ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/242367808

Mining Spatio-Temporal Association Rules, Sources, Sinks, Stationary Regions and Thoroughfares in Object Mobility Databases Technical Report 574

Article

CITATIONS
12

READS
14

2 authors:

Florian Verhein
Ludwig-Maximilians-University of Munich
15 PUBLICATIONS 406 CITATIONS
SEE PROFILE

Sanjay Chawla
Qatar Computing Research Institute
165 PUBLICATIONS 2,794 CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Urban Complex Systems View project

Project Anomaly Detection using Deep Learning Techniques View project

All content following this page was uploaded by Sanjay Chawla on 02 February 2017.

The user has requested enhancement of the downloaded file.

The University of Sydney

Mining Spatio-Temporal Association Rules, Sources, Sinks, Stationary Regions and Thoroughfares in Object Mobility Databases

Technical Report Number 574

October 2005

Florian Verhein and Sanjay Chawla

ISBN 1 86487 754 4

School of Information Technologies
University of Sydney NSW 2006

Mining Spatio-Temporal Association Rules, Sources, Sinks, Stationary Regions and Thoroughfares in Object Mobility Databases

Technical Report 574

Florian Verhein and Sanjay Chawla

School of Information Technologies, University of Sydney, NSW, Australia
{fverhein, chawla}@it.usyd.edu

Abstract. As mobile devices proliferate and networks become more location-aware, the corresponding growth in spatio-temporal data will demand analysis techniques to mine patterns that take into account the semantics of such data. Association Rule Mining (ARM) has been one of the more extensively studied data mining techniques, but it considers discrete transactional data (supermarket or sequential). Most attempts to apply this technique to spatial-temporal domains maps the data to transactions, thus losing the spatio-temporal characteristics. We provide a comprehensive definition of *spatio-temporal association rules (STARs)* that describe how objects move between regions over time. We define *support* in the spatio-temporal domain to effectively deal with the semantics of such data. We also introduce other patterns that are useful for mobility data; *stationary regions* and *high traffic regions*. The latter consists of *sources*, *sinks* and *thoroughfares*. These patterns describe important temporal characteristics of regions and we show that they can be considered as special STARs. We provide efficient algorithms to find these patterns by exploiting several pruning properties.

1 Introduction

The need for spatio-temporal data mining and analysis techniques is growing. Some specific examples include managing cell phone networks and dealing with the data generated by RFID tags. Mining such data could detect patterns for applications as diverse as intelligent traffic management, sensor networks, stock control and wildlife monitoring. For example, consider the movement of users between cells of a mobile phone (or similar) network. Being able to predict where users will go would make cell hand-over decisions easier and improve bandwidth management. Also, since most people own a mobile phone these days, the data could be used for fast and inexpensive population movement studies. Local governments would find the ability to answer questions such as “how much is this park being used?”, “which areas are congested?”, and “what are the main routes that people take through the city” useful. The latter query would help design better pedestrian and vehicle routes to take into account the main flows of people.

We therefore consider a set of regions, which may be any shape or size, and a set of objects moving throughout these regions. We assume that it is possible to determine which objects are in a region, but we do not know precisely where an object is in that region. We do not assume that objects are always somewhere in the region set, so in the example of a mobile phone network, turning the phone off poses no problems for our methods. We are interested in finding regions with useful temporal characteristics (thoroughfares, sinks, sources, and stationary regions) and rules that predict how objects will move through the regions (spatio-temporal association rules). A source occurs when the number of objects leaving a region is high enough. A sink has a high number of objects entering it. A thoroughfare is a region through which many objects move - that is, many objects enter and leave. A stationary region is where many objects tend to stay over time, while a STAR describes how an object moves between regions. Together, these patterns describe many mobility characteristics and can be used to predict future movements.

We take the approach of mining our patterns on a time window by time window basis. We think this is important because it allows us to see the changing nature of the patterns over time, and allows for interactive mining - including changing the mining parameters. Even though the patterns we consider occur in a spatial settings, they are all temporal patterns because they describe objects movements *over time*, as well as capturing *changes* in the way the objects move over time. To understand this, consider each pattern set ξ_i as capturing object movements over a 'short' period of time. In our algorithms this is the interval pair $[TI_i, TI_{i+1}]$. That is, ξ_i captures how the objects move between the time intervals TI_i and TI_{i+1}. Then, as the algorithm processes subsequent time intervals, the patterns mined at these points will in general change, forming a *sequence* of pattern sets $< \xi_i, \xi_{i+1}, ... >$. This change in the *patterns* that are output can be considered longer term changes. Such changes in the patterns describe the changes in the objects behavior over time. Another way to think about this is to consider the objects motion as a random process. If the process is stationary, we would expect the patterns to remain the same over time. If the process is not stationary, the patterns will change with time to reflect the change in the way the objects move.

There are a number of challenges when mining spatio-temporal data. First, dealing with the interaction of space and time is complicated by the fact that they have different semantics. We cannot just treat time as another spatial dimension, or vice versa. For example, time has a natural ordering while space does not. Secondly, we also need to deal with these spatio-temporal semantics effectively. This includes considering the effects of area and the time-interval width not only on the the patterns we mine, but also in the algorithms that find those patterns. Finally, handling updates efficiently in a dynamic environment is challenging - especially when the algorithm must be applied in real time. We adopt a *data stream* model where spatial data arrives continuously as a sequence of snapshots $S_1, S_2, ...$, and the model that we mine must keep up with this. The algorithms must therefore perform only a single pass in the temporal dimension. That is, the algorithm must not revisit S_i once it has started processing $S_i + 1$ - this means that the model must be incrementally update-able. Unless sampling techniques are used, such algorithms cannot do better than scale linearly with time. Since processing the spatial snapshots is expensive in general, we focus our attention there. We deal with exact techniques in this paper, but it is possible to use probabilistic counting techniques together with our algorithms, as demonstrated in one of our experiments.

2 Contributions

We make the following contributions:

- We give a rigorous definition of Spatio-Temporal Association Rules (STARs) that preserve spatial and temporal semantics. We define the concepts of *spatial coverage*, *spatial support*, *temporal coverage* and *temporal support*. Because these definitions retain the semantics of spatial and temporal dimensions, it allows us to mine data with regions of any size without skewing the results. That is, we successfully extend association rules to the spatio-temporal domain.
- We define useful spatio-temporal regions that apply to objects moving through such regions. These are *stationary regions* and *high traffic regions*. The latter may be further broken into *sources*, *sinks* and *thoroughfares*. We stress that these are temporal properties of a spatial region set, and show that they are special types of STARs. We also describe a technique for mining these regions efficiently by employing a pruning property.
- We propose a novel and efficient algorithm for mining the STARs by devising a pruning property based on the high traffic regions. This allows the algorithm to prune as much of the search space as possible (for a given dataset) before doing the computationally expensive part. If the set of regions is R, we are able to significantly prune R (to $A \subset R$ and $C \subset R$) resulting in a running time of $O(|R|) + O(|A' \times C'|)$ instead of $O(|R|^2)$, where $A' = A - A \cap C$ and $C' = C - A \cap C$. Our experiments show that this is a significant saving. Theoretically, it is also the most pruning possible without missing rules.

Our algorithms do not assume or rely on any form of index (such as an R-tree, or aggregated R-tree) to function or to obtain time savings. If such an index is available, the algorithms will perform even better. Our time savings come about due to a set of pruning properties, which are spatial in nature, based on the observation that only those patterns that have support above a threshold are interesting to a user (in the sense that they model the data).

The rest of the paper is organized as follows. In Section 3 we survey related work and place our contributions in context. In Section 4 we give several related definitions of STARs that highlight some of the differences in interpreting STARs. We then tackle the problem of extending support to the spatio-temporal domain. In Section 5 we define hot spots, stationary regions and high traffic regions. In Section 6 we describe our algorithm for mining STARs. The results of our experiments on STAR mining are described in Section 7. We conclude in Section 8 with a summary and directions for future work. The appendix contains proofs of the theorems we exploit.

3 Related Work

There has been work on Spatial association rules (examples include [1, 2]) and temporal association rules (examples include [3, 4]) but very little work has addressed both spatial and temporal dimensions. Most of the work that does can be categorised as traditional association rule mining [5] or sequential ARM *applied* to a spatio-temporal problem, such as in [6].

