Discovering SpatioTemporally Invariant Event Patterns From Mobility Data

Discovering SpatioTemporally Invariant Event Patterns from Mobility Data

Luca Colomba, Luca Cagliero, *Member, IEEE,* and Paolo Garza, *Member, IEEE*

***Abstract*—The discovery of sequential patterns from spatiotemporal data is known to be a very complex data mining task. The relevance of spatiotemporal patterns to study event correlations in mobility data is established. Prior works addressed either the separate analysis of spatial and temporal dependencies among data, such as the co-location of events, or the study of the joint spatiotemporal properties of the trajectories observed over a region of interest. The aim of this paper is instead to overcome existing approaches by extracting sequences of discrete events showing spatiotemporally invariant properties. For example, if *an arbitrary bike sharing station becomes full (all its docks are used)* then *we will observe an increase in the occupancy level of the bike sharing stations in the surrounding area within ten minutes*.**

We denote such a new pattern as a SpatioTemporally Invariant (STInv) event pattern because we observe several instances in the source data differing just in spatiotemporal shifts. We also propose a new algorithm to mine STInvs based on a prefix-projected sequential pattern growth approach and different quality metrics to quantify the contribution of the spatial invariance. The proposed approach is empirically evaluated on two mobility datasets related to a bike sharing system and traffic data. The results confirm the usability of the proposed solution in real-world scenarios.

***Index Terms*—Spatiotemporal Data, Mobility, Pattern Mining**

I. Introduction

With the incessant diffusion of IoT devices, analyzing spatio-temporal data has become particularly appealing in several application domains such as urban smart mobility [1], emergency management [2], and cybersecurity [3]. In the context of smart mobility, the use of spatiotemporal patterns to study the correlations among the observed events is established. For instance, in [4] and [5] the authors exploit spatiotemporal patterns to study the activities of bike sharing system users and perform short-term traffic congestion predictions, respectively.

Sequential pattern mining is an established data mining technique to discover relevant correlations among timestamped data. It extracts patterns representing recurrent instances of discrete event sequences [6], [7], [8]. They can provide actionable insights into data acquired in various domains, such as healthcare monitoring (e.g., [9]) and alarm management (e.g., [10]). Sequence mining allows discovering temporally correlated events [11]. To incorporate spatial information into event sequence mining, the following research lines have already been explored:

1) Discovering events/objects that are frequently located in close spatial proximity, namely the *co-location patterns* [12], [13], [14].
2) Exploring spatial trajectories reaching salient events/objects, such as the Points Of Interest (POI), and their transition times (e.g., from one POI to another) [15].
3) Characterizing the spatiotemporal neighborhood of an event by describing the dependencies between co-occurring and co-located geospatial entities using tree structures, namely the *propagation/influential patterns* [16].

Existing patterns do not provide an exhaustive view of spatiotemporal trends in data due to the following issues: (1) Co-location patterns neglect the temporal relations among events, i.e., they cope with static data snapshots rather than time-variant series; (2) Trajectory patterns embed only the temporal properties of restricted geospatial contexts, i.e., they do not describe relations among arbitrary located events; (3) Propagation/influential patterns focus on identifying chains of events, i.e., they potentially miss information about non-contiguous event sequences. A more thorough comparison with prior works is given in Section II.

The aim of the present work is to generalize geospatial event relations in mobility data by leveraging their spatiotemporal invariance. Given a (user-specified) *trigger event* of particular interest, we look for the associated sequences of events whose instances frequently show similar spatial and temporal relative gaps.

For example, if we are interested in studying the effects of a bike sharing station becoming full (i.e., all its docks are used and no further bikes can be returned) on the occupancy level of its neighboring stations, we define the bike sharing station becoming full as the relevant trigger event and automatically extract sequences of spatiotemporal event relations like the following one: If *an arbitrary bike sharing station becomes full*, then *we will observe an increase in the occupancy level of the bike sharing stations in the range from 250m to 500m from the station that becomes full within ten minutes*. Such a relation is defined by a new pattern, namely the SpatioTemporally Invariant pattern (STInv, in short), which can be expressed as follows:

$$\{\text{Station Full}\} \xrightarrow{10 \text{ mins}} \{\text{Increase occupancy level of neighbor station } (\delta s \in (250\text{m},500\text{m}]\})$$

L. Colomba, L. Cagliero, and P. Garza are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy.
E-mail: {luca.colomba,luca.cagliero,paolo.garza}@polito.it
Manuscript received ...

where *Station Full* is the spatiotemporal trigger event, whereas *Increase occupancy level of neighbor station* is the event that subsequently occurs after applying a spatiotemporal transformation. Notably, the transformation is characterized by a *relative spatial gap* (δs) ranging between 250m and 500m, and a *relative temporal gap* (δt) equal to 10 minutes, whereas is not constrained by any absolute spatial or temporal references.[1]

STInvs represent sequences of discrete events showing recurrent instances characterized by spatio-temporally invariant properties. Specifically, by applying a spatiotemporally invariant transformation to the trigger event (considered as the reference point), we can identify arbitrary instances. For example, moving 250m away from the location of the trigger event and/or waiting for 10 minutes after the trigger event occurrence time, we define the relative spatiotemporal dependence between the event occurrences. An STInv is deemed as relevant if the corresponding (spatiotemporally invariant) instances frequently occur in the sequential database.

