

Online Structure Learning for Traffic Management

✉Evangelos Michelioudakis¹, Alexander Artikis^{2,1}, and Georgios Paliouras¹

Institute of Informatics and Telecommunications, NCSR “Demokritos”¹
Department of Maritime Studies, University of Piraeus²
{vagmcs, a.artikis, paliourg}@iit.demokritos.gr

Abstract. Most event recognition approaches in sensor environments are based on manually constructed patterns for detecting events, and lack the ability to learn relational structures in the presence of uncertainty. We describe the application of $\text{OSL}\alpha$, an online structure learner for Markov Logic Networks that exploits Event Calculus axiomatizations, to event recognition for traffic management. Our empirical evaluation is based on large volumes of real sensor data, as well as synthetic data generated by a professional traffic micro-simulator. The experimental results demonstrate that $\text{OSL}\alpha$ can effectively learn traffic congestion definitions and, in some cases, outperform rules constructed by human experts.

Keywords: Markov Logic Networks, Event Calculus, Uncertainty

1 Introduction

Many real-world applications are characterized by both uncertainty and relational structure. Regularities in these domains are hard to identify manually, and thus automatically learning them from data is desirable. One framework that concerns the induction of probabilistic knowledge by combining the powers of logic and probability is Markov Logic Networks (MLNs) [12]. Structure learning approaches that focus on MLNs have been successfully applied to a variety of applications where uncertainty holds. However, most of these approaches are batch and cannot handle large training sets, due to their requirement to load all data in memory for inference in each learning iteration.

Recently, we proposed the $\text{OSL}\alpha$ [9] online structure learner for MLNs, which extends OSL [7] by exploiting a given background knowledge to effectively constrain the space of possible structures during learning. The space is constrained subject to the characteristics imposed by the rules governing a specific task, herein stated as axioms. As a background knowledge we are employing MLN-EC [14], a probabilistic variant of the Event Calculus [8, 10] for event recognition.

In event recognition [3, 1] the goal is to recognize *composite events* (CE) of interest given an input stream of *simple derived events* (SDEs). CEs can be defined as relational structures over sub-events, either CEs or SDEs, and capture the knowledge of a target application. Due to the dynamic nature of real-world applications, the CE definitions may need to be refined over time or the current

knowledge may need to be enhanced with new definitions. Manual curation of event definitions is a tedious and cumbersome process, and thus machine learning techniques to automatically derive the definitions are essential.

We applied $\text{OSL}\alpha$ to learning definitions for traffic incidents using over 3GiB of real data from sensors mounted on a 12km stretch of the Grenoble ring road, provided in the context of the SPEEDD project¹. The goal of SPEEDD is to develop a system for proactive, event-based decision-making, where decisions are triggered by forecast events. To allow for event recognition and forecasting, $\text{OSL}\alpha$ is employed to construct and refine the necessary CE definitions. Due to the high volume of the dataset, the learning process must employ an online strategy. To evaluate further the predictive accuracy of $\text{OSL}\alpha$, we employed a synthetic dataset generated by a professional traffic micro-simulator [13], developed by domain experts to allow for the systematic testing of the SPEEDD components.

The remainder of the paper is organized as follows. Section 2 presents $\text{OSL}\alpha$, while Section 3 describes the application of $\text{OSL}\alpha$ to the traffic domain. Section 4 summarizes the presented work and outlines further research directions.

2 $\text{OSL}\alpha$: An Online Structure Learner using Background Knowledge Axiomatization

$\text{OSL}\alpha$ extends OSL by exploiting a given background knowledge. Figure 1 presents the components of $\text{OSL}\alpha$. The background knowledge consists of the MLN-EC axioms (i.e., domain-independent rules) and an already known (possibly empty) hypothesis (i.e., set of clauses). Each axiom contains *query predicates* $\text{HoldsAt} \in \mathcal{Q}$ that consist the supervision, and *template predicates* InitiatedAt , $\text{TerminatedAt} \in \mathcal{P}$ that specify the conditions under which a CE starts and stops being recognized. The latter form the target CE definitions that we want to learn. $\text{OSL}\alpha$ exploits these axioms in order to create mappings of supervision predicates into template predicates and search only for explanations of these template predicates. Upon doing so, $\text{OSL}\alpha$ does not need to search over time sequences; instead it only needs to find appropriate bodies over the current time-point for the following definite clauses:

