

MAGIC: A Multi-Activity Graph Index for Activity Detection

Massimiliano Albanese¹

Andrea Pugliese²
Octavian Udrea¹

V.S. Subrahmanian¹

¹Univ. of Maryland Institute for Advanced Computer Studies, College Park, MD 20742, USA
{albanese,vs,udrea}@umiacs.umd.edu

²University of Calabria – DEIS department, Via P. Bucci, Rende, Italy
apugliese@deis.unical.it

Abstract

Suppose we are given a set \mathcal{A} of activities of interest, a set O of observations, and a probability threshold p . We are interested in finding the set of all pairs (a, O') , where $a \in \mathcal{A}$ and $O' \subseteq O$, that minimally validate the fact that an instance of activity a occurs in O with probability p or more. The novel contribution of this paper is the notion of the multi-activity graph index (MAGIC), which can index very large numbers of observations from interleaved activities and quickly retrieve completed instances of the monitored activities. We introduce two complexity reducing restrictions of the problem (which takes exponential time) and develop algorithms for each. We experimentally evaluate our exponential algorithm as well as the restricted algorithms on both synthetic data and a real (depersonalized) travel data set consisting of 5.5 million observations. Our experiments show that MAGIC consumes reasonable amounts of memory and can retrieve completed instances of activities in just a few seconds. We also report appropriate statistical significance results validating our experimental hypotheses.

1. Introduction

There are numerous applications which need to continuously monitor a body of data for the occurrence of certain activities. However, real world activities tend to be high level and can often be executed in many different ways. Models of activities that account for such uncertainty have been devised in various communities and are based on context free grammars [8, 9], graphical models [4, 6] and stochastic automata [1]. However, we are not aware of any work that addresses the problem of indexing large bodies of observations to make activity retrieval more efficient. For instance, web sites typically maintain very large web server logs to determine what activity a user is trying to perform. Detecting this early may allow the web site to prefetch data that it might need and identify methods of personalizing

data for her (predicted) goals. As many existing models of activities already exist in the literature, we start with a known stochastic automata-based activity model from the computer vision field [1]. Our choice was based on the fact that this model is clearly useful for some applications (in vision) and is relatively simple. Moreover, we show that its graph structure provides a good opportunity to track multiple activities simultaneously.

The contributions of this paper may be summed up as follows. We develop: (i) a structure called a *multi-activity graph* and a *multi-activity graph index* that allow multiple activity graphs to be merged together and indexed; (ii) algorithms to build such indexes; (iii) algorithms to solve two types of queries defined in [1]. The *Evidence* problem tries to find all sequences of observations that validate the occurrence of an activity with a minimum probability threshold. The *Identification* problem tries to identify the most probable activity occurring in an observation sequence. The first problem is exponential – hence we introduce two different restrictions that allow to effectively compute it. Finally, we evaluate MAGIC experimentally both on a synthetic and on a real, depersonalized travel data set, and show that the MAGIC index can be built fast, occupies reasonable amounts of memory, and solves efficiently the two problems above.

2. Stochastic Activity Modeling: Overview

We adapt the stochastic automaton activity model for video described in [1] to the case of observation data.

Definition 1 (Stochastic activity) A stochastic activity is a labeled graph (V, E, δ) where: V is a finite set of action symbols; E is a subset of $(V \times V)$; $\exists v \in V$ s.t. $\nexists v' \in V$ s.t. $(v', v) \in E$, i.e., there exists at least one start node in V ; $\exists v \in V$ s.t. $\nexists v' \in V$ s.t. $(v, v') \in E$, i.e., there exists at least one end node in V ; $\delta : E \rightarrow [0, 1]$ is a function that associates a probability distribution with the outgoing edges of each node, i.e., $\forall v \in V \sum_{\{v' \in V | (v, v') \in E\}} \delta(v, v') = 1$.

We now present a small example of a stochastic activity.