We introduce an ad hoc quality metric to quantify the contribution of spatial invariance.

Let us consider again the STInv $\{\text{Station Full}\} \xrightarrow{10 \text{ mins}} \{\text{Increase of the occupancy level of neighbor station } (\delta s \in (250\text{m},500\text{m}]\})$. We define its *Spatial Invariance Compression Ratio* (SICR) as the number of full station events that actually triggered an increase in the occupancy level of the bike sharing stations within the spatial range [250m,500m] from the triggering event. This is deemed as potentially relevant to bike sharing station occupancy monitoring activities. For instance, end-users can configure the desired level of spatiotemporal invariance in bike sharing mobility data by ranking or filtering the event patterns by decreasing SICR value.

We also propose a new algorithm to mine STInvs, namely STInv-Miner. It relies on prefix-projected sequential pattern growth [8], which is defined on top of a sequential representation of the spatiotemporal event instances. The algorithm is developed in Apache Spark [17] to enable efficient spatiotemporal data processing in a distributed environment.

We apply STInvMiner on two real-world datasets from bike sharing systems and traffic data. The main findings include a subset of spatiotemporal patterns providing complementary knowledge compared to any existing approach.

The main paper contributions can be summarized as follows:

- **Spatiotemporal invariance property**. Given a trigger event, we leverage the property of spatiotemporal invariance to count all the event sequences whose occurrences show a *relative* temporal and spatial dependence.
- **New pattern**. We define a new pattern denoting event sequences whose instances frequently show spatiotemporal invariance.
- **New algorithm and quality metrics**. We propose a new algorithm that tailors prefix-projected sequential pattern growth to STInv extraction. Furthermore, we introduce ad hoc quality metrics to rank and filter the extracted patterns based on their spatial invariance.
- **Extraction of complementary information in real case studies**. We highlight the complementary knowledge provided by the STInvs extracted from real-world datasets compared to those conveyed by existing patterns such as propagation patterns.

The rest of the paper is organized as follows. Section II overviews the state-of-the-art in the research field. Section III formally introduces the newly proposed pattern. Section IV presents the STInv Miner algorithm, whereas Section V describes the analyzed data and summarizes the main experimental outcomes. Finally, Section VI draws conclusions and discusses the future research extensions.

II. Related works

Table I reports a comparison between the newly proposed STInvs and the existing spatiotemporal patterns. The classical sequential patterns proposed in [8], [7] are neither spatially nor temporally annotated. Temporal annotations approaches to traditional sequence mining have been proposed in [11], [19]. Both of the aforementioned patterns neglect spatial information. Trajectory and STAR patterns [15], [18] are tailored to specific spatial routes thus ignoring spatiotemporally invariant features. Co-location patterns [12], [13], [14] identify events/objects that are frequently located together in proximate areas despite that they neglect the temporal event correlations. Propagation/influential patterns [16] verify the presence of co-located and co-occurring chains of events. However, in real data, the observed events are not always contiguous.

Summary of the main differences with prior works. In STInvs events are not necessarily located in close spatial proximity (unlike co-location patterns), are not constrained by a predefined trajectory (unlike trajectory patterns), and can represent non-contiguous sequences of co-occurring and co-located events (unlike propagation/influential patterns).

Differences with its preliminary conference version. The present paper is an extended version of the preliminary work presented in [20]. The new contributions are enumerated below:

- *Previously missing formalization*: a more formal definition of the newly proposed pattern and its main properties (see Sections III-B and III-C).
- *New quality metrics*: we introduce ad hoc pattern quality metrics to quantify the contribution of spatial invariance (see Section III-F).
- *Insights into the features and complexity of the algorithm*: a deeper analysis of the presented algorithm (with pseudocode) and its complexity (see Section IV).
- *Additional examples and datasets:* More examples of patterns extracted from data acquired from a bicycle sharing system, possible decision-making actions supported by the proposed patterns, and a new dataset, namely the Large-Scale Traffic and Weather Events dataset (see Section V-A).
- *Implementation details and configuration settings*: More details on project code and experiments' reproducibility (see Section V).

[1]For the sake of simplicity, the reported example of STInv involves just two events and a simple transformation. However, in STInvs the event sequences have arbitrary length and size, and the transformation can be an arbitrary combination of spatial and temporal constraints.

TABLE I: Comparison with prior works.

Papers	Time	Space	Pattern	Description
[8], [7]	No	No	$e \rightarrow e'$	Events e and e' occur in the specified sequential order.
[11]	Yes	No	$e \xrightarrow{\delta t} e'$	For a sufficient number of sequences in the source dataset $e \rightarrow e'$ holds and the temporal gap between the occurrences of events e and e' is close to δt.
[15]	Yes	Yes	$(x, y) \xrightarrow{\delta t} (x', y')$	Given the locations with coordinates (x, y) and (x', y'), there are many trajectories visiting the same sequence of locations $(x, y) \rightarrow (x', y')$ with similar transition times.
[16]	Yes	Yes	$gse \rightarrow gse'$	Given the geospatial entities gse and gse' consisting of quadruples (type, start-time, end-time, location) gse and gse' co-occur and are co-located.
[18]	Yes	Yes	$(r, TI) \xrightarrow{\delta t} (r', TI')$	A trajectory of objects in spatial grid from region r to region r' within specified time intervals TI and TI'.
[12], [13], [14]	No	Yes	$\{f_1, \ldots, f_k\}$	Subset of spatial features f_j, $j \in$[1,k] whose instances are frequently located together in proximate areas.
STInv (our)	Yes	Yes	$\{e^{tr}\} \xrightarrow{\delta t} \{\langle e', \delta s' \rangle\}$	A trigger event e^{tr} followed by a sequence of discrete events e' showing spatiotemporally invariant instances. E.g., after the relative temporal gap δt we see event e' is observed at a $\delta s'$ spatial distance from e^{tr}.