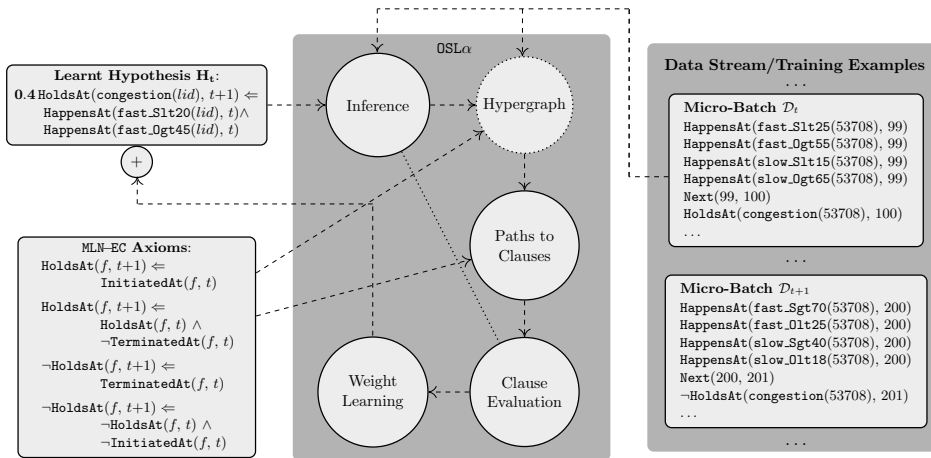
$$\begin{aligned} \text{InitiatedAt}(f, t) &\Leftarrow \text{body} \\ \text{TerminatedAt}(f, t) &\Leftarrow \text{body} \end{aligned}$$

Given the MLN-EC axioms, $\text{OSL}\alpha$ constructs a set \mathbf{T} that provides mappings of its axioms to the template predicates in \mathcal{P} that appear in their bodies. For instance, axiom (1) of \mathbf{T}

$$\text{HoldsAt}(f, t+1) \Leftarrow \text{InitiatedAt}(f, t) \wedge \text{Next}(t, t+1) \quad (1)$$

will be mapped to the template predicate $\text{InitiatedAt}(f, t)$ since the aim is to construct a rule for this predicate. The set \mathbf{T} is used during the search for structures (relational pathfinding) to find an initial search set \mathcal{I} of ground template

¹ <https://speedd-project.eu>

Fig. 1: The procedure of OSL α .

predicates, and search the space of possible structures for specific bodies of the definite clauses.

At any step t of the online procedure, a training example (micro-batch) \mathcal{D}_t arrives containing simple derived events (SDEs), e.g. a fast lane in a highway has average speed less than 25 km/hour and sensor occupancy greater than 55%. \mathcal{D}_t is used together with the already learnt hypothesis to predict the truth values \mathbf{y}_t^P of the composite events (CEs) of interest. This is achieved by (maximum a posteriori) MAP inference based on LP-relaxed Integer Linear Programming [6]. Then OSL α receives the true label \mathbf{y}_t and finds all ground atoms that are in \mathbf{y}_t but not in \mathbf{y}_t^P , denoted as $\Delta\mathbf{y}_t = \mathbf{y}_t \setminus \mathbf{y}_t^P$. Hence, $\Delta\mathbf{y}_t$ contains the false positives/negatives of the inference step. Given \mathcal{D}_t , OSL α constructs a hypergraph that represents the space of possible structures as graph paths. Constants appear in the graph as nodes and true ground atoms as hyperedges that connect the nodes appearing as its arguments.

Then for all incorrectly predicted CEs in $\Delta\mathbf{y}_t$, OSL α uses the set \mathbf{T} to find the corresponding ground template predicates for which the axioms belonging in \mathbf{T} are satisfied by the current training example. Consider, for instance, that one of these is axiom (1), and that we have predicted that the ground atom $\text{HoldsAt}(\text{CE}, 5)$ is false (false negative). OSL α substitutes the constants of $\text{HoldsAt}(\text{CE}, 5)$ into axiom (1). The result of the substitution will be the following partially ground axiom:

$$\text{HoldsAt}(\text{CE}, 5) \Leftarrow \text{Next}(t, 5) \wedge \text{InitiatedAt}(\text{CE}, t) \quad (2)$$

Since t represents time-points and Next describes successive time-points, there will be only one true grounding of $\text{Next}(t, 5)$ in the training data, having as argument the constant 4. OSL α substitutes the constant 4 into axiom (2) and adds $\text{InitiatedAt}(\text{CE}, 4)$ to the initial search set \mathcal{I} . This procedure essentially

reduces the hypergraph to contain only ground atoms explaining the template predicates. The pruning resulting from the template guided search is essential for learning in temporal domains.