The work by Tao et al. [7] is the only research found that addressed the problem of STARs (albeit briefly) in the Spatial-Temporal domain. As an application of their work they show a brute force algorithm for mining specific spatio-temporal association rules. They consider association rules of the form $(r_i, \tau, p) \Rightarrow r_j$, with the following interpretation: "If an object is in region r_i at some time t, then with probability p it will appear in region r_j by time $t + \tau$". Such a rule is aggregated over all t in the following way: if the probability of the rule occurring at any fixed t is above p, a counter in incremented. If the fraction of such occurrences is over another threshold c, the rule is considered important and output. The authors call p the *appearance probability*, and c the *confidence factor*. They do not discuss the reasons for or the consequences of this choice. The *confidence factor* is really the support with respect to time of the rule, and when interpreted in the traditional sense, p is really the *confidence threshold* of the rule. There is also no support defined. That is, the number of objects for which the rule applies is ignored. For each time-stamp, their algorithm examines each pair of regions in turn, and counts the number of objects that move between the regions. Their algorithm is a brute force technique that takes time quadratic in the number of regions. They use sketches (FM-PCSA) for speed, have a very simple STAR definition and ignore the spatial and temporal semantics of the data (such as the area of the regions or the time interval width).

Other interesting work that deals with spatio-temporal patterns in the spatio-temporal domain includes [8–11, 7]. Mamoulis et al. [11] mine periodic patterns in objects moving between regions. Wang et al. [9] introduce what they call *flow patterns*, which describe the changes of events over space and time. They consider events occurring in regions, and how these events are connected with changes in neighbouring regions as time progresses. So rather than mining a sequence of events in time, they mine a sequence of events that occur in specific regions over time and include a neighbourhood Ishikawa et al. [10] describe a technique for mining object mobility patterns in the form of markov transition probabilities from an indexed spatio temporal dataset of moving points. In this case, the transition probability p_{ij} of an (order 1) markov chain is $P(r_j | r_i)$ where r_i and r_j are regions, which is the confidence of a spatio-temporal association rule, although this is not mentioned by the authors. Tsoukatos et al. [8] mine frequent sequences of non spatio-temporal values for regions.

The work we have listed above is quite different from ours. Tao et al. [7] considers a simple spatio-temporal association rule definition, and the algorithm for finding the rules is brute force. Patterns that can be interpreted as STARs are considered by [10, 9], but they focus on very different research problems. The algorithm of [10] will find all transition probabilities, even if they are small. Amongst other things, our algorithm makes use of the fact that users will not be interested in rules below a support threshold, and uses this to prune the search space. And most importantly, none of the related work consider the spatial semantics of the regions, such as area, nor do they consider spatial support or similar concepts.

Dealing with the area of regions correctly is important for interpretation of the results. Many authors implicitly assume that the spatial regions can be specified to suit the algorithm. However, this is usually not the case. Cells in a mobile phone network are fixed, and have a wide range of sizes and geometries depending on geographic and population factors. Data mining applications have to be developed to work with the given region set, and we cannot ignore the influence of different sized regions. In the case of mining mobility patterns of moving objects (including sources, sinks, stationary regions, thoroughfares and STARs), ignoring area will skew the results in favour of larger regions because they will have more objects in them on average. By taking the region sizes into account, we avoid skewing the results and make our STARs comparable across different sized regions. Finally, although it is possible to scale most patterns by the area after they have been mined, this is undesirable because it it prevents pruning of the search space. Our algorithms deal with the spatio-temporal semantics such as area effectively throughout the mining process and prune the search space as much as possible.

No previous work could be found, despite our efforts, that considers sources, sinks, stationary regions and thoroughfares. We think these patterns are very important because they capture *temporal* aspects of the way that objects move in space.

4 Spatio-Temporal Association Rules

Given a dataset T of spatio-temporal data, define a language L that is able to express properties or groupings of the data (in both time, space, and object attributes). Given two sentences $\varphi_1 \in L$ and $\varphi_2 \in L$ that have no common terms, define a spatio-temporal association rule as $\varphi_1 \Rightarrow \varphi_2$. For example, the rule "late shift workers head into the city in the evening" can be expressed as $LateShiftWorker(x) \wedge InRegion(OutsideCity) \wedge Time(Evening) \Rightarrow InRegion(City) \wedge Time(Night)$. To evaluate whether such a spatio-temporal rule is interesting in T, a selection predicate $q(T, \varphi_1 \Rightarrow \varphi_2)$ maps the rule to $\{true, false\}$. The selection predicate will in general be a combination of support and confidence measures. For example, if the support and confidence of a rule R_1 are above their respective thresholds, then $q(T, R_1)$ evaluates to true.

The language L can be arbitrarily complex. We consider the special case where objects satisfying a query move between regions. A query q allows the expression of predicates on the set of non spatio-temporal attributes of the objects. We explore a number of definitions of such STARs in this section to highlight subtleties. We deal only with the STAR of Definition 2 outside this section, so the reader can safely focus on this on the first reading, without missing the main ideas of the paper.

Definition 1. STAR (alternatives): *Objects in region r_i satisfying q at time t will:*
(a) **appear** *in region r_j* **for the first time at** *time* $t + \tau$. Notation: $(r_i, t, @\tau, q) \Rightarrow r_j$.
(b) **be in region** r_j **at time** $t + \tau$. Notation: $(r_i, t, \tau, q) \Rightarrow r_j$.
(c) **appear** *in region r_j* **by (at or before) time** $t + \tau$. Notation: $(r_i, [t, \tau], q) \Rightarrow r_j$.

Note that (a) distinctly defines the time in r_j at which the objects must arrive. (b) is less rigid and allows objects that arrived earlier than time $t + \tau$ to be counted as long as they are still present *at* time $t + \tau$. (c) counts the objects as long as they have made an appearance in r_j at any time within $[t, t + \tau]$. We generalise (c) in our final definition:

Definition 2. STAR: *Objects appearing in region r_i satisfying q during time interval TI_s will appear in region r_j during time interval TI_e, where $TI_s \cap TI_e = \emptyset$ and TI_s is immediately before*[1] TI_e. Notation: $(r_i, TI_s, q) \Rightarrow (r_j, TI_e)$.

Note that all the definitions are equivalent when $TI_s = t$, $TI_e = t + 1$ and $\tau = 1$. We are interested in the rules that have a high confidence and high support . We will use the notation $r_i \Rightarrow r_j$ or ζ for a STAR when we are not concerned with its exact definition. We will consider the problem of more rigorous support definitions that are more appropriate in a spatio-temporal setting later.

Definition 3. *Define* **support** *of a rule ζ, denoted by $\sigma(\zeta)$, to be the number of objects that follow the rule, and the support (with respect to q) of a region r during TI, denoted by $\sigma(r, TI, q)$, to be the number of distinct objects within r during TI satisfying q.*

Definition 4. *Define the* **confidence** *of a rule ζ whose antecedent contains region r_i during TI, denoted by $c(\zeta)$, as the conditional probability that the consequent is true given that the antecedent is true. This is the probability that the rule holds and is analogous to the traditional definition of confidence and is given by $c(\zeta) = \sigma(\zeta)/\sigma(r_i, TI)$.*

We illustrate the above definitions with an example.

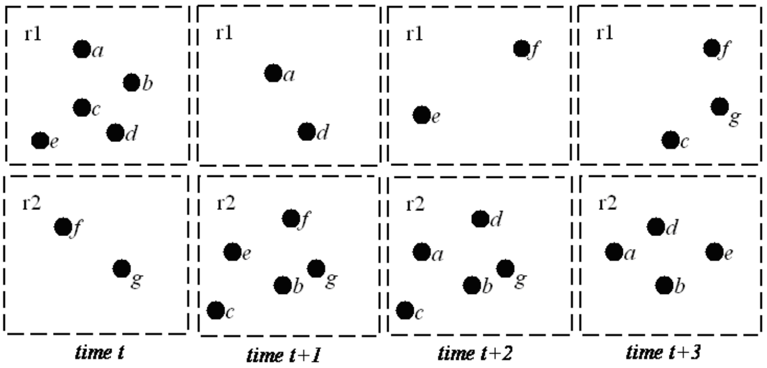


Fig. 1. Example data for spatio-temporal association rule mining. See example 1.

Example 1. Consider Figure 1 which shows the movement of the set of objects $S = \{a, b, c, d, e, f, g\}$ in the time-frame $[t, t + 3]$ captured at the four snapshots $t, t + 1, t + 2, t + 3$. Assume that $q = `true'$ so that all objects satisfy the query.

Consider the STAR $\zeta = (r_1, t, @1, q) \Rightarrow r_2$. From the diagram, $\{b, c, e\}$ follow this rule, so the support of the rule is $\sigma(\zeta) = 3$. Since the total number of objects that started in r_1 is $5 = \sigma(r_1, t) = |\{a, b, c, d, e\}|$, the confidence of the rule is $c(\zeta) = \frac{3}{5}$. For $\zeta = (r_1, t, @2, q) \Rightarrow r_2$ we have $\sigma(\zeta) = 2$ because $\{a, d\}$ follow the rule, and

[1] That is, there does not exist a time-stamp t that is *between* the time intervals in the sense that $(t < t_e \forall t_e \in TI_e) \wedge (t > t_s \forall t_s \in TI_s)$.

