DISTRIBUTED MINING OF SPATIO-TEMPORAL EVENT PATTERNS IN SENSOR NETWORKS*

Kay Römer
Institute for Pervasive Computing
ETH Zurich, Switzerland
roemer@inf.ethz.ch

Abstract Many sensor network applications are concerned with discovering interesting patterns among observed real-world events. Often, only limited apriori knowledge exists about the patterns to be found eventually. Here, raw streams of sensor readings are collected at the sink for later offline analysis – resulting in a large communication overhead. In this position paper, we explore the use of in-network data mining techniques to discover frequent event patterns and their spatial and temporal properties. With that approach, compact event patterns rather than raw data streams are sent to the sink. We also discuss various issues with the implementation of our proposal and report our experience with preliminary experiments.

Keywords: sensor networks, data mining, association rules, spatio-temporal event patterns

1. Introduction

Wireless sensor networks have been successfully applied for detailed observation of a variety of real-world phenomena. Many of these applications are of a highly exploratory nature, where only a very rough idea of the expected findings exists before the experiment. In this case, streams of raw sensor readings from every node are typically delivered to a central sink for later offline analysis [28] in order to find interesting patterns in the data – resulting in a large data volume that has to be delivered through the network (e.g., [27, 29]).

A similar approach is used in the context of monitoring and debugging sensor networks [24]. Experience has shown that subtle real-world influences and large scale are the cause of numerous bugs and indeterministic behavior of sensor networks. Again, since the nature of these problems is often unknown in advance, testbeds [10, 33] and deployment support networks [3] have been proposed to deliver high-volume event logs from every sensor node to a central sink for off-line analysis in order to identify patterns that lead to failure.

*The work presented in this paper was supported (in part) by the Swiss National Science Foundation under grant number 5005-67322 (NCCR-MICS).

In the above cases, missing advance knowledge about the patterns to be found eventually in the data limits the applicability of sophisticated in-network data processing and reduction techniques. Rather, raw data streams are delivered to the sink for later analysis. The resulting large data volume is a serious obstacle for deploying long-lived and large-scale sensor networks.

In this position paper, we explore the use of data mining techniques to discover potentially interesting data patterns in a sensor network for the above type of applications. Rather than transmitting raw data streams from every sensor to the sink, only compact patterns mined at sensor nodes are transmitted to the sink, thus contributing to long-lived and large-scale sensor network deployments.

To limit the potentially huge search space for the discovery of such interesting patterns, we apply a *constrained* form of data mining, where a user provides certain hints to the system on what kind of patterns to look for. Specifically, the user must define a limited set of *events* which are of potential interest. In a volcano monitoring application [32], for example, these events might be defined by the occurrence of seismic shocks, infra-sound shocks, excessive emission of certain indicative gases, or the occurrence of an eruption itself. The system will then discover frequent spatio-temporal patterns among these events, such as “within 20 minutes after an eruption event, there are frequent seismic shock events with high probability”, or “in the neighborhood of gas emission events, seismic events are likely to occur” etc.

To constrain the search space further, the user must additionally specify lower bounds for the frequency and confidence of patterns he is interested in. Also, the user is required to specify a maximal spatial and temporal scope. The system will then only consider and report patterns among the given events occurring within the spatial and temporal scopes with the given minimum frequency and confidence.

Before we can present our approach in detail in Sect. 1.3, we need to review some basic data mining techniques that we are going to apply.

2. Background: Mining of Association Rules

Our approach to discover event patterns is based on a data mining technique known as *association-rule mining* [1, 13] which was originally conceived to find patterns among items that are frequently shopped together such as *“if buy milk then also buy bread with high probability”*. More formally, we consider items $i \in \mathcal{I}$. We are given a database $\mathcal{D}$ of recorded transactions $T \subseteq \mathcal{I}$. That is, each transaction T represents the contents of a shopping basket at checkout. In the simplest case, we are then interested in finding rules of the form

$$I \Rightarrow i\,[S, C] \tag{1}$$

where $I \subseteq \mathcal{I}$ and $i \in \mathcal{I}$, meaning that if the items in I are in a basket, then item i is also likely to be in the basket. Here, “likely” is formally specified by support S and confidence C, where S is the percentage of the transactions in the database that contain $I \cup i$, and C is the percentage of the transactions containing $I \cup i$ among the transactions containing I. The user specifies minimum values $minsupp$ for S and $minconf$ for C, expressing interest only in association rules that satisfy these minimum values.

