Learning Automaton Based On-Line Discovery and Tracking of Spatio-temporal Event Patterns

Anis Yazidi[1], Ole-Christoffer Granmo[1], Min Lin*, Xifeng Wen*,
B. John Oommen[1,2], Martin Gerdes[3], and Frank Reichert[1]

[1] Dept. of ICT, University of Agder, Grimstad, Norway
[2] School of Computer Science, Carleton University, Ottawa, Canada**
[3] Ericsson Research, Aachen, Germany

Abstract. Discovering and tracking of spatio-temporal patterns in noisy sequences of events is a difficult task that has become increasingly pertinent due to recent advances in ubiquitous computing, such as community-based social networking applications. The core activities for applications of this class include the sharing and notification of events, and the importance and usefulness of these functionalites increases as event-sharing expands into larger areas of one's life. Ironically, instead of being helpful, an excessive number of event notifications can quickly render the functionality of event-sharing to be obtrusive. Rather, any notification of events that provides redundant information to the application/user can be seen to be an unnecessary distraction. In this paper, we introduce a new scheme for discovering and tracking noisy spatio-temporal event patterns, with the purpose of suppressing reoccurring patterns, while discerning novel events. Our scheme is based on maintaining a collection of hypotheses, each one conjecturing a specific spatio-temporal event pattern. A dedicated Learning Automaton (LA) – the *Spatio-Temporal Pattern LA* (STPLA) – is associated with each hypothesis. By processing events as they unfold, we attempt to infer the correctness of each hypothesis through a real-time guided random walk. Consequently, the scheme we present is computationally efficient, with a minimal memory footprint. Furthermore, it is ergodic, allowing adaptation. Empirical results involving extensive simulations demonstrate the STPLA's superior convergence and adaptation speed, as well as an ability to operate successfully with noise, including both the erroneous inclusion and omission of events. Additionally, the results included, which involve a so-called "*Presence Sharing*" application, are both promising and in our opinion, impressive. It is thus our opinion that the proposed STPLA scheme is, in general, ideal for improving the usefulness of event notification and sharing systems, since it is capable of significantly, robustly and adaptively suppressing redundant information.

Keywords: Learning Automata, Spatio-Temporal Pattern Recognition.

* Former student. Can be contacted at: C/o Dr. Ole-Christoffer Granmo, Dept. of ICT, University of Agder, Grooseveien 36, 4876 Grimstad, Norway.

** *Chancellor's Professor*; *Fellow : IEEE* and *Fellow : IAPR*. The Author also holds an *Adjunct Professorship* with the Dept. of ICT, University of Agder, Norway.

B.-T. Zhang and M.A. Orgun (Eds.): PRICAI 2010, LNAI 6230, pp. 327–338, 2010.

1 Introduction

Presence Sharing is a ubiquitous service in which distributed mobile devices periodically broadcast their identity via short-range wireless technology such as BlueTooth or WiFi [1]. The whole problem of *Presence Sharing* is intricately bound to the issue of the recording and processing of "events" involving the entities included within the social network. Applications that utilize *Presence Sharing* have been used in social contexts to maintain an "in touch" feeling strengthening social relations [2], as well as in work environments to enhance collaboration between colleagues [3].

Typically, "events" occurring in the real world can be characterized as being in one of two classes, i.e., "Stochastically Episodic" (SE) and "Stochastically Non-Episodic" (SNE). This is a distinction that is especially pertinent in simulation, where it is customary for one to model the behaviour of accidents, telephone calls, network failures etc. using their respective probability distributions, even though they follow no known pattern. Indeed, events of these families happen all the time, and so can be termed as being "stochastically non-episodic". As opposed to this, there is a whole class of events that can stochastically occur in a non-anticipated manner. These so-called "stochastically episodic" events include earthquakes, nuclear explosions etc. The difficulty with modelling SE events is that most of the observations appear as noise. However, when the SE event does occur, its magnitude and features far overshadow the background, as one observes after a seismic event. The modelling and simulation of such SE events in the presence of a constant stream of SNE events is a relatively new field [4,5], where the authors model the SE and SNE events *simultaneously* in such a way that the effect of an SE event is perceived through the "lens" of the underlying background of SNE events.

