Congestion Control of Vehicle Traffic Networks by Learning Structural and Temporal Patterns

SooJean Han SOOJEAN@CALTECH.EDU

Soon-Jo Chung SJCHUNG@CALTECH.EDU

Johanna Gustafson[1] JOHANNA@INTEGER.SE

Computing and Mathematical Sciences, California Institute of Technology.

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

For many network control problems, there exist natural spatial structures and temporal repetition in the network that can be exploited so that controller synthesis does not spend unnecessary time and energy redundantly computing control laws. One notable example of this is vehicle traffic flow over metropolitan intersection networks: spatial symmetries of the network arise from the grid-like structure, while temporal symmetries arise from both the structure and from human routine. In this paper, we propose a controller architecture based on *pattern-learning with memory and prediction (PLMP)*, which exploits these natural symmetries to perform congestion control without redundant computation of light signal sequences. Memory is implemented to store any patterns (intersection snapshots) that have occurred in the past "frequently enough", and redundancy is reduced with an extension of the state-of-the-art episodic control method which builds equivalence classes to group together patterns that can be controlled using the same traffic light. Prediction is implemented to estimate future occurrence times of patterns by predicting vehicle arrivals at subsequent intersections; that way, we schedule light signal sequences in advance. We compare periodic baselines to various implementations of our controller model, including a version of PLMP with prediction excluded called *pattern-learning with memory (PLM)*, by evaluating their performance according to three congestion metrics on two traffic datasets with varying arrival characteristics.

Keywords: Control design, Pattern analysis, Road traffic control, Networked control systems

1. Introduction

In our previous work Han et al. (2023), we introduced the concept of *pattern-learning for prediction (PLP)*: considering repeating jump patterns in the behavior of jump stochastic systems allows for more efficient controller design by eliminating computation time and redundancy by preserving past patterns into memory and predicting the future occurrence of patterns. In the context of traffic networks, patterns can be defined according to the temporal repetition and structural symmetry that arises naturally in a variety of ways. For example, many metropolitan road networks are typically arranged like a rectangular grid; in America, T-junctions and X-junctions (4-way intersections) are highly common. These kinds of repeated topological structures fundamentally impact the travel behavior of vehicles and consequently, congestion level (Xie and Levinson, 2007). Repetition can be observed in the traffic density over time not only due to the grid structure of the network, but also due

1. Caltech SURF student. Currently with the Department of Automatic Control, Faculty of Engineering, Lund University, Sweden.

to human routine: rush hours during the weekdays are a notable example of this. Even for special events that do not occur regularly (e.g., traffic jams near the venue of a music concert), a certain level of congestion can be predicted if this special event was planned beforehand (Kwoczek et al., 2014). Additional predictions can also be made by understanding the nature of the planned event: for a music concert that lasts three hours, a second wave of congestion would be expected approximately three hours after the first wave due to people leaving the venue. These natural spatial and temporal structures in most urban vehicle traffic flow problems suggest that a congestion control mechanism designed around some suitable choice of ‘’pattern” can improve the time and computation efficiency at which light signal sequences are designed.

Related Work One of the most common methods of modeling vehicle traffic flow is via queueing theory (Miller, 1961; Lioris et al., 2017; Muralidharan et al., 2015). While queuing-based results are very useful for benchmarking performance, they often rely on assumptions that are not reflective of real-world traffic (e.g., Poisson arrivals). Another class of models encompass discrete-time ODE dynamics (Coogan et al., 2015), which allow for the explicit formulaic construction of control laws, but sometimes rely on the knowledge of parameters (e.g., turning proportion) whose values may be difficult to obtain in practice. Recently, data-driven architectures such as neural networks are gaining traction as suitable methods for vehicle congestion control due to their ability to accommodate realistic traffic characteristics and complex network topologies. For example, Zhang and Taylor (2006) developed a framework for automated incident detection based on Bayesian networks, with an emphasis on being able to flexibly incorporate domain-specific knowledge into an otherwise all-data-driven approach. More recently, works such as Yu et al. (2018) and Li et al. (2018) have considered variations on GNN architectures to predict the spatiotemporal behavior of traffic spread across complex networks. However, many of these neural network architectures are designed to account for general topologies, and may be less efficient when considering environments where structural symmetry could be leveraged. To take advantage of environment repetition, methods based on explicit rule-based construction have also emerged (Lu et al., 2014; Dion and Hellinga, 2002). For vehicle routing, reinforcement learning methods are especially suitable (Rivière and Chung, 2022), and for repetitive environments like urban grid intersection networks, experience replay approaches can be used. One such approach is called episodic control (Lengyel and Dayan, 2007; Blundell et al., 2016; Pritzel et al., 2017), which incorporates episodic memory (Botvinick et al., 2019) into traditional learning techniques with the goal of speeding up training by recalling specific instances of highly rewarding experiences.