- *Exploration of further research*: a study of the effect of spatial invariance property on the frequency of occurrence of the extracted patterns (see Section III-G) and a comparison with propagation patterns [16] (see Section V-C2b).

III. The SpatioTemporally Invariant Pattern

This section introduces the notation used and the preliminary concepts in Sections III-A, III-B, and III-C. Then, it formalizes the newly proposed pattern, its spatiotemporal invariance property, and the addressed mining task (see Sections III-D, III-E, and III-G).

A. Summary of the notation used

- A: the geographical area under analysis (e.g., the urban area where the bike sharing system is active).
- l: a geographical location within A (e.g., the location of a docking station).
- r: spatial resolution (i.e., the radius of the circular neighborhood) of a geographical location l (e.g., 500m can be chosen as the radius of the neighborhood of the bike sharing stations).
- td: temporal resolution (i.e., the duration of the considered time slots).
- $length$: max sequence length (i.e., the maximum number of time slots that appear in an input sequence).
- δs: spatial gap (multiple of r).
- δt: temporal gap (multiple of td).
- $\mathbf{E}$: set of event types under analysis (e.g., car accident, very low level of occupancy of a bike sharing station).
- $\mathbf{E}^{tr}$: trigger event types ($\mathbf{E}^{tr} \subseteq \mathbf{E}$).
- $\langle e,t,l \rangle$: occurrence of event type e at timestamp t in location l.

B. Preliminaries

We describe the occurrences of a set of discrete events of interest according to their spatiotemporal properties.

For example, the manager of a bike sharing system can be interested in recording and analyzing critical occupancy levels of the stations in terms of percentage of available docks [21].

Domain experts are asked to (1) Define a set of possible event types E and (2) Mark a subset of them, hereafter denoted by *trigger events*, as particularly relevant to effectively support decision-making. For example, when the occupancy level of a station is too high, a trigger event can be recorded. It is deemed particularly relevant because system managers can be interested in implementing specific actions to prevent service disruption.

When an arbitrary event is observed, we keep track of the corresponding location l and time t. The location indicates the geographical position where the event of type e took place. Similar to co-location patterns [12], locations are also characterized by a local neighborhood, which indicates the potential area of influence of an event occurrence. In our context, the local neighborhood of a location l is defined as a circular area of radius r centered in l. To enable a multi-resolution spatial analysis, we define various amplitudes of neighborhood represented by concentric circles of radius $k \cdot r$ centered in l ($k \in \mathbb{Z}^+$). Hereafter, it will be also denoted as *spatial resolution*.

The timestamp of occurrence of a discrete event t is discretized into discrete time slots of equal duration td. td indicates the temporal resolution at which we consider the occurrences of discrete events.

Example. In Figure 1 the temporal resolution is 15 minutes. All the events occurring in the same time slot (e.g., e_1 and e_2) are assumed to be concurrent. Domain experts are asked to set a temporal resolution suited to the domain under analysis.

A spatiotemporally annotated occurrence of event e at timestamp t in location l is denoted by a triplet $\langle e,t,l \rangle$. For the sake of brevity, we will also denote an event occurrence $\langle e_i, t_j, l_z \rangle$ where $e_i \in E$, $t_j \in T$, and $l_z \in A$ simply by occ_{ijz} whenever it is clear from the context.

C. The event sequence mining task

We explore temporal sequences of discrete events [7].

Definition 1: An *event sequence* of length n (also denoted by event n-sequence) is an ordered list of items, each one consisting of an event occurrence: [$occ_{e_1 t_1 l_1}$, $occ_{e_2 t_2 l_2}$, ..., $occ_{e_n t_n l_n}$], where $t_1 \leq t_2 \ldots \leq t_n$.□