Since events are almost omnipresent, one has to consider the observation due to Garlan *et al.* [6], who state that the most precious resource in a computer system is no longer its processor, memory, disk, or network, but rather human attention. Thus, our aim in this paper is to address a fundamental challenge concerning the above class of applications: *How can one harvest the benefit of event-sharing without distracting the application user with redundant notifications*? The solution we propose is to try to discern the nature of the events encountered[1]. Of course, the events may not be drastically SE or SNE, as in the case of earthquakes or nuclear explosions. However, if we can discern that an event is repeating (even though this repetition is non-periodic), it is still of a SNE nature which must be given less weight, while non-repeating events (which are in one sense, SE) must be assigned a greater weight. Thus, the question we resolve involves demonstrating how we can enhance the *Presence Sharing* experience by weighting the SE and SNE events appropriately.

[1] To exemplify the usefulness of such a strategy, consider the nuisance caused by being notified every time one meets a colleague at work, which is a repeating pattern, or a SNE event. In contrast, it would be far more useful to be promptly notified whenever the same colleague unexpectedly appears in your vicinity after a travel abroad. This would be non-repeating pattern, or an SE event.

1.1 Related Work

A number of earlier studies have investigated techniques for discovering the periodicity of time patterns, such as the episode[2] discovery algorithm found in [7]. However, episode discovery, and other related approaches, suffer from the limitation that they assume unperturbed patterns that exhibit an exact periodicity. Unfortunately, the real-life unfolding of events is typically noise ridden. On the one hand, regular events may get cancelled, introducing what we define as *omission noise*, and on the other, events may arise spontaneously and unexpectedly, without being part of a periodic pattern, introducing *inclusion noise*. A pioneering work which was reported in [8], introduced the concept of *off-line* mining of partially periodic events. Nevertheless, deciding whether to suppress event notifications must often be done instantaneously, as the events are unfolding. Indeed, we argue that any realistic scheme should discover and adapt to patterns as they appear and evolve in an on-line manner, without relying on extensive off-line data mining.

1.2 Paper Contribution and Organization

The paper is organized as follows. In Section 2, we present our overall approach to on-line discovery and tracking of spatio-temporal event patterns, in which the so-called Learning Automata (LA) plays a crucial role. The scheme is designed to deal with noisy spatio-temporal event patterns, when event patterns are evolving with time. We continue in Section 3 by evaluating our scheme using an extensive range of static and dynamic *noisy* event patterns. The experiments demonstrate the scheme's superior convergence and adaptation speed, as well as an excellent ability to operate successfully with noise, including both erroneous inclusion and omission of events. In order to highlight the applicability of our scheme, we present a "*Presence Sharing*" application prototype in Section 4 where we also summarize some initial user experiences. Finally, Section 5, concludes the paper and also provide pointers for further work.

2 On-Line Discovery and Tracking of Spatio-temporal Event Patterns

The method which we propose is based on the theory of LA. Since space does not permit a detailed overview of this theory, this is included elsewhere [9]. However, in all brevity, we state that our scheme is based on maintaining a collection of hypotheses, each one conjecturing a specific spatio-temporal event pattern. A dedicated LA, which we coin the *Spatio-Temporal Pattern LA* (STPLA), is associated with each hypothesis. The STPLA decides whether its corresponding hypothesis is true by observing events as they unfold, processing evidence for and/or against the correctness of the hypothesis. To explain this, we first address hypothesis management, and then proceed with the details of the STPLA.

[2] The expression "episode" used in this setting must not be confused with the class of SE and SNE events described in the earlier paragraph.