Contributions Our paper proposes a learning-based controller architecture for vehicle traffic congestion control by implementing *pattern-learning with memory and prediction (PLMP)*, which is an extension of the PLP architecture from Han et al. (2023) via the explicit implementation of a memory component. Here, the “patterns” at each intersection are the intersection’s snapshots, e.g., traffic camera photos which display the distribution of vehicles present in each lane and direction. In contrast to the neural network architectures described in the related works above, PLMP is designed to explicitly leverage the natural spatial and temporal symmetries of a given traffic network, e.g., the rectangular grids often found in metropolitan cities. Congestion control via PLMP employs two ways to eliminate unnecessary time and energy spent redundantly computing light signal sequences. First, *memory* is implemented in the form of a table that maps patterns and light signal sequences to rewards; our architecture employs an approach which extends the state-of-the-art episodic control methods (e.g., Blundell et al. (2016)) by building *equivalence classes* to group patterns that can be controlled using the same sequence of light signals. Second, *prediction* is implemented with a one-timestep lookahead that augments, to the original pattern, the distribution of vehicles in the adjacent links of the intersection and schedules future light signal sequences in advance. In addition to the extension with equivalence classes, our PLMP method differs from pure episodic control by the inclusion of this prediction component. We apply our model to two synthetic datasets, one synthesized from scratch and one synthesized from real-world data, and compare two periodic baseline light signals to variations of our PLMP controller, including a version without prediction called *pattern-learning with memory (PLM)*. We evaluate the performance of each controller on a variety of traffic scenarios according to three different congestion metrics: 1) average waiting time per vehicle, 2) average time deviation away from the optimal travel duration, and 3) the number of vehicles that have not yet reached the end of their routes. We find that, on average, PLM outperforms the periodic baselines while PLMP outperforms PLM with mild variation among the different implementations.

2. System Setup and MDP Formulation

2.1. The Grid Network of Signalized Intersections

A $H \times L$ rectangular grid network of 4-way intersections is represented by a graph $\mathcal{G} = (\mathcal{I}, \mathcal{N}, \mathcal{E})$, with set of intersections $\mathcal{I}$, nodes $\mathcal{N}$, and directed edges $\mathcal{E}$ that connect between two nodes. Each intersection is denoted with a tuple $I := (h, i) \in \mathcal{I}$ marking its location in the grid, $h \in \{0, \cdots, H-1\}$ and $i \in \{0, \cdots, L-1\}$. Each node is represented as a tuple $(I, \texttt{D}, \chi, f) \in \mathcal{N}$, where I is the intersection ID, $\texttt{D}$ is one of the four directions $\{\texttt{E}, \texttt{N}, \texttt{W}, \texttt{S}\}$, $\chi \in \{1, 0\}$ indicates whether vehicles are incoming (1) or outgoing (0) at the node. The variable $f \in \{0, 1\}$ indicates whether the node is located at the fringes of the network or not; we partition the set of nodes $\mathcal{N}$ into the set $\mathcal{N}_F$ of *fringe nodes* and the set $\mathcal{N}_I := \mathcal{N} / \mathcal{N}_F$ of *intermediate nodes*. Each intersection I is controlled by a traffic light signal; let $m \in \mathcal{M}$ be the mode of the traffic light and $\mathcal{M}$ be the set of possible modes. We assume there are $|\mathcal{M}| = 8$ possible modes each signal can take: 1) `E-W` forward green, 2) `E-W` left-turn green, 3) `N-S` forward green, 4) `N-S` left-turn green, 5) `E` forward and left, 6) `N` forward and left, 7) `W` forward and left, 8) `S` forward and left. Right-turns are permitted whenever.

2.2. Vehicle Arrival Processes

Let $\mathcal{V}_A[t]$ represent the time-varying set of vehicle arrivals from the fringes of the network, i.e., $\mathcal{V}_A[t] = \varnothing$ if no vehicles entered the grid at time t and $\mathcal{V}_A[t] = \{v_1, \cdots, v_K\}$ if some number $K \in \mathbb{N}$ vehicles v_1 to v_K have entered at time t. Note that $\mathcal{V}[t] := \cup_{s=0}^{t} \mathcal{V}_A[t]$ is the total number of vehicles that are in the grid network by time t. Let $\mathcal{V}_D[t] \subseteq \mathcal{V}[t]$ be the set of departed vehicles (i.e., vehicles that have reached their destination) and let $\mathcal{V}_C[t] := \mathcal{V}[t] \backslash \mathcal{V}_D[t]$ be the set of circulating vehicles. For all sets of the form $\mathcal{V}_*[t]$, we use $V_*[t] = |\mathcal{V}_*[t]|$ to represent its cardinality.

All vehicles are identical with some length and travel at a constant speed, meaning they travel to and across each intersection at a constant amount of time. Let Δt_L be the time it takes a single vehicle to travel an uncongested link between intersections, and let Δt_I be the time it takes to cross an intersection. Define $\mathcal{R}(H, L)$ to be the entire combinatorial set of all routes from fringe to fringe of a grid network with dimensions $H \times L$. We assume each $v \in \mathcal{V}[t]$ is traversing the grid network according to a pre-determined route $\mathbf{r}_v \in \mathcal{R}(H, L)$ that starts at a node of entry $e_v \in \mathcal{N}_F$ and ends at a node of departure $d_v \in \mathcal{N}_F$. We represent $\mathbf{r}_v$ as an alternating sequence of nodes and

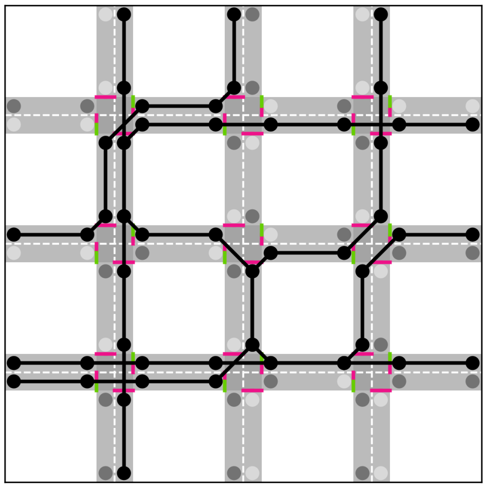

Figure 1: Grid network visualization of intersection graph $\mathcal{G}$ for $H = L = 3$. Nodes are distinguished by incoming ($\chi = 1$, light gray dots) or outgoing ($\chi = 0$, dark gray dots). Sample routes for five vehicles are also shown as an alternating sequence of nodes (black dots) and links (black lines). In this snapshot (e.g., traffic camera photos which display the distribution of each intersection in the grid), forward-going `E`/`W` traffic are allowed to pass through each intersection (red and green line segments); over time, and as the vehicles trace their respective routes, these traffic light colors would change.

