TRAMESINO: Traffic Memory System for Intelligent Optimization of Road Traffic Control*

Cristian Axenie[1], Rongye Shi[2](✉), Daniele Foroni[1], Alexander Wieder[1], Mohamad Al Hajj Hassan[1], Paolo Sottovia[1], Margherita Grossi[1], Stefano Bortoli[1], and Götz Brasche[1]

[1] Intelligent Cloud Technologies Lab, Huawei Munich Research Center
Riesstrasse 25, 80992 Munich, Germany
cristian.axenie@huawei.com

[2] EI Intelligence Twins Program, Huawei Cloud BU, Shenzhen, China
shirongye@huawei.com

Abstract. Whether efficient road traffic control needs accurate modelling is still an open question. Additionally, whether complex models can dynamically adapt to traffic uncertainty is still a design challenge when optimizing traffic plans. What is certain is that the highly non-linear and unpredictable real-world road traffic situations need timely actions. This study introduces TRAMESINO (TRAffic Memory System INtelligent Optimization). This novel approach to traffic control models only relevant causal action-consequence pairs within traffic data (e.g. green time - car count) in order to store traffic patterns and retrieve plausible decisions. Multiple such patterns are then combined to fully describe the traffic context over a road network and recalled whenever a new, but similar, traffic context is encountered. The system acts as a memory, encoding and manipulating traffic data using high-dimensional vectors using a spiking neural network learning substrate. This allows the system to learn temporal regularities in traffic data and adapt to abrupt changes, while keeping computation efficient and fast. We evaluated the performance of TRAMESINO on real-world data against relevant state-of-the-art approaches in terms of traffic metrics, robustness, and run-time. Our results emphasize TRAMESINO's advantages in modelling traffic, adapting to disruptions, and timely optimizing traffic plans.

1 INTRODUCTION

Solving traffic congestion in urban agglomerations is still a problem resistant to straightforward solutions despite the large amount of research and systems developed to analyze [20], model [23], and control road traffic [26]. Systems deployed in real-world [10,14,6] use a traffic model [24,7] that heavily influences the run-time performance of the overall system. Basically, the role of the traffic model is to describe the dynamics of the traffic flow and to cope, eventually, with

* C. Axenie, R. Shi, D. Foroni, A. Wieder, M. A. H. Hassan, P. Sottovia, M. Grossi and S. Bortoli—Authors contributed equally to this research.

unforeseen deviations (i.e. disruptions) in traffic patterns [24]. But, in order to achieve that, the system needs to optimize multiple metrics, such as spatial and temporal traffic demand, traffic volume [3]. This implies a substantial computational cost that might hinder the overall real-time capabilities of the system and increase the cost of large scale traffic optimization. It is the system designer's duty to make a trade-off between two dimensions, namely performance and execution time. The present study addresses the problem of such costly optimization routines and explores a novel approach to speed-up traffic control, named TRAMESINO (TRAffic Memory System INtelligent Optimization). At the core of TRAMESINO is the capability to exploit the similarity and invariant features of traffic flow patterns. Basically, by storing relevant causal patterns (i.e. action/consequence: allocated green time/measured car count) in traffic flows, one can bypass the costly constrained optimization routines typically employed in traffic control systems. Such patterns can be correlated in time to fully describe the traffic context in an entire region. Given "cues" of traffic data (i.e. current car count), the system can "recover" a plausible cause (i.e. the allocated green time). To achieve this TRAMESINO uses:

- an efficient encoding scheme for traffic timeseries covariates;
- a mechanism storing associations among traffic timeseries covariates;
- an efficient learning framework to natively process the encoded quantities and implement the association dynamics.

In the remainder of this section, we ground our contribution and emphasize those relevant features and drawbacks of adaptive traffic control systems motivating our study.

1.1 Optimization-based Adaptive Traffic Control Systems

Traditionally, flow optimization for coordinated traffic signals is based on average travel times between intersections and average traffic volumes at each intersection [9]. However, most of these approaches do not consider the stochastic nature of high-resolution field traffic data or capture it through computationally expensive processes, such as Markov Decision Processes (MDP)[22]. Beyond stochasticity, the community also explored the use of mixed-integer linear programming (MILP) for optimizing the control of traffic signals, in particular, offsets, split times, and phase orders [11]. The approach provided optimal results but with a high computational cost. Additionally, such systems couldn't handle changes of the controlled variables in real-time due to the optimization process that needs to iterate to convergence. In a first attempt to exploit the periodic nature of the traffic signals, the work in [18] formulated the traffic light optimization into a continuous optimization problem without integer variables, by modeling traffic flow as sinusoidal. The system solved a convex relaxation of the non-convex problem using a tree decomposition reduction with very good performance in simulations, but it lacked the capability to scale and adapt to traffic disruptions. Finally, relying on predicting arrivals at coordinated signal approaches the work in [2] proposed the link pivot algorithm that assumed nearest-neighbor interactions between signals in cyclic flow profiles to model traffic flows. Despite its well performing optimization, the algorithm couldn't handle unpredictable changes in platoon shapes (i.e. occasionally caused by platoon splitting and merging) or prediction during saturated conditions (i.e. traffic jams, accidents) limiting its use in real-world deployments. As we briefly emphasized hitherto, aspects such as stochasticity, simultaneous traffic assignment and traffic signal calculation, periodicity, regional scaling, and real-time constraints, describe real-world traffic situations. Each system excels in handling a sub-set of these aspects only and cannot capture their combined impact on traffic dynamics.