Numerous algorithms (e.g., [2, 12]) have been proposed to efficiently discover association rules from large databases of transactions, perhaps the most celebrated one being the so-called *apriori algorithm* [2]. The algorithm proceeds in two stages. First, it discovers all *frequent itemsets* $I_F \subseteq \mathcal{I}$, where an itemset is called frequent if it has at least support $minsupp$ in the database. For this, the algorithm first generates frequent 1-itemsets with $|I_F| = 1$. It then merges frequent 1-itemsets to obtain frequent 2-itemsets and so on until no more frequent itemsets can be found. The algorithm exploits the fact that a $k + 1$-itemset cannot be frequent if any of its subsets of size k is not frequent.

In the second stage, every frequent itemset I_F generated in the first phase is split in any possible way into a rule antecedent $I \subseteq \mathcal{I}$ and a rule consequent $i \in \mathcal{I}$ such that $I \cup i = I_F$ and $I \cap i = \emptyset$. For each such rule candidate $I \Rightarrow i$, the confidence is computed. The rule is output if the confidence is above $minconf$.

The apriori algorithm typically makes several passes over the whole database in order to generate frequent itemsets. This is impractical in the context of sensor networks as it implies that all data has to be stored somewhere. However, recently there has been a growing amount of work on discovering frequent itemsets from a *stream* of transactions such that every transaction is only considered once and can be deleted afterwards (e.g., [6, 9][15, 19, 21]).

Another issue of relevance for our work is the memory consumption of algorithms for mining association rules as we intend to run such algorithms in a sensor network. The memory footprint of these algorithms is typically linear in the number of frequent itemsets, which can be very large for small values of $minsupp$. To reduce the memory footprint, two approaches have been applied.

Firstly, techniques for approximate mining of frequent itemsets trade off memory for accuracy by discovering itemsets whose frequency is larger than $minsupp - \epsilon$ for a given error bound ϵ with memory footprint being a function of ϵ (e.g., [15, 19, 21]). Secondly, rather than discovering all frequent itemsets, well-defined subsets of the latter can be discovered. One example are *closed itemsets* [30, 34]. A closed itemset is a frequent itemset that has only proper supersets with smaller support than itself. The number of closed itemsets is often significantly smaller than the number of frequent itemsets. For small values of $minsupp$, the number of closed itemsets can still be very large. Here, one can resort to *maximal itemsets* [11, 16]. A maximal itemset is a frequent itemset that has no proper supersets which are frequent. The number of maximal itemsets is often again significantly smaller than the number of closed itemsets.

3. Mining Spatio-Temporal Event Patterns

Our approach is based on the observation that events in a sensor network that are somehow correlated often occur in spatial and/or temporal proximity. Depending on the actual application, "proximity" may be a small as a few meters and seconds or as large as kilometers and days. We assume that the user can specify upper bounds $maxscope$ (measured in meters or network hops) and $maxhistory$ (measured in seconds), expressing the fact that he is interested in patterns among events that occur within a distance not larger than $maxscope$ and within a time frame not larger than $maxhistory$.

Based on the given bounds, every sensor node in the network continuously collects event notifications occurring at nodes that are within a distance $maxscope$ of itself and keeps a history with size $maxhistory$ of these events. Details regarding event collection are discussed in Sect. 1.3.4 below. The node then executes a mining algorithm for discovering patterns among these collected events (Sect. 1.3.5). More specifically, every node n mines patterns of the following form

$$A_1 \wedge ... \wedge A_m \Rightarrow E\ [S, C] \tag{2}$$

meaning that an event of type E occurs at node n with support S and confidence C given that antecedents A_i all hold true. Every antecedent is of the form

$$A_i = (E_i, D_i, T_i, N_i) \tag{3}$$

meaning that A_i is true iff a certain type of event E_i occurred N_i times at a distance D_i from node n and T_i time units before E. D_i, T_i, and N_i usually denote intervals such as $T_i =$ *"more than five minutes ago"*, $D_i =$ *"less than 20 meters away"*, or $N_i =$ *"between one and five times"*. Details about these intervals are discussed in Sect. 1.3.3. We show in Sect. 1.3.5 below how this special type of association rules can be mined with standard association rule mining algorithms discussed in Sect. 1.2.