links $\mathbf{r}_v = [e_v, \ell_v^{(e)}, n_1, \ell_1, \cdots, n_{k-1}, \ell_{k-1}, n_k, \ell_v^{(d)}, d_v]$, where $k \in \mathbb{N}$ is the route length, $n_i \in \mathcal{N}_I$ for $i \in \{1, \cdots, k\}$, and $\ell_v^{(e)}, \ell_v^{(d)}, \ell_i \in \mathcal{E}$ for $i \in \{1, \cdots, k-1\}$. We distinguish $\ell_v^{(e)}$ and $\ell_v^{(d)}$ from the other links ℓ_i as the fringe links of the route, i.e., links that connect to or from a fringe node $(I, \mathtt{D}, \chi, 1)$. To keep the paper focused, we defer the treatment of heterogeneous traffic and route optimization to future work. A sample visualization of $\mathcal{G}$ with vehicle routes and light signals is in Figure 1.

Definition 1 (Vehicle Quantities) *Let $T_{sim} \in \mathbb{N}$ be the time duration of the experiment. Each vehicle $v \in \mathcal{V}[T_{sim}]$ locally keeps track of two congestion quantities. First, $W_v[t] \in \mathbb{N}$ is its cumulative waiting time by time $t \in [0, T_{sim}]$, which increments by 1 for each timestep it waits at an intersection on a red light. Second, if $v \in \mathcal{V}_D[T_{sim}]$, $D_v \in \mathbb{N}$ is the total time it took to travel its entire route.*

2.3. The Vehicle MDP (VMDP) Formulation

States: The state space $\mathcal{S} := \mathcal{S}_N \times \mathcal{S}_L$ is composed of two distinct parts. First, $\mathcal{S}_N \subseteq (\mathbb{Z}^{\geq 0})^{12HL}$ denotes the number of vehicles at each incoming intermediate node $\{(I, \mathtt{D}, 1, 0) : \mathtt{D} \in \{\mathtt{E}, \mathtt{N}, \mathtt{W}, \mathtt{S}\}\}$, partitioned by direction and turn (right, left, or forward); the elements of each $\mathbf{s}_{t,N} \in \mathcal{S}_N$ are ordered [`E`-rt, `E`-lft, `E`-fwd, `N`-rt, $\cdots$, `W`-rt, $\cdots$, `S`-rt, `S`-lft, `S`-fwd] where rt, lft, and fwd are shorthand for right, left, and forward, respectively. Second, $\mathcal{S}_L \subseteq (\mathbb{Z}^{\geq 0})^{3(4HL-2H-2L)}$ represents the number of vehicles that are present in each link, partitioned again by turn, and each $\mathbf{s}_{t,L} \in \mathcal{S}_L$ is ordered in the same way as $\mathbf{s}_{t,N}$. The full state vector is concatenated as $\mathbf{s}_t = [\mathbf{s}_{t,N}^\top, \mathbf{s}_{t,L}^\top]^\top \in \mathcal{S}$.

Actions: The action space $\mathcal{A} := \mathcal{M}^{HL}$ describes the mode of each light signal at each intersection.

Transition Function: The transition function $\mathscr{T}(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$ for two states $\mathbf{s}_t, \mathbf{s}_{t+1} \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$ is defined by the constraints of vehicle movement along the grid (i.e., to get from intersection $(0,0) \to (1,1)$, take either $(0,0) \to (0,1) \to (1,1)$ or $(0,0) \to (1,0) \to (1,1)$). We assume the time spent in each link is directly proportional to the level of congestion: if a vehicle enters a link with $X \in (\mathbb{Z}^{\geq 0})$ vehicles inside, it takes $(X+1)\Delta t_L$ timesteps to travel it if the link is between two intersections and $(X+1)\Delta t_I$ timesteps if the link is across an intersection. For simplicity, we let $v^* \in \mathbb{N}$ be the maximum number of vehicles per turn that can cross an intersection in one timestep.

Rewards: The reward function $\mathscr{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) := \mathbf{1}^\top(\mathbf{s}_{t+1,N} - \mathbf{s}_{t,N})$ is the *rate of intersection clearance*, which computes the total number of vehicles that are removed from each intersection through an action $\mathbf{a}_t$ that drives $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. Here, $\mathbf{1} \in (\mathbb{Z}^{\geq 0})^{12HL}$ is the vector of all ones.

Definition 2 (Congestion Metrics) *Let $T_{sim} \in \mathbb{N}$ be the time duration of the experiment, and define $D_v^* \leq D_v$ to be the optimal travel time of each vehicle $v \in \mathcal{V}_D[T_{sim}]$ (i.e., the time taken to reach its destination assuming an empty network and all-green light signals). With the vehicle quantities described in Definition 1, we use the following metrics to evaluate the performance of our controller. First, define the average cumulative waiting time to be $W := (1/V_D[T_{sim}]) \sum_{v \in \mathcal{V}_D[T_{sim}]} W_v[T_{sim}]$. Second, define the average travel deviation to be $D := (1/V_D[T_{sim}]) \sum_{v \in \mathcal{V}_D[T_{sim}]} (D_v - D_v^*)$. Third, we keep track of $V_C[t]$, the number of vehicles that did not reach their destinations by $t \in [0, T_{sim}]$.*

3. Pattern-Learning with Memory and Prediction

We now describe the controller architecture based on *pattern-learning with memory and prediction (PLMP)* for the VMDP by describing the specific implementation of the memory and prediction components. With $I := (h, i)$, let the set $\Psi_I[t] = \{\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_{K[t]}\}$ be the collection of patterns for intersection I at time t, where $K[t] \in \mathbb{N}$ is the number of patterns currently recorded and each $\boldsymbol{\psi}_k$ represents a pattern. Note that for any $0 < s < t$, $\Psi_I[s] \subseteq \Psi_I[t]$. In our VMDP, the "patterns" of intersection I correspond to the distribution of vehicles in its local snapshot; for concreteness, we choose $\boldsymbol{\psi}_k \in (\mathbb{Z}^{\geq 0})^8$ to be a projection of a state $\mathbf{s}_{t,N} \in \mathcal{S}_N$ down to left and forward turns per direction; since we allowed vehicles to turn right whenever, they are not considered in the pattern.