Example 1 Figure 1 shows the stochastic activity associated with ordering products from an online store. A user first accesses the product catalog page (catalog) and either inspects the details of desired items (itemDetails) or continues with a previously saved cart (cart). The checkout process requires the user to select a shipping method (shippingMethod), choose a payment method (paymentMethod), review the final details of the order (review) and finally confirm it (confirm). At each stage during the payment process, the user can cancel the sequence and return to the cart and from there on to one of the item details. The probabilities labeling the edges have the following intuitive meaning: once the catalog action has been taken, there is a .9 probability that the user will check details of some item (itemDetails) and a .1 probability that she will continue with a previously saved shopping cart (cart).



Figure 1. Online purchase stochastic activity

Definition 2 (Activity instance) Let (V, E, δ) be a stochastic activity. An instance of (V, E, δ) is a sequence $\langle v_1, \dots, v_n \rangle$ with $v_i \in V$ such that: (i) v_1 is a start node and v_n is an end node in (V, E, δ) ; (ii) $\forall i \in [1, n - 1]$, $(v_i, v_{i+1}) \in E$. Thus, an activity instance is a path from a start node to an end node in (V, E, δ) . The probability of the instance is $\text{prob}(\langle v_1, \dots, v_n \rangle) = \prod_{i \in [1, n-1]} \delta(v_i, v_{i+1})$.

Intuitively, an activity instance is a path from a start node to an end node – the probability of this activity instance occurring is the product of the edge probabilities on the path. **Assumptions.** Throughout this paper, we assume that each node in an activity is an observable event. We also assume that the probability of taking an action at any time only depends on the previous event. Thus the probability labeling an edge (v_1, v_2) can be interpreted as the conditional probability of observing action v_2 given that action v_1 has been observed. These assumptions allow to compute the probability of an instance as a product of probabilities.

3. The Evidence and Identification Problems

We assume that all observations are stored in a single relational database table. Each row (or tuple) o in the table denotes a single action, denoted $o.action$, which is observed at a given time denoted $o.ts$.

Definition 3 (Activity occurrence) Let (V, E, δ) be a stochastic activity and let O be an observation table. An occurrence of (V, E, δ) in O with probability p is a set $\{o_1, \dots, o_n\} \subseteq O$ s.t. $o_1.ts \leq o_2.ts \dots \leq o_n.ts$ and

Table 1. Example web log

id	ts	action	id	ts	action
1	1	catalog	10	7	paymentMethod
2	2	catalog	11	8	itemDetails
3	2	itemDetails	12	9	cart
4	3	cart	13	9	review
5	4	itemDetails	14	10	confirm
6	5	itemDetails	15	11	shippingMethod
7	5	shippingMethod	16	11	paymentMethod
8	6	cart	17	11	review
9	7	shippingMethod	18	12	confirm

$\langle o_1.action, \dots, o_n.action \rangle$ is an instance of (V, E, δ) with $\text{prob}(\langle o_1.action, \dots, o_n.action \rangle) \geq p$. Moreover, we define the span of the occurrence above as the time interval $\text{span}(\{o_1, \dots, o_n\}) = [o_1.ts, o_n.ts]$.

Intuitively, an activity occurrence is a sequence $\{o_1, \dots, o_n\}$ of observations in O corresponding to the nodes of an activity instance and the probability of this occurrence is the probability of the instance.

Example 2 The activity in Figure 1 occurs in the web server log of Table 1, as the latter contains the sequence of observations with identifiers 1, 4, 7, 10, 13, and 14, corresponding to the activity instance $\{\text{catalog}, \text{cart}, \text{shippingMethod}, \text{paymentMethod}, \text{review}, \text{confirm}\}$. The span of this activity occurrence is $[1, 10]$.

Proposition 1 Given an observation table O and a stochastic activity A , the problem of finding all occurrences of A in O takes exponential time, w.r.t the size of O .

The reason for this is that there may be an exponential number of identifiable occurrences of A in O , due to interleaving of activities. This result tells us two things. First, it is not feasible in practice, to try to find all occurrences. Instead, we will attempt to impose restrictions on what constitutes a valid occurrence in order to greatly reduce the number of possible occurrences. We will propose two constraints applicable in most real-world scenarios. Second, due to the size of the search space, it is important to have a data structure that enables very fast searches for activity occurrences. We will propose the MAGIC index structure that allows us to answer the *Evidence* and *Identification* problems efficiently.