2.1 Hypothesis Management

The premise of our discussions is the following: In order to reduce distraction, events should only be signalled when they are SE. This means that they cannot be anticipated, obey no known stochastic distribution, and possess an element of "surprise", i.e., they can not be easily predicted by the recipient[3]. An event can either be sporadic, arising spontaneously, or it can be part of a spatio-temporal pattern, making it occur regularly. In either case, if it cannot be explained by any of the spatio-temporal patterns that are known by the recipient, the recipient should be notified. However, when the event constitutes a part of an ongoing spatio-temporal pattern, it is really non-episodic (or SNE) in nature. We require that this phenomenon be discovered as soon as possible, so that the events generated from this pattern can be suppressed before the pattern loses its novelty to the recipient.

In our proposed scheme, when an event is observed, all potentially interesting patterns that *could* have produced the event are identified. We refer to these potential patterns as *hypotheses*. The reader will thus observe that our approach is based on the concept of predefined pattern structures, as advocated in [10], rather than trying to look for patterns with unknown structure. Thus, in this spirit, we consider a discrete world of m spatial location primitives $L = \{l_1, l_2, \ldots, l_m\}$ and of n discrete time primitives $T = \{t_1, t_2, \ldots, t_n\}$ of appropriate granularity. By way of example, the location primitives could be "Home", "Office", or "Abroad", while the time primitives could be "Mondays", "Tuesdays", "Weekends", and so on. The location and time primitives are combined from their cross-product spaces to produce spatio-temporal patterns. Thus, the resulting spatio-temporal pattern space would (or could) be an exhaustive enumeration of relevant combinations such as "Mondays at Office", "Weekends at Office", and so on. Each spatio-temporal pattern of the latter form is seen as a hypothesis, conjecturing that the respective pattern specifies an ongoing stream of events. In the following, we assume that there are r such hypotheses, represented as a set $H = \{h_1, h_2, \ldots, h_r\}$. Observe that although the cardinality of this set might get large, the computational efficiency and small memory footprint of our LA (as seen presently), effectively handles the size of the state space.

Note too that the novelty of this present work is not the above indicated structuring of the spatio-temporal pattern space, which is a well-known approach used in typical calendar systems. Rather, it is the learning scheme we propose[4] for determining whether a given spatio-temporal event pattern can be found in a stream of events, in an on-line manner, and under noisy conditions.

[3] Events should, of course, also match the interest profile of the recipient. We will, in this paper, assume that all events are of interest, as long as they are novel. On-line adaptive learning of interest profiles will be addressed in another forthcoming paper.

[4] Using the techniques presented in [4,5], we are currently investigating how one-class classifiers can be used to learn the most appropriate hypothesis. This would assume that the patterns which can be anticipated constitute the SNE events, and the set of SE events, which cannot be anticipated, constitutes the one-class to be recognized.

2.2 Learning Automaton Based On-Line Discovery and Tracking of Spatio-temporal Event Patterns

We base our work on the principles of LA [9,11]. LA have been used to model biological systems [12], and have recently attracted considerable interest because they can learn the optimal actions when operating in (or interacting with) unknown stochastic environments. Furthermore, they combine rapid and accurate convergence with low computational complexity.

Generally stated, an LA chooses a sequence of actions offered to it by a random *environment*. The environment can be seen as a generic *unknown* medium that responds to each action with some sort of reward or penalty, usually *stochastically*. Based on the responses from the environment, the aim of the LA is to find the action that minimizes the expected number of penalties received. Before we proceed with describing the STPLA itself, it is necessary for us to first define the environment that we are dealing with.

Spatio-Temporal Pattern Environment: The purpose of the Spatio-Temporal Pattern Environment is to provide feedback to the individual STPLA about the validity of their respective hypotheses.