1.2 Beyond Optimization

Historically, neural networks were employed in traffic control to exploit its intrinsic temporal dynamics. A reference work in this category is the study of [16] which proposed a Hopfield network-based system designed to capture temporal patterns. Opposite to optimization approaches, such a system exploited the interaction between neurons whose dynamics modelled traffic signals state changes and stochasticity. These first steps away from optimization, were extended in [17] by emphasizing the purpose of feedback loops for decreasing the differences of the conflicting flows, measured during a congestion or large number of waiting vehicles. This solution enabled regional scaling and simultaneous traffic assignment and traffic signal calculation using the same network, by exploiting the capability to describe and solve a constraint satisfaction problem of Hopfield networks[8]. Using a simplified, linear Hopfield neural network, the study in [12] proposed a system capable of solving an arbitrary set of (linear) equations through online learning. Interestingly, the typical Hopfield network was augmented with an additional feed-forward layer used to compute the Moore-Penrose Generalized Inverse (i.e. pseudoinverse) of the weight connection matrix. Calculating the pseudoinverse, allowed the system to actually compute a "best fit" (in least squares sense) solution to the evolving traffic dynamics model (i.e. unexpected disruptions "move" the optimum in solution space) in a parallel fashion. Addressing the scaling problem, the study in [27] employed an augmented Hopfield network to solve mixed integer programming. The approach exploited the temporal dynamics of the Hopfield network to find better solutions than Lagrangian relaxation and only very rarely converged to unfeasible solutions. Finally, despite the advantages that systems, such as Hopfield networks, have, there are some known barriers to deploy them to real-world scenarios. A first aspect refers to the limited memory capacity of Hopfield networks, and the actual computational cost of storing a large number of memories - which increases with the number of neurons. Another strong limitation is the pattern orthogonality assumption, which limits the recall accuracy, especially at scale. Considering the unique and optimal recall of Hopfield networks, there are strong limitations due to the existence of local minima and spurious states of attraction. Such a limitation is stronger in the case of storing high-dimensional traffic contexts, where due to increased similarity, identifying the discrepancy is difficult, especially in the presence of a large number of states of attraction.

1.3 Motivation and Contributions

Besides the relevant aspects already mentioned, a significant drawback of existing traffic control systems is that they fail to fully exploit the causal coupling (or associations) between traffic control signals and traffic flow dynamics. It is known that, despite being highly nonlinear, traffic dynamics is regular on certain timescales. Such regularities together with available sensory data can be used to judiciously extract traffic contexts that can be subsequently used in optimizing future traffic situations. Basically, the associations among the control signals (i.e. green time / red time) and the measured outcomes (i.e. flow of cars) capture the dynamics of traffic on a road, intersection, or region. Obviously, in order to optimize flow and minimize delay time, the traffic control system would need to find the best traffic light timing. This functionality is described in Figure 1.

In this context, our contribution focuses on four main points:

– optimizing the time for decision-making and ”short-circuit” re-computation of a control signal (i.e. green time allocation) by exploiting previously learnt patterns of traffic context (i.e. traffic flow – green time pair). Metaphorically, TRAMESINO accumulates wisdom over traffic optimization, and uses the acquired knowledge to bypass possibly computationally complex decision-making processes based solely on the ongoing traffic perception.
– representing traffic contexts (i.e. regional traffic flow, local allocated green times, etc.) as a ”memory”, basically a high-dimensional numeric vector depicting the traffic state at a certain moment in time. Additionally, such memories can be stored and recovered using a learning system, which is at the core of TRAMESINO. This way TRAMESINO can exploit the descriptive power of pairs of actions and their outcomes in order to learn memories from historical data. Such memories of associations speed-up operation, when facing new traffic situations by recalling the most similar (previously seen) traffic context. To support this speed-up, the contexts are represented using high-dimensional vectors, which map the complex dynamics to simple (algebraic) operations in high-dimensional spaces.
– exploiting learned context associations among patterns of traffic (i.e. traffic flow – allocated green time) to infer what would be the most plausible traffic flow when a control signal (i.e. green time) is available and what would be the green time for measured flow values. In other words, given a partial context, TRAMESINO recalls the most similar context learnt in the past by restoring the missing part of the context (i.e. either green time for measured flow or flow for applied green time) - similarly to an autoassociative memory.
– release of a new real-world dataset, used in the TRAMESINO experiments, which contains 74 days of real urban road traffic data from 8 crosses in a city in China.

The main problems the proposed system solves are:

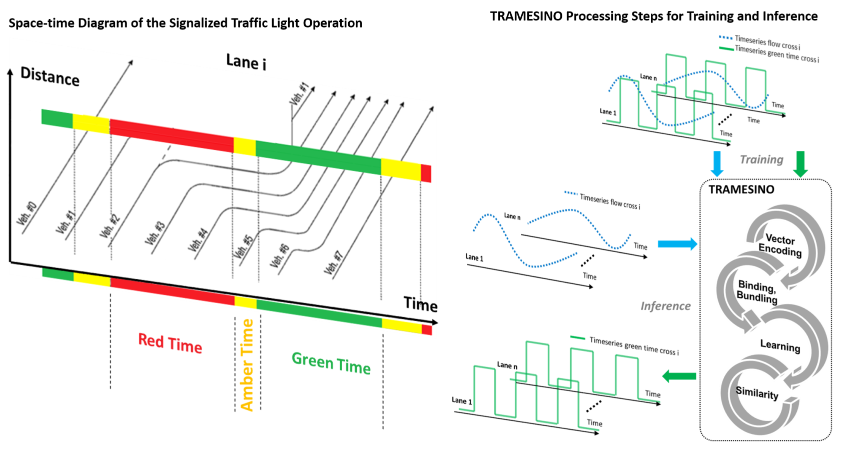


Fig. 1. TRAMESINO System functionality overview.