$c(\zeta) = \frac{2}{5}$. For $\zeta = (r_1, t, @3, q) \Rightarrow r_2$ we have $\sigma(\zeta) = 0$ because no object appears in r_2 for the first time at time $t + 3$.

The STAR $\zeta = (r_1, t, 1, q) \Rightarrow r_2$ is equivalent to $\zeta = (r_1, t, @1, q) \Rightarrow r_2$. But for $\zeta = (r_1, t, 2, q) \Rightarrow r_2$ we have $\sigma(\zeta) = 4$ because $\{a, b, c, d\}$ follow the rule (for this STAR definition we count them as long as they are still there at time $t + 2$), and $c(\zeta) = \frac{4}{5}$. For $\zeta = (r_1, t, 3, q) \Rightarrow r_2$ we have $\sigma(\zeta) = 4$ since $\{a, b, d, e\}$ follow the rule (we don't count c because it is no longer in r_2 at time $t + 3$), and $c(\zeta) = \frac{4}{5}$.

$(r_1, [t, 1], q) \Rightarrow r_2 = (r_1, t, 1, q) \Rightarrow r_2 = (r_1, t, @1, q) \Rightarrow r_2$. For $\zeta = (r_1, [t, 2], q) \Rightarrow r_2$ we have $\sigma(\zeta) = 5$ because $\{a, b, c, d, e\}$ satisfy the rule. e satisfies even though it has left by $t + 2$. Since all objects from r_1 have made an appearance in r_2 by $t + 2$ we must have $\sigma((r_1, [t, k], q) \Rightarrow r_2) = 5$ for all $k \geq 2$. For $\zeta = (r_1, [t + 1, 1], q) \Rightarrow r_2$ we have $\sigma(\zeta) = 2$ and $c(\zeta) = \frac{2}{2} = 1$

The STAR $\zeta = r_1, [t, t], q) \Rightarrow (r_2, [t + 1, t + k]$ is equivalent to $(r_1, [t, k], q) \Rightarrow r_2$ for $k \geq 1$. For the STAR $\zeta = r_1, [t, t + 1], q) \Rightarrow (r_2, [t + 2, t + 3]$ we have 5 distinct objects ($\{a, b, c, d, e\}$) appearing in r_1 during $[t, t + 1]$ and 6 distinct objects ($\{a, b, c, d, e, g\}$) appearing in r_2 during $[t + 2, t + 3]$. The objects following the rule are $\{a, b, c, d, e\}$ so the support of the rule is 5 and its confidence is $\frac{5}{5} = 1$. For $\zeta = r_1, [t + 1, t + 2], q) \Rightarrow (r_2, [t + 3]$ we have $\sigma(\zeta) = 3$ and $c(\zeta) = \frac{3}{4}$.

Counting the objects that move between regions is a simple task. The main idea is that if S_1 is the set of objects in r_1 at time t and S_2 is the set of objects in r_2 at time $t + 1$ then the number of objects moving from r_1 to r_2 during that time is $|S_1 \cap S_2|$ (assuming objects don't appear in more than one region at a time).

4.1 Extending Support into the Spatio-Temporal Setting

Defining support in a spatio-temporal setting is more complicated than we have considered so far. Specifically, the size of any spatial atom or term in the rule should affect the support. That is, given the support in Definition 3, two rules whose only difference is the area of the region in which they apply will have identical support. Consider Figure 2 where $r_1 \subset R_1$, and objects $\{a, b, c, d\}$ move from r_1 to r_2 between time t and $t + 1$. Then the rules $r_1 \Rightarrow r_2$ and $R_1 \Rightarrow r_2$ have the same support[2]. However, among these sets of equivalent rules we would prefer the rule covering the smallest area because it is more precise. A similar issue arises when we wish to compare the support of rules that cover different sized regions. Consider again Figure 2. The support of $r_1 \Rightarrow r_2$ is 4 and $\sigma(r_3 \Rightarrow r_4) = 2$ while $\sigma(R_1 \Rightarrow R_2) = 6$ which is higher than the other rules but only because it has a greater coverage. This leads to the conclusion that support should be defined in terms of the coverage of a rule.

Definition 5. *The* **spatial coverage** *of a spatio-temporal association rule ζ, denoted by $\phi_s(\zeta)$, is the sum of the area referenced in the antecedent and consequent of that rule. Trivially, the spatial coverage of a region r_i is defined as $\phi_s(r_i) = area(r_i)$.*

For example, the coverage of the rule $r_1 \Rightarrow r_2$ is $area(r_1) + area(r_2)$. This remains true even if $r_1 = r_2$ so that STARs with this property are not artificially advantaged

[2] Since $\{a, b, c, d\}$ follow the rules, the support is 4 in both cases.

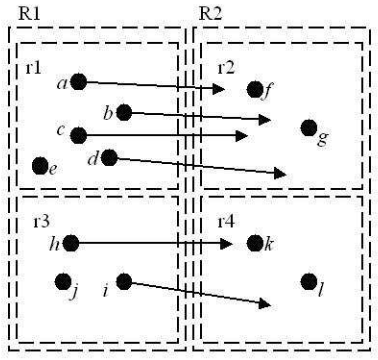


Fig. 2. Example data showing objects moving from time t to $t + 1$.

over the others. We solve the problem of different sized regions by scaling the support $\sigma(\zeta)$ of a rule by the area that it covers, to give *spatial support.*

Definition 6. *The* **spatial support**, *denoted by $\sigma_s(\zeta)$, is the spatial coverage scaled support of the rule. That is, $\sigma_s(\zeta) = \sigma(\zeta)/\phi_s(\zeta)$. The spatial support of a region r_i during TI and with respect to q is $\sigma_s(r_i, TI, q) = \sigma(r_{i,TI,q})/\phi_s(r_i)$.*

Consider again Figure 2 and assume the r_i have unit area and the R_i are completely composed of the r_i they cover. Then we have $\sigma_s(r_1 \Rightarrow r_2) = \sigma(r_1 \Rightarrow r_2)/\phi_s(r_1 \Rightarrow r_2) = 4/2 = 2$, $\sigma_s(r_3 \Rightarrow r_4) = 2/2 = 1$ and $\sigma_s(R_1 \Rightarrow R_2) = \sigma(R_1 \Rightarrow R_2)/\phi_s(R_1 \Rightarrow R_2) = 6/4 = \frac{3}{2}$. The rule $R_1 \Rightarrow R_2$ no longer has an advantage, and in-fact its spatial support is the weighted average of its two composing rules.

We do not need to scale confidence because it does not suffer from these problems. Indeed, increasing the size of the regions in a rule will on average increase both $\sigma(\zeta)$ and $\sigma(r_i, TI_s)$, so larger regions are not advantaged. Confidence is also a (conditional) probability, so scaling it by spatial coverage would remove this property.

In some applications it is desirable to find rules that are prevalent throughout the temporal axis of the dataset, rather than knowing exactly when the rule applied. For example, we may wish to avoid sporadic rules which might not be very useful. In these situations it will be useful to let t range over all values in the database. This is a *simple* way of aggregating the rules mined by our algorithms. Because the time the rules apply is now no longer 'fixed', it gives rise to the rule representations $(r_i, @\tau, q) \Rightarrow r_j$, $(r_i, \tau, q) \Rightarrow r_j$, $(r_i, [\tau], q) \Rightarrow r_j$ and $(r_i, [\tau_s], q) \Rightarrow (r_j, [\tau_e])$ respectively. The first three are interpreted as "objects in r_i satisfying q will be in region r_j (for the first time / at / by) τ time units later", while the latter is interpreted as "objects in r_i during a time interval of width τ_s and satisfying q will be in region r_j at some time during the subsequent time interval of width τ_e".

Aggregating the rules over time like this requires us to consider *temporal support.*

Definition 7. *The* temporal coverage *of a rule ζ, denoted by $\phi_t(\zeta)$, is the total length of the time intervals in the rule definition.*

For example, the temporal coverage of the rule $(r_i, TI_s, q) \Rightarrow (r_j, TI_e)$ is $|TE_s| + |TE_e|$ where $|TE|$ is an appropriate measure of the time interval width.