For all ground template predicate in \mathcal{I} , the hypergraph is searched, guided by path mode declarations [7] using relational pathfinding [11] up to a predefined length, for definite clauses explaining the CEs. The search procedure recursively adds to the path hyperedges (i.e., ground atoms) that satisfy the mode declarations. The search ends when the path reaches the specified length or when no new hyperedges can be added.

The paths discovered during the search correspond to conjunctions of true ground atoms connected by their arguments and can be generalized into definite clauses by replacing constants in the conjunction with variables. Then, these conjunctions are used as a body to form definite clauses using as head the template predicate present in each path. The resulting set of formulas is converted into clausal normal form and evaluated.

Evaluation takes place for each clause c individually. The difference between the number of true groundings of c in the ground-truth world $(\mathbf{x}_t, \mathbf{y}_t)$ and those in predicted world $(\mathbf{x}_t, \mathbf{y}_t^P)$ is then computed (note that \mathbf{y}_t^P was predicted without c). Only clauses whose difference in the number of groundings is greater than or equal to a predefined threshold μ will be added to the MLN:

$$\Delta n_c = n_c(\mathbf{x}_t, \mathbf{y}_t) - n_c(\mathbf{x}_t, \mathbf{y}_t^P) \geq \mu \quad (3)$$

The intuition behind this measure is to add to the hypothesis clauses whose coverage of the ground-truth world is significantly (according to μ) greater than that of the clauses already learned. Finally, the weights of the retained clauses are then optimized by the AdaGrad online learner [5], the weighted clauses are appended to the current hypothesis \mathcal{H}_t , and the procedure is repeated for the next training example \mathcal{D}_{t+1} .

Our implementations of OSL α , AdaGrad, and MAP inference based on LP-relaxed Integer Linear Programming, are contributed to LoMRF², an open-source implementation of MLNs written in Scala. LoMRF enables knowledge base compilation, parallel and optimized grounding, inference and learning.

3 Empirical Evaluation

We applied OSL α to traffic management using real data from magnetic sensors mounted on the southern part of the Grenoble ring road (Rocade Sud), that links the city of Grenoble from the south-west to the north-east. In addition to sustaining local traffic, this road has a major role, since it connects two highways: the A480, which goes from Paris and Lyon to Marseilles, and the A41, which goes from Grenoble to Switzerland. Furthermore, the mountains surrounding Grenoble prevent the development of new roads, and also have a negative impact on pollution dispersion, making the problem of traffic regulation on this road even

² <https://github.com/anskar1/LoMRF>

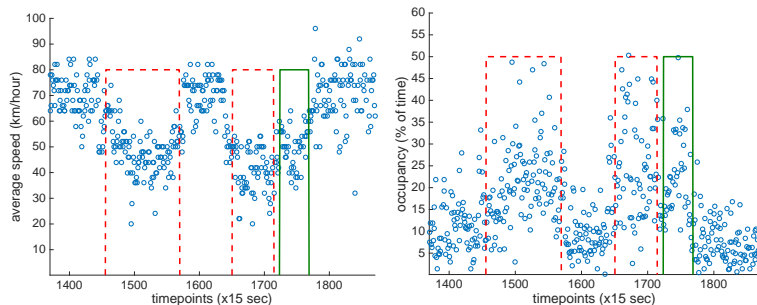


Fig. 2: Location 353708, fast lane: average speed (left) and occupancy (right). The blue points indicate the average speed (occupancy), the green windows indicate the congestion annotated by human experts, and red (dashed) windows the potentially missing annotations.

more crucial [4]. The dataset was made available by CNRS-Grenoble, our partner in the SPEEDD project, and consists of approximately 3.3GiB of sensor readings (one month data). Sensors are placed in 19 collection points along a 12km stretch of the highway. Each collection point has a sensor per lane. Sensor data are collected every 15 seconds, recording the total number of vehicles passing through a lane, average speed and sensor occupancy. Annotations of traffic congestion are provided by human traffic controllers, but only very sparsely.