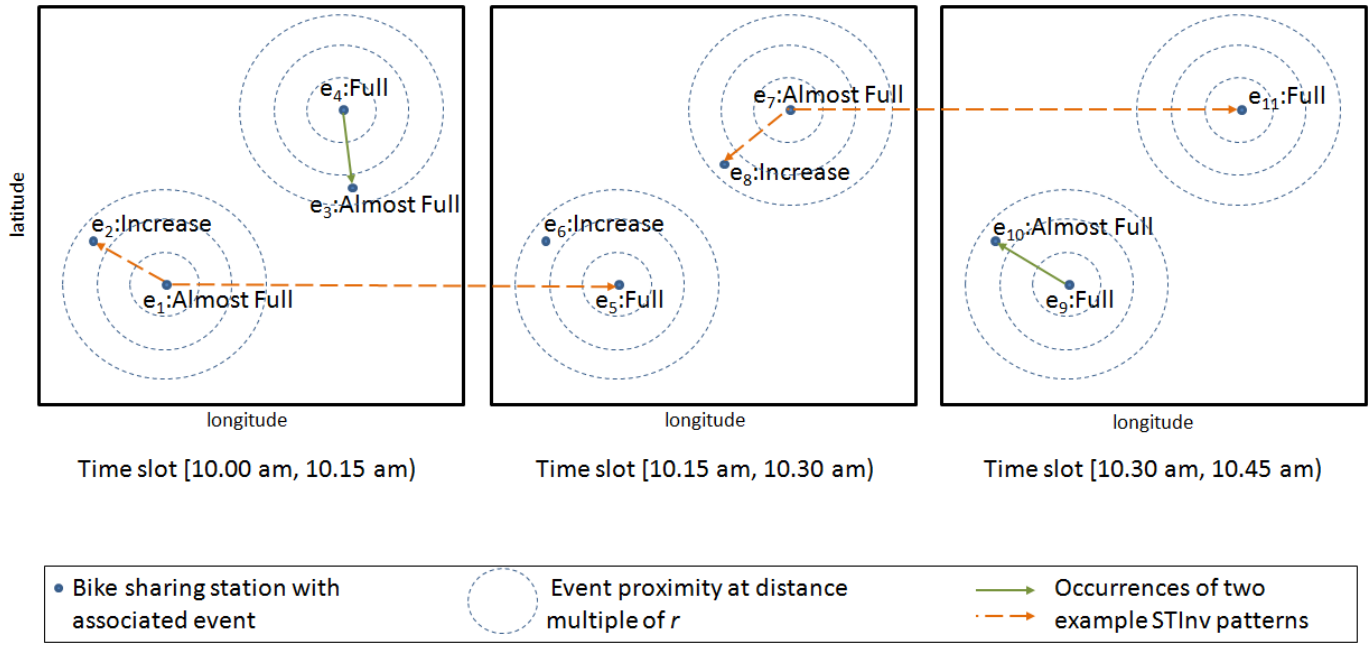


Fig. 1: Examples of occurrences of two SpatioTemporally Invariant patterns. The figure graphically shows the levels of occupancy of four bicycle sharing stations in three consecutive time slot snapshots (i.e., time slots [10.00 am, 10.15am), [10.15 am, 10.30am), and [10.30am, 10.45am)). The boxes in the upper part represent the 2D space where stations are located. The dashed circles indicate the level of proximity of stations with respect to the reference trigger events. Events indicate discrete states associated with stations' occupancy levels (e.g., Almost Full = above 80% of occupancy, Full = 100% of occupancy). The arrows represent instances of spatiotemporally invariant patterns. We use continuous and dashed lines to graphically differentiate the occurrences of two different example STInv patterns across the time slot snapshots. For example, the dashed lines are the occurrences of an STInv indicating that given two spatially nearby stations, one almost full (trigger event) and one with increasing occupancy at a given time slot, the one that is almost full will become full in the next time slot (likely because bikers in the neighborhood tend to saturate the almost fully occupied station).

An arbitrary sequence may include events of different types and contemporary events happening in different locations.

Example. The sequence $[occ_{e_1t_1l_1}, occ_{e_2t_1l_2}, occ_{e_3t_2l_2}]$ may consist of the following items: $occ_{e_1t_1l_1}$=*a bike sharing station in location* l_1 *becomes full at time* t_1, $occ_{e_2t_1l_2}$=*a bike sharing station in location* l_2 *becomes almost full at time* t_1, $occ_{e_3t_2l_2}$=*the bike sharing station in location* l_2 *becomes full at the following time* t_2. The first two events co-occur in two different locations at the same time, whereas the latter occurs later in the same place as the second event.

We are interested in extracting recurrent portions of the event sequences, commonly denoted by *subsequences*.

Definition 2: Let S_1 and S_2 be two distinct event sequences of length n and m, respectively. S_1 is a *subsequence* of S_2 if and only if (1) $n < m$, (2) every item contained in S_1 appears also in S_2 as well.□

For our purposes, we extract subsequences characterized by a (user-specified) starting event, hereafter denoted by *trigger event*. It indicates the event triggering a sequence of temporarily ordered events reported in the subsequence.

All the timestamps associated with the event occurrences are discretized into time slots (e.g., [10am, 10.15am)) with fixed duration td. Hereafter, we will also denote the user-specified duration td by the *temporal resolution* (which is typically defined by the domain expert).

To collect a sequence database consisting of a representative set of event sequences, we apply the classical time windowing approach [22]. It entails defining a time window of size n, which slides over the timeline. For every sliding step, it generates a new sequence of the event database that includes all the events occurring within the corresponding time window.

Example. Let us consider the time slot TS_1 including timestamps t_1 and t_2, time slot TS_2 including timestamps t_3, and time slot TS_3 including t_4. Let $occ_{e_1t_1l_1}$, $occ_{e_2t_1l_2}$, $occ_{e_2t_2l_1}$, $occ_{e_4t_2l_3}$, $occ_{e_1t_3l_1}$, and $occ_{e_5t_4l_4}$ be the observed events.

Let us define a window of size $2 \cdot td$ sliding over the time span and including pairs of consecutive time slots. The corresponding sequence database contains the following sequences $S_1, \ldots, S_3$ respectively corresponding to $TS_1, \ldots, TS_3$:

S_1: $[occ_{e_1t_1l_1}, occ_{e_2t_1l_2}, occ_{e_2t_2l_1}, occ_{e_4t_2l_3}, occ_{e_1t_3l_1}]$

S_2: $[occ_{e_1t_3l_1}, occ_{e_5t_4l_4}]$

S_3: $[occ_{e_5t_4l_4}]$

The sequence database D=$\cup_{i=1}^{3} S_i$ collects all the sequences generated by the sliding process.