Definition 8. *The* temporal support *of a rule ζ, denoted by $\sigma_t(\zeta)$, is the number of time interval pairs $TI' = [TI_s, TI_e]$ over which it holds.*

Note that we did not perform scaling by temporal coverage. In short, this is because we view the temporal coverage as being defined by the user and so each rule mined will necessarily have the same temporal coverage. A more complicated reason presents itself when we consider mining the actual rules. For example, assume the temporal coverage of a rule ζ is τ. We have at least two options, either we count the temporal support of the rule during $[t, t + \tau], [t + 1, t + 1 + \tau], [t + 2, t + 2 + \tau], ...$ or during $[t, t + \tau], [t + \tau, t + 2\tau], [t + 2\tau, t + 3\tau], ...$ Scaling by temporal coverage would only make sense in the second case. If we assume an open timescale (one that has no end, or is sufficiently large that we can assume this) then the number of opportunities to gain a support count (that is, for the rule to hold) in the first case does not depend on the size of τ. That is, the temporal coverage is not a factor.

Note that *temporal support* only applies to the case where a user is interested in which STARs re-occur over time (and hence that STARs which rarely occur are not interesting). The reader should note that the definitions of STARs that we give apply to a specific time interval and describe how objects move during that time (indeed our algorithms look at each pair of time intervals TI' only once). This is quite general in the sense that the mined STARs can be analysed for changes over time, for periodicities, or simply aggregated in time to find recurrent patterns.

Both temporal and spatial coverage are defined by the user (or by the application). Spatial coverage is inherent in the size of the regions. Temporal coverage is more flexible and determines the window for which rules must be valid, but this choice is the same for all rules. When mining STARs we attempt to find rules that have a *spatial support* above a threshold, $minSpatSup$ and *confidence* above $minConf$. If the user is interested in summarising STARs over time, we additionally output only those rules with *temporal support* above $minTempSup$.

5 Hot-Spots, High Traffic Areas and Stationary Regions

Definition 9. *A region r is a dense region or* **hot spot** *with respect to q during TI if $density(r, TI) \equiv \sigma_s(r, TI, q) \geq minDensity$. We also define r as dense during $TI' = [TI_i, TI_{i+1}]$ if it is dense during both both TI_i and TI_{i+1}.*

We define a region r to have high traffic (with respect to some query q) if the number of objects that satisfy q and are entering and/or leaving the region is above some threshold. A stationary region is one where enough objects remain in the region. These patterns are a special type of STAR. They are also easy to find. Consider two *successive* time intervals TI_1 and TI_2. Then the number of objects (satisfying q) that are in r in TI_2 that were not there in TI_1 is the number of objects that entered r between TI_1 and TI_2. Let S_1 be the set of objects (satisfying q) that are in r during TI_1, and let S_2 be the corresponding set for TI_2. We are clearly looking for $|S_2 - S_1|$, where $-$ is the set difference operation. Similarly, the number of objects leaving r is $|S_1 - S_2|$ and the number of objects that remain in r for both TI_1 and TI_2 is $|S_1 \cap S_2|$.

Example 2. Consider again Figure 1 during $TI' = [[t,t],[t+1,t+1]] \equiv [t,t+1]$ and assume the threshold is 3. $\{e,b,c\}$ enter r_2 during TI', so r_2 is a sink and because they came from r_1, r_1 is a source. During $TI' = [t+1,t+2]$, $\{g,b,c\}$ remain in r_2 so it is a stationary region during TI'. If the threshold is 2, it would also be a thoroughfare because $\{a,d\}$ enter while $\{e,f\}$ leave during TI'. r_2 is also a stationary region during $[t=2,t+3]$ because $\{a,b,d\}$ stay there.

To express high traffic regions as STARs, note that if we let $*$ be the the set of regions excluding r but including a special region r_{else}, then the number of objects entering r during $TI' = [TI_i, TI_{i+1}]$ is just the support of $(*, TI_i, q) \Rightarrow (r, TI_{i+1})$. We need r_{else} to cater for the case that an object 'just' appears (disappears) from the region set. We model this as the object coming from (going to) r_{else}. This situation would happen in the mobile phone example when a user turns his/her phone on (off). We now formally define high traffic areas and stationary regions.

Definition 10. *A region r is a* **high traffic region** *with respect to query q if the number of objects (satisfying q) entering r (n_e) or leaving r (n_l) during $TI' = [TI_i, TI_{i+1}]$ satisfy*

$$\frac{\alpha}{\phi_s(r)} \geq minTraffic \,:\, \alpha = n_e \, or \, n_l$$

where $minTraffic$ is a given density threshold and ϕ_s is given by definition 5. Note that $n_e \equiv \sigma((*, TI_i, q) \Rightarrow (r, TI_{i+1}))$ and $n_l \equiv \sigma((r, TI_i, q) \Rightarrow (*, TI_{i+1}))$.

Such regions can be further subdivided. If $n_e/\phi_s(r)$ is above $minTraffic$ we call that region a **sink.** *If $n_l/\phi_s(r)$ is above $minTraffic$ we call it a* **source,** *and if a region is classified as both a sink and a source we call it a* **thoroughfare.**

Definition 11. *If the number of objects remaining in r, denoted by n_s, satisfies $\frac{n_s}{\phi_s(r)} \equiv \sigma((r, TI_i, q) \Rightarrow (r, TI_{i+1}))/\phi_s(r) \geq minTraffic$ then we call r a* **stationary region***. A stationary region may or may not be a high traffic region.*

Note that if we define $area(*) = 0$ then the definition of high traffic areas is a statement about the *spatial support* of special types of STARs. For stationary regions however, we get as a consequence of Definition 5 that $\frac{n_s}{\phi_s(r)} = 2 \cdot \sigma_s((r, TI_i, q) \Rightarrow (r, TI_{i+1}))$. We define $n_\alpha/\phi_s(r) \,:\, \alpha \in \{e, l, s\}$ as the *spatial support* of these patterns.

The following theorem allows us to prune the search space for finding high traffic regions and stationary regions.

Theorem 1. *If $minTraffic \geq minDensity$ then:*
1. The set of sources during $[TI_i, TI_{i+1}]$ are a subset of dense regions during TI_i
2. The set of sinks during $[TI_i, TI_{i+1}]$ are a subset of dense regions during TI_{i+1}
3. The set of stationary regions during $[TI_i, TI_{i+1}]$ are a subset of the regions that are both dense during TI_i and during TI_{i+1}.

Proof. See the appendix.

As a consequence, if $minTraffic \geq minDensity$ then the set of thoroughfares during $[TI_i, TI_{i+1}]$ is a subset of the regions that are dense at time TI_i and at time TI_{i+1}. These properties prune the search for high traffic regions, so we can find high traffic regions by setting $minDensity = minTraffic$, mining all hot-spots and then mining the hot-spots for high traffic areas.

6 Mining Spatio-Temporal Association Rules

In this section we exploit the high traffic area mining techniques to develop an efficient STAR mining algorithm for the STARs of definition 2 (omitting q for simplicity).

As a motivation to why this is useful, assume a set of regions R. Then there are $|R|^2$ possible rules for each TI', since there are $|R|$ starting regions and $|R|$ finishing regions. Using a brute force method, this would require $|R|^2$ counts of the number of objects in a region. Our algorithms address this quadratic time component.

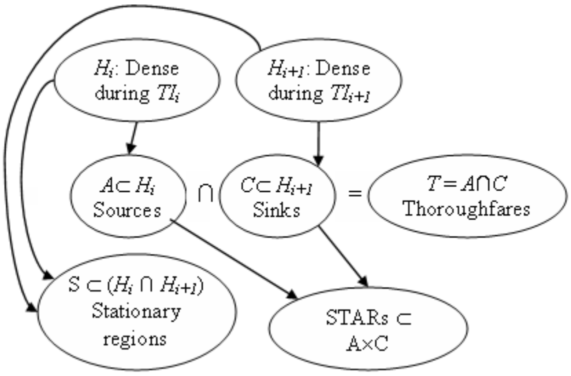

H_i: Dense during TI_i
H_{i+1}: Dense during TI_{i+1}
$A \subset H_i$ Sources
∩
$C \subset H_{i+1}$ Sinks
=
$T = A \cap C$ Thoroughfares
$S \subset (H_i \cap H_{i+1})$ Stationary regions
STARs ⊂ A×C

Fig. 3. Illustration of the complete mining procedure.