Consider the activity in Figure 1 and the following set of observations: $\{\text{catalog}, \text{cart}, \text{shippingMethod}, \text{paymentMethod}, \text{review}, \text{confirm}, \text{confirm}\}$. This leads to two activity occurrences, one for each of the *confirm* actions. Our first restriction requires that if two occurrences O_1 and O_2 are found in the observation sequence and the span of O_2 is contained **within** the span of O_1 , we will discard O_1 from the set of results. This is called the *minimal span restriction* (MS for short).

Definition 4 (MS restriction) Let $O_1 = \{o_1, \dots, o_k\} \subseteq O$ and $O_2 = \{o'_1, \dots, o'_l\} \subseteq O$ be two occurrences of the same activity. We say that the span of O_1 is less than or

equal to the span of O_2 (written $\text{span}(O_1) \leq \text{span}(O_2)$) iff $o_{1.ts} \geq o'_{1.ts}$ and $o_{k.ts} \leq o'_{k.ts}$. Under the MS restriction, we only consider occurrences that are minimal w.r.t. span.

The MS restriction, by itself, may still allow exponentially many occurrences of an activity. This is because multiple occurrences may have the same span.

Our *earliest action* (EA for short) restriction requires that when looking for the next action symbol in an activity occurrence, we always choose the first possible successor in the sequence. For instance, consider the activity definition $v_1 \xrightarrow{1} v_2 \xrightarrow{1} v_3$ and the observation sequence $\{v_1^1, v_2^2, v_1^3, v_2^4, v_3^5\}$ where v_j^i denotes the fact that action v_j was observed at time i . There are two occurrences starting with v_1^1 , namely $\{v_1^1, v_2^2, v_3^5\}$ and $\{v_1^1, v_2^4, v_3^5\}$. Under the EA restriction we only consider $\{v_1^1, v_2^2, v_3^5\}$, since v_2^2 is the first possible successor to v_1^1 . This restriction makes the search space linear in the size of the observation sequence.

Definition 5 (EA restriction) An activity occurrence $\{o_1, \dots, o_n\} \subseteq O$ is said to have the earliest action property if $\forall i \in [2, n], \nexists w_i \in O$ s.t. $w_i.ts < o_i.ts$ and $\{o_1, \dots, o_{i-1}, w_i, o_{i+1}, \dots, o_n\}$ is an occurrence of the same activity.

We now formally state the types of queries we are interested in. The *Evidence* problem is stated as: given an observation table O , a set of activities \mathcal{A} , a time interval (t_s, t_e) , and a probability threshold p_t , compute all the (possibly restricted) occurrences X in O of activities in \mathcal{A} such that X occurs within the interval (t_s, t_e) and $\text{prob}(X) \geq p_t$.

The *Identification* problem is stated as: given an observation table O , a set of activities \mathcal{A} , and a time interval (t_s, t_e) , find the activity which occurs in O in the interval (t_s, t_e) with maximal probability among the activities in \mathcal{A} . A solution to the *identification* problem could be biased by the fact that shorter activity occurrences will generally tend to have higher probabilities. To remedy this, we normalize occurrence probabilities as defined in [1], by introducing the *relative probability* p^* of an occurrence X of activity A as $p^*(X) = \frac{\text{prob}(X) - p_{\min}}{p_{\max} - p_{\min}}$, where p_{\min}, p_{\max} are the lowest and highest possible probabilities of any occurrence of A .

4. Multi-Activity Graph Index Creation

In order to monitor observations for occurrences of multiple activities, we first merge all activity definitions from $\mathcal{A} = \{A_1, \dots, A_k\}$ into a single graph. We use $\text{id}(A)$ to denote a unique identifier for activity A and $I_{\mathcal{A}}$ to denote the set $\{\text{id}(A_1), \dots, \text{id}(A_k)\}$.

Definition 6 (Multi-activity graph) Let $\mathcal{A} = \{A_1, \dots, A_k\}$ be a set of stochastic activities, where $A_i = (V_i, E_i, \delta_i)$. A Multi-Activity Graph is a triple $G = (V_G, I_{\mathcal{A}}, \delta_G)$ where: (i) $V_G = \cup_{i=1, \dots, k} V_i$ is a set of action symbols and (ii) $\delta_G : V_G \times V_G \times I_{\mathcal{A}} \rightarrow [0, 1]$ is a function that associates a triple $(v, v', \text{id}(A_i))$ with $\delta_i((v, v'))$, if $(v, v') \in E_i$ and 0 otherwise.