In all brevity, at each time instant matching the time primitive t_i, if an STPLA predicts the presence of an event at location l_j, it informs the environment about this prediction. Conversely, if the STPLA predicts the absence of an event at the same location, this too is submitted to the environment. The environment, in turn, responds with a *Reward* if an event took place (or did not take place) as predicted. If the prediction is incorrect, on the other hand, the environment responds with a *Penalty* instead. That is, the STPLA is penalized if an event takes place, but none was predicted, or if an event is predicted, but does not take place. The latter reward policy is illustrated in Fig. 1.

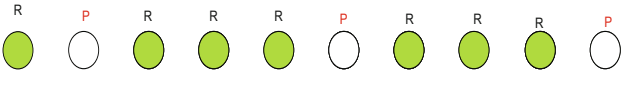

Fig. 1. Feedback for a daily event hypothesis (R-Reward, P-Penalty)

The figure illustrates events generated from a daily meeting. The STPLA that hypothesizes a daily meeting will be rewarded each day a meeting takes place (green circle) because of its ability to correspondingly predict the daily event. An important challenge that we address in this paper, however, is how to deal with spatio-temporal event patterns that are affected by noise. In the figure, for example, some of the daily meetings may be cancelled (depicted by white circles) due to external conditions, such as when the participants are unavailable. Thus, when meetings are cancelled, the STPLA maintaining the daily meeting hypothesis will get penalized because of its prediction, despite the fact that its hypothesis is true. In a similar vein, so-called "straggler" events, not being part of any periodic pattern, can also occur in a sporadic and spontaneous manner.

From the above example it can be seen that we face two kinds of noise:

Omission Error: This is an error which occurs when an event that forms a part of a periodic spatio-temporal pattern is randomly left out. In other words, the event was supposed to have taken place according to the pattern, but did not. Notice the SE nature of this event – it is not something that could have been anticipated.

Inclusion Error: This is an error which occurs when an event that occurs is not part of a periodic (anticipated) pattern, but rather arises sporadically and spontaneously. Again, one must observe the SE nature of this event.

By way of example, Alice may cancel a regular meeting with Bob due to ill health. However, Alice may still meet Bob sometime outside of the regular meeting schedule – purely by chance (e.g., an accidental meeting in the canteen). In this manner, we can appropriately model both these kinds of noise.

The Spatio-Temporal Pattern Learning Automaton (STPLA): We now introduce the STPLA that we have designed to discover and track spatio-temporal patterns. In brief, the task of an STPLA is to decide whether a specific spatio-temporal pattern hypothesis is true. By observing events as they unfold, the correctness of an hypothesis is decided.

The STPLA can be designed to model arbitrarily general SE and SNE events. But due to space limitations, in this paper, we confine our design and implementation details to events which can be characterized *deterministically*.

The STPLA is inspired by so-called family of fixed structured LA [13]. Accordingly, a STPLA can be defined in terms of a quintuple [9]:

$$\{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, \mathcal{F}(\cdot,\cdot), \mathcal{G}(\cdot,\cdot)\}.$$

Here, $\underline{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_s\}$ is the set of internal automaton states. $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of automaton actions. Further, $\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of inputs that can be given to the automaton. An output function $\alpha_t = \mathcal{G}[\phi_t]$ determines the action performed (or chosen) by the automaton given the current automaton state. Finally, a transition function $\phi_{t+1} = \mathcal{F}[\phi_t, \beta_t]$ determines the new state of the automaton from: (1) The current state of the automaton and (2) The response of the environment to the action performed (or chosen) by it.

Based on the above generic framework, the crucial issue is to design automata that can learn the optimal action when interacting with the environment. Several designs have been proposed in the literature, and the reader is referred to [9] for an extensive treatment. In this paper, since we target the learning of spatio-temporal patterns, our goal is to design an LA that is able to discover and track such patterns over time. Briefly stated, we construct an automaton with

- States: $\underline{\Phi} = \{1, 2, \ldots, N_1, N_1 + 1, \ldots, N_1 + N_2 + 1\}$.
- Actions: $\underline{\alpha} = \{\mathit{Notify}, \mathit{Suppress}\}$.
- Inputs: $\underline{\beta} = \{\mathit{Reward}, \mathit{Penalty}\}$.