D. The spatiotemporal invariance property

We look for the sequences of discrete events showing invariant spatiotemporal properties. To this end, we first introduce the concepts of temporal and spatial gaps, which indicate the sequential variation of the recorded timestamps and locations, respectively.

Definition 3: Let S=[$occ_{e_1t_1l_1}$, $occ_{e_2t_2l_2}$, …, $occ_{e_nt_nl_n}$] be an event n-sequence. The *temporal gap sequence* relative to S, $\delta t(S)$ in short, is the (n-1)-sequence [$t_2 - t_1$, $t_3 - t_1$, …, $t_n - t_1$]. □

The temporal gaps $t_j - t_1$ are multiples of the reference temporal resolution td.

Definition 4: Let S=[$occ_{e_1t_1l_1}$, $occ_{e_2t_2l_2}$, …, $occ_{e_nt_nl_n}$] be an event n-sequence. The *spatial gap sequence* relative to S, $\delta s(S)$ in short, is the (n-1)-sequence [dist(l_2,l_1), dist(l_3,l_1), …, dist(l_n,l_1)], where

$$dist(l_x, l_y) = \left\lceil \frac{haversine\_distance(l_x, l_y)}{r} \right\rceil$$

□

The spatial distance between two event occurrences depends on the chosen spatial resolution r.

Example. In the left-hand side picture of the toy example in Figure 1, the (discretized) spatial gap between events e_1 (trigger event) and e_2 is equal to $3 \cdot r$ because event e_2 lies in the third concentric circle of the trigger event's neighborhood.

Two n-sequences are spatially/temporally invariant if their corresponding spatial/temporal gap sequences are coincident.

Definition 5: Let S_1 and S_2 be two event n-sequences and let $\delta t(S_1)$ and $\delta t(S_2)$ be the corresponding temporal gap sequences. S_1 and S_2 are *temporally invariant* if and only if $\delta t(S_1)$=$\delta t(S_2)$. □

Definition 6: Let S_1 and S_2 be two event n-sequences and let $\delta s(S_1)$ and $\delta s(S_2)$ be the corresponding spatial gap sequences. S_1 and S_2 are *spatially invariant* if and only if $\delta s(S_1)$=$\delta s(S_2)$. □

Example. Let us consider the following sequences: S_1: *a bike sharing station in location l_1 becomes almost full at 9 am*, then *the same bike sharing station located in l_1 becomes full at 9:15 am*. S_2: *a bike sharing station in location l_2 becomes almost full at 11 am*, then *the same bike sharing station located in l_2 becomes full at 11.15 am*.

S_1 and S_2 are spatiotemporal invariant sequences because the corresponding pairs of events occurred within the same position (spatial gap=0) with a temporal gap of 15 minutes.

The key idea is to leverage the spatiotemporal invariance in sequential data to identify event correlations that are worth considering for decision-making.

E. The new pattern

We introduce a new type of pattern, namely the SpatioTemporally Invariant (STInv) pattern. The main purpose is to identify and characterize the most relevant spatiotemporally invariant trends in sequence databases. An STInv summarizes a set of event sequences S_1, S_2, …, S_q such as all the pairs of summarized sequences (S_i, S_j) $1 \leq i, j \leq q$ are spatiotemporally invariant.

Definition 7: Let S_1, S_2, …, S_q be a set of q spatiotemporally invariant sequences sharing the same trigger event as first event type ($e_1 \in \mathbf{E}^{tr}$). The STInv is a sequence of triplets [⟨ e_1, δs_1, δt_1 ⟩, ⟨ e_2, δs_2, δt_2 ⟩, …, ⟨ e_n, δs_n, δt_n ⟩] where e_x is x-th event type in S_1, S_2, …, S_q, δs_x and δt_x are the x-th spatial and temporal gap values in the corresponding sequences. □

Example. Figure 1 shows an example of event sequences related to the monitoring of the occupancy of the bike sharing stations in an urban environment. Specifically, the monitored events, i.e., *fully occupied, almost fully occupied, and increasing occupancy level* indicate the levels of occupancy of the stations of a bike sharing system, expressed in terms of percentage of available docks. We consider as temporal resolution the 15-minute time slots depicted as consecutive time frames. To model the neighborhood of trigger events e_1 and e_4, we set the spatial resolution to r=100m and plot surrounding circles indicating the spatial distances $1 \cdot r$=100m, $2 \cdot r$=200m, and $3 \cdot r$=300m, respectively.

The two STInvs in Figure 1 are depicted as oriented splines connecting different occurrences of spatiotemporal events. Specifically, they are represented as a dashed line and a continuous line, respectively.