To deal with this issue, and test further $OSL\alpha$, we used a synthetic dataset generated by a professional traffic micro-simulator [13], developed in the context of SPEEDD. The simulator is based on AIMSUN³ — a widely used transport modeling software that uses a microscopic model simulating individual vehicle movement, based on statistical laws from car-following and lane-changing theories. Typically, vehicles enter a transportation network using a statistical distribution of arrivals. The microscopic model incorporates sub-models for acceleration, speed adaptation, lane-changing, etc, to describe how vehicles move, interact with each other and the infrastructure. The synthetic dataset concerns the same location — the Rocade Sud — and consists of 6 simulations of one hour each (≈ 18.6 MiB). The simulator has been calibrated using real traffic data. Unlike the real dataset, artificial sensors exist in 98 collection points of the highway, there is no distinction between (fast, queue, etc) lanes, and sensor measurements additionally include vehicle density. Furthermore, the synthetic dataset is much better annotated than the real dataset.

3.1 Learning Challenges

Both datasets used for learning traffic congestions exhibit several challenges. Concerning the real dataset, the first challenge is its size, making the use of batch learners, as well as online learners such as OSL that cannot make use

³ <http://www.aimsun.com/>

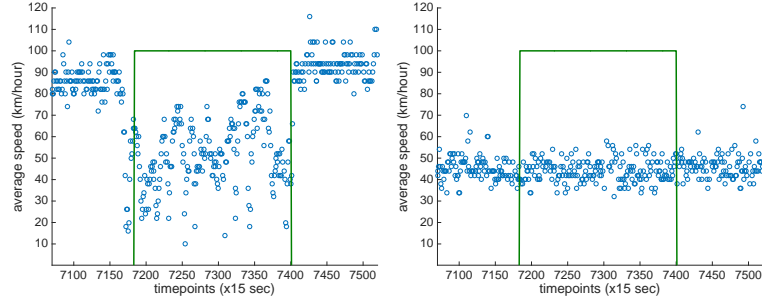


Fig. 3: Location 347549: fast lane (left) vs queue lane (right). The blue points indicate the average speed while green windows indicate the congestion annotated by human experts.

of background knowledge, prohibitive. For instance, in the empirical evaluation presented in [9], OSL was tested on a much smaller training set ($\approx 2.6\text{MiB}$) and required ≈ 25 hours to process just 40% of the data. Second, as mentioned above, traffic congestion annotation is largely incomplete, leading to the incorrect penalization of good rules. This issue is illustrated in Figure 2.

Third, the quality of information of each sensor differs considerably. This issue is illustrated in Figure 3, that displays the average speed of the fast lane and the queue lane at the same location, as well as the congestion annotation. Figure 3 shows that the information provided by the sensors of the queue lane is largely uninformative.

Fourth, generic, location- and lane-agnostic rules are not sufficient. Consider, for example, a simple rule defining traffic congestion for any possible location regardless of the lane type:

$$\begin{aligned} \text{InitiatedAt}(\text{congestion}(lid), t) \Leftarrow \\ \text{HappensAt}(\text{aggr}(lid, \text{occupancy}, \text{avgspd}), t) \wedge \\ \text{avgspd} < 50 \wedge \text{occupancy} > 25 \end{aligned}$$

According to the above rule, a congestion in some location is said to be initiated if the average speed is below 50 km/hour and the occupancy is greater than 25%. Similar rules, not shown here to save space, terminate the recognition of congestion. The optimization of the weights of these rules had large fluctuations along the learning steps, leading to zero crossings, indicating that the rules correctly capture the concept of traffic congestion in a few locations, and completely fail in others. To deal with this issue, location- and lane-specific rules must be constructed.

On the other hand, the synthetic dataset introduces different types of challenge. Density and occupancy measurements are mostly noisy, while there are a lot of zero values in all sensor measurements, leading to detection errors. These issues are illustrated in Figure 4. Furthermore, although supervision is more complete compared to the real dataset, there are cases of missing annotation.

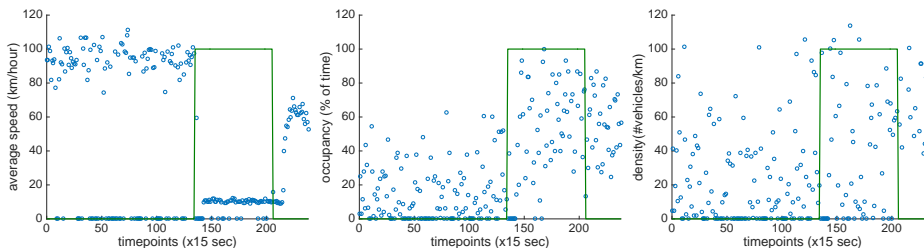


Fig. 4: Simulation 2 at location 1320: average speed (left), occupancy (middle) and density (right). The green windows indicate the congestion supervision.