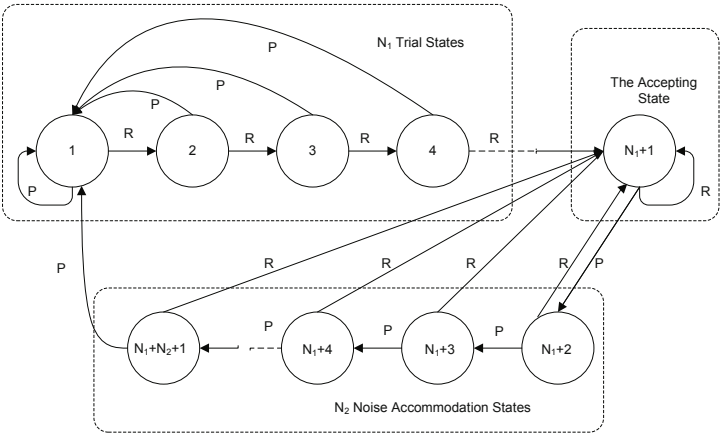

Fig. 2. The state transition map and the output function of a STPLA

Fig. 2 specifies the state space of STPLA as well as the $\mathcal{G}$ and $\mathcal{F}$ matrices. The $\mathcal{G}$ matrix can be summarized as follows. If the automaton state lies in the set $\{1, \ldots, N_1\}$, which we refer to as the *Pattern Evaluation States*, then the LA will choose the action "Notify". If, on the other hand, the state is either $N_1 + 1$ or one of the states in the set $\{N_1 + 2, \ldots, N_1 + N_2 + 1\}$, it will choose the action "Suppress". We refer to the state $N_1 + 1$ as the *Pattern Acceptance State*, and the states $\{N_1 + 2, \ldots, N_1 + N_2 + 1\}$ as the *Pattern Tracking States* for reasons explained presently. Note that since we initially do not know whether a pattern is present, we set the initial state of our automaton to 1.

The state transition matrix $\mathcal{F}$ determines how the learning proceeds. In brief, the learning is divided into three parts:

Pattern Evaluation: In the Pattern Evaluation part, the goal of the LA is to discover the presence of the spatio-temporal event pattern associated with the maintained hypothesis, without being distracted by omission and inclusion errors. In this phase, the state transitions illustrated in the figure are such that any deviance from the hypothesized pattern, modelled as a Penalty (P), causes a jump back to state 1. Conversely, only a systematic presence of the pattern hypothesized, modelled as a pure sequence of Rewards (R), will allow the LA to pass into the Pattern Acceptance part.

Pattern Acceptance: In the Pattern Acceptance part, consisting of state $N_1 + 1$, the hypothesized pattern has been confirmed with high probability.

Pattern Tracking: In the Pattern Tracking part, consisting of states $\{N_1 + 2, \ldots, N_1 + N_2 + 1\}$, the goal is to detect when the discovered pattern disappears, without getting distracted by omission errors. Thus, this part is the "opposite" of the Pattern Evaluation part in the sense that a pure sequence of Penalties is required to "throw" the LA back into the Pattern Evaluation part again, while a single Reward reconfirms the pattern, returning the LA to the Pattern Acceptance part of the state space.

In other words, the automaton attempts to incorporate past deterministic responses when deciding on a sequence of actions.

We define the "*Ensemble*" characteristic of a set of STPLA as follows: *An event is only signalled to the recipient when all of the STPLA that maintain hypotheses that are consistent with the event, collectively find themselves in the Pattern Evaluation part of the state space.* As soon as one of the STPLA can deterministically[5] explain an event as being part of the corresponding hypothesized spatio-temporal event pattern, that particular event will be suppressed and no notification will be issued to the recipient.