3.1. Learning from Spatial Patterns

The VMDP implements the memory part of the PLMP controller architecture by storing any patterns that have frequently occurred in the past. This is motivated by the spatiotemporal symmetries that are likely to be prevalent throughout the rectangular grid, e.g., an intersection snapshot containing X number of vehicles in the North-South lanes and no vehicles in the East-West lanes is likely to occur again later in time. As opposed to "softer" methods of constraining these symmetries by adding inductive biases to the learning method, this implementation focuses on "harder" methods which hard-code a separate memory buffer to recall specific experiences and their rewards. This concept was inspired by *episodic memory* in the human brain (Lengyel and Dayan, 2007); some common implementations of episodic memory for control are Blundell et al. (2016), and Pritzel et al. (2017).

In our VMDP, episodic memory is implemented for each intersection I with a *memory table* $\mathcal{Q}_I : \mathbb{Z}^{\geq 0} \times \mathtt{Eq}(\Psi_I[t]) \times \mathcal{M} \to \mathbb{R}$, which maps patterns and light signal modes to best rewards. Compared to previous episodic memory approaches, each memory table in our VMDP also uses *equivalence classes* so that its size does not grow linearly with each new pattern. The original pattern collection $\Psi_I[t]$ is divided into multiple classes such that all patterns in a class are assigned the same optimal traffic light. We define $\mathtt{Eq}(\Psi_I[t]) \subseteq \Psi_I[t]$ to be the *unique keys* of the equivalence classes for $\Psi_I[t]$. Each entry $\mathcal{Q}_I(t, \boldsymbol{\psi}, m) = r$ means that as of time t, the best reward of r can be obtained by applying mode m to intersection I if the given pattern is $\boldsymbol{\psi}$.

For each intersection I, equivalence classes are constructed in the following simple way. For the first pattern $\boldsymbol{\psi}_1 \in \Psi_I[0]$, $\boldsymbol{\psi}_1$ is placed inside $\mathtt{Eq}(\Psi_I[t])$ and its associated equivalence class is

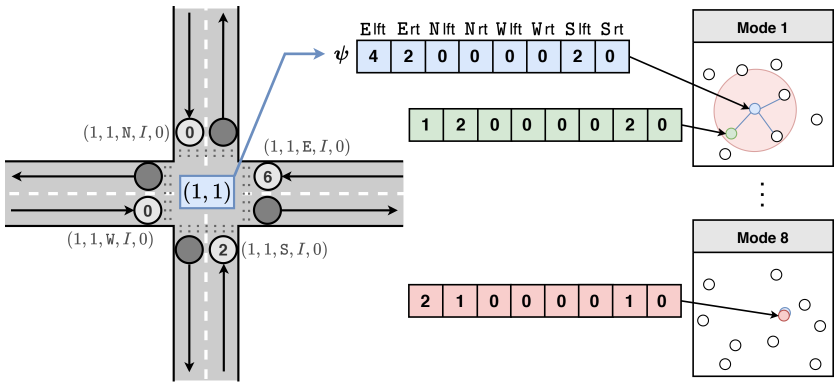

Figure 2: Example memory table $\mathcal{Q}_I$ for intersection $I := (1,1)$ with current pattern $\boldsymbol{\psi} = [4,2,0,0,0,0,2,0]$ (blue), $v^* = 2$, and $k = 3$ nearest neighbors. Entries in $\mathrm{Eq}(\Psi_I[t])$ are marked with white circles. For mode 1, $\boldsymbol{\psi}$ does not exist in $\mathcal{Q}_I$, so the 3 nearest patterns (large red ball) are used during lookup; one example of a "near" pattern is in green, where the left-turn lane in the East direction has three less vehicles. For mode 8, an entry for $\boldsymbol{\psi}$ already exists because it is equivalent to the red pattern, which is $\boldsymbol{\psi}/2$.

constructed as $\mathrm{Eq}(\boldsymbol{\psi}_1) = \mathrm{Eq}_0(\boldsymbol{\psi}_1)$, where

$$\mathrm{Eq}_0(\boldsymbol{\psi}_j) := \{v \cdot \boldsymbol{\psi}_j, v \in \{2,\cdots,v^*\}\} \cup \{[v_1 + \psi_{j,1},\cdots,v_8 + \psi_{j,8}], [v_1,\cdots,v_8] \in \{0,\cdots,v^*\}^8 \backslash \mathbf{0}\} \quad (1)$$

where $\cdot$ denotes multiplication by a scalar, v^* is from Section 2.3, and $\mathbf{0} \in \mathbb{R}^8$ is the all-zeros vector. This means $\mathrm{Eq}_0(\boldsymbol{\psi}_j)$ contains the following two types of elements: 1) every elementwise multiple of $\boldsymbol{\psi}_j$ up to a factor of v^*, 2) every nonzero additive variation of the entries of $\boldsymbol{\psi}_j$ up to v^*.