Every node in the network infrequently sends a subset of the discovered patterns to the sink along with the location of the node and the time frame during which these rules have been collected. As discussed in Sect. 1.3.6, the sink may perform data mining on these patterns in order to create a more global perspective – both with respect to time and space.

3.1 Preliminaries

We assume that all nodes in the sensor network share a common physical time and have knowledge of their locations. Time is divided into intervals of fixed length $epochlen$. The duration of an epoch is application specific and depends on the change rate of the measured physical quantities. Typical values for $epochlen$ are several tens of seconds. Time durations such as $maxhistory$ are then measured in integral numbers of epochs. A point in time is given by the number of epochs that have elapsed since startup.

Distances such as $maxscope$ can either be given as Euclidean distances or as a number of network hops. The former is typically used for reasoning about real-world events, while the latter is often more appropriate for monitoring aspects of the sensor network itself.

3.2 Events

In the context of our work, an event is simply an identifier such as "eruption" plus the epoch when it occurred and the location of the node where it was observed. In each epoch, a node can generate at most one instance of each event type. If there are more occurrences in one epoch, they are treated as a single instance. If this is not acceptable, the epoch duration must be reduced.

The generation of events is up to the application, which simply has to inform the mining system executing on the node of the occurrence of an event. In general, the generation of events will be either based on sensor output or on certain system parameters (e.g., low memory). It must be emphasized that sensor nodes are highly unreliable systems that can produce wrong sensor readings that would result in spurious events. Care must be taken when mining patterns among such events. In general, data cleaning methods [14] should be applied to reduce the probability of such false information. Note that these are not part of our system. However, in some cases our data mining approach could even be used to deliberately discover patterns among such spurious events, for example, to pinpoint possible causes.

3.3 Discretization

As we will see in Sect. 1.3.5, we need to discretize continuous quantities such as distance, time offset, or number of occurrences of an event in order to apply the pattern mining algorithm. Essentially, this means that we have to partition the domain of each continuous variable into a small set of non-overlapping intervals. The mining algorithm will then output a set of intervals in the form of antecedents as in Eq. 3. The partitioning has to be specified by a user as input to our system and is a critical issue as it affects the patterns that will be found eventually.

More specifically, a user has to specify a partitioning of distances, time offsets, and event frequency for every type of event. In practice, we expect that these partitionings will often be identical for some or all event types. For ease of exposition, we will assume in the remainder of this paper that partitionings are identical for all event types.

See below for a set of example partitions. Parenthesis indicate open interval ends and square brackets denote closed interval ends. Note that a partition can also consist of multiple non-continuous intervals. We will refer to this example in the remainder of the paper.

```
// distance (meters)
near: (0, 5m]
far: (5m, maxscope]

// time (epochs)
now: 0
recent: (0, 5]
old: (5, maxhistory]

// event frequency
none: 0
some: [1, infinity]
```

Below we will refer to the number of event types as $\#E$, to the number of distance partitions as $\#D$, to the number of time partitions as $\#T$, and to the number of frequency partitions as $\#N$ in accordance with the variable names in Eq. 3.

3.4 Event Distribution

As stated earlier, each node in the network collects event notifications from nodes in a neighborhood of size $maxscope$. This can be achieved by applying a framework that supports neighborhood abstractions such as Abstract Regions [31] or Logical Neighborhoods [22]. These tools allow a node in the network to define a *neighborhood* that consists of a set of nodes that fulfill certain conditions such as to be within a given distance of the node. Primitives for collecting data from the nodes in a neighborhood are provided. Using the discretization of distance discussed in the previous section, in-network data aggregation is applied to collect the frequency of each event for each distance partition in the neighborhood of the node. Using the partitions from Sect. 1.3.3, we would count the occurrence of each event at distances “near” and “far”. We need $\#E \times \#D$ integer variables to collect the information for one epoch from a neighborhood that contains an arbitrary number of nodes. We will refer to such a set of variables as an *event summary*.

An important issue is how often to collect event notifications. Since we are interested in event patterns over longer periods of time, it is typically not necessary to collect event summaries once per epoch. Rather, it often pays off to collect event summaries for a number of epochs using a single message.