3.2 Experimental Setup and Results

Sensor readings constitute the simple derived events (SDEs), while traffic congestion is the target CE. The data are stored in a PostgreSQL database and the training sequence for each micro-batch, as shown in Figure 1, is constructed dynamically by querying the database. A set of first-order logic functions is used to discretize the numerical data (speed, occupancy) and produce input events such as, for instance, `HappensAt(fast_S1t55(53708), 100)`, representing that the speed in the fast lane of location 53708 is less than 55 km/hour at time 100. The CE supervision indicates when a traffic congestion holds in a specific location. Each training sequence is composed of input SDEs (`HappensAt`) over the first-order logic functions and the corresponding CE annotations (`HoldsAt`). The total length of the training sequence in the real data case consists of 172,799 time-points, and we consider only SDEs from fast lanes. In the synthetic data the total training sequence length consists of 238 time-points and there is no distinction between lanes.

In the experiments presented below, we compare:

- $OSL\alpha$ starting with an empty hypothesis.
- $OSL\alpha$ starting with manually constructed traffic congestion definitions developed in collaboration with domain experts.
- The AdaGrad [5] online weight learner operating on the aforementioned hand-crafted definitions.

The evaluation results were obtained using MAP inference [6] and are presented in terms of F_1 score. In the real dataset, all reported statistics are micro-averaged over the instances of recognized CEs using 10-fold cross validation over the entire dataset, using varying batch sizes. At each fold, an interval of 17,280 time-points was left out and used for testing. In the synthetic data, the reported statistics are micro-averaged using 6-fold cross validation over 6 simulations by leaving one out for testing, using varying batch sizes. The experiments were performed on a computer with an Intel i7 4790@3.6GHz processor (4 cores and 8 threads) and 16GiB of RAM, running Ubuntu 16.04.

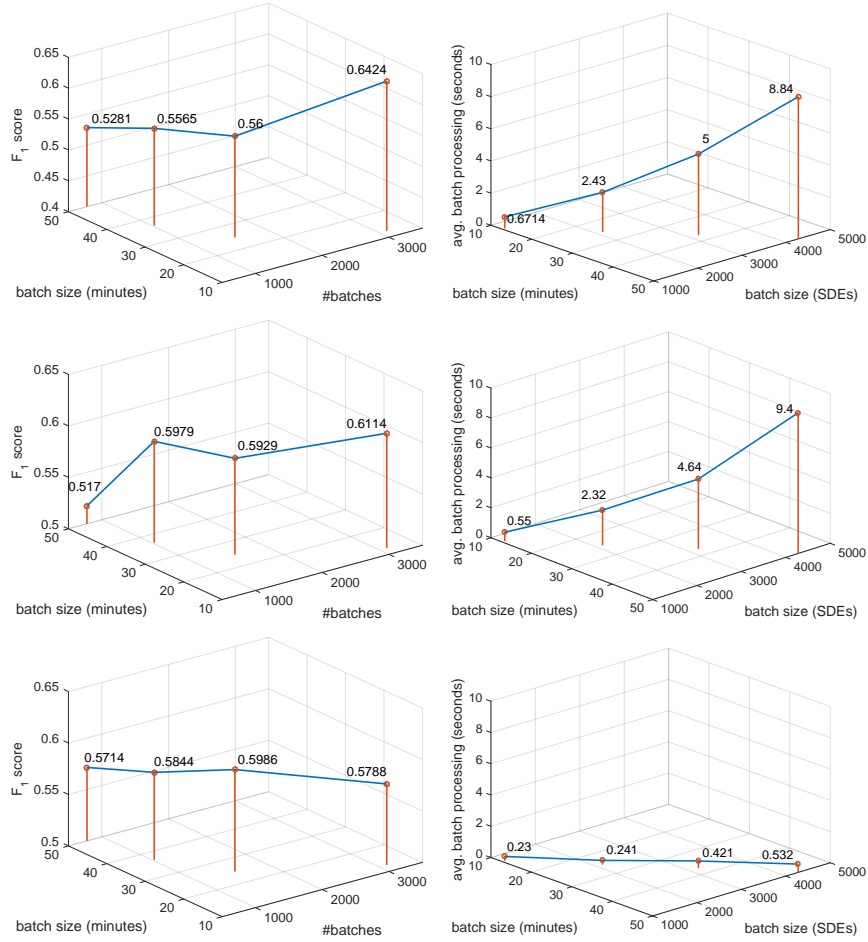


Fig. 5: Real dataset: F_1 score (left) and average batch processing time (right) for $\text{OSL}\alpha$ starting with an empty hypothesis (top), $\text{OSL}\alpha$ starting with manually constructed rules (middle) and AdaGrad operating on the manually constructed rules (bottom). In the left figures, the number of batches (see the Y axes) refers to number of learning iterations.