A multi-activity graph merges a number of stochastic activities. It can be graphically represented by labeling nodes with action symbols and edges with the id's of activities containing them, along with the corresponding probabilities (we do not explicitly represent edges with probability 0). Note that for a set \mathcal{A} of activities, the multi-activity graph can be computed in time polynomial in the size of \mathcal{A} . Figure 2 shows two stochastic activities, A_1 and A_2 , and the corresponding multi-activity graph.

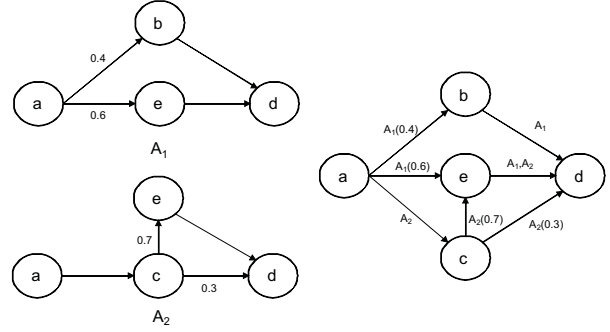


Figure 2. Stochastic activities and multi-activity graph

If $A = (V, E, \delta)$ is an activity and $v \in V$, we use $A.p_{\max}(v)$ to denote the maximum product of probabilities on any path in E between v and an end node. The *multi-activity graph index* data structure allows us to efficiently monitor activity occurrences as new observations occur.

Definition 7 (Multi-activity graph index)

Let $\mathcal{A} = \{A_1, \dots, A_k\}$ be a set of stochastic activities, where $A_i = (V_i, E_i, \delta_i)$, and let $G = (I_{\mathcal{A}}, V_G, \delta_G)$ be the multi-activity graph built over \mathcal{A} . A Multi-Activity Graph Index is a 6-tuple $I_G = (G, \text{start}_G, \text{end}_G, \text{max}_G, \text{tables}_G, \text{completed}_G)$, where:

- $\text{start}_G : V_G \rightarrow 2^{I_{\mathcal{A}}}$ is a function that associates with each node $v \in V_G$, the set of activity-id's for which v is a start node;
- $\text{end}_G : V_G \rightarrow 2^{I_{\mathcal{A}}}$ is a function that associates with each node $v \in V_G$, the set of activity id's for which v is an end node;
- $\text{max}_G : V_G \times I_{\mathcal{A}} \rightarrow [0, 1]$ is a function that associates a pair $(v, \text{id}(A_i))$ with $A_i.p_{\max}(v)$, if $v \in V_i$ and 0 otherwise;
- For each $v \in V_G$, $\text{tables}_G(v)$ is a set of tuples of the form $(\text{current}^\uparrow, \text{activityID}, t_0, \text{probability}, \text{previous}^\uparrow, \text{next}^\uparrow)$, where current^\uparrow is a reference (pointer) to an observation, $\text{activityID} \in I_{\mathcal{A}}$, $t_0, \text{probability} \in \mathbb{R}^+$, previous^\uparrow and next^\uparrow are references to tuples in tables_G ;
- $\text{completed}_G : I_{\mathcal{A}} \rightarrow 2^{\mathcal{P}}$ where \mathcal{P} is the set of tuples in tables_G , is a function that associates an activity identifier $\text{id}(A)$ with a set of references to tuples in tables_G corresponding to a completed instance of activity A .