- efficiently representing traffic context using the measured data (i.e. traffic flow) and control signals (i.e. allocated green time) in a system capable of learning multiple associations among such causal data encoded in efficient high-dimensional vectors;
- avoiding costly optimization methods and control signal re-computation by exploiting previously learnt patterns of traffic data and infer, given partial information (i.e. either traffic flow or allocated green time), what would be the best corresponding full context corresponding to the partial information;
- scalability through storing multiple memories (i.e. multiple full traffic contexts) and deployment at different granularity (e.g. per lane, per direction, per intersection);
- the efficient computation of traffic control signals (i.e. green time) that embed and exploit the intrinsic traffic constraints and physics without an explicit need to model the constraints;

2 MATERIALS AND METHODS

In this study, we introduce TRAMESINO, a flexible framework and system capable to learn and store associations among measured traffic data (e.g. traffic flow) and the corresponding traffic control signal generating it (i.e. allocated green time). In order to speed up computation in similar, but novel, traffic situations, the system recalls the most plausible learnt association.

2.1 Introducing TRAMESINO

TRAMESINO is an associative memory system for traffic flow optimization. The system builds a vector description of the current traffic context from timeseries of specific traffic data (i.e. flow of cars, green time, traffic density). The key ingredient of TRAMESINO is the Holographic Reduced Representation (HRR)[19], responsible for the traffic data encoding, learning, and computation with the encoded quantities. Such a framework demonstrated already that structured vector-representations are able to capture relations and mutual influence between multiple traffic context data [15].

HRR are a type of Vector Symbolic Architectures (VSAs) [5] that describe a family of modelling approaches to represent physical quantities by mapping them to (high-dimensional) vectors. Beside the numerical structure underlying the vectors, the core computational components of a VSA are a measure of similarity and typically two algebraic operations, namely superposition and binding. Superposition implements the basic addition and combines multiple vectors to create a vector similar to the input vectors. Binding implements the basic multiplication in order to produce highly dissimilar response to both input vectors. A very important aspect is the fact that binding is invertible and preserves distance metrics which support the associative memory implementation. Within TRAMESINO, superposition allows storing multiple traffic contexts defined by available traffic data (i.e. flow, green time, cycle time, phase length), whereas binding provides the core mechanism to recall previously stored contexts given a similarity metric. An important property of the high-dimensional vector space in TRAMESINO is that with a very high probability all stored vectors are dissimilar to each other (i.e. quasi-orthogonal). This enables the system to implement the associative memory behavior using simple operations in high dimensions. Finally, unlike many traditional neural networks, HRR do not rely on backpropagation but rather on algebraic operations on high-dimensional vectors which are embarrassingly parallel operations that can be performed efficiently (in principle, in constant time).

Data representation TRAMESINO uses high-dimensional HRR vectors and operations to represent traffic contexts (i.e. action-consequence pairs) and do computation (i.e. associative memory). Intuitively, for practical use, TRAMESINO needs to store multiple such contexts as memories to be able to handle arbitrary new contexts. As mentioned, in order to store multiple vectors encoding traffic contexts, TRAMESINO utilizes bundling, which accounts for an element-wise addition of the vectors. For the recall phase, TRAMESINO utilizes binding, which is basically implementing a circular convolution. We now introduce the formalism behind the specific HRR operations in TRAMESINO.

HRR allow for complex vector values, i.e., $N \subseteq \mathbb{C}$ and use a multiplication operation $\circledast$ based on circular convolution. For any two vectors of size D, $\mathbf{x}, \mathbf{y} \in V_D(N)$, circular convolution $\circledast$ is defined as

$$\mathbf{z} = \mathbf{x} \circledast \mathbf{y} \qquad \text{with } z_j = \sum_{k=0}^{D-1} x_k y_{(j-k) mod D}. \tag{1}$$

Circular convolution can efficiently be computed using the the Discrete Fourier Transform (DFT) [1] defined as the function

$$DFT : \mathbf{C}^D \to \mathbf{C}^D, \mathbf{x} \to \left(\textstyle\sum_{j=0}^{D-1} x_j \zeta_D^{-jk} \right)_{k=0}^{D-1} \qquad \text{with } \zeta_D = \exp\left(\frac{i2\pi}{D}\right). \tag{2}$$

Similarly, the Inverse Discrete Fourier Transform (IDFT) is defined as the function

$$IDFT : \mathbf{C}^D \to \mathbf{C}^D, \mathbf{x} \to \left(\frac{1}{D} \sum_{j=0}^{D-1} x_j \zeta_D^{jk}\right)_{k=0}^{D-1}. \tag{3}$$

In TRAMESINO, we make use of the fact, that circular convolution can be written as a combination of the DFT, IDFT, and element-wise multiplication $\odot$ [19]. Using the convolution theorem, we can calculate the circular convolution of any two vectors $\mathbf{v}, \mathbf{w} \in V_D(N)$ by

$$\mathbf{v} \circledast \mathbf{w} = IDFT\left(DFT(\mathbf{v}) \odot DFT(\mathbf{w})\right), \tag{4}$$

with $\odot$ denoting element-wise multiplication in this case. This induces that circular convolution obeys the same commutative and associative rules as element-wise multiplication. Additionally, we define the convolutive power as the real part of the transformed vector

$$\mathbf{v}^p := \Re\left(IDFT\left(DFT\left(\mathbf{v}\right)^p\right)\right), \tag{5}$$

This operation is used when recalling a traffic memory. This involves building the HRR vector of a partial context (i.e. traffic flow car count), bundling and binding it to existing memories, and then computing the similarity.