3.5 Local Pattern Mining

We show how our mining problem can be transformed into a traditional item-based representation which is suitable for itemset association rule mining algorithms discussed in Sect. 1.2. In our description, we will take on the perspective of a node in the network that executes the mining algorithm at a certain epoch. We will refer to this as *the node* and *the epoch*.

Essentially, we will create one transaction $T \subseteq \mathcal{I}$ for every epoch. $\mathcal{I}$ contains two types of items: $i(e)$ and $j(e, d, t, n)$. Item $i(e)$ is contained in the transaction iff event e occurred at the node during the epoch. That is, we have $\#E$ distinct items of this type.

Item $j(e, d, t, n)$ is contained in a transaction iff event e occurred n times in the neighborhood of the node at distance d and time offset t. Here, d, t, n refer to partitions for distance, time, and frequency, respectively, as described in Sect. 1.3.3. For example, item $j(\mathrm{eruption}, \mathrm{near}, \mathrm{now}, \mathrm{none})$ is contained in the transaction if no occurrence of event “eruption” was recorded during the epoch at nodes closer than 10 meters. In other words, we have $\#E \times \#D \times \#T \times \#N$ distinct items of this type.

To generate the transactions, every node collects event summaries for every epoch as described in Sect. 1.3.4 and stores the event summaries of the last $maxhistory$ epochs in an internal table with one row per epoch. As discussed in Sect. 1.3.4, every event summary contains one integer variable for each pair of event type and distance partition, resulting in a table with $\#E \times \#D$ columns.

For every epoch, a copy of this table is created. The rows of this copied table are now grouped and summed up column-wise according to the given time partitions. For example, for the time partition "recent" we would sum up rows 1 through 5. In the resulting grouped table with $\#T$ rows, the contents of every cell (an event frequency) is now converted to the respective frequency partition identifier.

In the resulting table, there is one column for each pair (e, d) of event and distance partition and one row for every time partition t. We now enumerate all cells (e, d, t) containing n in the table and include the respective items $j(e, d, t, n)$ in the transaction. In addition, we include an item $i(e)$ for each event e that occurred at the node during the epoch.

The resulting stream of transactions (one for each epoch) is then fed to an algorithm for mining frequent, closed, or maximal itemsets. The resulting itemsets are converted to association rules as described in Sect. 1.2, resulting in rules of the form given in Eq. 2.

Note that there are several optimizations that can be applied to reduce the number of itemsets. For example, we can prune all itemsets that do not contain exactly one item of type $i(e)$. Also, all itemsets that contain $j(e, d, t, n)$ and $j(e, d, t, n')$ with $n \neq n'$ for any e, d, t are invalid and can be pruned since frequency partitions must be disjoint. We will examine the number of resulting itemsets in more detail in Sect. 1.4.

3.6 Global Patterns

All nodes in the network infrequently report discovered patterns or subsets thereof (e.g., ranked by $S \times C$) to the sink along with the location of the reporting node and the time interval during which these patterns were discovered. Note that a pattern can be efficiently represented as a bit vector of items plus support and confidence values. Using the reverse of the mapping described in the previous section, the sink can reconstruct patterns in the form of Eq. 2.

The sink may now perform a secondary mining procedure over the reported patterns [26] in order to construct a more global picture. For spatial integration, the sink may apply clustering techniques on the rules to identify sets of nodes that discovered similar rules. For temporal integration, the sink can discover rules that do not change over time or segment time into partitions during which the rules do not change significantly [4].

4. Implementation Issues

Clearly the major challenge in implementing our proposal is that of fitting the mining approach described in the previous section into the constrained communication, computational, and memory resources of a sensor node. In this section, we point out and discuss the major challenges in this regard.

With respect to communication overhead, we mainly have to consider the event distribution scheme in Sect. 1.3.4. Similar schemes for in-network data collection and aggregation have been successfully implemented before [20], so we can be confident that the proposal in Sect. 1.3.4 can be implemented. However, in previous applications, the sink collects data from the whole network, whereas in our case, every node in the network collects aggregated data from its neighborhood (with limited size). In other words, there are many overlapping neighborhoods where event collection is being performed concurrently, leading to interesting scheduling and aggregation issues.