3 Experiments

In order to evaluate our scheme, we have applied it to both an event simulation system as well as to a real world prototype. This section reports the results obtained using the simulation, while the next section covers the prototype.

Since one of our main aims is handling noisy patterns, we intend to impose "stress" onto our scheme by using a wide range (percentage or degrees) of omission and inclusion errors. We will use q to denote the probability of event omission, while p denotes the probability of event inclusion. We also investigate how the number of states N_1 and N_2 affect the LA's speed and the accuracy.

As a performance criterion, we have chosen the probability of issuing a notification (alert) when an event takes place. We refer to this probability as P_1. Intriguingly, when a spatio-temporal pattern produces events, P_1 should be minimized, while when events are novel, P_1 should be maximized. We will presently see that our scheme achieves both. For instance, consider an event that occurs daily, with the possibility, however, that events may get cancelled (causing omission errors). In that case, our scheme should quickly stop alerting the user about these events. In contrast, when novel sporadic events occur, even on a daily basis, our scheme should rather always produce alerts, so that the user is notified about these novel events. Thus, by monitoring our scheme in terms of the index P_1 using various scenarios, we can capture its overall performance.

3.1 Performance after Convergence

Table 1 summarizes the performance after convergence, with a wide range of event inclusion probabilities, p, event omission probabilities, q, *Pattern Evaluation States*, N_1, and *Pattern Tracking States*, N_2. The resulting performance is then reported in terms of P_1, with P_1 being estimated by averaging over $1,000$ experiments, each consisting of $100,000$ iterations.

In the case of daily patterns, we have varied the omission error probabilities from $q = 0.05$ to $q = 0.2$, thus covering a spectrum of small to high degrees of omission noise. In the case when no patterns are present, we have allowed random encounters to appear with probabilities from $p = 0.05$ to $p = 0.2$.

[5] The system can easily be generalized for SE and SNE events by rendering the transitions stochastic.

Table 1. Alert probability P_1 under varying conditions

	Daily Pattern			No Underlying Pattern		
	$q = 0.05$	$q = 0.1$	$q = 0.2$	$p = 0.05$	$p = 0.1$	$p = 0.2$
(N_1, N_2)						
$(1, 5)$	1.5E-8	9.9E-7	6.4E-5	0.735	0.531	0.262
$(2, 5)$	3.2E-8	2.1E-6	1.4E-4	0.983	0.925	0.680
$(3, 5)$	4.9E-8	3.3E-6	2.4E-4	0.999	0.992	0.916
$(4, 5)$	6.7E-8	4.7E-6	3.6E-4	0.999	0.999	0.982
$(5, 5)$	8.6E-8	6.2E-6	5.2E-4	0.999	0.999	0.996
$(5, 4)$	1.7E-6	6.2E-5	2.6E-3	0.999	0.999	0.997
$(5, 3)$	3.4E-5	6.2E-4	0.012	0.999	0.999	0.998
$(5, 2)$	6.9E-4	6.2E-3	0.062	0.999	0.999	0.998
$(5, 1)$	0.0137	0.059	0.254	0.999	0.999	0.999

From Table 1, we see that for the best configuration, $N_1 = N_2 = 5$, we get very high accuracy, with the scheme producing a negligible number of superfluous notifications to the user, while alerting the user of almost all novel events, even with high degrees of both omission and inclusion errors.

