# Flash in Action: Scalable Spatial Data Analysis Using Markov Logic Networks

Ibrahim Sabek
Dept. of Computer Science
University of Minnesota, USA
sabek@cs.umn.edu

Mashaal Musleh
Dept. of Computer Science
University of Minnesota, USA
musle005@cs.umn.edu

Mohamed F. Mokbel*
Qatar Comp. Research Inst.
HBKU, Qatar
mmokbel@hbku.edu.qa

## ABSTRACT

The current explosion in spatial data raises the need for efficient spatial analysis tools to extract useful information from such data. However, existing tools are neither generic nor scalable when dealing with big spatial data. This demo presents *Flash*; a framework for *generic* and *scalable* spatial data analysis, with a special focus on spatial probabilistic graphical modelling (SPGM). *Flash* exploits Markov Logic Networks (MLN) to express SPGM as a set of declarative logical rules. In addition, it provides spatial variations of the scalable RDBMS-based learning and inference techniques of MLN to efficiently perform SPGM predictions. To show *Flash* effectiveness, we demonstrate three applications that use *Flash* in their SPGM: (1) Bird monitoring, (2) Safety analysis, and (3) Land use change tracking.

## 1. INTRODUCTION

Spatial data analysis has grabbed significant attention from both industry and academia (see [20] for a comprehensive survey). The main objective is to extract insights and useful patterns from spatial data (e.g., satellite images [25], medical images [9], geotagged tweets [16]). Spatial data analysis has been employed in many crucial applications in different domains. For example, environmentalists analyze geotagged tweets to predict the people who might need help during disasters [23]. Epidemiologists use spatial analysis techniques to identify cancer clusters [18]. As a result, researchers and practitioners worldwide have released many spatial analysis systems and libraries (e.g., [15, 13, 22]).

Existing spatial analysis solutions suffer from two main issues. First, they can not scale beyond implementing prototypes over small spatial datasets (e.g., see [5, 13]) (*scalability issue*). The scalability challenge is mainly because these solutions were not originally designed for the huge amounts of spatial data being generated at the moment (e.g., there are 10 Million geotagged tweets issued every day [16]). Second, these solutions are specifically tailored for domain-specific applications (e.g., a spatial hidden markov model for animal tracking [22], and a statistical learning approach for crime analysis [15]) (*non-generic issue*). As a result, to use a spatial analysis technique in a new application, a developer would need to re-implement and optimize such technique at the application layer. This is inconvenient for a non-expert application developer who might not be quite familiar with efficient implementations of spatial analysis techniques.

In this paper, we demonstrate *Flash*; a framework for *generic* and *scalable* spatial data analysis. *Flash* achieves orders of magnitudes scalability gain over existing solutions while preserving the same accuracy. For example, *Flash* is at least two orders of magnitude faster than ngspatial [13] when implementing autologistic regression. *Flash* focuses on building a major class of spatial analysis techniques, called *spatial probabilistic graphical modelling* (SPGM), which uses probability distributions and graphical representations (e.g., spatial Bayesian networks [6]) to describe spatial phenomena and make predictions about them [20]. SPGM has many applications including health care [12], risk analysis [2], and environmental science [11].

*Flash* exploits Markov Logic Networks (MLN) [19] (a framework that combines first-order logic rules [10] with probabilistic models) to express SPGM with logical semantics, and allow developers to implement their applications using a set of rules instead of thousands of lines of code. To support *scalability*, *Flash* translates the generated MLN rules of any SPGM application into SQL queries using a grounding technique from [7], and then executes these queries inside scalable database engines (e.g., PostgreSQL). In addition, *Flash* provides spatial variations of the RDBMS-based learning and inference algorithms of MLN [17] to perform scalable SPGM predictions (e.g., predictions over models with millions of nodes). Using *Flash*, a myriad of spatial applications can be built without the need to worry about the underlying SPGM computation.

To show the effectiveness of *Flash*, we built three spatial analysis applications, where *Flash* is used to implement their underlying SPGM: (1) *Bird monitoring*: an application that uses spatial Markov random fields [4] to predict the existence of a specific bird species, namely Barn Swallow, over North America. This application uses Ebird dataset [24] containing more than 360 Million bird observations at 84K location cells. (2) *Safety analysis*: an application that uses spatial hidden Markov models [12] to determine the safety level at different locations based on the reported incidents. As a case study, we assess the safety in Chicago based on

*Also affiliated with University of Minnesota, MN, USA.