Pattern example 1. The first occurrence of the STInv with trigger event type (*Full*) is observed in the time slot [10.00am, 10.15am) (at the upper-right corner of the spatial area). The pattern indicates the correlation between events e_4 (*Full*) and e_3 (*Almost full*), which co-occur in the same time slot at a spatial distance ranging between $2 \cdot r$=200m and $3 \cdot r$=300m, i.e., (200m, 300m]. Notice that the same pattern has another spatiotemporally invariant occurrence in the time frame [10.30am, 10.45am) (at the lower-left corner of the respective spatial area). The aforesaid pattern can be expressed as follows:

[⟨ Full, δs=0m, δt=0 ⟩, ⟨ Almost Full, δs=(200m,300m], δt=0 ⟩]

Since, by construction, the trigger event *Full* is the origin of the spatiotemporal hyperspace under consideration, for the sake of readability hereafter we will simplify the notation as follows:

[Full, ⟨ Almost Full, δs=(200m,300m], δt=0 ⟩]

Pattern example 2. Let us consider now the STInv pattern with trigger event (*Almost full*). It is observed in the time slot [10.00am, 10.15am). It shows a spatial correlation between events e_1 (*Almost full*) and e_2 (*Increase*). More specifically, e_2 occurs in the neighborhood of the trigger event at a distance ranging between $2 \cdot r$=200m and $3 \cdot r$=300m, i.e., (200m, 300m]. Next, 15 minutes after the occurrence of the trigger event e_1, we can also observe in the same location the occurrence of event e_5 (*Full*). Such a combination of spatiotemporal events (e_1-e_2 and then e_5) is worth considering because another spatiotemporally invariant occurrence can be observed at the upper-right corner of the spatial areas in time slots [10.15am, 10.30am) and [10.30am, 10.45am) (events

e_7-e_8 and then e_{11}). The latter STInv can be formulated as follows:

[Almost Full, ⟨ Increase, δs=(200m,300m], δt=0 ⟩, ⟨ Full, δs=0m, δt=15mins ⟩]

To further summarize the pattern expression and simplify human exploration, we can omit the temporal gap indication at the event occurrences while reporting the relative temporal gap δt observed between the event sets occurring in different time slots, e.g.,

{Almost Full, Increase(δs=(200m,300m])} $\xrightarrow{15 \text{ mins}}$ {Full(δs=0m)}

Notice that (1) Adopting the above notation is feasible because we focus on *relative* time variations rather than on absolute timestamps; (2) The former pattern cannot be reformulated using the same compact form because all the reported events occur within the same time slot.

In summary, the STInvs depicted in Figure 1 can be interpreted as follows:

- *Pattern example 1*: If *a bike sharing station is full* then *the occupation level of at least one of its nearby stations (between 200m and 300m) is contemporary almost full*.
- *Pattern example 2*: If *a bike sharing station is almost full and the occupation level of at least one of its nearby stations (between 200m and 300m) is increasing* then *within 15 minutes the occupancy of the same station will get full*.

F. The quality measures

a) Support: Similar to traditional sequence mining techniques [7], the STInv mining process is driven by frequency-based quality indices. The *support* index counts the number of occurrences of an STInv in a sequence database.

Definition 8: Let D be a sequence database. The *support* of STInv in D is the number of corresponding event sequences in D.□

Let us consider the STInv {Almost Full(δs=0m)} $\xrightarrow{15 \text{ mins}}$ {Full(δs=0m)} depicted in Figure 2. According to the windowing approach, its corresponding event sequences are

S_1: [occ_{e_1,TS_1,l_1}, $occ_{e_3TS_2l_1}$]
S_2: [occ_{e_2,TS_1,l_2}, $occ_{e_4TS_2l_2}$]

So the STInv support is 2. Notice that the sequence of temporally consecutive events *Almost Full* and *Full* described by the STInv is spatially invariant, meaning that the support counts both the occurrences in l_1 and l_2.

b) Location-conditioned support: By relaxing the spatial invariance property, we can redefine the concept of support measure conditioned to a specific sequence of locations.

Definition 9: Let D be a sequence database and let S_L be a sequence of locations in which any event sequence in D occur. The *support* of STInv in D *conditioned to* S_L is the number of event sequences in D corresponding to both the STInv and S_L.□

For example, recalling the previous example the support of the STInv conditioned to the location sequence S_L: [l_1, l_1] is 1 because the occurrence of the STInv in l_2 is disregarded. Notice that by enforcing the spatiotemporal invariance the support of the STInv counts the pattern occurrences in any location sequence in D.

$$\text{support}(STInv, D) = \sum_{S_L \text{ in } D} \text{location-conditioned support}(STInv, D, S_L)$$

Thus, spatial invariance generalizes the traditional concept of temporal sequence toward different (yet comparable) spatial contexts.

c) Spatial Invariance Compression Ratio: Given an STInv, we can quantify the spatial invariance compression contribution by computing the number of location sequences associated with the event sequences associated with STInv.

Definition 10: Let [⟨ e_1, δs_1, δt_1 ⟩, ⟨ e_2, δs_2, δt_2 ⟩, …, ⟨ e_n, δs_n, δt_n ⟩] be an STInv and let $\mathbf{S}$ be the set of corresponding event sequences. Let $\mathbf{S}_L$ be set of location sequences S_L in $\mathbf{S}$. The Spatial Invariance Ratio is defined as follows.

$$SICR(STInv) = |\mathbf{S}_L|$$

□

For example, let us consider the STInv {Almost Full(δs=0m)} $\xrightarrow{15 \text{ mins}}$ {Full(δs=0m)} again and the events reported in Figure 2. The two location sequences [l_1, l_1] and [l_2, l_2] are associated with the example STInv. Hence, its SICR(STInv) is 2. Hence, thanks to its spatial invariance property, the example STInv compactly represents a sequence of events occurring in two distinct location sequences.