For each time $t+1$ when a new pattern $\boldsymbol{\psi}_k \notin \mathrm{Eq}(\Psi_I[t])$ is observed at intersection I, its equivalence class is constructed iteratively as:

$$\mathrm{Eq}(\boldsymbol{\psi}_k) := \begin{cases} \varnothing & \text{if } \exists\, \boldsymbol{\psi}_j \in \mathrm{Eq}(\Psi_I[t]) \text{ s.t. } \boldsymbol{\psi}_k \in \mathrm{Eq}(\boldsymbol{\psi}_j) \\ f(\mathrm{Eq}_0(\boldsymbol{\psi}_k), \{\mathrm{Eq}(\boldsymbol{\psi}_j)\}_{j=1}^{K[t]}, \mathrm{Eq}(\Psi_I[t])) & \text{else} \end{cases} \quad (2)$$

where Eq_0 is defined in (1). The function f is designed to check if every $\boldsymbol{\psi} \in \mathrm{Eq}_0(\boldsymbol{\psi}_k)$ is already in the pattern collection, whether as a unique key or an equivalence class member:

$$f(\mathrm{Eq}_0(\boldsymbol{\psi}_k), \{\mathrm{Eq}(\boldsymbol{\psi}_j)\}_{j=1}^{K[t]}, \mathrm{Eq}(\Psi_I[t])) := \left\{\boldsymbol{\psi} \in \mathrm{Eq}_0(\boldsymbol{\psi}_k) : \nexists \boldsymbol{\psi}_j \in \mathrm{Eq}_0(\boldsymbol{\psi}_k) \text{ s.t. } \left(\boldsymbol{\psi} = \boldsymbol{\psi}_j \wedge \boldsymbol{\psi} \in \mathrm{Eq}(\boldsymbol{\psi}_j)\right)\right\} \quad (3)$$

This construction allows all elements of $\Psi_I[t]$ to be partitioned into its unique keys and disjoint equivalence classes for all time t, i.e., $\Psi_I[t] = \mathrm{Eq}(\Psi_I[t]) \cup \mathrm{Eq}(\boldsymbol{\psi}_1) \cup \cdots \cup \mathrm{Eq}(\boldsymbol{\psi}_{K[t]})$. Looking up Q-values then amounts to looking through only $\mathrm{Eq}(\Psi_I[t])$ instead of the entire collection $\Psi_I[t]$, which reduces memory compared to other episodic control approaches. The update method of each intersection's memory table follows similarly to episodic control. At specific intersection I, suppose $\boldsymbol{\psi}$ is the current pattern snapshot observed at time t. If $\boldsymbol{\psi} \notin \Psi_I[t]$, the Q-value is approximated with $\hat{Q}_I$, which averages the Q-values of the k-nearest-neighbor (kNN) patterns in $\mathrm{Eq}(\Psi_I[t])$:

$$\hat{Q}_I(t,\boldsymbol{\psi},m) := \begin{cases} \frac{1}{k}\sum_{j=1}^{k} \mathcal{Q}_I(t,\hat{\boldsymbol{\psi}}_j,m) & \text{if } \boldsymbol{\psi} \notin \Psi_I[t] \\ \mathcal{Q}_I(t,\boldsymbol{\psi},m) & \text{else} \end{cases} \quad (4)$$

where $\{\hat{\psi}_j\}_{j=1}^k \subseteq \mathtt{Eq}(\Psi_I[t])$ are the k unique keys with the nearest distance to ψ at time t. Here, "nearest" is measured with ℓ_1-norm difference, modulus the structure of the equivalence classes: $d(\psi_k, \psi_j) := \left\|(\{\psi_k\} \cup \mathtt{Eq}(\psi_k)) - (\{\psi_j\} \cup \mathtt{Eq}(\psi_j))\right\|_1$ where we briefly abuse notation to denote $\|\mathcal{B}_1 - \mathcal{B}_2\|_1 := \min\{\|b_1 - b_2\|_1, b_1 \in \mathcal{B}_1, b_2 \in \mathcal{B}_2\}$. During training, the Q-values of the memory table are updated by comparing the existing value with the Bellman update. Denote $\overline{\psi} \in \mathcal{S}_N$ to be the expansion of ψ where zeros are placed in the positions of right-turning vehicles. Suppose the pair (ψ, m) at time t transitions to the pattern ψ^* via transition function $\mathscr{T}_I(\overline{\psi}^* | \overline{\psi}, m)$ and yields reward $\mathscr{R}_I(\overline{\psi}, m, \overline{\psi}^*)$, where $\mathscr{T}_I$ and $\mathscr{R}_I$ are dimension-reduced versions of $\mathscr{T}$ and $\mathscr{R}$ (from Section 2.3) for individual intersections. Then define:

$$r^* := (1-\alpha)\hat{Q}_I(t, \psi, m) + \alpha(\mathscr{R}_I(\overline{\psi}, m, \overline{\psi}^*) + \gamma \hat{Q}_I(t, \psi^*, m^*)) \tag{5}$$

Here, $\hat{Q}_I$ is the estimated Q-value computed through (4), $\alpha \in [0,1]$ is the learning rate, and $\gamma \in [0,1]$ is the reward discount rate. Mode m^* is the optimal light signal mode from pattern ψ^* (and varies by algorithm, e.g., Q-learning, SARSA). The update for entry (ψ, m) is performed as follows:

$$\mathcal{Q}_I(t+1, \psi, m) \leftarrow \begin{cases} \max\{\mathcal{Q}_I(t, \psi, m), r^*\} & \text{if } (t, \psi, m) \in \mathcal{Q}_I \\ r^* & \text{else} \end{cases} \tag{6}$$

The action $\mathbf{a}_t \in \mathcal{A}$ is then constructed by putting together all the optimal modes m^* of each intersection into a single vector. The PLMP algorithm with only memory implemented (without prediction) will henceforth be called *pattern-learning with memory (PLM)*; note that it differs from episodic control by implementation of the equivalence classes. For concreteness and variety, we consider two different ways of choosing the optimal mode m given pattern ψ. First, *greedy exploitation* uses transition function $\mathscr{T}_I$ to approximate the next ψ^* and chooses the mode m that maximizes the immediate reward $\mathscr{R}_I(\overline{\psi}, m, \overline{\psi}^*)$. Second, *episodic control (EC) exploitation* chooses the action m^* which maximizes (4). We also enable *exploration* with some probability $\epsilon \in [0,1)$.