The reader may find it useful to refer to Figure 3 while reading the following. We exploit the following theorem:

Theorem 2. *Let* $sizeFactor = \frac{\max_k(area(r_k)) + \min_k(area(r_k))}{\max_k(area(r_k))}$.
If $sizeFactor \cdot minSpatSup \geq minTraffic$ *then during* $TI' = [TI_i, TI_{i+1}]$*:*
1. The set of consequent regions of STARs with spatial support above $minSpatSup$ *is a subset of the set of sinks,* C*.*
2. The set of antecedent regions of STARs with spatial support above $minSpatSup$ *is a subset of the set of sources,* A*, and*
3. The set of STARs whose consequent and antecedent are the same and have a spatial support above $minSpatSup$ *correspond to a subset of stationary points, with equality when* $2 \cdot minSpatSup = minTraffic$*.*

Proof. See the appendix.

The steps to mine STARs of definition 2 with spatial support above $minSpatSup$ and confidence above $minConf$ during the time interval $TI'_i = [TI_i, TI_{i+i}]$ are as follows:

1. Set $minDensity = minSpatSup \cdot sizeFactor$ and mine all *hot-spots* during TI_i and during TI_{i+1} to produce the set H_i and H_{i+1}.
2. Set $minTraffic = minSpatSup \cdot sizeFactor$ and find the set of *high traffic areas* and *stationary regions* from H_i and H_{i+1}. Denote the set of *sources* by A, the set of *sinks* by C, the set of *thoroughfares* by T and the set of *stationary regions* by S. Recall from Theorem 1 that $A \subset H_i$, $C \subset H_{i+1}$, $S \subset H_i \cap H_{i+1}$ and $T = A \cap C \subset H_i \cap H_{i+1}$.
3. By Theorem 2, A contains all candidates for the antecedent of STARs, C contains all the candidates for consequents of STARs and S contains all the STARs where the antecedent is the same as the consequent. Using this we evaluate the rules corresponding to the elements of $A \times C - S \times S$ and S for spatial support and confidence[3]. We keep all rules that pass these tests.

We then apply the above procedure for the next successive set of timestamps $TI'_{i+1} = [TI_{i+1}, TI_{i+2}]$ and so on. We therefore generate a sequence of pattern sets (hot-spots, high traffic areas, stationary regions and STARs) $< \xi_i, \xi_{i+1}, \xi_{i+2}, ... >$ over time. If desired, we aggregate the patterns by counting the number of intervals TI' for which each of the the patterns hold. If the total number of these (its temporal support as defined earlier) is above the threshold $minTempSup$ after the procedure is complete, we output the pattern. The TI are given by a *schedule algorithm* that splits up the timestamps into a stream of time intervals. There are many possible choices for this algorithm, two examples of which we have considered in section 4.1.

If regions are of different sizes, then in the worst case situation where a very large region and a very small region exist the pruning will be least efficient. In the limiting case we obtain the choice which gives the lower bound $\min_{choice\, of\, region\, geometry} sizeFactor = 1$. On the other hand, the best pruning occurs when all the regions are the same size, in which case $sizeFactor = 2$ and set of stationary regions corresponds exactly to the set of STARs with the same antecedents and consequents. In this case we set $2 \cdot minSpatSupport = minTraffic = minDensity$ in the procedure above and we don't need to check the rules corresponding to S for support.

An optional pruning method may be applied that takes into account an objects maximum speed or other restrictions on its movement. That is, a STAR will not exist between two regions r_i and r_j if they are so far apart that it is impossible to reach r_j from r_i during the time interval TI'. Define a neighbourhood relation $N(R_i, R_j)$ that outputs the subset S of $R_i \times R_j$ such that $S = \{r_i, r_j : N(r_i, r_j) = 1, r_i \in R_i, r_j \in R_j\}$. Here, R_i, R_j are sets of regions, and $N(r_i, r_j)$ is 1 if and only if r_i and r_j are neighbours. By 'neighbours' we mean that r_j can be reached from r_j during TI'. This relation allows restrictions such as 'one way' areas, inaccessible areas, and maximum speed of objects

[3] Note that S may or may not be contained in $A \cup C$ and may in fact be disjoint. This is why we need to evaluate all of S for STARs. Since some overlap may occur, we save repeated work by evaluating $A \times C - S \times S$ rather than $A \times C$.

to be exploited for further pruning the search space. If such a relation is available, we now need only to evaluate $N(A, C) - S \times S$.

The reader should note that $|A \times C - S \times S| \leq |R \times R|$, and that the amount by which it is smaller depends on the data and $minSpatSup$. We effectively prune the search space *as much as possible given the dataset and mining parameters* before doing the computationally expensive part. Our experiments show that this time saving is large in practice, even for very small spatial support thresholds.

Finally, to tie up the theory, we define the temporal support of a *hot-spot* as the number of TI's for which the region is dense. Consequently, the reader should note that the *hot-spots, stationary regions, high traffic areas* and *STARs* all have spatial and temporal support defined for them, and apply over two successive time intervals $TI' = [TI_i, TI_{i+1}]$.

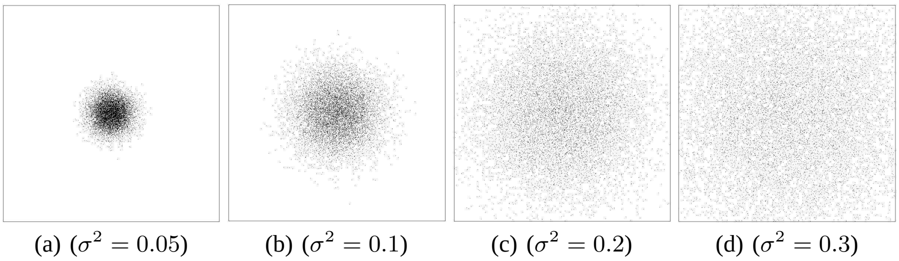
7 Experiments

In this section we present the results of some experiments on our STAR mining algorithm. We show that our algorithm gives significant time savings and scales well. We then use FM-PCSA sketches [12] in our algorithm, and evaluate the resulting performance. Finally, we show how well a pattern can be mined under noisy conditions.

7.1 Efficiency and Scalability

The Datasets We used a modified[4] version of the well known GSTDTOOL to generate the data for this experiment. Theodoridis et al. [13] proposed the GSTD ("Generate Spatio-Temporal Data") algorithm for building sets of moving points or rectangular objects. For each object o the GSTD algorithm generates tuples of the form $(id, t, x, y, ...)$ where id is a unique id, t is a time-stamp and (x, y) are the coordinates of the point. Object movements are configurable by specifying the distribution of changes in location. We generated four datasets for our experiments consisting of $10,000$ points, each moving through $[0, 1)^2$ at timestamps $0, 0.01, 0.02, ..., 1$ according to the rules $x \leftarrow x + X$, $y \leftarrow y + Y$ where $X \sim uniform(-0.01, 0.05)$ and $Y \sim uniform(-0.01, 0.1)$. That is, the points changed their location every 0.01 time units which can be thought of as sampling the location of continuously moving objects every 0.01 time units. We use the following partitioning of timestamps to generate our intervals: $TI' = [t, t+1], [t+1, t+2], [t+2, t+3], ...$ That is, the time intervals discussed in our algorithms become individual timestamps and each successive pair of time stamps is used for STAR mining. *Toroidal* adjustment was used so objects wrap around the unit square when they reach its boundary. The initial distributions were *Gaussian* with mean 0.5.Our four datasets differed only in the variance of the initial distributions, with $\sigma^2 \in \{0.05, 0.1, 0.2, 0.3\}$ corresponding to what we shall call our *Compact, Medium, Sparse* and *Very Sparse* datasets. The initial distributions are displayed in Figure 4. For the parameters listed above, the objects make about 4.5 loops of the workspace during the 100 timestamps in the dataset. The speed

[4] The GSTD algorithm does not output data in time order.



(a) ($\sigma^2 = 0.05$) (b) ($\sigma^2 = 0.1$) (c) ($\sigma^2 = 0.2$) (d) ($\sigma^2 = 0.3$)

Fig. 4. Initial Distribution of the Four Datasets (*Compact, Medium, Sparse and Very Sparse*)

and the randomness of the motion has the effect of spreading the objects out rather quickly over the workspace, so all the datasets become sparse toward the end of the timestamps. The compact dataset is thus not as easy for our STAR mining algorithm as it might first appear. Indeed, the datasets were chosen to thoroughly exercise our algorithm. We used six region configurations in $n \times n$ grids where the number of regions was varied $(36, 81, 144, 225, 324, 441)$, while keeping the total area covered constant at $15^4 = 50625$ (the output of the GSTD algorithm was scaled to this area).