G. The pattern mining task

We look for the STInv patterns that summarize a large enough set of event sequences in the input sequence database, namely the *frequent* STInv.

Definition 11: Let $minsup$ be a (user-specified) minimum support threshold and D be a sequence database. A *frequent* STInv in D is an STInv whose support in D is above $minsup$.□

Given a sequence database consisting of a large set of event sequences, our purpose is to automatically extract all the frequent STInv patterns.

IV. The STInv-Miner algorithm

In this section, we describe a new algorithm, namely STInv-Miner, that is used to address the pattern mining task defined in Section III-G. It takes as input the sequence database D, a (user-specified) minimum support threshold $minsup$, the spatial and temporal resolutions r and td, and mines all frequent STInv patterns.

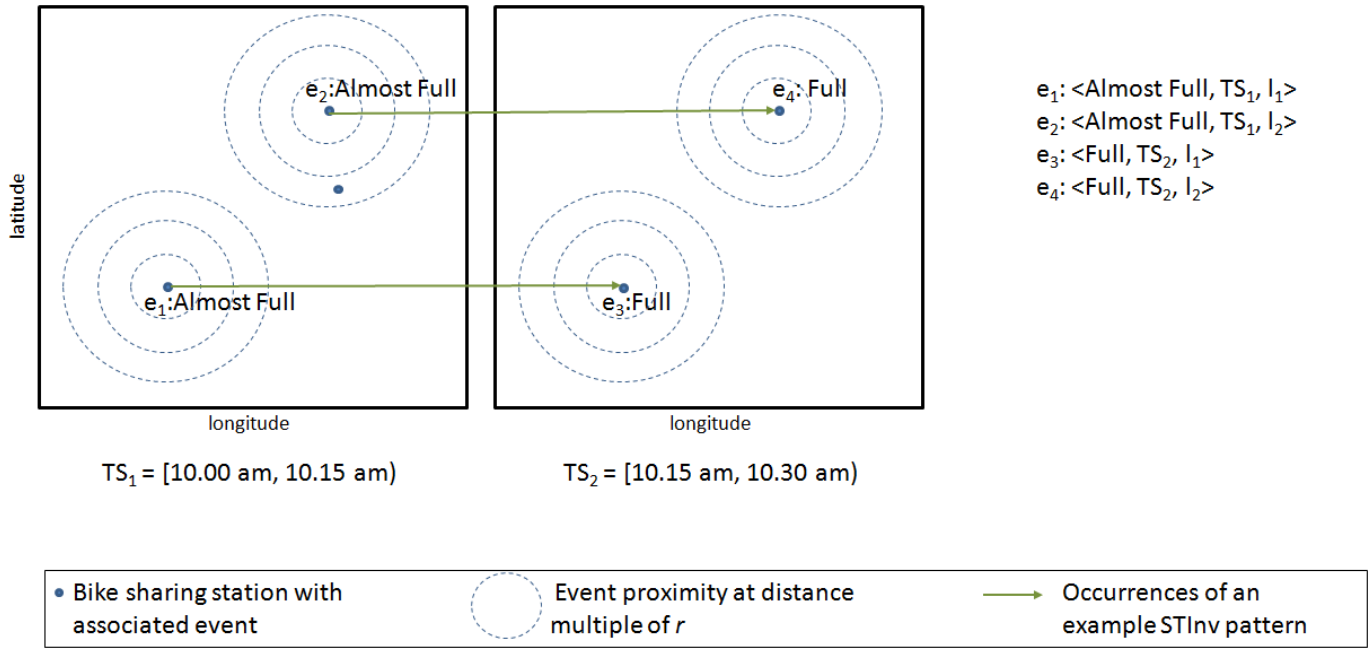


Fig. 2: Occurrences of an example STInv.

Algorithm 1 The STInv-Miner algorithm

1: **Input**: Raw event sequence database D, spatial resolution r, temporal resolution td, minimum support threshold $minsup$
2: **Output**: Set of frequent STInv $FSTinv$
3: **procedure** STINV-MINER(D, r, td, $minsup$)
/ Sequence projection step */*
/ Project the input sequences from D to the projected sequence database D^p */*
4: $D^p = \emptyset$
5: **for all** sequences S_i in D **do**
6: **for all** trigger events e_j in $E^{tr}(S_i)$ **do**
7: $D^p = D^p \cup S_j^p$ ▷ S_j^p is the projected sequence with respect to the trigger event e_j of S_i
8: **end for**
9: **end for**
/ Mine STInvs from D^p using any traditional sequence mining algorithm. */*
10: $FSTinv$ = SequenceMining(D^p) ▷ We use PrefixSpan to implement SequenceMining
11: **end procedure**

STInv-Miner is based on a two-step approach that leverages a projection-based procedure combined with any traditional sequence mining algorithm to efficiently mine STInv patterns (see Algorithm 1). Thanks to the first step (Lines 4-9 of Algorithm 1), which tailors the concept of sequence projection to the STInv case, the second step of the mining procedure can leverage the efficiency of any centralized or parallel sequence mining algorithm proposed in the past (Line 10) to mine STInv patterns. In the current implementation, we rely on a Spark-based parallel implementation of the state-of-the-art PrefixSpan [8] sequence mining algorithm to scale on large datasets, i.e., the SequenceMining procedure (Line 10 of Algorithm 1) is based on PrefixSpan.