	Algorithm $insert(t_{new}, I_G, p_t)$ Input: New observation to be inserted t_{new} , multi-activity graph index I_G , threshold probability p_t Output: Updated multi-activity graph index I_G		Algorithm $MAGIC-evidence(I_G, \mathcal{A}, p_t)$ Input: Multi-activity graph index I_G , set of activity identifiers I_A , time interval (t_s, t_e) , threshold probability p_t Output: Set of pairs $(id(A), R)$ where $id(A) \in I_A$ and R is a set of sequences of observations that satisfy activity A with probability above p_t in the time interval (t_s, t_e)
1	// Look at start nodes	1	$S \leftarrow \emptyset$
2	for each tuple $t \in tables_G(t_{new}.action)$	2	for each $id \in I_A$
3	if $t.activityID \in start_G(t_{new}.action)$ and $t.next = \perp$	3	$R \leftarrow \emptyset$
4	$t.current^{\uparrow} \leftarrow t_{new}$	4	for each $t^{\uparrow} \in completed_G(id)$
5	end if	5	if $t.probability \geq p_t$ and $t.t_0 \geq t_s$
6	end for	6	and $t.current.ts \leq t_e$
7	for each activity id for which no tuple was updated in the previous loop	7	$L \leftarrow \{t\}$
8	add $(t_{new}^{\uparrow}, id, t_{new}.ts, 1, \perp, \perp)$ to $tables_G(t_{new}.action)$	8	repeat
9	end for	9	$p^{\uparrow} \leftarrow t.previous^{\uparrow}$
10	// Look at intermediate nodes	10	$t \leftarrow p$
11	for each action symbol $v \in V_G$ s.t. $\exists id \in I_A, \delta_G(v, t_{new}.action, id) \neq 0$	11	$dt^{\uparrow} \leftarrow t.current^{\uparrow}$
12	for each tuple $t \in tables_G(v)$ with $t.next = \perp$ and having maximal t_0	12	add dt to L
13	w.r.t. the set of tuples in $tables_G(v)$ with the same activityID	13	until $t.previous^{\uparrow} = \perp$
14	$a \leftarrow t_{new}.action, id \leftarrow t.activityID$	14	add $reverse(L)$ to R
15	if $\delta_G(v, a, id) \neq 0$ and $t.probability \cdot \delta_G(v, a, id) \cdot max_G(a, id) \geq p_t$	15	end if
16	$t' \leftarrow (t_{new}^{\uparrow}, id, t.t_0, t.probability \cdot \delta_G(v, a, id), t^{\uparrow}, \perp)$	16	end for
17	add t' to $tables_G(a)$	17	add (id, R) to S
18	$t.next \leftarrow t'^{\uparrow}$	18	end for
19	// Look at end nodes	19	return S
20	if $id \in end_G(a)$		
21	add t'^{\uparrow} to $completed_G(id)$		
22	end if		
23	end if		
24	end for		
25	end for		

Figure 3. (a) Multi-activity graph index maintenance algorithm (MS, EA); (b) Algorithm *MAGIC-evidence*

Note that $G, start_G, end_G, max_G$ can be computed a-priori. All the tables that are part of the index will be initially empty. As new observations are added, the index tables will be updated accordingly. The MAGIC index tracks information about which nodes are start and/or end nodes for the original activities. For each node, it also stores (i) the maximum probability of reaching an end node for each activity in \mathcal{A} , (ii) a table that tracks *partially-completed* activity instances where each tuple points to an observation whose action is part of the activity instance, as well as to the previous and successor tuples. In addition, each tuple stores the probability of the activity occurrence so far, and the time at which the partial occurrence began.

4.1. MAGIC Insertion Algorithm

This section describes an algorithm to update the MAGIC index (when new observations occur) under the MS and EA restrictions. We briefly discuss the changes necessary to implement the unrestricted case. Figure 3(a) shows the algorithm under the MS, EA restrictions.

Example 3 Figure 4 shows the evolution of the index for the multi-activity graph of Figure 2, as new observations are collected. The first observation denotes action a. Since both activities A_1 and A_2 have a as their start node, two tuples are added to the index table associated with action a, i.e. $tables_G(a)$. The second observation denotes again action a; to apply the minimal span restriction, the tuples in $tables_G(a)$ are updated to point to the new observation. The third observation has action b, so the insertion algorithm looks at $tables_G(a)$, whose corresponding action is the only predecessor of b in the multi-activity graph, to check whether the new observation can be linked to a partially-completed occurrence. The observation can in fact be linked to the first tuple in $tables_G(a)$ (for activity