3.2. Learning from Temporal Patterns

The VMDP implements the prediction part of the PLMP controller architecture by approximating future occurrences of patterns so that future light signal sequences can be scheduled in advance. Because the objective is to demonstrate the advantage of enabling prediction, we use a simple one-timestep lookahead assuming that all predictions are accurate due to sensors being abundantly placed throughout the grid; we defer the treatment of noisy predictions to future work.

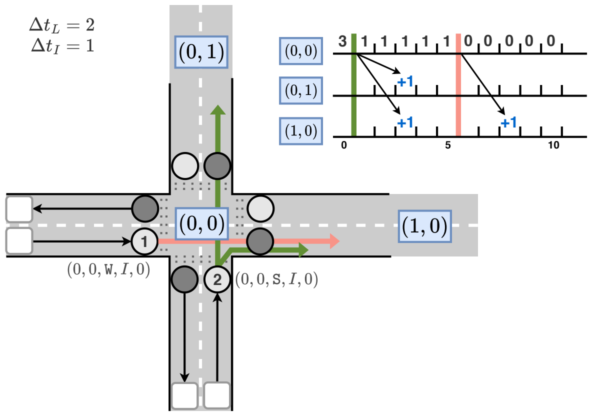

Figure 3: Sample prediction procedure for intersection $(0,0)$ and its neighbors $(0,1)$ and $(1,0)$. Here, $\Delta t_L = 2$ and $\Delta t_I = 1$. There are a total of three vehicles at $(0,0)$ at time 0: two vehicles (one right-turning, one forward-going) at direction $\mathtt{S}$ are given the green light to pass at time 0 while one vehicle (forward-going) at direction $\mathtt{W}$ is given the green light to pass at time 6. Here, there are no other vehicles in the system, so each vehicle takes $\Delta t_I + \Delta t_L = 3$ timesteps to reach their next intersection.

We employ an *augmented pattern* representation $\boldsymbol{\phi}_k = [\boldsymbol{\psi}_k^\top, \boldsymbol{\zeta}_k^\top]^\top \in (\mathbb{Z}^{\geq 0})^{16}$ associated with each original pattern $\boldsymbol{\psi}_k \in \Psi_I[t]$. The eight additional entries $\boldsymbol{\zeta}_k \in (\mathbb{Z}^{\geq 0})^8$ contains the counts of incoming vehicles in its adjacent links, and can be viewed as a projection of state $\mathbf{s}_{t,L} \in \mathcal{S}_L$ down to left and forward turns per direction. Define $\mathcal{P} : (\mathbb{Z}^{\geq 0})^{16} \to (\mathbb{Z}^{\geq 0})^8$ to be a *projection mapping* such that $\mathcal{P}(\boldsymbol{\phi}_k)$ is equal to the pattern which will occur in the next timestep. Because a vehicle's transition time from a link to an incoming node depends on the number of other vehicles that are currently present on the link, we do not write the explicit form of $\mathcal{P}$; essentially, we achieve accurate predictions by enabling one-timestep lookahead using the augmented pattern. For example, when $\Delta t_L = 1$ and there are no other vehicles in the left-turn lane of the link to the East of intersection I, we get $\mathcal{P}([\mathbf{0}^\top, \mathbf{e}_1^\top]^\top) = \mathbf{e}_1^\top$, where $\mathbf{e}_1$ is the first standard basis vector of $(\mathbb{Z}^{\geq 0})^8$.

We conclude this section with a side-by-side comparison of the algorithm pseudocode for vehicle traffic congestion control with PLM and PLMP. We emphasize that many design choices made in this section were chosen for concrete comparison between architectures with and without learning patterns; optimizing these design choices is a topic of future work.

Algorithm 1 Congestion Control via PLM

```
1:  Initialize VMDP.
2:  Initialize pattern tables $\{\Psi_I[0]\}$.
3:  Create next pattern $\boldsymbol{\psi}$.
4:  Create next traffic light from $\boldsymbol{\psi}$.
5:  for $t = 1 : T_{\text{sim}}$ do
6:      Propagate 1 step.
7:      Add any new vehicle arrivals.
8:      Update VMDP state.
9:      Update pattern tables $\{\Psi_I[t]\}$.
10:     Create next pattern $\boldsymbol{\psi}$.
11:     Create next traffic light from $\boldsymbol{\psi}$.
12: end for
```

Algorithm 2 Congestion Control via PLMP

```
1:  Initialize VMDP.
2:  Initialize pattern tables $\{\Psi_I[0]\}$.
3:  Predict next pattern $\boldsymbol{\psi}^* = \mathcal{P}(\boldsymbol{\phi})$.
4:  Create next traffic light from $\boldsymbol{\psi}^*$.
5:  for $t = 1 : T_{\text{sim}}$ do
6:      Propagate 1 step.
7:      Add any new vehicle arrivals.
8:      Update VMDP state.
9:      Update pattern tables $\{\Psi_I[t]\}$.
10:     Predict next pattern $\boldsymbol{\psi}^* = \mathcal{P}(\boldsymbol{\phi})$.
11:     Create next traffic light from $\boldsymbol{\psi}^*$.
12: end for
```

4. Numerical Simulations

We demonstrate the performance of various implementations of Algorithms 1 and 2. We compare the two ways from Section 3.1 in which actions are chosen: exploration with probability ϵ together with greedy or EC exploitation. We distinguish the way in which each intersection updates its memory table by varying the learning rate α: an *episodic control (EC) update* computes the new potential Q-value as in (5) with $0 < \alpha < 1$ (specifically chosen $\alpha = 0.9$), while a *greedy update* uses $\alpha = 0$. We also consider a *periodic baseline* controller, where the light at each intersection cycles through the modes repeatedly with some cycle duration $C \in \mathbb{N}$.