3.2 Performance in Dynamic Environment

To investigate the ability of our scheme to track spatio-temporal patterns that change with time, we have conducted several experiments in dynamic environments. In all brevity, we report here a representative configuration, where spatio-temporal patterns end after a certain time period, while new ones are introduced every 200^{th} iteration. We modelled this by using an omission error probability

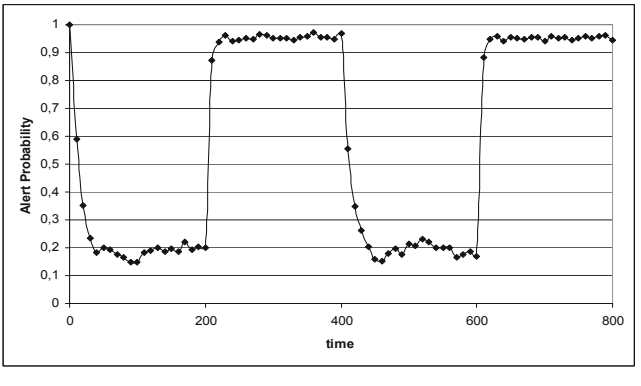

Fig. 3. Evolution of the alert probability in a dynamic environment

of $q = 0.2$ when a pattern was present, and with an inclusion error probability of $p = 0.2$ when no pattern was present.

Fig. 3 depicts how the STPLA scheme adapts to the presence and absence of patterns over time. For instance, prior to time instant 200, the probability q was equal to 0.2, implying the presence of a daily pattern. As seen, the STPLA quickly learns to suppress these events, albeit, with some error due to the high omission error probability. When the pattern disappears after 200 time steps, being replaced with novel events only, we observe how quickly the STPLA changes from suppressing the events to alerting the user of them.

We thus conclude by stating that the empirical results confirm the power of STPLA both in noisy and dynamic environments.

4 Prototype

In addition to the empirical results presented in the previous section, we have also implemented a social networking application and conducted real-life tests.

A key requirement of our community based social networking application demands that users can be made aware of the *Presence* of their friends at anytime and anywhere using their mobiles sensing capabilities. The latter requirement is akin to the field of pervasive computing where ad-hoc mode-based architectures are recognized to be a better alternative than infrastructure-based architecture.

We now provide a brief description of our prototype, the details of whose implementation can be found in [14]. Our prototype system consists of two mobile phones: *HTC P3300* and *Sony Ericsson X1*, both of which are equipped with Wi-Fi modules. An ad-hoc network is established to provide a communication platform where our proposed solution for a "Friend Reminder" service runs.

This design is based on the "SmokeScreen" architecture [1], which introduces an effective approach to resolve privacy issues of *Presence Sharing*. The signal generation procedure[6] referred in [1] is depicted in Fig. 4. However, we have added novel enhancements to the "SmokeScreen" approach, by introducing mechanisms that allow a finer level of privacy control. In brief, we allow the user to specify exactly which of his friends can see the signal of his *Presence*. Accordingly, we let every pair of friends share a symmetric key. This is in contrast to the results presented in [1] where a user shares the information of his *Presence* with his social network at the granularity of his group. A major disadvantage of the latter approach is thus that the user cannot apply a finer privacy control by preventing a specific member of the group from sensing the information of his *Presence* (unless the user does not broadcast the signal of his *Presence*). From a privacy perspective, we believe that the control of the user-related information should be fully under his own control. Thus, every user should be able to authorize the specific people who have the right to reveal his user-related information, and to also isolate other users.

The users must be synchronized to independently update the *Presence* signal and broadcast it periodically. Note that the update is deterministic so that every

[6] As in [1], we use md5 to compute the signal and sha1 to update the secret key.

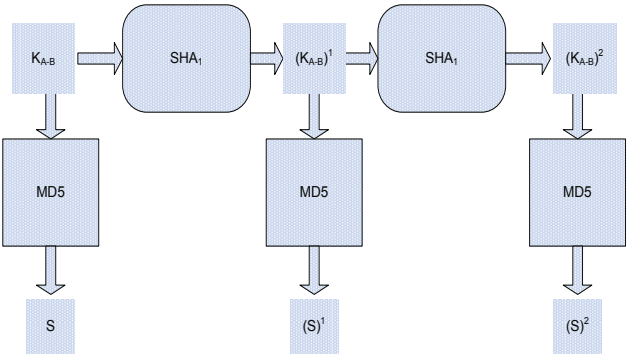

Fig. 4. The signal and key generation over time proposed by [1] and used by us, where k_{A-B} stands for the symmetric key