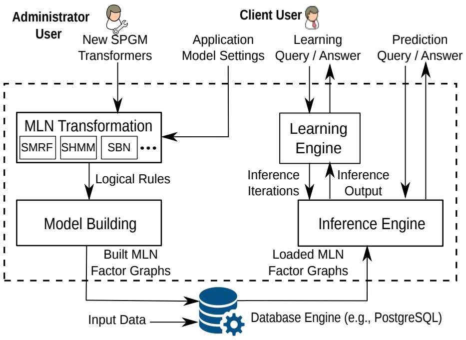


**Figure 1: *Flash* System Overview.**

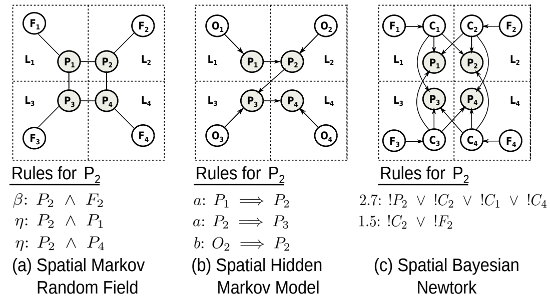


**Figure 2: Logical Rules for SPGM in *Flash*.**

its official crime data repository [8], that has around 7 Million reported incidents. (3) *Land use change tracking*: an application that uses spatial Bayesian networks [5] to analyze where the change in land use is most likely to occur. This application uses a grid dataset containing one Million cells of land cover distribution over Minnesota state, and is compiled from national land cover data repository [26].

## 2. FLASH OVERVIEW

Figure 1 depicts the system architecture of *Flash*. It has two types of users; *administrator* and *client*. An *administrator* should have expertise with both MLN and SPGM to provide user-defined functions for transforming spatial graphical models into a set of first-order logical rules [10]. A *client* can be either application developer or casual user. She can build the SPGM of any application by specifying some settings (e.g., model type, graphical topology) as input. The built model will be stored in a relational database (e.g., PostgreSQL) as a factor graph [27]. A client can also issue learning and prediction queries over the built models. Learning queries can fit the parameters of a specific model to input application data (e.g., hidden Markov model parameters). Prediction queries can answer relevant questions about the model (e.g., what is the probability of a specific event to happen?). As depicted in Figure 1, *Flash* consists of the following four main modules:

**MLN Transformation.** For any SPGM input, this module is responsible on generating an equivalent set of *weighted* rules containing logical predicates (e.g., bitwise-AND, and imply). These weights represent the original SPGM parameters. The generated rules follow the syntax of an efficient Datalog-like logic programming framework, called DDlog [21]. *Flash* chooses DDlog because of its DBMS-friendly schema declaration and rules syntax that can be efficiently processed during the model building module. Currently, *Flash* supports transformation for three spatial graphical models; spatial Markov random fields (SMRF) [4], spatial hidden Markov models (SHMM) [12], and spatial Bayesian networks (SBN) [5] (Details are in Section 3).

**Model Building.** The generated logical rules from the MLN transformation module are considered templates for constructing factor graphs [27]. As a result, *Flash* adapts a scalable factor graph grounding technique from [7] to efficiently translate these rules into SQL queries, and then apply such queries on the input application data to obtain the final output that is equivalent to the SPGM input.

**Inference Engine.** *Flash* evaluates prediction queries using Gibbs sampling-based inference algorithms over factor graphs [17]. However, such algorithms perform sequential sampling over the factor graph nodes which results in slow convergence to the inference answer in case these nodes have spatial dependencies as in SPGM applications [14]. To overcome this limitation, *Flash* employs a variation of Gibbs Sampling that exploits a *concliques*-based traversal pattern [14] to efficiently sample spatially-dependent nodes. A conclique is a set of nodes such that no two nodes in this set are spatially neighbours. The main idea behind defining concliques is ensuring the neighbouring independence between nodes in the same conclique set, and hence these nodes can be sampled in parallel.

**Learning Engine.** *Flash* employs a pseudo-likelihood learning algorithm to learn any unknown weights of the generated MLN rules (i.e., SPGM parameters) from the factor graph. This algorithm repeatedly uses the proposed spatial variation of Gibbs sampling-based inference algorithm in the inference engine to compute the gradient of the SPGM pseudo-likelihood and then determine the weights using an efficient gradient descent optimization technique.

## 3. DEMO APPLICATIONS

In this section, we describe three spatial analysis applications that will be presented during the demo session.