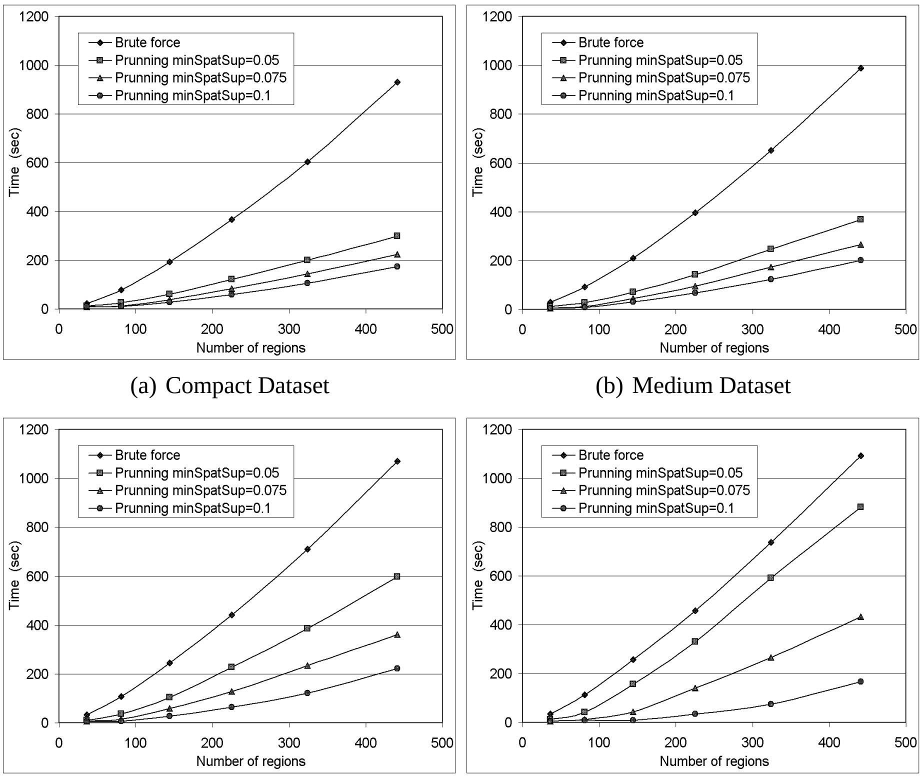
Evaluating the STAR Mining Algorithm We evaluate the performance gains of the algorithm over a brute force algorithm (the current[5] approach in literature) performing the same task on the various datasets, using different parameter settings. Recall that the STAR mining algorithm first found dense regions, then used these to find sources, sinks, thoroughfares and stationary regions. It then used the sources as potential antecedents (A), the sinks as potential consequents (C) and evaluated all STARs corresponding to a subset of the cross product $A \times C$ for spatial support and confidence. The brute force mining algorithm simply evaluates all STARs corresponding to the cross product $R \times R$ where R is the set of all regions. We used a simple brute force technique to find the dense regions. We did *not* use a neighbourhood relation to further prune the search space in these experiments. We varied the spatial support threshold: $minSpatSup \in \{0.05, 0.075, 0.1\}$. Due to our region configuration (which gives $sizeFactor = 2$), and using the results from the theory, we always have $minSpatSup \cdot 2 = minDensity = minTraffic$. We used $minConf = 0.0$ and $minTempSup = 1$.

The choice of a *very low* spatial support threshold ($minSpatSup = 0.05$) was made so that many rules were mined for all the datasets, and that a high proportion of regions would be dense and high traffic regions according to the corresponding thresholds. For example, for the 15 by 15 region set, $minSpatSup = 0.05$ corresponds to a region being dense (high traffic) if at least 22.5 objects occupy it (move into it or out of it). Since there are $10,000$ objects and only $15^2 = 225$ regions, on average this means that if the points were spread out evenly, (which is almost the case for the very sparse dataset, and all datasets at the end of the 101 timestamps) each region would have more than 44 objects in it, more than sufficient for the support thresholds. And since objects

[5] The problem of STARs is quite new. Recall that the only other work to address STARs was by Tao et al. [7], who used a brute force technique.

will move on average more than $2/3$ of the way across a region during each timestamp, there will be plenty of objects moving between regions to provide support for high traffic regions and STARs. With the finer region configurations ($18 \times 18, 21 \times 21$) objects almost move an entire region width each timestamp. If the algorithm performs well on very low setting of support, then it is guaranteed to perform even better for higher settings. The other support settings are still quite low but are large enough to demonstrate how big the benefits of pruning are. For the highest ($\sigma_s = 0.1$) in the 15×15 region configuration the support threshold works out to be 45, barely greater than the average number of objects per region.

Results As expected, the rules mined by our STAR mining algorithm, which is labeled in the figures as the 'pruning' technique, were identical to those mined by the brute force algorithm. The time taken to mine the rules using our algorithm was far superior to the brute force algorithm, as can be seen in Figure 5.



(a) Compact Dataset

(b) Medium Dataset

(c) Sparse Dataset

(d) Very Sparse Dataset

Fig. 5. Results on the different datasets with different support settings.

Recall that the benefit of our algorithm is to prune the search space of STARs in order to avoid the quadratic time explosion that occurs when the the antecedent and consequent of a rule both vary over the set of all regions. In our algorithm, the quadratic time component occurs only for a subset of the total regions. Since the size of this subset depends on the support thresholds and the spread of the data (assuming of course that the points are sufficiently dense, which was deliberately the case for our datasets), the more spread out the data is, the more regions become potential sources and sinks (and hence potential antecedents and consequents) and the more regions must be examined for STARs. This effect is demonstrated in the results for the four datasets used.

For the compact and medium dataset case (Figure 5(a) and (b)), the time taken by the pruning algorithm for all support thresholds grows significantly slower than the brute force approach. For the very sparse dataset (Figure 5(d)), the time for both algorithms grow at a similar rate for $minSpatSup = 0.05$, but the pruning algorithm is still significantly faster. Recall that for this *low* setting of the support threshold, almost every region becomes dense and a high traffic region. For the higher support thresholds, the pruning algorithm is able to prune more regions and subsequently performs much better. The other datasets fall between these two cases. Recall that the more regions are pruned, the smaller the quadratic component of the algorithm is. In all cases it can be seen that pruning is very beneficial and that the amount the search space can be pruned is determined by the support thresholds. Since users will be looking for patterns with high support, this is ideal.

7.2 Performance when using Sketches to Handle Real-Time Streaming Data

In this experiment we use FM-PCSA sketches [12] in place of the exact counting algorithm used in the other experiments. Recall that in the previous section we had 10,000 objects. We used a bit vector to represent the sets in all the regions, which made it easy and fast to calculate the set operations we require. However, this approach will no longer be feasible if we need to deal with millions of objects - especially if it is in a streaming environment. Infact, any exact techniques will not scale well. We therefore evaluate the effectiveness of using sketches.

Flajolet and Martin [12] proposed a sketch and associated algorithms for finding the approximate cardinality of a multi-set efficiently. The basic data structure is known as the FM sketch, consisting of r bits. By using a number (m) of these sketches in a technique known as Probabilistic Counting With Stochastic Averaging (PCSA), Flajolet and Martin improve the estimate produced by the algorithm. The resulting algorithm is known as the FM-PCSA algorithm, and its data structure (synopsis, sketch) consists of a m by r bit array. It has the property that $E(a(S_1)\,OR\,a(S_2)) = E(a(S_1 \cup S_2))$ where E is the function that maps a sketch to the resulting estimate and a is a function that creates the sketch from a set S. This property allows us to use one sketch for each region, and combine them together as desired when we need to calculate the number of objects moving between or staying in regions. We do this by expressing set difference and intersection in terms of set union, which the above property allows us to estimate.

Using a probabilistic counting algorithm means that the set operations we need to evaluate are not exact, and we will get approximate answers and results. The benefit obtained is a faster running time and lower space requirements. Note that an assumption of the STAR mining algorithm is that the underlying counting algorithm is exact. We can thus not expect the results of brute force mining and the pruning algorithm to return the same rules if a probabilistic counting algorithm is used. This is because the prunning properties do not apply if the estimates provided by the counting algorithm have errors. Indeed, this is the case for our results. The reader should note that this disparity may in-fact be useful. with pruning enabled, the results are *consistent*. In contrast, if it is disabled, then there will exist STARs where the antecedent is *not* a source or where the consequent is *not* a sink - that is, there are inconsistencies. The results we present were obtained with the pruning enabled.

We used the *compact* dataset in these experiments. For all the experiments r was set to 10 and m was selected from the set $\{28, 48, 80\}$. The choice of r is determined by how high one wishes to count. The FM-PCSA algorithm requires a good, uniform hashing function. Simple multiplicative hash functions are acceptable for small values of m, but have poor performance for larger m due to their relatively poor statistical properties. We used a multiplicative hash for $m = 28$, and the SHA-1 secure hash function for $m \in \{48, 80\}$.