A_1), thus a new tuple is added to $tables_G(b)$ with probability equal to the product of the probability of the tuple in $tables_G(a)$ and the probability of the edge from a to b. The fourth observation has action b; to apply the earliest action restriction we do not link it to the first tuple in $tables_G(a)$ that already has a successor. The fifth observation has action c, and can be linked to the second tuple in $tables_G(a)$ (for activity A_2), so a new tuple is added to $tables_G(c)$. The sixth observation has action d, so the insertion algorithm looks at $tables_G(b)$, $tables_G(e)$, and $tables_G(c)$, and adds 2 new tuples to $tables_G(d)$, the first linked to the one in $tables_G(b)$ (for activity A_1) and the second to the one in $tables_G(c)$ (for activity A_2). Moreover, as d is an end node for both activities A_1 and A_2 , pointers to the newly added tuples are added to $completed_G(id(A_1))$ and $completed_G(id(A_2))$, respectively.

Proposition 2 Algorithm $insert$ runs in time $\mathcal{O}(|\mathcal{A}| \cdot max_{(V,E,\delta) \in \mathcal{A}}(|V|) \cdot |O|)$, where O is the set of observations indexed so far.

For space reasons, we omit a detailed description of how the algorithm in Figure 3(a) can be changed to compute unrestricted occurrences¹.

The *MAGIC-evidence* algorithm (Figure 3(b)) finds all minimal sets of observations that validate the occurrence of activities in \mathcal{A} with a probability exceeding a given threshold. For reasons of space, we omit a formal description of the *MAGIC-id* algorithm that solves the *Identifica-*

¹Briefly, when looking at observations corresponding to start nodes, we no longer need to update start nodes, so we remove lines 1-5 and add a tuple to $tables_G(t_{new}.action)$ for any activity $id \in start_G(t_{new}.action)$. When looking at observations corresponding to inner nodes in the activity graph, we can have multiple successors for a single MAGIC tuple, hence we remove the condition $t.next = \perp$. Moreover, when iterating over $tables_G(v)$, we relax the condition of having maximal t_0 .

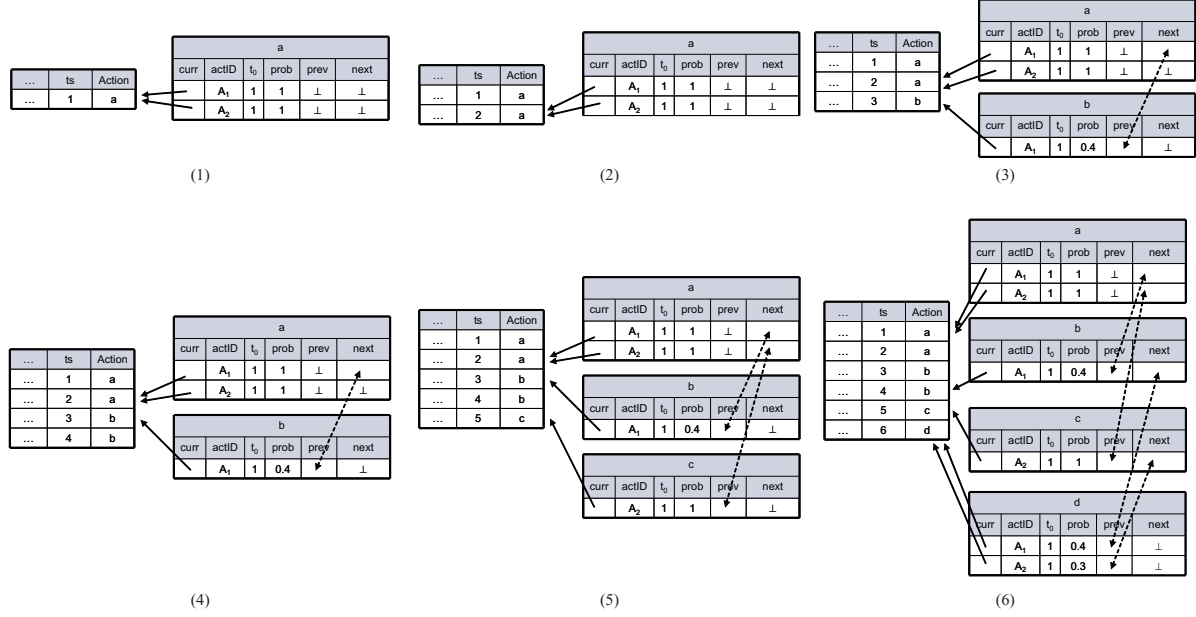


Figure 4. Evolution of a multi-activity graph index

tion problem. *MAGIC-id* simply identifies those tuples in $completed_G$ (and hence the set of associated activity IDs) that have maximum probability and are within the required time span.