Next, we describe the encoding, its constraints, and the considerations to handle temporal aspects (i.e. traffic contexts are timeseries of various traffic measured quantities). TRAMESINO uses the unitary base of vectors $\mathbf{b}$ for encoding (real-valued) scalar traffic quantities (i.e. flow of cars, green time) in high-dimensional HRR vectors, which are in fact combinations of basis vectors using simple algebraic operations. Additionally, it uses $C_0, \dots, C_D$ to represent each type of traffic data and $T_0, \dots, T_D$ for encoding the temporal structure (i.e. timestamps). As a design choice, we use unitary vectors u, since they have some desirable properties, namely $|\mathbf{u}| = 1$, $\mathbf{u}^p$ is still unitary for any $p \in \mathbb{R}$, and convolution with unitary vectors preserves the norm, i.e., $|\mathbf{v}| = |\mathbf{v} \otimes \mathbf{u}|$ for any other vector $\mathbf{v}$. We can now create actual HRR vectors V_i of different traffic quantities values v_i as

$$V_i = \sum_{j=1}^{D} C_j \otimes \mathbf{b}^{v_j \cdot s}, \tag{6}$$

where s is a scaling factor. To additionally encode the temporal structure, we simply bind each traffic quantity vector V_i to a vector T encoding the timestamp,

$$V_T = \sum_{i=1}^{D} \left(\sum_{j=1}^{D} C_j \otimes \mathbf{b}^{v_j \cdot s}\right) T_i. \tag{7}$$

Learning and inference In TRAMESINO, the HRR traffic data representation and the HRR binding and bundling operations, are implemented in the Neural Engineering Framework (NEF) [4]. NEF offers a systematic method of "compiling" high-level descriptions, such as vector convolution, correlation, and similarity, into synaptic connection weights between populations of spiking neurons with efficient learning capabilities. In NEF, neural populations represent time-varying signals, such as traffic flow data, through their spiking activity. Such signals drive neural populations based on each neuron's tuning curve, which describes how much a particular neuron will fire as a function of the input signal.

Formally, we consider A a population of $N \in \mathbb{N}$ neurons encoding a subset V of a real-valued vector space, i.e., $V \subseteq \mathbb{R}^n$, representing measurable traffic quantities. Given a function $\mathbf{x}(\mathbf{t}) : \mathbb{R} \to V$, we can write the activity a_i at time t of the i-th neuron in a neural population encoding a time-varying vector (e.g. traffic flow data) $\mathbf{x}(t)$ as a spike train,

$$a_i\left(\mathbf{x}(t)\right) = \sum_{j=1}^{m_i} \delta(t - t_j) = G_i(\alpha_i \langle \mathbf{e}_i, \mathbf{x}(t) \rangle + J_i) \quad \text{for } 1 \leq i \leq N, \tag{8}$$

where G_i is the neural non-linearity, α_i is the gain of the neuron, $\mathbf{e}_i$ is the neuron's preferred encoding vector, J_i describes the neural background activity, and t_j are the m_i spike-times of the i-th neuron, and $\langle . \rangle$ is the inner product. To decode the traffic quantities $\mathbf{x}(t)$ back out of the neural population A, the spike train is convolved $*$ with an exponentially decaying filter $h : \mathbb{R} \to \mathbb{R}$ resulting in

$$\tilde{a}_i\left(\mathbf{x}(t)\right) = \sum_{j=1}^{m_i} h(t) * \delta(t - t_j) = \sum_{j=1}^{m_i} h(t - t_j). \tag{9}$$

We consider here the exponential decaying filter given by $h : \mathbb{R} \to \mathbb{R}, t \to e^{\frac{-t}{\tau_p}}$, where τ_p is the post-synaptic time constant. Through filtering we obtain an estimation $\hat{\mathbf{x}}(t)$ of the original input $\mathbf{x}(t)$ as a weighted sum with some decoder values $\mathbf{d}_i$

$$\hat{\mathbf{x}}(t) = \sum_{i=1}^{N} \tilde{a}_i\left(\mathbf{x}(t)\right) \mathbf{d}_i. \tag{10}$$

To calculate the optimal decoders $\mathbf{d}_i$, the system needs to minimize the error between input $\mathbf{x}(t)$ and decoded output $\hat{\mathbf{x}}(t)$

$$E = \int \left(\mathbf{x}(t) - \sum_{i=1}^{N} \tilde{a}_i\left(\mathbf{x}(t)\right) \mathbf{d}_i \right)^2 d\mathbf{x}(t). \tag{11}$$

NEF solves for the decoders $\mathbf{d}_i$ by default using an efficient least squares optimization [4].

Encoding and decoding operations on NEF neural populations representations allow us to encode traffic flow signals over time, and decode transformations (i.e. mathematical functions) of those signals. In fact, NEF allows us to decode arbitrary transformations of the input traffic data by computing functions across the connections between the populations of neurons encoding the traffic data. For instance, if we consider A resp. B populations of N resp. M

neurons encoding a time-varying vector $\mathbf{x}(t) \in V \subset \mathbb{R}^n$ (e.g. traffic flow) resp. $\mathbf{y}(t) \in W \subset \mathbf{R}^m$ (e.g. traffic density) and a function $f : V \rightarrow W \subset \mathbf{R}^m$. In order to approximate the function f (i.e. traffic flow - density dependency) across a connection from population A to population B, TRAMESINO calculates a set of decoder values $\mathbf{d}_i^f$ for population A by minimizing the error

$$E = \int \left(f(\mathbf{x}(t)) - \sum_{i=1}^{N} \tilde{a}_i\left(\mathbf{x}(t)\right) \mathbf{d}_i^f \right)^2 d\mathbf{x}(t). \tag{12}$$