Real Dataset. Figure 5 presents the experimental results on the real dataset. AdaGrad and $\text{OSL}\alpha$ (when a starting with a non-empty hypothesis) were given location-specific rules defining traffic congestion in terms of speed and occupancy. The predictive accuracy of the learned models, both for $\text{OSL}\alpha$ and AdaGrad, is low. This arises mainly from the largely incomplete supervision. In $\text{OSL}\alpha$, the predictive accuracy increases (almost) monotonically as the learning iterations increase. On the contrary, the accuracy of AdaGrad is more or less constant. $\text{OSL}\alpha$, starting with or without the manually constructed rules, outperforms AdaGrad

in terms of accuracy. (OSL α starting without (respectively with) the hand-crafted rules achieves a 0.64 (resp. 0.61) F_1 score, while the best score of AdaGrad is 0.59.) This is a notable result. The aid of human knowledge can help OSL α — see the two middle points (batch size/learning iterations) in the two top left diagrams of Figure 5. However, OSL α achieves the best score when starting with an empty hypothesis. The absence of proper supervision penalizes the hand-crafted rules, compromising the accuracy of the learning techniques that use them. OSL α starting with an empty hypothesis is not penalized in this way, and is able to construct rules with a better fit in the data, given enough learning iterations. For some locations of the motorway, OSL α has constructed rules with different thresholds for speed and occupancy than those of the hand-crafted rules.

With respect to efficiency (see the right diagrams of Figure 5), unsurprisingly AdaGrad is faster and scales better to the increase in the batch size. At the same time, OSL α processes data batches efficiently — for example, OSL α takes less than 10 seconds to process a 50-minute batch including 4,220 SDEs.

Synthetic Dataset. To test the behavior of OSL α under better supervision, we made use of a synthetic dataset produced by a professional traffic micro-simulator. The dataset concerns the same location: the southern part of the Grenoble ring road. Figure 6 presents the experimental results using only SDEs for average speed. As mentioned in Section 3.1, density and occupancy measurements are mostly noisy in the synthetic data. Consequently, AdaGrad and OSL α (when a starting with a non-empty hypothesis) were given rules defining traffic congestion only in terms of speed. These rules were location-agnostic since the artificial sensors do not distinguish between lanes. Not surprisingly, the predictive accuracy of the learned models in these experiments is much higher as compared to real dataset. Moreover, the accuracy of OSL α and AdaGrad is affected mostly by the batch size: accuracy increases as the batch size increases. The synthetic dataset is smaller than the real dataset and thus, as the batch size decreases, the number of learning iterations is not large enough to improve accuracy. The best performance of OSL α and AdaGrad is almost the same (approximately 0.89). In other words, OSL α starting with an empty hypothesis can match the performance of techniques taking advantage of rules crafted by human experts. This is another notable result.

The right diagrams of Figure 6 report the average batch processing times. These diagrams verify that AdaGrad is more efficient than OSL α , and that OSL α achieves a good performance, processing batches much faster than their duration. For example, OSL α takes less than 3 seconds to process a 25-minute batch.

To evaluate further the behavior of OSL α , we performed additional experiments using the synthetic dataset, this time keeping the noisy occupancy measurements. The aim of the experiments was to test OSL α in a well-annotated setting with noisy SDEs. The evaluation results are shown in Figure 7. The hand-crafted rules defining traffic congestion combined speed with occupancy. Figure 7 shows that the noisy occupancy readings have affected the accuracy of OSL α and AdaGrad. However, OSL α starting with the manually constructed rules is affected much less, outperforming significantly both OSL α starting with

an empty hypothesis and AdaGrad. OSL α has augmented the hand-crafted rules with additional clauses that focus on speed, and reduced the weight values of rules combining speed with occupancy. This way, OSL α was able to minimize the effect of noisy SDEs.