With respect to computational and memory overheads, we mainly have to consider the local pattern mining approach described in Sect. 1.3.5 and specifically the approach for mining frequent itemsets discussed in Sect. 1.2.

Regarding computational overhead, we are considering itemset-mining algorithms that operate on data streams rather than on traditional databases. Hence, an important point here is the amount of processing required to deal with a newly arriving transaction. The required amount of processing implies an upper bound on the arrival rate of new transactions and thus also a lower bound on the duration of an epoch. However, since typical epoch durations in sensor networks are in the order of several seconds, we expect that our proposal can be actually implemented on sensor nodes with respect to computational overhead.

With respect to memory footprint, the critical part in our proposal is the algorithm used for the discovery of frequent itemsets. As discussed in Sect. 1.2, the memory footprint of such algorithms is largely a function of the number of the itemsets discovered from the data stream. Hence, we will examine the number of such itemsets below. First, we will derive some bounds on the number of itemsets for a specific problem instance. We will then turn to some preliminary experiments on traces of real sensor data collected at Intel Research Lab Berkeley [35].

From the description in Sect. 1.3.5, the number of different items in an itemset follows immediately:

$$|\mathcal{I}| = \#E \times (1 + \#T \times \#D \times \#N) \tag{4}$$

This also means that each itemset can be represented by a bit vector of size $|\mathcal{I}|$. For $\#E = 2$ types of events and with the example partitions from Sect. 1.3.3, we have $|\mathcal{I}| = 26$ bits or 4 bytes.

Based on this, one could expect $2^{|\mathcal{I}|}$ possible different itemsets. However, as discussed at the end of Sect. 1.3.5, we are only interested in a subset of these itemsets. In particular, we are only interested in itemsets that contain exactly one local event $i(e)$. Also, all itemsets that contain $j(e, d, t, n)$ and $j(e, d, t, n')$ with $n \neq n'$ for any e, d, t are invalid and can be pruned. One can easily verify that this results in the following number of different itemsets:

$$\#E \times (1 + \#N)^{\#T \times \#D \times \#E} \tag{5}$$

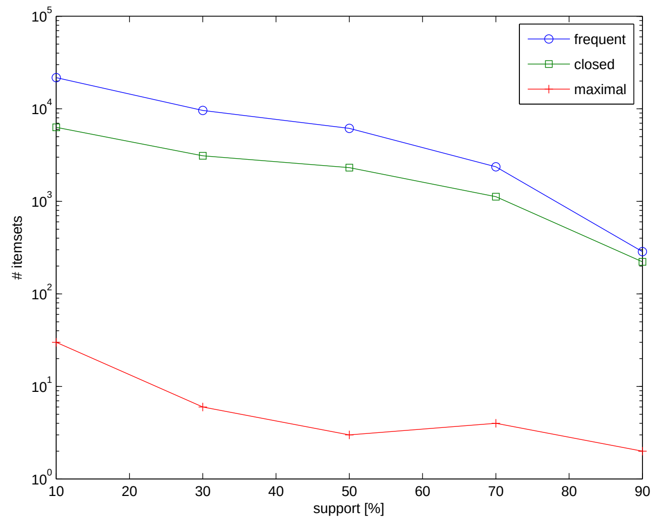

Figure 1. Number of frequent, closed, and maximal itemsets for different support values.

For the above example, we obtain about one million different itemsets. However, the number of *frequent* (or *closed* or *maximal*) itemsets can be expected to be much smaller in practice. To verify this, we performed an experiment using sensor data collected during one month from 54 sensor nodes in the Intel Research Lab Berkeley [35]. This dataset was collected with an epoch duration of about 30 seconds (resulting in a total of about 65000 epochs) and contains, among others, temperature and light readings.

In our experiment, we consider two types of events: *temperature* and *light* events. Each sensor node with a temperature reading > 23 degrees Celsius in an epoch emits a temperature event in this epoch. Every sensor node with a light reading > 300 Lux emits a light event. Otherwise, we use the partition definitions from the example given in Sect. 1.3.3 with $maxscope = 10$ meters and $maxhistory = 10$ epochs.

For our experiment, we consider the mining algorithm that would be executed on mote with ID 1 in the dataset. With the above settings, mote 1 generates a temperature event in about 23% of all epochs and a light event in about 14% of all epochs. We would expect a strong correlation of the occurrence of these events on mote 1 with the light and temperature events in the neighborhood of the mote.