4.1. Dataset Preprocessing

We apply each variation of our proposed architecture to the following two datasets.

Pure Synthetic The number $V_A[t]$ of vehicles arriving into the network from the fringes as a function of time t is described as follows. Vehicles enter into the fringe intersections as platoons. Let T_n be the time of arrival for the nth platoon. Interarrival times $T_n - T_{n-1}$ are generated independently from a Geometric distribution with a time-varying parameter $p[t] \in (0, 1)$. At each arrival

Strategy	Description
Periodic16	Periodic with $C = 16$.
Periodic8	Periodic with $C = 8$.
EC(0)	PLM with EC exploitation, $\epsilon = 0$ exploration, EC update
PLM(0,1,1)	PLM with greedy exploitation $\epsilon = 0$ exploration, greedy update
PLM(5e-3,1,0)	PLM with greedy exploitation $\epsilon = 5\text{e-}3$ exploration, EC update
PLMP(0,0,0)	PLMP with EC exploitation $\epsilon = 0$ exploration, EC update
PLMP(0,1,1)	PLMP with greedy exploitation $\epsilon = 0$ exploration, greedy update
PLMP(5e-3,1,0)	PLMP with greedy exploitation $\epsilon = 5\text{e-}3$ exploration, EC update

Table 1: The eight different controller implementations compared in the experiment: two periodic baselines and six different versions of Algorithms 1 and 2. PLM$(0, 0, 0)$ is equivalent to episodic control (EC), but with equivalence classes implemented.

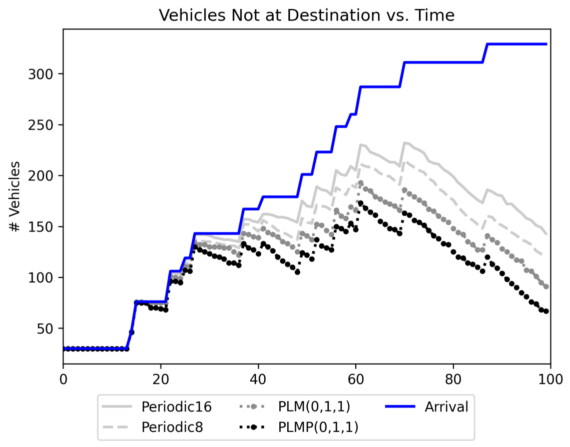

Figure 4: Congestion metric $V_C[t]$ from Definition 2 plotted over time until $t = 100$ for four strategies from Table 1, applied to the pure synthetic dataset. The cumulative number of vehicles in the system over time, $V[t]$, is shown in solid blue.

time T_n, the size of each platoon is a random nonzero integer generated between some minimum and maximum size. We choose experiment duration $T_{\text{sim}} = 200$, $p[t] \in [0.1, 0.25]$, and each platoon generated has some size between 30 to 50 vehicles.

Intersections in Hangzhou, China We use real-world data of traffic flowing through several single intersections in Hangzhou, China (provided in TSCC (2019)). For our grid network setting, we assign the behavior of one intersection to each of the four fringes of the grid: intersections on the East fringe of the grid behave according to the `kn-hz` intersection, the North fringe is according to the `qc-yn` intersection, West to `sb-sx`, and South to `tms-xy`. Each intersection in the original dataset consists of two one-hour arrivals of vehicles; we add the two arrival processes together and discretize arrivals into 5-second bins, i.e., $T_{\text{sim}} = 3600/5 = 720$. The original vehicle routes are also modified to respect the constraints of being a fringe intersection, e.g., traffic emerging from the East side of `sb-sx` are rerouted to emerge from the West side (because `sb-sx` corresponds to fringe intersections of the form $(0, i)$ for $i \in \{0, \cdots, L-1\}$).

4.2. Results

We evaluate eight different controller implementations, using the congestion metrics from Definition 2, on a grid network with dimensions $H = L = 5$. Each implementation is described in Table 1. Our results for $v^* = 2$, $\Delta t_L = 2$, $\Delta t_I = 1$, $k = 3$ nearest neighbors, and $\gamma = 0.1$ averaged over 20 Monte-Carlo trials, are demonstrated in Table 2 for the synthetic dataset (left subtable) and the Hangzhou dataset (right subtable). A sample plot of one trial of $V_C[t]$ (the number of circulating vehicles metric from Definition 2) for the synthetic dataset until $t = 100$ timesteps is shown in Figure 4; the figure for the Hangzhou dataset yielded a similar trend, but with smaller values because the arrival process is much thinner than the synthetic dataset despite being over longer time interval.