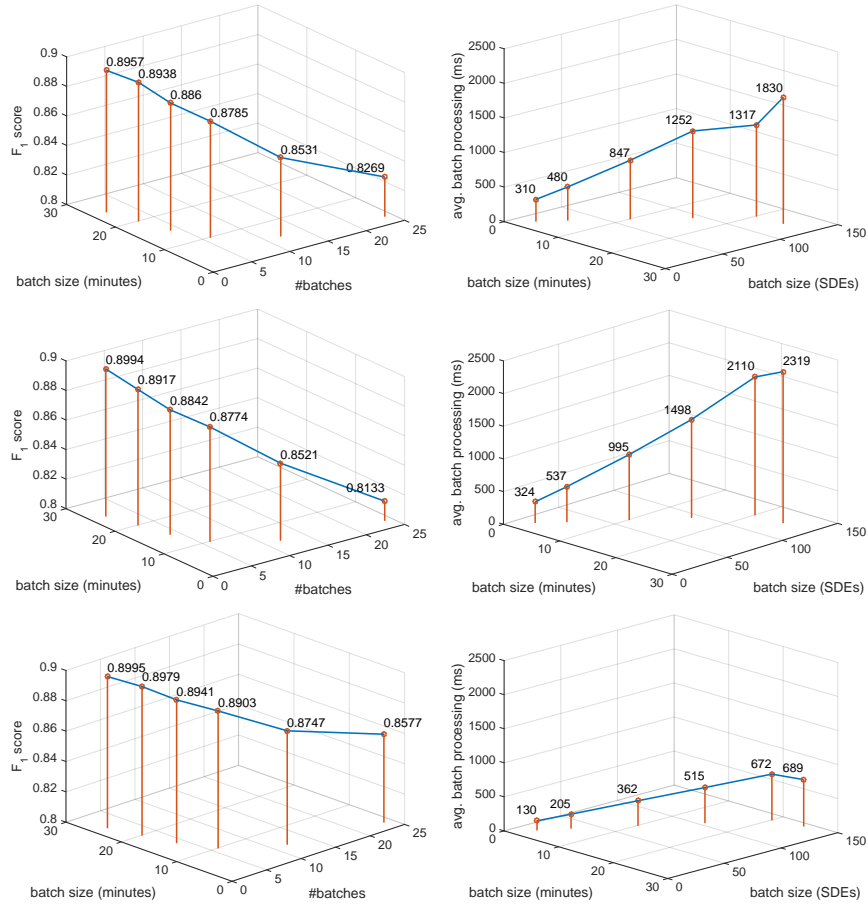


Fig. 6: Synthetic dataset with speed measurements: F_1 score (left) and average batch processing time (right) for OSL α starting with an empty hypothesis (top), OSL α starting with manually constructed rules (middle) and AdaGrad operating on the manually constructed rules (bottom).

For completeness, the right diagrams of Figure 7 report the average batch processing times of OSL α and AdaGrad.