Applying the method given in Sect. 1.3.5, we create a stream of transactions which is then fed to an apriori algorithm to discover frequent, closed, and maximal itemsets for different values of $minsupp$. In contrast to the definition of support given in Sect. 1.2, we use a different notion of support here to eliminate the dependence on the absolute number of transactions in the stream that contain an item of type $i(e)$. In particular, for calculation of the support of an itemset containing $i(e)$, we do only consider transactions in the stream that do also contain item $i(e)$ (rather than the number of all transactions in the data stream).

The results shown in Figure 1.4 are encouraging as the number of maximal itemsets is very small over the whole range of considered support values and the number of closed (and even) frequent itemsets is small enough for larger support values to handle them on a sensor node. Recall that a single itemset in this experiment can be represented with 26 bits.

For $minsupp = 90\%$ we obtain two maximal itemsets that result in the following patterns for node 1 in the format of Eq. 2:

$$(\mathrm{t}, \mathrm{now}, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{t}, \mathrm{recent}, *, \mathrm{some}) \wedge (\mathrm{t}, \mathrm{old}, *, \mathrm{some}) \Rightarrow t\ [96\%, 38\%]$$
$$(\mathrm{t}, *, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{l}, \mathrm{now}, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{l}, \{\mathrm{old}, \mathrm{recent}\}, *, \mathrm{some}) \Rightarrow l\ [92\%, 32\%]$$

Here, “t” and “l” refer to temperature and light events as defined above. The notations “{...,...}” and “*” mean that the enclosing antecedent is valid for the set of given partition identifiers or for all possible partition identifiers, respectively. The rules indicate that – as expected – the occurrence of temperature/light events at mote 1 is correlated with the occurrence of these events in the neighborhood of the node.

5. Related Work

There is a large amount of related work in the areas of distributed data mining [23], spatial data mining [8], and temporal data mining [25]. In particular, distributed algorithms for mining association rules have been been devised (e.g., [23]). However, these assume that the set of transactions is distributed in the network. In our case, however, every single transaction is itself generated from distributed sources. Hence, these algorithms are not directly applicable here. Also, centralized algorithms for mining spatial (e.g., [18]) and temporal (e.g., [5]) association rules have been devised.

Recently, techniques from distributed data mining have also been applied, adopted, or specifically developed for sensor networks with constrained and unreliable resource availability [17]. We are not aware, however, of any proposals for in-network discovery of spatio-temporal event patterns in sensor networks.

In a more general context, there is a growing interest in applying data mining techniques to sensor data [28]. However, most of the existing techniques operate on a centralized data set rather than providing mechanisms for in-network mining.

6. Conclusions and Future Work

In this position paper we have examined the use of data mining techniques to discover frequent event patterns and their spatio-temporal relationships within a sensor network, such that compact patterns rather than raw data streams would have to be transmitted from nodes to the sink. In particular, such a system would be helpful to support exploratory settings, where only a rough idea of the actual findings exists before the experiment.

We have also discussed the challenges in implementing this proposal on sensor nodes. In particular, we have identified the memory consumption of itemset discovery algorithms and the communication overhead of event collection as major issues. With respect to the former, we have performed an experiment with real-world data to motivate that an implementation is feasible. Besides the above issues, there are a number of open question that have to be addressed, such as a specification language for formulating "mining queries", or mechanisms for finding global patterns and making them accessible to humans. Last but not least, the value of the proposal for concrete, large-scale, and complex applications has to be studied.

References

[1] R. Agrawal, T. Imielinsko, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *VLDB 1993*, Dublin, Ireland, August 1993.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *VLDB 1994*, Santiago de Chile, Chile, September 1994.

[3] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald. Next-Generation Prototyping of Sensor Networks. In *Sensys 2004*, Baltimore, USA, November 2004.

[4] S. Chakrabarti, S. Sarawagi, and B. Dom. Mining Surprising Patterns Using Temporal Description Length. In *VLDB 1998*, New York City, USA, August 1998.

[5] X. Chen, I. Petrounias, and H. Heathfield. Discovering Temporal Association Rules in Temporal Databases. In *IADT 1998*, Berlin, Germany, July 1998.