pair of participating users (for example Alice and Bob) can predict and interpret the time varying broadcast *Presence* signal. The *Presence* signal might vary on the hour and is known only to Alice and Bob, thus preventing impersonation attacks. As alluded to previously, we employed a symmetric key per pair of social contacts. Consequently, the size of the broadcast *Presence* signal increases linearly with the number of social contacts. In order to alleviate this problem, we have used Bloom filters to reduce the size of the *Presence* signal [15], and thus the operation of *Presence* detection reduces to the Bloom filter match operation.

Based on the above architecture, we implemented our STPLA scheme on each mobile phone, allowing suppression of *Presence* notification when the *Presence* is part of a regular pattern. In all brevity, the STPLA scheme made the "Friend Notification Service" less obtrusive by only alerting the user of novel events, but suppressed alerts for regular meetings (e.g., for weekly lectures).

5 Conclusion

In this paper, we have presented the *Spatio-Temporal Pattern Learning Automaton* (STPLA) for the on-line discovery and tracking of patterns in noisy event streams. Our scheme is based on a team of finite automata, rendering it computationally efficient with a minimal memory footprint. The advantages of our approach was demonstrated through extensive simulations, as well as a prototype running on mobile devices. The scheme demonstrated excellent performance under different noise levels and in various dynamic settings. We thus believe the STPLA forms an ideal framework for notification suppression in event notification based systems. As a future work, we intend to formally analyze the behaviour of the STPLA, as well as to extend our prototype to learning interest profiles and adaptive service recommendations.

References

1. Cox, L.P., Dalton, A., Marupadi, V.: SmokeScreen: flexible privacy controls for presence-sharing. In: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, San Juan, Puerto Rico, pp. 233–245. ACM, New York (2007)
2. Eagle, N., Pentland, A.: Social serendipity: mobilizing social software. IEEE Pervasive Computing 4(2), 28–34 (2005)
3. Holmquist, L., Falk, J., Wigstrm, J.: Supporting group collaboration with interpersonal awareness devices. Personal and Ubiquitous Computing 3(1), 13–21 (1999)
4. Bellinger, C.: Simulation and pattern classification for rare and episodic events. Master's thesis, Carleton University, Ottawa, Ontario (2010)
5. Bellinger, C., Oommen, B.J.: On simulating episodic events against a background of noise-like non-episodic events. In: Proceedings of 42nd Summer Computer Simulation Conference, SCSC 2010, Ottawa, Canada, July 11-14 (to appear 2010)
6. Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.: Project aura: toward distraction-free pervasive computing. IEEE Pervasive Computing 1(2), 22–31 (2002)
7. Youngblood, G., Cook, D.: Data mining for hierarchical model creation. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(4), 561–572 (2007)
8. Ma, S., Hellerstein, J.: Mining partially periodic event patterns with unknown periods. In: Proceedings of 17th International Conference on Data Engineering, pp. 205–214 (2001)
9. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, Englewood Cliffs (1989)
10. Lee, C., Chen, M., Lin, C.: Progressive partition miner: an efficient algorithm for mining general temporal association rules. IEEE Transactions on Knowledge and Data Engineering 15(4), 1004–1017 (2003)
11. Thathachar, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic Publishers, Dordrecht (2004)
12. Tsetlin, M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press, London (1973)
13. Narendra, K.S., Thathachar, M.A.L.: Learning automata: an introduction. Prentice-Hall, Inc., Englewood Cliffs (1989)
14. Lin, M., Wen, X.: Handling Relations in a Ubiquitous Computing Environments. Master's thesis, University of Agder, Norway (June 2009)
15. Mitzenmacher, M., Broder, A.: Survey: Network applications of bloom filters: A survey. Internet Mathematics 1 (2003)