Given encoders $\mathbf{e}_j^B$ and gain α_j^B for $1 \leq j \leq M$ of population B, we can derive a weight matrix for the connection from A to B approximating the function f by

$$w_{ij} = \alpha_j^B \mathbf{d}_i^f L \mathbf{e}_j^B \quad \text{for } 1 \leq i \leq N \text{ and } 1 \leq j \leq M, \tag{13}$$

where L is a $M \times N$ linear operator. Here, NEF makes the assumption, that connection weights can be factored into encoders, decoders, and a transform. Finally, in order to implement the associative memory behavior, we need to describe the dynamics of such an operation. But first we introduce how can we implement such dynamics in populations of spiking neurons. If we consider A

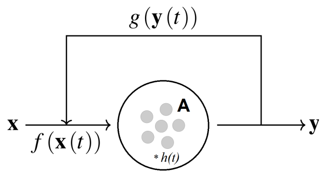


Fig. 2. Dynamics implementation of TRAMESINO associative memory.

a population of neurons with an incoming connection approximating the function $f : V \rightarrow W \subset \mathbb{R}^m$ and a recurrent connection approximating the function $g : W \rightarrow W$ (cf. Fig. 2). Thus, the overall function the population is approximating is

$$\mathbf{y}(t) = h(t) * (f(\mathbf{x}(t)) + g(\mathbf{y}(t))) \tag{14}$$

with exponential decaying filter function $h : \mathbb{R} \rightarrow \mathbb{R}, t \rightarrow e^{\frac{-t}{\tau}}$. By setting the functions $g(\mathbf{y}(t)) = \tau a(\mathbf{y}(t)) + \mathbf{y}(t)$ and $f(\mathbf{x}(t)) = \tau b(\mathbf{x}(t))$ - with a and b arbitrary nonlinear functions - we obtain a neural model approximating a dynamical system. The learning rule implementing the autoassociative memory needs to modify the encoding vectors of active neurons to be selective to an input vector (i.e. a partial context, traffic flow). Basically, this operation adjusts the connection weights so that a small number of distinct neurons respond to each such partial traffic context - by triggering the memory most similar to it. For this we used the three layer neural autoassociator using NEF spiking neurons from [25]. Given a traffic context vector x encoded by the activity of the input neural population, the filtered activity $a(t)$ of neurons in the middle layer, and the matrix $\mathbf{e}$ whose rows are the "preferred traffic context" vectors of the middle layer neurons, we modify the "preferred traffic context" vectors of the middle layer neurons according to:

$$\frac{\partial \mathbf{e}(t)}{\partial t} = -\eta a(t)\mathbf{e}(t) + \eta(a(t)\mathbf{x}^T(t), \tag{15}$$

where η is the learning rate. Changing the "preferred traffic context" vectors corresponds to changing the connection weights using a local learning rule in Equation 15. This system has been proven to have high accuracy, a fast, feed-forward recall process, and efficient scaling, requiring a number of neurons linear in the number of stored associations.

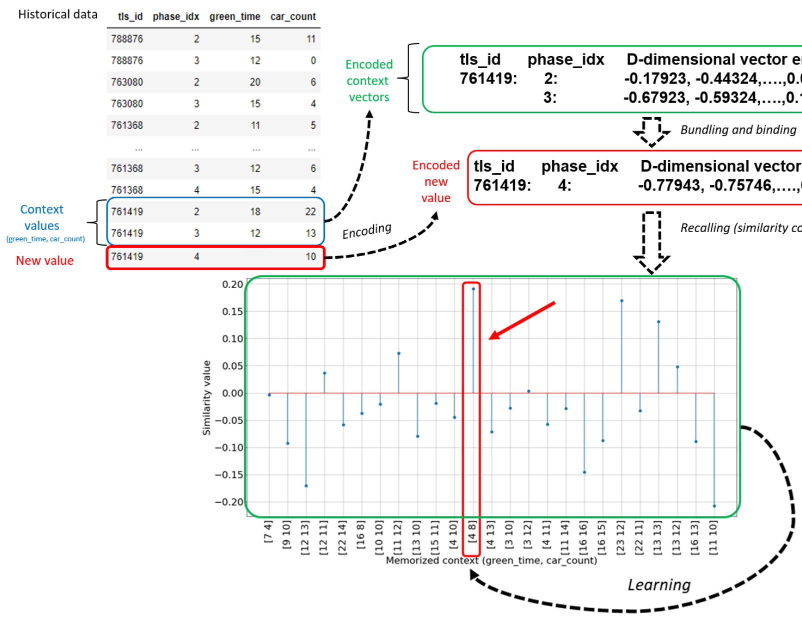
Parametrization In all our experiments, within TRAMESINO[3], traffic flow readings from an urban region were concatenated in a context vector (i.e. memory) of size $D = 1024$, each encoding neural population had a size of 100 neurons, a new memory was stored for each traffic light and each phase every $n = 10$ traffic light cycles (i.e. accounting for a memory every approx. 5 minutes), and a new green time was recalled at the generation of each new plan (i.e. approx. every 2 minutes). Note that, increasing the number of neurons (i.e ≥ 1000) provides a more accurate encoding and, hence results, but the computation time increase supra-linearly. Our parametrization reflects the trade-off to make for the computation time gain. Figure 3 provides an overview on the traffic context data, the encoding process, and the similarity calculation processes, respectively.