Proposition 3 *Algorithms MAGIC-evidence and MAGIC-id terminate and return the correct answers.*

We omit a proof of the above proposition for space reasons.

5. Experimental Results

Our MAGIC implementation consists of approximately 5400 lines of C code running on a Pentium 4 3.2Ghz with 2 GB of RAM running SuSE 9.3. All experiments were averaged over 10 independent runs. We use paired t -tests and z -score tests (where applicable) to determine statistical significance.

Data sets. We evaluated MAGIC on (i) a synthetic dataset of 5 million observations generated automatically, using two separate Java programs²; (ii) a third party depersonalized dataset consisting of travel information such as hotel reservations, passport and flight information, containing approximately 5.5 million observations. We enhanced the second dataset by adding bogus credit card transactions for purchases, transportation, etc. The final dataset consists of 7.5 million observations and requires 1.05 GB of memory/disk

²The first program generates random activities by taking as an input the set of action symbols, a probability distribution (one of uniform, Gauss and Zipf) and generates an activity graph in which the edges and probability labels are based on that distribution. The second tool generates observation sequences by using the set of activity definitions, the desired level of noise and the amount of interleaving between occurrences (the maximum number of instances that are executing at any given timestamp).

space. We manually developed 30 activity definitions from activities that occur in the dataset.

MAGIC Index Creation Time and Memory Requirements. Figure 5(a) and (b) show the time to build the index and the memory used for the index for the synthetic dataset. Note that the x-axis is on a log-scale. As expected, it took approximately 10 minutes and 588 MB to build the unrestricted MAGIC structure for 50,000 observations – we stopped running experiments on the unrestricted version on MAGIC at this time. The three restrictions we defined in the paper yield significantly better results. We are able to process the entire dataset in 86 and less than 1.5 seconds for the MS and EA restrictions respectively. For reasons of space, we do not show similar figures for the travel data set – however, with 7.5M observations, the MAGIC structure is built in under 3 minutes with 150MB memory under the MS restriction.

Query Time. Finally, we look at the average query time on the synthetic dataset. We generated 75 *evidence* and 75 *identification* queries. Each query was run on 10 intervals generated uniformly at random encompassing between 1% and 75% of the data. Each *evidence* query was also run with 5 distinct thresholds selected uniformly at random. The running times reported in Figure 5(c) are an average over the entire set of queries. Query answering times are always below 2 seconds for any restrictions, showing that the MAGIC structure handles activity occurrences efficiently.

6. Related Work and Conclusions

There is a large body of work in the AI community on plan and activity recognition, a large portion of which

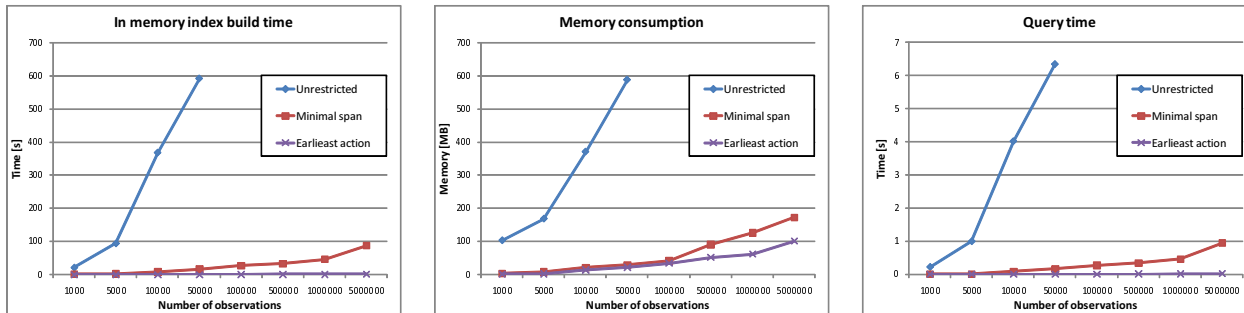


Figure 5. (a) MAGIC build time variation with the number of observations in the synthetic dataset; (b) MAGIC memory requirements for the synthetic dataset; (c) MAGIC query time