### 3.1 Bird Monitoring Application

This application predicts the existence of a bird species across a certain area. Ornithologists model this problem using autologistic regression [13] as shown in [1], where the area is divided by a two-dimensional grid. Each grid cell holds a binary prediction variable indicating the presence or absence of the bird at this cell, and a set of feature variables that help predicting the value of this prediction variable. Then, the prediction at any cell is determined based on the values of feature variables at this cell along with a set of predicted or observed values at neighbouring cells. As a case study, we use the daily distribution of a certain bird species, namely Barn Swallow, from Ebird dataset [24], which contains more than 360 Million observations collected over North America. We define a grid of 84K cells over North America, and map each observation to one cell. Then, we predict the bird existence at cells with no observations.

**SPGM.** Autologistic regression can be represented as a spatial Markov random field (SMRF) as shown in [4]. Figure 2(a) gives an example of an equivalent SMRF graphical representation to an autologistic regression (with one feature $F$) defined over 4-cells grid, where the neighbourhood of any cell $l$ is assumed to be cells that share edges with $l$ only. In this example, a prediction variable $P_l$ at each cell $l$ has undirected edges with feature $F_l$ at this cell and each neighbouring prediction variable. For example, $P_2$ is connected with feature $F_2$ and neighbours $P_1$ and $P_4$. *Flash* provides an equivalent weighted *bitwise-AND* predicate for each pair of connected variables (theoretical foundation is omitted due to space constraints), where these weights correspond to the regression parameters to be learned. Figure 2(a) shows all logical rules defined over the prediction variable $P_2$ in the example.

## 3.2 Safety Analysis Application

The objective of this application is to infer the safety level (e.g., low, medium and high) at a bunch of neighbouring locations simultaneously based on reported incidents at these locations. As a case study, we use the official Chicago crime dataset [8], which contains around 7 Million reported incidents (i.e., observations) over 500K locations. We predict the safety level for each of these locations.

**SPGM.** This application has been usually represented with spatial hidden Markov models (SHMM) [12] as shown in [3], where the safety level at each location $l$ is considered a hidden state $P_l$ to be predicted and the reported incident at $l$ is an observation $O_l$ that affects the value of $P_l$. SHMM imposes an ordered spatial dependence among neighbouring locations. Figure 2(b) gives an example of the directed graphical representation of SHMM defined over 4-cells grid, where we use z-curve ordering technique to build a sequence that preserves the spatial dependence. *Flash* provides an equivalent weighted *imply* predicate for any state/state or observation/state pair, where these weights correspond to the SHMM parameters. Figure 2(b) shows all logical rules defined over the hidden state $P_2$ in the example.

## 3.3 Land Use Change Tracking Application

The objective of this application is to determine whether there will be a change in the land use or not. For example, the land in a location $l$ could be suitable for agriculture, however, given certain factors (e.g., crowded neighbourhoods), it is expected to be for human use soon. As a case study, we use a grid dataset containing one Million cells of land cover distribution over Minnesota state, and is compiled from national land cover data repository [26].

**SPGM.** The state-of-the-art work in this application uses spatial Bayesian networks (SBN) [5] as shown in [11]. Figure 2(c) gives an example of the directed graphical representation of SBN defined over 4-cells grid, where the change $P_l$ to be predicted at each cell $l$ is affected by the current status at this location (represented by two variables $C_l$ and $F_l$) and its neighbours. *Flash* provides an equivalent weighted combination of *bitwise-OR* and *negation* predicates for each causality relation (i.e., directed edge). The weights of these predicates are calculated from the input prior probabilities of SBN. Figure 2(c) shows all logical rules defined over the prediction variable $P_2$ in the example.

## 4. DEMO SCENARIO

Our demo attendees would be able to test *Flash* functionality and interact with any of its applications (see Section 3) through one or more of the following scenarios.

## 4.1 Scenario 1: Basic Learning and Prediction

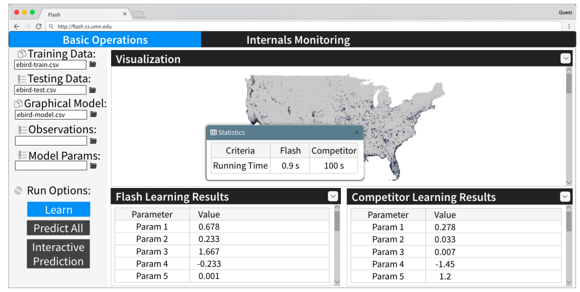


**Figure 3: Main User Interface of *Flash*.**

In this scenario, the demo attendees will explore how to perform learning and prediction in SPGM applications that are empowered by *Flash*. We will interactively run this scenario for the three applications described before.