3 Experiments and Results

In our experiments, we used the SPRING-MUSTARD (Spring season Multi-cross Urban Signalized Traffic Aggregated Region Dataset) real-world dataset, which contains 74 days of real urban road traffic data from 8 crosses in a city in China[4]. The road network layout is depicted in Figure 4 a. In order to perform experiments and evaluate the system, we simulated the real-world traffic flows in the Simulator for Urban Mobility (SUMO) [13]. The realistic vehicular simulator generates routes, vehicles, and traffic light signals that reproduce the real car flows in the real-world dataset. In order to evaluate the adaptation capabilities, we systematically introduced progressive flow magnitude disruptions over the 74 days of traffic flow data. Such degenerated traffic conditions describe non-recurrent events such as sport events, accidents or adverse weather, for instance. More precisely, accidents and adverse weather typically determine a decrease in the velocity which might create jams, whereas, special activities such as football matches or beginning/end of holidays increase the flow magnitude. Using the

[3] Codebase at: https://github.com/omlstreaming/aaltd2021

[4] The SPRING-MUSTARD real-world dataset used in our experiments is available at: http://doi.org/10.5281/zenodo.5025264



tls_id	phase_idx	green_time	car_count
788876	2	15	11
788876	3	12	0
763080	2	20	6
763080	3	15	4
761368	2	11	5
...	...	...	...
761368	3	12	6
761368	4	15	4
761419	2	18	22
761419	3	12	13
761419	4		10

tls_id	phase_idx	D-dimensional vector encoding
761419:	2:	-0.17923, -0.44324,….,0.03454
	3:	-0.67923, -0.59324,….,0.12454

tls_id	phase_idx	D-dimensional vector encoding
761419:	4:	-0.77943, -0.75746,….,0.565254

Fig. 3. TRAMESINO System Encoding and Similarity Mechanisms.

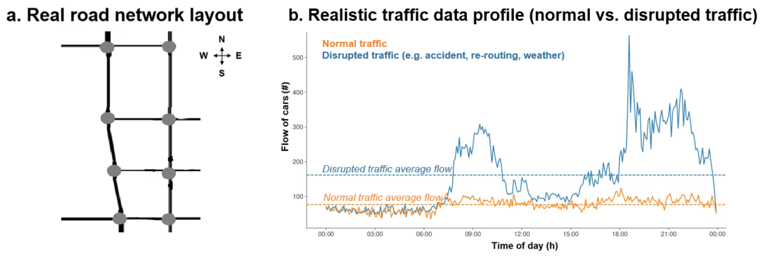


Fig. 4. Real-world Road Network Layout and Normal vs. Disrupted Data.

real-world flow in the dataset and SUMO, we reproduce the traffic flow behavior when disruption occurs starting from normal traffic flow data by reflecting the disruption effect on vehicles speed and/or network capacity and demand. We sweep the disruption magnitude from normal traffic up to 3 levels of disruption (i.e. low, medium, high) reflected over all the 8 crosses over 24 hours. The evaluated systems are the following:

- BASELINE: static traffic planning that uses pre-stored timing plans computed offline using historic data.
- MILP: Mixed-Integer Linear Programming traffic optimization implementation inspired from [18].
- HOPFIELD: Hopfield neural network implementation inspired from [16].
- TRAMESINO: instantiation of our system per each traffic light installed in each direction of each of the 8 crosses in the road network.

For the evaluation of the different approaches (i.e. BASELINE, MILP, HOPFIELD, and TRAMESINO), we followed the next procedure:

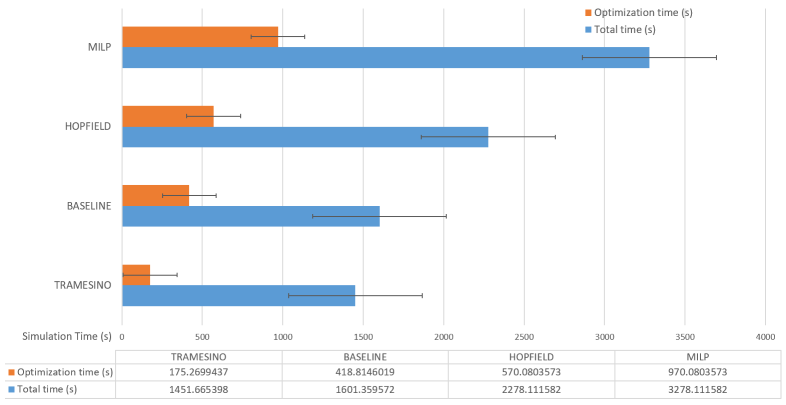
- Simulate real-world SPRING-MUSTARD flows in SUMO and store the results (without disruptions and with the 3 levels of progressive disruptions) for each of the five approaches.
- Compute relevant traffic aggregation metrics (i.e. average trip duration, average speed, and waiting time, respectively).
- Rank experiments depending on performance.
- Perform statistical tests (i.e. a combination of omnibus ANOVA and posthoc pairwise T-test with a significance $p = 0.05$) and adjust ranking depending on significance.
- Evaluate best algorithms depending on ranking for subsets of relevant metrics (i.e. the metrics with significant difference).