The basic trend for both datasets is that PLMP does better on average than PLM and PLM does better on average than the periodic baseline. In Figure 4, all four controllers experience at least

Strategy	W	D
Periodic16	12.21145	44.37445
Periodic8	20.25	58.79762
EC(0)	1.2622	24.27622
PLM(0,1,1)	1.10247	23.75265
PLM(5e-3,1,0)	4.62031	30.0
PLMP(0,0,0)	0.41584	22.19802
PLMP(0,1,1)	0.38158	21.92434
PLMP(5e-3,1,0)	2.42907	28.37716

Strategy	W	D
Periodic16	12.13208	38.20283
Periodic8	6.87660	26.78723
EC(0)	0.89105	14.80156
PLM(0,1,1)	0.89412	14.93334
PLM(5e-3,1,0)	0.90551	14.79528
PLMP(0,0,0)	0.22509	9.98524
PLMP(0,1,1)	0.23443	9.17216
PLMP(5e-3,1,0)	0.25368	9.49632

Table 2: The average cumulative waiting time W and the average travel deviation D (from Definition 2) for each of the two datasets. [Left] Pure synthetic. [Right] Hangzhou.

a 15-timestep delay after time 0 from which vehicles begin to reach their respective destinations; thus, each line follows the cumulative number of vehicles in the system (blue line) precisely until timestep 15. Afterwards, however, PLMP consistently begins to drop first, followed by PLM, then finally the periodic baseline, indicating that PLMP enables vehicles to reach their destination the fastest on average and periodic enables the slowest. This is also consistent with the magnitude of the measurements in Table 2. For the periodic baselines applied to the synthetic dataset (heavier traffic), the left subtable in Table 2 shows that a smaller period (i.e., faster light signal switching) can cause more congestion than relief. For PLM and PLMP, the average waiting time per vehicle (W) and average travel deviation (D) are mostly consistent to how they increase or decrease in value together. Adding exploration causes both PLM and PLMP to perform worse, which is expected because traffic in a structured, predictable setting like a rectangular grid leaves very little chance that choosing a random mode will perform better than pure exploitation.

5. Conclusion

This paper presented new controller architectures based on pattern-learning with memory and prediction (PLMP) and pattern-learning with memory (PLM) for vehicle traffic congestion control over a metropolitan grid of signalized intersections. The architectures exploited the natural spatial symmetries and temporal repetition in the traffic network to perform control without redundant computation of light signal sequences. In particular, the memory component used an extension of episodic memory which builds equivalence classes to group together patterns that are controlled using the same light signals. In addition, accurate predictions are incorporated with a one-timestep lookahead that augments vehicle counts in the adjacent links of the intersection and schedules light signals in advance. We demonstrated the performance of multiple implementations of PLM and PLMP with respect to three congestion metrics over two different traffic scenarios, and found that on average, PLM outperforms the periodic baselines while PLMP outperforms PLM with mild variation among the different implementations. Future work includes noisy predictions, predictions over longer horizons, and comparing with other traffic optimization architectures (e.g., GNNs, rule-based).

Acknowledgments

This paper is based on work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1745301 and the Aerospace Corporation.

References

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-Free Episodic Control. *ArXiv preprint, arXiv:1606.04460*, 2016.

Matthew Botvinick, Sam Ritter, Jane X. Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement Learning, Fast and Slow. *Trends in Cognitive Sciences*, 23(5):408–422, 2019.

Samuel Coogan, Ebru Aydin Gol, Murat Arcak, and Calin Belta. Controlling a Network of Signalized Intersections from Temporal Logical Specifications. In *2015 American Control Conference (ACC)*, pages 3919–3924, 2015.

François Dion and Bruce Hellinga. A Rule-Based Real-Time Traffic Responsive Signal Control System with Transit Priority: Application to an Isolated Intersection. *Transportation Research Part B: Methodological*, 36(4):325–343, 2002.

SooJean Han, Soon-Jo Chung, and John C. Doyle. Predictive Control of Linear Discrete-Time Markovian Jump Systems via the Analysis of Recurrent Patterns. *Automatica*, 2023. URL https://arxiv.org/abs/2305.05587. To appear.

Simon Kwoczek, Sergio Di Martino, and Wolfgang Nejdl. Predicting and Visualizing Traffic Congestion in the Presence of Planned Special Events. *Journal of Visual Languages & Computing*, 25(6):973–980, 2014.

Máté Lengyel and Peter Dayan. Hippocampal Contributions to Control: The Third Way. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*, 2018.

Jennie Lioris, Ramtin Pedarsani, Fatma Yildiz Tascikaraoglu, and Pravin Varaiya. Platoons of Connected Vehicles Can Double Throughput in Urban Roads. *Transportation Research Part C: Emerging Technologies*, 77:292–305, 2017.

Guangquan Lu, Lumiao Li, Yunpeng Wang, Ran Zhang, Zewen Bao, and Haichong Chen. A Rule Based Control Algorithm of Connected Vehicles in Uncontrolled Intersection. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 115–120, 2014.

Alan J Miller. A Queueing Model for Road Traffic Flow. *Journal of the Royal Statistical Society: Series B*, 23(1):64–90, 1961.

Ajith Muralidharan, Ramtin Pedarsani, and Pravin Varaiya. Analysis of Fixed-time Control. *ArXiv preprint, arXiv:1408.4229*, 2015.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural Episodic Control. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2827–2836. PMLR, 2017.

Benjamin Rivière and Soon-Jo Chung. H-TD2: Hybrid Temporal Difference Learning for Adaptive Urban Taxi Dispatch. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10935–10944, 2022.

TSCC. Reinforcement Learning for Traffic Signal Control: Benchmark Dataset for the Traffic Signal Control Competition (TSCC), 2019. [Link].

Feng Xie and David Levinson. Measuring the Structure of Road Networks. *Geographical Analysis*, 39(3):336–356, 2007.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 3634–3640. AAAI Press, 2018.

Kun Zhang and Michael A.P. Taylor. Effective Arterial Road Incident Detection: A Bayesian Network Based Algorithm. *Transportation Research Part C: Emerging Technologies*, 14(6):403–417, 2006.