Figure 3 depicts the main user interface of *Flash*. The user interface allows the user to upload: (1) input datasets, which can be either training data (e.g., as in bird monitoring application) or observations data (e.g., as in safety analysis application), and testing data (e.g., locations need to predict their existence values), and (2) graphical model representation (e.g., model type, model grid size, nodes' dependencies) and its parameters (e.g., prior probability value for each model node, if known as in SBN). Recall that *Flash* currently supports three spatial graphical models (i.e., SMRF, SHMM, and SBN) only. Thus, any input dataset (e.g., training dataset) should be pre-processed by the user to match the settings of one of these three models. The application then waits for the user to select one of three running options, namely, "Learn, "Predict All", and "Interactive Prediction". The "Learn" option facilitates users to learn the model parameters (i.e., the weights of logical rules), if unknown (e.g., regression parameters), while "Predict All" and "Interactive Prediction" (covered in Scenario 2) perform a prediction over either all or selected testing data which is visualized in the "Visualization" area. Once the user clicks on the selected running option, the application then submits its data and configurations to *Flash* back-end to produce the output in the "Results" area. For each supported SPGM model in *Flash*, we provide the most popular state-of-the-art implementations (named as "Competitors") of the learning and prediction operations of this model to compare *Flash* with (e.g., [13] for SMRF, [22] for SHMM, and [5] for SBN). This is important for demo attendees to investigate how the outputs look like. In addition, to judge the efficiency, we calculate some measurements (e.g., running time, prediction accuracy) for both *Flash* and the competitor, and present these measurements in the "Statistics" box for comparison. Figure 3 shows an example of the learned regression parameters in the bird monitoring application, and the running times of both *Flash* and the competitor, where *Flash* achieves at least two orders of magnitude performance gain over the competitor.

## 4.2 Scenario 2: Interactive Prediction

In this scenario, we show the ability of *Flash* to perform an interactive prediction, where the prediction results are changing instantly based on user selection of variables to be predicted. This is an important scenario for spatial data analysts as their SPGM applications are aligned with geo-

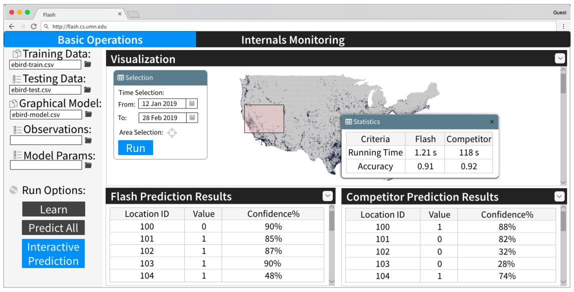


**Figure 4: Interactive Prediction in *Flash*.**

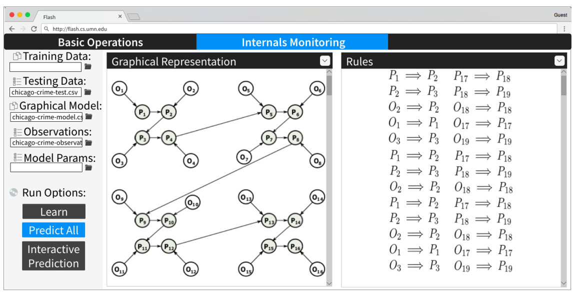


**Figure 5: Monitoring Internals in *Flash*.**

graphical areas (i.e., prediction variables are spatially distributed over geographic areas). Due to the scalability of *Flash* inference engine, demo attendees would be able to try multiple prediction queries and get results very fast. To do that, instead of using the "Predict All" option, the demo attendee should: (1) click the "Interactive Prediction" button, (2) navigate to the spatial range of interest through the map, (3) optionally, in case of having many variables with different time stamps defined at the same location, select variables within a certain time interval using the "Selection" box, (4) click the "Run" button to obtain the predictions in the result area. Note that we can also perform interactive prediction using the competitor implementation, yet, this incurs very high latency in case of large selections due to the competitor scalability issue. Figure 4 shows an example of an interactive prediction query from the bird monitoring application. In this example, we give the predictions (i.e., 0 or 1) for variables within the selection rectangle and specified time range. We also report measurements for running time and prediction accuracy (i.e., calculating ratio of correctly predicted variables using ground truth).