[6] W. Cheung and O. R. Zaiane. Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraints. In *IDEAS 2003*, Hong Kong, China, July 2003.

[7] Y. Chi, H. Wang, P. S. Yu, and R. M. Muntz. Moment: Mainatining Closed Frequent Itemsets over a Stream Sliding Window. In *ICDM 2004*, Brighton, UK, November 2004.

[8] M. Ester, H.-P. Kriegel, and J. Sander. Algorithms and Applications for Spatial Data Mining. In H. J. Miller and J. Han, editors, *Geographic Data Mining and Knowledge Discovery*. 2001.

[9] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In *NSF Workshop on Next Generation Data Mining 2002*, Baltimore, USA, November 2002.

[10] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer. A System for Simulation, Emulation, and Deployment of Heterogeneous Sensor Networks. In *Sensys 2004*, Baltimore, USA, November 2004.

[11] K. Gouda and M. J. Zaki. Efficiently Mining Maximal Frequent Itemsets. In *ICDM 2001*, San Jose, USA, November 2001.

[12] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *SIGMOD 2000*, Dallas, USA, May 2000.

[13] J. Hipp, U. Guetzer, and G. Nakhaeizadeh. Algorithms for Association Rule Mining - A General Survey and Comparison. *SIGKDDEx*, 2(1):58–64, 2000.

[14] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative Support for Sensor Data Cleaning. In *Pervasive 2006*, Dublin, Ireland, May 2006.

[15] R. Jin and G. Agrawal. An Algorithm for In-Core Frequent Itemset Mining on Streaming Data. In *ICDM 2005*, New Orleans, USA, November 2005.

[16] R. J Bayardo Jr. Efficiently Mining Long Patterns from Databases. In *SIGMOD 1998*, Seattle, USA, June 1998.

[17] H. Kargupta. Distributed Data Mining for Sensor Networks (Tutorial). In *ECML/PKDD 2004*, Pisa, Italy, September 2004.

[18] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Systems. In *SSD 1995*, Portland, USA, August 1995.

[19] H.-F. Li, S.-Y. Lee, and M.-K. Shan. An Efficient Algorithm for Mining Frequent Itemsets over the entire History of Data Streams. In *First Intl. Workshop on Knowledge Discovery in Data Streams 2004*, Pisa, Italy, September 2004.

[20] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *OSDI 2002*, Boston, USA, December 2002.

[21] G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. In *VLDB 2002*, Kong Kong, China, August 2002.

[22] L. Mottola and G. P. Picco. Programming Wireless Sensor Networks with Logical Neighborhoods. In *INTERSENSE 2006*, Nice, France, May 2006.

[23] B.-H. Park and H. Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. In N. Ye, editor, *Data Mining Handbook*. 2002.

[24] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the Sensor Network Debugger. In *Sensys 2005*, San Diego, USA, November 2005.

[25] J. F. Rodick and M. Spiliopoulou. A Survey of Temporal Knowledge Discovery Paradigms and Methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750–767, July 2002.

[26] M. Spiliopoulou and J. F. Roddick. Higher Order Mining: Modelling and Mining the Results of Knowledge Discovery. In *Data Mining 2000*, 2000.

[27] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a Sensor Network Expedition. In *EWSN 2004*, Berlin, Germany, January 2004.

[28] P.-N. Tan. Knowledge Discovery from Sensor Data. *Sensors*, 23(3):14–19, March 2006.

[29] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A Macroscope in the Redwoods. In *Sensys 2005*, San Diego, USA, November 2005.

[30] J. Wang, J. Han, and J. Pei. Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *SIGKDD 2003*, Washington, USA, August 2003.

[31] M. Welsh and G. Mainland. Programming Sensor Networks Using Abstract Regions. In *NSDI 2004*, Boston, USA, March 2004.

[32] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network. In *EWSN 2005*, Istanbul, Turkey, January 2005.

[33] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: A Wireless Sensor Network Testbed. In *IPSN/SPOTS 2005*, Los Angeles, USA, April 2005.

[34] M. J. Zaki and C. Hsiao. An Efficient Algorithm for Closed Itemset Mining. In *ICDM 2002*, Maebashi City, Japan, December 2002.

[35] Intel Lab Sensor Data. http://berkeley.intel-research.net/labdata/.