Note that we are using very little space for the sketch in each of our experiments – 280, 480 and 800 bits per region as opposed to the 10,000 we used previously. *That is, we are using only 2.8%, 4.8% and 8% respectively of the space required for exact mining.*

The estimator used in the FM-PCSA sketches are known to have non-linearities when the number of objects counted is small [12], and indeed the algorithm tends to overestimate in such circumstances because the scaling factor in the algorithm is calculated from an asymptotic bound, and is thus not suitable for small estimates. We thus vary the spatial support in order to counteract the non-linearities of the FM-PCSA algorithm when it is applied for small cardinalities.

Results Figure 6 shows the precision and recall for the three values of m [6]. As the number of returned objects increases (in our experiment, as the spatial support is decreased), we expect the recall to increase (in the limit all rules are returned and we have a recall of 1), while precision can be expected to decrease as it becomes difficult to return only correct rules. The results show that the precision and recall tend to increase as we increase m (notice form $m = 80$ the two curves meet before the curves for other values of m). This is to be expected because increasing m increases the accuracy of the estimates within the STAR mining procedure. Secondly, as spatial support increases, precision increases at a faster rate than recall falls. This leads us to believe that the overestimates caused by the false positive rates (which is a property of the FM-PCSA sketch) can be ameliorated, to a certain extent, by increasing the spatial support.

[6] *Precision* is defined as the fraction of rules found that are correct, that is, $\frac{TP}{TP+FP}$ where TP is the number of true positives and FP is the number of false positives (as determined by the exact mining algorithm). *Recall* is defined as the proportion of of correct rules that are found relative to the total number of rules that in the dataset. Recall is thus given by the equation $\frac{TP}{TP+FN}$ where FN is the number of false negatives. It is generally not possible to maximise both precision and recall as there is a trade-off.

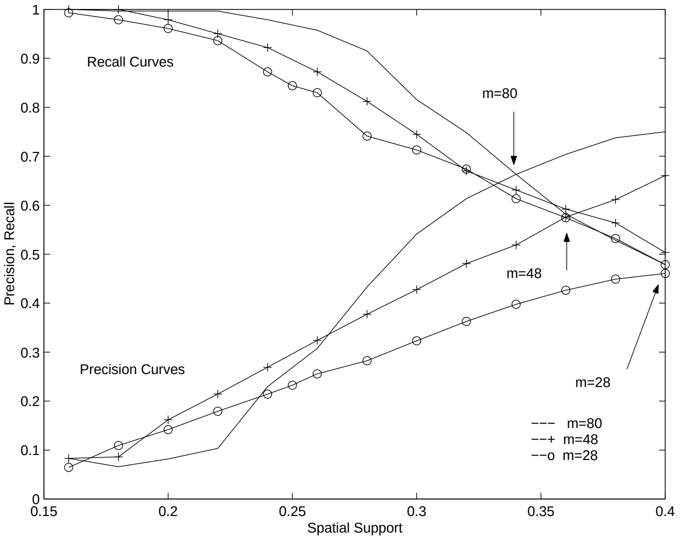


Fig. 6. Precision and Recall for the three different settings of the number of sketches, m. The curve was obtained by varying the spatial support.

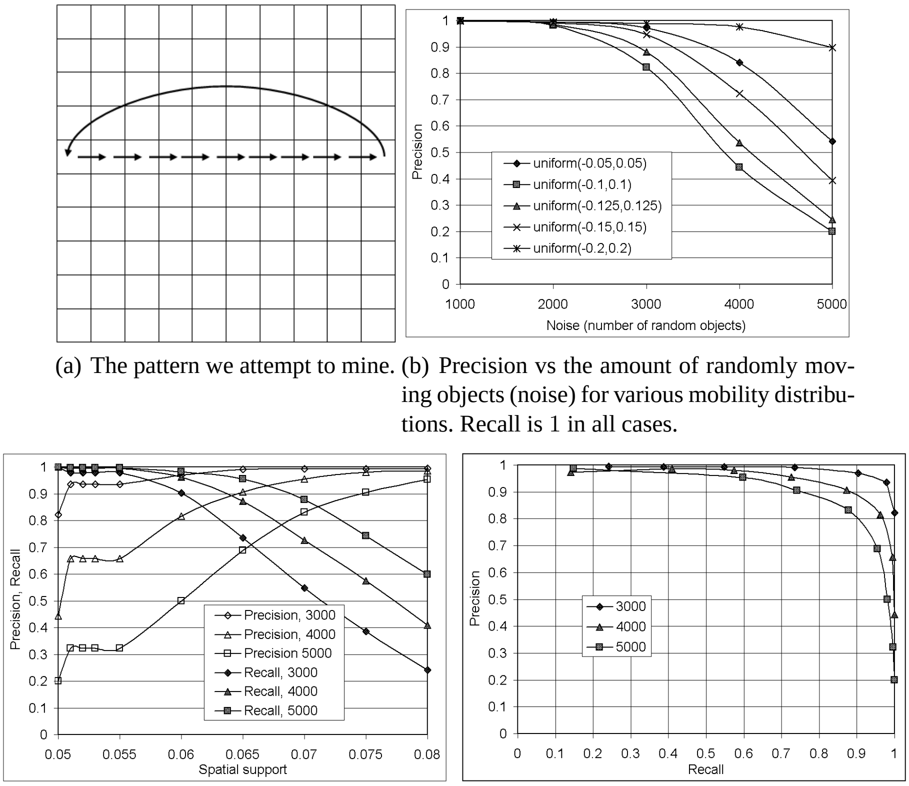
There were significant time savings obtained by using the probabilistic counting algorithm. As percentages of the time required by the exact technique, these savings are 36.7%, 36.6% and 37.8% for $m = 28, 48,$ and 80 respectively.

In summary therefore, we were able to achieve precision and recall values of 67% using only 8% of the space and less than 38% of the time of exact methods. Since the size of FM-PCSA sketches scale logarithmically in the number of objects, they can be used to handle millions of objects. By increasing their size, errors can be reduced, and hence precision and recall will increase.

7.3 Finding Patterns in Noisy Data

Datasets In this experiment we generate a dataset with a known pattern and add noise to it. We used a 10×10 region layout. 100 objects were spread evenly (10 per region) over one row of the grid and made to move with a constant velocity of one region (0.1 of the unit square) per time stamp to the right, as illustrated in Figure 7(a). The grid was toroidal, so objects leaving on one side emerge on the other. We then added various numbers N of uniformly distributed objects to the region layout, moving with various speeds. We used $N \in \{1000, 2000, 3000, 4000, 5000\}$, and the distribution defining the objects' change in location was $X \sim Y \sim uniform(-\alpha, \alpha)$ with $\alpha \in \{0.05, 0.1, 0.15, 0.2\}$. The datasets were generated in the unit square, which was scaled up so that each (square) region had an area of 100. With a $minSpatSup \leq 0.05$ therefore, this creates the same 10 STARs at each time interval. We set $minSpatConf = 0$.

Results We first set $minSpatSup = 0.05$ (and hence $minTraffic = minDensity = 0.1$), thus ensuring a recall of 1, and examined the effect of the 'noisy objects' on precision. As illustrated in Figure 7(b), precision fell for all datasets as the number of objects was increased. We also note that the way the noisy objects moved impacted on the false positives. Objects that moved fast or slow created fewer false positive STARs. The worst drop in precision (supported by further experiments with finer grained variation of α) was for objects moving with $uniform(-0.1, 0.1)$. This meant that they moved enough on average to move between regions, but not enough to move too far away and thus 'dilute' their support.



(a) The pattern we attempt to mine.

(b) Precision vs the amount of randomly moving objects (noise) for various mobility distributions. Recall is 1 in all cases.

(c) Varying $minSpatSup$.

(d) Precision vs Recall for $uniform(-0.1, 0.1)$, obtained by varying the $minSpatSup$.

Fig. 7. Results for mining a pattern in noisy data.

Overall, the results were quite good: for a recall of 1 we achieved precision above 0.98 for all datasets with 2000 or fewer noisy objects. That is, when the number of noisy objects was twenty times the number of objects creating the target pattern (this meant there were two times as many noisy objects in regions of the STARs than objects supporting the desired pattern). For most of the datasets the performance for 3000 or more noisy objects was not very good. We therefore sacrifice our perfect recall score for some increases in precision by increasing $minSpatSup$. Generally, increasing $minSpatSup$ will reduce recall but increase precision. We use the *hardest* dataset for this. That is, with the noisy objects moving with $uniform(-0.1, 0.1)$. Figure 7(c) shows the precision and recall for the various $minSpatSup$, while Figure 7(d) shows the resulting precision-recall curve. This curve is has excellent characteristics - it shows that by treating $minSpatSup$ as a tuning parameter we can achieve high recall and precision, with a relatively small trade-off at high values. Clearly, even for the 5000 noisy object case we are able to achieve precision and recall above 0.85, which is an excellent result. That is, with noisy objects 50 times the number of objects supporting the desired pattern, moving over an area 10 times the area of the desired pattern, with α set to maximise the false positive rate, we are still able to get precision and recall above 0.85.