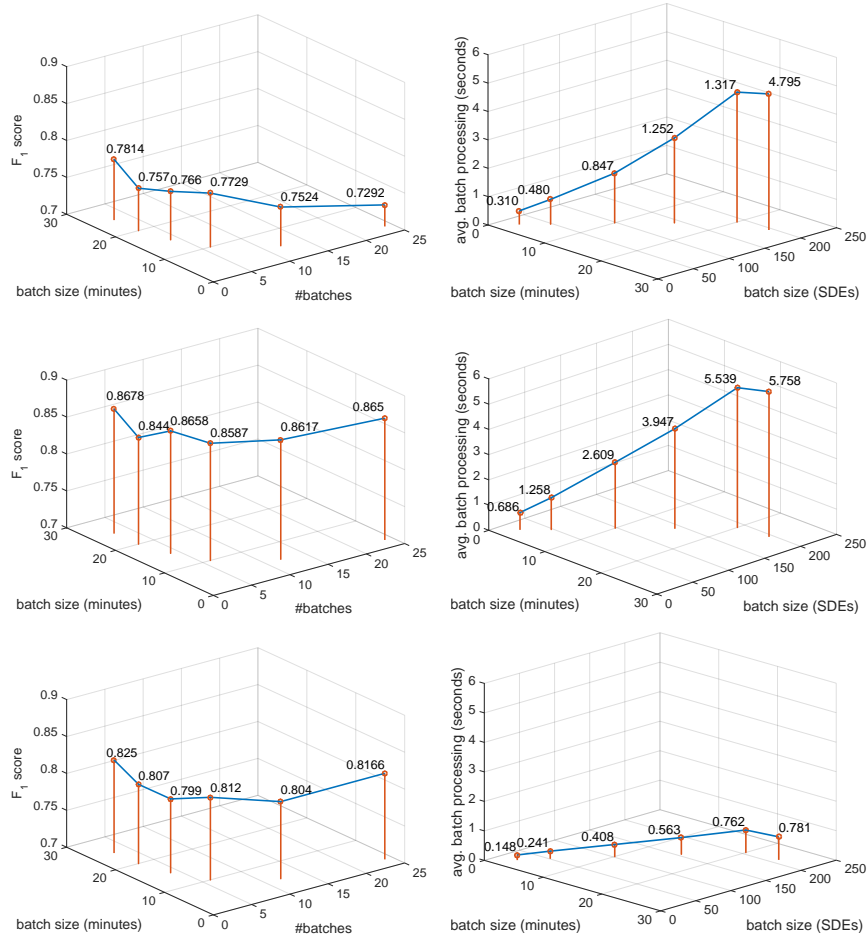


Fig. 7: Synthetic dataset with speed and occupancy measurements: F_1 score (left) and average batch processing time (right) for OSL α starting with an empty hypothesis (top), OSL α starting with manually constructed rules (middle) and AdaGrad operating on the manually constructed rules (bottom).

4 Summary and Further Work

We presented the application of OSL α , a recently proposed structure learner for Markov Logic Networks that exploits background knowledge in the form of Event Calculus theories, to complex event recognition for traffic management. We performed an extensive empirical evaluation using over 3GiB of real data, allowing us to test the scalability of our approach, and a synthetic dataset that enabled us to test systematically the predictive accuracy of the structure learner. The experimental evaluation showed that OSL α , without the aid of hand-crafted knowledge, performs at least as good as the AdaGrad weight learner operating on

rules constructed by human experts. The aid of hand-crafted rules allows OSL α to outperform significantly AdaGrad in the presence of noisy SDEs. With respect to efficiency, OSL α processes data batches much faster than their duration.

There are several directions for further work. We aim to extend OSL α in order to handle effectively the absence of annotation. We are also performing a human factors evaluation involving traffic controllers — see [2] for the initial results.

Acknowledgments. Funded by EU FP7 project SPEEDD (619435).

References

1. A. Artikis, A. Skarlatidis, F. Portet, and G. Paliouras. Logic-based event recognition. *Knowledge Engineering Review*, 27(4):469–506, 2012.
2. C. Baber, S. Starke, N. Morar, A. Howes, A. Kibangou, M. Schmitt, C. Ramesh, J. Lygeros, F. Fournier, and A. Artikis. Deliverable 8.5 – Intermediate evaluation report of SPEEDD prototype for traffic management. SPEEDD Project. http://speedd-project.eu/sites/default/files/D8.5_-_Intermediate_Evaluation_Report-revised.pdf.
3. G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):15, 2012.
4. C. Canudas de Wit, I. Bellicot, F. Garin, P. Grandinetti, A. Ladino, R. Singhal, A. Kibangou, F. Morbidi, M. Schmitt, A. Hempel, C. Baber, and N. Cooke. Deliverable 8.1 – User requirements and scenario definition. SPEEDD project. http://speedd-project.eu/sites/default/files/D8.1_User_Requirements_Traffic_updated_final.pdf.
5. J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, July 2011.
6. T. N. Huynh and R. J. Mooney. Max-Margin Weight Learning for Markov Logic Networks. In *Proceedings of ECML PKDD*, pages 564–579. Springer, 2009.
7. T. N. Huynh and R. J. Mooney. Online Structure Learning for Markov Logic Networks. In *Proceedings of ECML PKDD*, volume 2, pages 81–96, 2011.
8. R. Kowalski and M. Sergot. A Logic-based Calculus of Events. *New Generation Computing*, 4(1):67–95, 1986.
9. E. Michelioudakis, A. Skarlatidis, G. Paliouras, and A. Artikis. Online Structure Learning using Background Knowledge Axiomatization. In *Proceedings of ECML-PKDD*, volume 1, pages 242–237, 2016.
10. E. T. Mueller. Event Calculus. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 671–708. Elsevier, 2008.
11. B. L. Richards and R. J. Mooney. Learning relations by pathfinding. In *Proceedings of AAAI*, pages 50–55. AAAI Press, 1992.
12. M. Richardson and P. M. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
13. R. Singhal, A. Andreev, and A. Kibangou. Deliverable 8.4 – Final Version of Micro-Simulator. SPEEDD project. http://speedd-project.eu/sites/default/files/SPEEDD-D8-4_Final_Version.pdf.
14. A. Skarlatidis, G. Paliouras, A. Artikis, and G. A. Vouros. Probabilistic Event Calculus for Event Recognition. *ACM Transactions on Computational Logic*, 16(2):11:1–11:37, February 2015.