## 4.3 Scenario 3: Internals Monitoring

In addition to performing learning and prediction operations as described before, the demo attendees would be able to see the generated SPGM models and their equivalent logical rules. Figure 5 shows a partial graphical model and its rules from the safety analysis application, visualized in a screen labeled with "Internals Monitoring". Moreover, we show the intermediate outputs of different modules in *Flash* (e.g., how large factor graphs of built models are efficiently generated and stored). Monitoring the system internals is deemed important for demo attendees as it helps in understanding the concepts behind *Flash*, analyzing the system bottlenecks and thinking of future research directions.

# 5. REFERENCES

[1] R. Ambrosini et al. Modelling the Progression of Bird Migration with Conditional Autoregressive Models Applied to Ringing Data. *PLOS ONE*, 9(7), 2014.

[2] S. Balbi et al. A Spatial Bayesian Network Model to Assess the Benefits of Early Warning for Urban Flood Risk to People. *Natural Hazards and Earth System Sciences*, 2016.

[3] F. Bartolucci et al. A Latent Markov Model for Detecting Patterns of Criminal Activity. *Royal Statistical Society Journal*, 170(1), 2007.

[4] J. Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *Royal Statistical Society Journal*, 1974.

[5] bnspatial: Spatial Implementation of Bayesian Networks. `cran.r-project.org/web/packages/bnspatial`, 2019.

[6] S. Chawla et al. Modeling Spatial Dependencies for Mining Geospatial Data. In *SIAM*, 2001.

[7] Y. Chen and D. Z. Wang. Knowledge Expansion over Probabilistic Knowledge. In *SIGMOD*, 2014.

[8] Chicago Crime Data, 2019. `data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2/`.

[9] M. Gasthuber et al. Online and Offline Data Storage and Data Processing at the European XFEL Facility. *Journal of Physics: Conference Series*, 2017.

[10] M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, 1987.

[11] J. Gonzalez-Redin et al. Spatial Bayesian Belief Networks as a Planning Decision Tool for Mapping Ecosystem Services Trade-offs on Forested Landscapes. *Environmental Research*, 2016.

[12] P. J. Green and S. Richardson. Hidden Markov Models and Disease Mapping. *JASA*, 2002.

[13] J. Hughes. ngspatial: A Package for Fitting the Centered Autologistic and Sparse Spatial Generalized Linear Mixed Models for Areal Data. *The R Journal*, 2014.

[14] M. Kaiser et al. Goodness of Fit Tests for a Class of Markov Random Field Models. *The Annals of Statistics*, 2012.

[15] N. Levine. *CrimeStat: A Spatial Statistical Program for the Analysis of Crime Incidents*. Springer Publishing, 2017.

[16] Making The Most Detailed Tweet Map Ever., 2014. `blog.mapbox.com/making-the-most-detailed-tweet-map-ever-b54da237c5ac`.

[17] F. Niu et al. Tuffy: Scaling Up Statistical Inference in Markov Logic Networks Using an RDBMS. In *VLDB*, 2011.

[18] L. Pickle et al. The Crossroads of GIS and Health Information. *International Journal of Health Geographics*, 5(1):51, 2006.

[19] M. Richardson and P. M. Domingos. Markov Logic Networks. *Machine Learning*, 2006.

[20] S. Shekhar et al. Identifying Patterns in Spatial Information: A Survey of Methods. *WIRES: Data Mining and Knowledge Discovery*, 2011.

[21] J. Shin et al. Incremental Knowledge Base Construction Using DeepDive. In *PVLDB*, 2015.

[22] shmm: An R Implementation of Spatial Hidden Markov Models. `github.com/mawp/shmm`, 2019.

[23] J. Singh et al. Event Classification and Location Prediction from Tweets During Disasters. *Annals of Operations Research*, 2017.

[24] B. Sullivan et al. eBird: A Citizen-based Bird Observation Network in the Biological Sciences. *Biological Conservation*, 2009.

[25] Telescope Hubble Essentials: Quick Facts., 2019. `hubble.stsci.edu/the_telescope/hubble_essentials/`.

[26] USGS National Land Cover. `archive.usgs.gov/archive/sites/landcover.usgs.gov/landcoverdata.html`, 2019.

[27] M. Wick et al. Scalable Probabilistic Databases with Factor Graphs and MCMC. In *PVLDB*, 2010.