8 Conclusion

We have introduced rigorous definitions of important spatio-temporal patterns while retaining the semantics of space and time. Most notably, we have effectively extended association rule mining to the spatio-temporal domain. We also defined other patterns: *hot spots* , *stationary regions*, *high traffic areas*, *sources*, *sinks* and *thoroughfares*. These can be used to prune the search space of STARs, but are also interesting in their own right. Finally, we presented efficient algorithms for finding these patterns in object mobility datasets. These algorithms prune the search space as much as possible before doing the computationally expensive part. By mining the patterns on a time interval by time interval basis, we can not only find current patterns in streaming data, but also see how these patterns evolve over longer periods of time. Future work will be directed at analysing changes in the patterns over time.

References

1. Shekhar, S., Huang, Y.: Discovering spatial co-location patterns:a summary of results. In: Proceedings of the 7th International Symposium on Spatial and Temporal Databases SSTD01. (2001)
2. Huang, Y., Xiong, H., Shekhar, S., Pei, J.: Mining confident co-location rules without a support threshold. In: Proceedings of the 18th ACM Symposium on Applied Computing ACM SAC. (2003)
3. Ale, J.M., Rossi, G.H.: An approach to discovering temporal association rules. In: SAC ’00: Proceedings of the 2000 ACM symposium on Applied computing, ACM Press (2000) 294–300
4. Li, Y., Ning, P., Wang, X.S., Jajodia, S.: Discovering calendar-based temporal association rules. Data Knowl. Eng. **44** (2003) 193–218
5. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases VLDB, Morgan Kaufmann (1994) 487–499

6. Mennis, J., Liu, J.: Mining association rules in spatio-temporal data. In: Proceedings of the 7th International Conference on GeoComputation. (2003)
7. Tao, Y., Kollios, G., Considine, J., Li, F., Papadias, D.: Spatio-temporal aggregation using sketches. In: 20th International Conference on Data Engineering, IEEE (2004) 214–225
8. Tsoukatos, I., Gunopulos, D.: Efficient mining of spatiotemporal patterns. In: SSTD '01: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases, London, UK, Springer-Verlag (2001) 425–442
9. Wang, J., Hsu, W., Lee, M.L., Wang, J.T.L.: Flowminer: Finding flow patterns in spatio-temporal databases. In: ICTAI. (2004) 14–21
10. Ishikawa, Y., Tsukamoto, Y., Kitagawa, H.: Extracting mobility statistics from indexed spatio-temporal datasets. In: STDBM. (2004) 9–16
11. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, indexing, and querying historical spatiotemporal data. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2004) 236–245
12. Flajolet, P., Martin, G.: Probabilistic counting algorithms for data base applications. Journal of Computer Systems Science **31** (1985) 182–209
13. Theodoridis, Y., Silva, J., Nascimento, M.: On the generation of spatio-temporal datasets. In: 6th International Symposium of Large Spatial Databases (SSD'99), Springer-Verlag (1999) 147–164

Appendix: Proofs

Proof of Theorem 1: Let $n(TI_i)$ be the number of objects in r during TI_i, let $n_l(TI_i)$ be the number of objects leaving r during TI_i (compared with $n(TI_{i-1})$), and similarly, let $n_e(TI_i)$ be the number of objects entering r during TI_i. The set of objects leaving a region r during TI_{i+1} is a subset of the objects within that region during TI. So $n_l(TI_{i+1}) \leq n(TI_i)$. Similarly, the number of objects entering r during TI_{i+1} is a subset of the number of objects in r during TI_{i+1}, giving $n_e(TI_{i+1}) \leq n(TI_{i+1})$. For r to be classified as a source during $TI' = [TI_i, TI_{i+1}]$ we must have $n_l(TI_{i+1})/\phi_s(r) \geq minTraffic$. So if $minTraffic \geq minDensity$ then $density(r, TI_i) = n(TI_i)/\phi_s(r) \geq n_l(TI_{i+1})/\phi_s(r) \geq minTraffic \geq minDensity$ so r must be dense during TI_i. Similarly, for sinks we have $density(r, TI_{i+1}) = n(TI_{i+1})/\phi_s(r) \geq n_e(TI_{i+1})/\phi_s(r) \geq minTraffic \geq minDensity$.

The set of stationary regions is the intersection of the objects that are in the region during TI_i (S_1) and the set of objects that are in the region during TI_{i+1} (S_2). The stationary regions are thus clearly a subset of both S_1 and S_2. Therefore r can be a stationary region only if r is dense during both TI_i and TI_{i+1} with respect to the threshold $minTraffic$. ■

Proof of Theorem 2: In the following the time interval $TI' = [TI_i, TI_{i+1}]$ is implicit. Consider two regions r_i and r_j where i and j range over all possible values. Let n_l be the number of objects leaving r_i during TI', let n_e be the number of objects entering r_j during TI' and let n_m be the number of objects moving from r_i to r_j. Clearly $n_m \leq n_e$ and $n_m \leq n_l$. For the rule $\zeta = (r_i, TI_i, q) \Rightarrow (r_j, TI_{i+1})$ to have support of at least $minSpatSup$ we have $\sigma_s(\zeta) = \frac{n_m}{area(r_i)+area(r_j)} \geq minSpatSup$. We then have $\frac{n_l}{area(r_i)+area(r_j)} \geq \sigma_s(\zeta)$ and $\frac{n_e}{area(r_i)+area(r_j)} \geq \sigma_s(\zeta)$ so $\frac{n_l}{area(r_i)} \geq$

$\sigma_s(\zeta) \cdot \frac{area(r_i)+area(r_j)}{area(r_i)}$ and $\frac{n_e}{area(r_j)} \geq \sigma_s(\zeta) \cdot \frac{area(r_i)+area(r_j)}{area(r_j)}$. Therefore by definition 10 on page 10, r_i is a source if $\sigma_s(\zeta) \cdot \frac{area(r_i)+area(r_j)}{area(r_i)} \geq minTraffic$ and r_j is a sink if $\sigma_s(\zeta) \cdot \frac{area(r_i)+area(r_j)}{area(r_j)} \geq minTraffic$ since in those cases we have $\frac{n_l}{area(r_i)} \geq \sigma_s(\zeta) \cdot \frac{area(r_i)+area(r_j)}{area(r_i)} \geq minTraffic$ and $\frac{n_e}{area(r_j)} \geq \sigma_s(\zeta) \cdot \frac{area(r_i)+area(r_j)}{area(r_j)} \geq minTraffic$. For both the source and sink to be found we must thus have $\min\left(\frac{area(r_i)+area(r_j)}{area(r_i)}, \frac{area(r_i)+area(r_j)}{area(r_j)}\right) \geq minTraffic$.

Now because we require the relationship to hold for all r_i and r_j we must minimise $\min\left(\frac{area(r_i)+area(r_j)}{area(r_i)}, \frac{area(r_i)+area(r_j)}{area(r_j)}\right)$ so that the bound is strict.

Since $\frac{area(r_i)+area(r_j)}{area(r_i)}$ is minimised for the choice of regions $r_i = \max_k(area(r_k))$ and $r_j = \min_k(area(r_k))$, the value we require is $sizeFactor$.

The last case is a consequence of the definition of the spatial support - that is, even if the antecedent and consequent are the same they are both still counted in the scaling factor. In this case we have $r_i = r_j$. Let n_s be the number of objects remaining in region $r = r_i = r_j$. By the definition of spatial support we have $\sigma_s(\zeta) = \frac{n_s}{2 \cdot area(r)}$ and for ζ to have sufficient spatial support we require $\sigma_s(\zeta) = \frac{n_s}{2 \cdot area(r)} \geq minSpatSup$. Now r is a stationary region if $\frac{n_s}{area(r)} \geq minTraffic$ so all rules ζ where $r_i = r_j$ will be found if $2 \cdot minSpatSup \geq minTraffic$, with the set of such STARs being exactly equal to the set of stationary points if $2 \cdot minSpatSup = minTraffic$. The theorem follows by noting that $\max_{choice\ of\ region\ geometry} sizeFactor = 2$. ■

View publication stats