Our evaluation results are given in Table 1 where each of the approaches is ranked across the disruption magnitude scale (no disruption (N) to max disruption (H)) over the specific metrics (i.e. average trip duration, average speed, and waiting time, respectively). For flow magnitude disruptions, the level of disruption (i.e. low (L), medium (M), and high (H)) is a factor used to adjust the number of vehicles during the disruption. As one can see in Table 1, TRAMESINO over-

System/ Disruption level	N	L	M	H	Ranking	Deviation
Average trip duration(s)						
BASELINE	168.805	181.217	265.546	270.167	4	49.86%
MILP	118.336	132.406	167.173	167.673	1	0.0%
HOPFIELD	151.281	151.381	223.017	257.464	3	32.28%
TRAMESINO	156.379	157.371	203.775	236.224	2	28.44%
Average speed(km/h)						
BASELINE	58.15	56.78	49.38	47.50	4	10.95%
MILP	59.30	60.00	59.40	59.10	1	0.0%
HOPFIELD	59.48	59.97	49.28	46.18	3	9.84%
TRAMESINO	59.78	59.02	52.08	48.28	2	8.14%
Waiting time(s)						
BASELINE	16.45	18.53	32.59	35.13	4	7.02%
MILP	13.98	16.14	15.14	15.07	1	0.0%
HOPFIELD	13.98	14.96	29.32	37.29	3	5.84%
TRAMESINO	14.95	14.57	22.16	29.01	2	2.96%

Table 1. Performance evaluation of the different systems in normal traffic (N) and with varying disruption levels: low(L), medium(M), and high(H). Besides absolute ranking we take the average performance deviation from the optimal solution of MILP.

comes both HOPFIELD and BASELINE, but deviated from the optimal MILP solution with under 30% in average trip duration and waiting time, and just under 3% in average speed. This is due to the optimal solution that MILP finds given the constraints that the values of the traffic quantities rely upon. However, this performance decreases in the typical metrics is successfully compensated by the run-time analysis in Figure 5. Here, when simulating one day of traffic, TRAMESINO demonstrates that storing and recalling traffic context memories is almost 2× faster than BASELINE and up to 5× faster than MILP, when considering the actual optimization time (i.e. for TRAMESINO store and recall based on similarity, constrained optimization convergence for MILP). Additionally, the overall TRAMESINO processing only took around 12% from the total simulation time of a single day (i.e. approx. 24 minutes). A specific anal-



	TRAMESINO	BASELINE	HOPFIELD	MILP
🟧 Optimization time (s)	175.2699437	418.8146019	570.0803573	970.0803573
🟦 Total time (s)	1451.665398	1601.359572	2278.111582	3278.111582

Fig. 5. Run-time performance evaluation for the real-world flows in the simulator.

ysis and evaluation for TRAMESINO is the accuracy and robustness of the high-dimensional encoding. We explored how does the size of the encoding D influence the encoding and decoding of each memory (i.e. in the storing and recall processes of TRAMESINO). Intuitively, a higher dimension of the encoding will support more accurate representations. This is visible in Figure 6, where we stored and recalled a varying number of traffic data memories of different dimensions. Please recall that this process describes the entire functionality pipeline of TRAMESINO described in Figure 1.

4 DISCUSSION

Traffic optimization and control is a complex multi-factorial problem. Such a problem requires accurate models, robust control, and, above all, efficient computation, to meet real-world constraints. But, there is a trade-off to be made in order to accommodate all these objectives. Combining human expertise, simple models, and heuristics, the typical static plans (i.e. BASELINE) are the

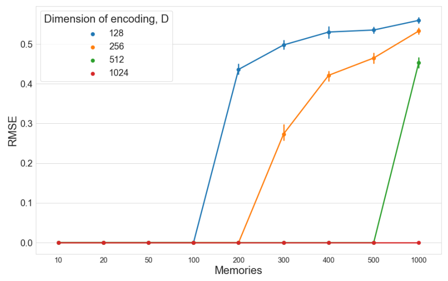


Fig. 6. Encoding/decoding accuracy of TRAMESINO memories.

best choice when there is a predictable traffic demand and no dynamic changes in the flow (i.e. accounts for a look-up-table query). Such models fail to capture and accommodate sudden changes in the traffic context and are typically used as fall-back mechanisms. Increasing the price of modelling and computation with mathematical programming and constrained optimization, adaptive systems (i.e. MILP[18]) are the choice for accurate responses to abrupt changes in traffic dynamics. As computing new traffic light plans is required very often (e.g. every 5 mins), optimization-based systems reach their limitation at scale, when controlling large urban networks. Constrained optimization might provide the optimal solution but miss the timing. Trying to balance accuracy and computation efficiency, while exploiting the regularity in traffic patterns for robust control, optimization-free methods [16] were developed. Such class of methods, of which TRAMESINO is a member, try to exploit temporal regularities in the traffic data to store relevant patterns of action-consequence (i.e. green time / flow of cars) to be able to bypass expensive optimization.

Looking at the evaluation in Table 1 we see that the accuracy trade-off is visible, optimization-free methods (i.e. HOPFIELD and TRAMESINO) ranking worse than MILP in the traffic specific performance metrics. This is due to the fact that MILP's constrained optimization focuses on satisfying all dependencies among traffic data quantities (i.e. traffic flow, green time, phase offset) in order to provide green time values that minimize trip duration, maximize speed, and reduces waiting time, respectively. The power of such an approach is visible also when progressive disruptions are introduced over the daily traffic patterns. TRAMESINO outperforms the HOPFIELD model due to its efficient computation using NEF and spiking neural networks. This allows for an efficient high-dimensional data representation, simple algebraic operations, and memory dynamics, that can exploit traffic data regularities. These regularities captured by TRAMESINO's memory yield fast adaptation to sudden changes in the traffic flow patterns. As shown in Table 1 despite the increasing disruption levels TRAMESINO is still providing the second best speed, average trip duration, and waiting time. Finally, due to its efficient computation TRAMESINO dominates in terms of run-time (see Figure 5). Thanks to its learning and adaptation capabilities, TRAMESINO captures traffic regularities when storing new context memories and overcomes the judiciously-parametrized static plan of the BASELINE system. This offers a serious gain in execution time, avoiding relaxation of HOPFIELD and the optimal convergence of MILP.

