Schmitz, and C. Ressel, "Empowering people with disabilities using urban public transport," *Procedia Manufacturing*, vol. 3, pp. 2349–2356, 2015.

[11] B. Abdulhai and L. Kattan, "Reinforcement learning: Introduction to theory and potential for transport applications," *Canadian Journal of Civil Engineering*, vol. 30, no. 6, pp. 981–991, 2003.

[12] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[13] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of learning, inference and control of multi-agent systems (at NIPS 2016)*, vol. 8, pp. 21–38, 2016.

[14] F. Rasheed, K.-L. A. Yau, R. M. Noor, C. Wu, and Y.-C. Low, "Deep reinforcement learning for traffic signal control: A review," *IEEE Access*, vol. 8, pp. 208016–208044, 2020.

[15] S. H. Li, Y. Yu, N. I. Miguel, D. Calderone, L. J. Ratliff, and B. Açıkmeşe, "Adaptive constraint satisfaction for markov decision process congestion games: Application to transportation networks," *Automatica*, vol. 151, p. 110879, 2023.

[16] D. Calderone, E. Mazumdar, L. J. Ratliff, and S. S. Sastry, "Understanding the impact of parking on urban mobility via routing games on queue-flow networks," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 7605–7610, 2016.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.