rely on Hidden Markov Models (HMMs) and their variants. Luhr et al. [6] apply Hierarchical HMMs to learn the probabilistic nature of simple sequences of activities. Duong et al. [4] introduce the Switching Hidden Semi-Markov Model (S-HSMM), a two-layered extension of the Hidden Semi-Markov Model (HSMM). The bottom layer represents atomic activities and their duration using HSMMs, while the top layer represents a sequence of high-level activities, defined in terms of atomic activities. [7] uses non-stationary HSMMs to model the dependency of transition probabilities on the duration an agent has spent in a given state. Dynamic Bayesian networks can also be used for tracking and recognizing multi-agent activities [5]. The CFG-based representation of human activities [8, 9] and interactions enables to formally define complex activities based on simple actions. The problem of recognizing multiple interleaved activities has been studied in [2], where the authors propose a symbolic plan recognition approach, relying on hierarchical plan-based representation and a set of algorithms that can answer a variety of recognition queries.

However, to the best of our knowledge, the problem of indexing large amounts of observations for the purpose of quickly retrieving completed instances of activities has not been addressed before. Indexing has primarily been used as a method to retrieve past activities or plans to recognize activities in a new set of observations (or a new video). For instance, [3] proposes recognition of human activities in videos on top of multidimensional index structures that store body pose vectors in different frames.

Our work brings two important novel contributions. First, we define the *multi-activity graph* which captures many activity in a single labeled stochastic automata. Second, we define the MAGIC index structure that can index millions of observations from interleaved activities and retrieve completed instances of these in just a few seconds. We also introduce two reasonable restrictions that reduce the overall complexity of the activity recognition problem to a manageable level. Finally, we experimentally evaluate

our algorithms on both a synthetic and a third-party dataset and show that MAGIC is very fast and has reasonable memory consumption.

Acknowledgments. Researchers funded in part by grant N6133906C0149, ARO grant DAAD190310202, AFOSR grants FA95500610405 and FA95500510298, and NSF grant 0540216.

References

- [1] M. Albanese, V. Moscato, A. Picariello, V. Subrahmanian, and O. Udrea. Detecting Stochastically Scheduled Activities in Video. In *Proc. of IJCAI-07*, pages 1802–1807, 2007.
- [2] D. Avrahami-Zilberbrand, G. Kaminka, and H. Zarosim. Fast and Complete Symbolic Plan Recognition: Allowing for Duration, Interleaved Execution, and Lossy Observations. In *Proc. of MOO-05*, 2005.
- [3] J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram. Human Activity Recognition Using Multidimensional Indexing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(8):1091–1104, 2002.
- [4] T. Duong, H. Bui, D. Phung, and S. Venkatesh. Activity Recognition and Abnormality Detection with the Switching Hidden Semi-Markov Model. In *Proc. of IEEE CVPR-05*, volume 1, pages 838–845, 2005.
- [5] R. Hamid, Y. Huang, and I. Essa. ARGMode Activity Recognition Using Graphical Models. In *Proc. of IEEE CVPR-03*, volume 3, pages 38–43, 2003.
- [6] S. Luhr, H. Bui, S. Venkatesh, and G. West. Recognition of Human Activity through Hierarchical Stochastic Learning. In *Proc. of PCC-03*, pages 416–422, 2003.
- [7] E. Marhasev, M. Hadad, and G. A. Kaminka. Non-stationary Hidden Semi-Markov Models in Activity Recognition. In *Proc. of MOO-06*, 2006.
- [8] R. Nevatia, T. Zhao, and S. Hongeng. Hierarchical Language-based Representation of Events in Video Streams. In *Proc. of Workshop on Event Mining (in conj. with IEEE CVPR)*, 2003.
- [9] V.-T. Vu, F. Brémond, and M. Thonnat. Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition. In *Proc. of IJCAI-03*, pages 1295–1302, 2003.