5 CONCLUSIONS

In order to exploit regularities in road traffic patterns and avoid expensive optimization techniques, TRAMESINO stands out as a good candidate for efficient traffic control. The system exploits the causal relation among action - consequences pairs (i.e. traffic light green time - flow of cars) in time in order to store relevant contexts. Such traffic context memories are subsequently recalled in new situations, but similar, traffic situations bypassing a new traffic plan re-computation. Our experiments on real-world data demonstrate that such an approach provides a good trade-off between accuracy and robustness overcoming the static plans heuristics and the expensive optimization through a superior gain in run-time. This behavior benefits from the rather deterministic daily traffic profile but, as our experiments demonstrate, can also accommodate sudden disruptions of increasing magnitudes.

References

1. Bracewell, R.N., Bracewell, R.N.: The Fourier transform and its applications, vol. 31999. McGraw-Hill New York (1986)
2. Day, C.M., Bullock, D.M.: Optimization of traffic signal offsets with high resolution event data. Journal of Transportation Engineering, Part A: Systems **146**(3), 04019076 (2020)
3. Dhamija, S., Gon, A., Varakantham, P., Yeoh, W.: Online traffic signal control through sample-based constrained optimization. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 30, pp. 366–374 (2020)
4. Eliasmith, C., Anderson, C.H.: Neural engineering: Computation, representation, and dynamics in neurobiological systems. MIT press (2003)
5. Gayler, R.W.: Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. arXiv preprint cs/0412059 (2004)
6. Henry, J.J., Farges, J.L., Tuffal, J.: The prodyn real time traffic algorithm. In: Control in Transportation Systems, pp. 305–310. Elsevier (1984)
7. Hoogendoorn, S.P., Bovy, P.H.: Generic gas-kinetic traffic systems modeling with applications to vehicular traffic flow. Transportation Research Part B: Methodological **35**(4), 317–336 (2001)
8. Hopfield, J.J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the national academy of sciences **81**(10), 3088–3092 (1984)
9. Hu, H., Liu, H.X.: Arterial offset optimization using archived high-resolution traffic signal data. Transportation Research Part C: Emerging Technologies **37**, 131–144 (2013)
10. Hunt, P., Robertson, D., Bretherton, R., Royle, M.C.: The scoot on-line traffic signal optimisation technique. Traffic Engineering & Control **23**(4) (1982)

11. Köhler, E., Strehler, M.: Traffic signal optimization: Combining static and dynamic models. Transportation science **53**(1), 21–41 (2019)
12. Lendaris, G.G., Mathia, K., Saeks, R.: Linear hopfield networks and constrained optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **29**(1), 114–118 (1999)
13. Lopez, P.A., et al., M.B.: Microscopic traffic simulation using sumo. In: The 21st IEEE International Conference on Intelligent Transportation Systems. IEEE (2018), `https://elib.dlr.de/124092/`
14. Lowrie, P.: Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. Roads and Traffic Authority NSW, Traffic Control Section (1990)
15. Mirus, F., Blouw, P., Stewart, T.C., Conradt, J.: An investigation of vehicle behavior prediction using a vector power representation to encode spatial positions of multiple objects and neural networks. Frontiers in neurorobotics **13**, 84 (2019)
16. Nishikawa, I., Iritani, T., Sakakibara, K.: Improvements of the traffic signal control by complex-valued hopfield networks. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings. pp. 459–464. IEEE (2006)
17. Nishikawa, I., Kuroe, Y.: Dynamics of complex-valued neural networks and its relation to a phase oscillator system. In: International Conference on Neural Information Processing. pp. 122–129. Springer (2004)
18. Ouyang, Y., Zhang, R.Y., Lavaei, J., Varaiya, P.: Large-scale traffic signal offset optimization. IEEE Transactions on Control of Network Systems **7**(3), 1176–1187 (2020)
19. Plate, T.A.: Holographic Reduced Representation: Distributed representation for cognitive structures. CSLI Lecture Notes (2003)
20. Punzo, V., Simonelli, F.: Analysis and comparison of microscopic traffic flow models with real traffic microscopic data. Transportation Research Record **1934**(1), 53–63 (2005)
21. Sánchez, C.S., Wieder, A., Sottovia, P., Bortoli, S., Baumbach, J., Axenie, C.: Gannster: Graph-augmented neural network spatio-temporal reasoner for traffic forecasting. In: International Workshop on Advanced Analytics and Learning on Temporal Data. pp. 63–76. Springer (2020)
22. Sun, J., Liu, H.X.: Stochastic eco-routing in a signalized traffic network. Transportation Research Procedia **7**, 110–128 (2015)
23. Treiber, M., Kesting, A.: Traffic flow dynamics. Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg (2013)
24. Treiber, M., Kesting, A., Helbing, D.: Understanding widely scattered traffic flows, the capacity drop, and platoons as effects of variance-driven time gaps. Physical review E **74**(1), 016123 (2006)
25. Voelker, A.R., Crawford, E., Eliasmith, C.: Learning large-scale heteroassociative memories in spiking neurons. Unconventional Computation and Natural Computation **7**, 2014 (2014)
26. van Wageningen-Kessels, F., Van Lint, H., Vuik, K., Hoogendoorn, S.: Genealogy of traffic flow models. EURO Journal on Transportation and Logistics **4**(4), 445–473 (2015)
27. Walsh, M.P., Flynn, M.E., O'Malley, M.J.: Augmented hopfield network for mixed-integer programming. IEEE transactions on neural networks **10**(2), 456–458 (1999)