The key idea of the first step is to transform the original event sequences, annotated with absolute temporal and spatial information, into a set of equivalent spatiotemporally invariant sequences containing their corresponding projected versions including relative time and spatial gaps with respect to each trigger event. This is instrumental for efficiently verifying the presence of temporally invariant patterns in the event sequences leveraging any traditional sequence mining algorithm in the second step.

For each trigger event in the input sequence, we generate a projected sequence from the original one that contains the trigger event and the non-trigger events annotated with their relative spatial and temporal distance with respect to the trigger event, i.e., the spatial and temporal gaps previously defined in Section III-D. The projected sequences corresponding to all event sequences in D are stored in a projected sequence database D^p. A more formal definition of projected sequence follows.

Definition 12: Let S_i be an event n-sequence and let $E^{tr}(S_i)$ be the set of the q trigger events in the first time slot of S_i (which are a subset of the whole event set $E(S_i)$ occurring in S_i). The projection function maps each S_i to a set of q distinct sequences S_1^p, S_2^p, …, S_q^p, namely the *projected sequences*. Each projected sequence corresponds to a distinct trigger event in $E^{tr}(S_i)$. The j-th projected sequence S_j^p is defined as follows:

$$[\langle\ e^{tr},\ \delta s = 0,\ \delta t = 0\ \rangle,\ \langle\ e_2,\ \delta s_2,\ \delta t_2\ \rangle,\ \ldots,\ \langle\ e_m,\ \delta s_m,\ \delta t_m\ \rangle]$$

where e^{tr} is the j-th trigger event, e_2, …, e_m are the events in S_i that are spatially correlated with e^{tr}, and the corresponding

δs and δt are spatial and temporal gaps relative to the trigger event. □

Algorithm 2 Parallel version of the Sequence projection step

1: **Input**: Raw sequence database D, spatial resolution r, temporal resolution td
2: **Output**: Projected sequence database D^p
3: **procedure** MAP(Identifier of S_i in D, sequence of events S_i)
4: **for all** trigger events e_j in $E^{tr}(S_i)$ **do**
5: Initialize the projected sequence S_j^p as an empty list of lists
6: **for all** occurrences of events e in S_i **do**
7: **if** e is spatially correlated with trigger event e^j **then**
8: Compute spatial gap δs
9: Compute temporal gap δt
10: item = $\langle e, \delta s, \delta t \rangle$
11: $S_j^p[\delta t] = S_j^p[\delta t] \cup$ [item]
12: **end if**
13: **end for**
14: emit(S_j^p, NULL) ▷ D^p is the union of all the returned projected sequences
15: **end for**
16: **end procedure**

To enable efficient STInv pattern mining, a parallel version of the STInv-Miner algorithm was implemented using the Apache Spark framework [17]. A pseudocode of the parallel and distributed sequence projection step (first step of STInv-Miner), equivalent to the sequential code reported in Lines 4-9 of Algorithm 1, is described in Algorithm 2 using the MapReduce-based paradigm. It processes multiple raw input sequences in parallel by assigning a set of input sequences to each server used to run our application. Specifically, the Map procedure is executed once per input sequence S_i and returns the corresponding set of projected sequences (one for each trigger event). Finally, the second step uses the parallel version of PrefixSpan available in Spark to parallelize that step as well.

The implementation of the parallel version of the code can be found at https://github.com/lccol/stinv-miner (latest access: May 2025).

A. Complexity analysis

The sequence projection step described by Algorithm 1 entails the extraction of all the projected sequences from an input sequence database D. The size of the projected sequence database can be estimated as $|D| \cdot E^{tr}_{avg}$, where E^{tr}_{avg} corresponds to the average number of trigger events present in an original event sequence. Hence, an estimate of the time complexity of the projection step is O($|D| \cdot E^{tr}_{avg}$).

The second step of the STInvMiner algorithm relies on the PrefixSpan algorithm [8] run on top of the projected database (see Algorithm 1). The complexity of PrefixSpan is linear with the number of extracted sequences. Given the set Tr of distinct triplets $\langle e, \delta s, \delta t \rangle$ in D, in the worst case scenario (minsup=1) the time complexity is O($2^{|Tr|}$) [23][2]. Both the aforesaid steps can be parallelized over n workers. An empirical analysis of the algorithm scalability on real-world datasets is reported in Section V.

V. EXPERIMENTS

This section is organized as follows. Section V-A describes the bike sharing and traffic data analyzed and the pre-processing steps applied to the raw data. Section V-B provides details on the reproducibility of our research work. Section V-C reports a characterization and analysis of the extracted patterns, including a comparison with traditional sequences and more recent spatiotemporal patterns [16]. More specifically, the analyses encompass the extraction of patterns within a bike sharing system over five cities, followed by a comparison between the developed patterns and tree-based ones [16] applied on a traffic congestion dataset over the entirety of the USA and three specific cities.

A. Datasets

We applied our new patterns in two diverse mobility domains: the bike sharing domain and the traffic flow context. Two distinct datasets, one for each domain, have been used. Moreover, two distinct campaigns of experiments have been conducted to show the generality of the proposed pattern. The two datasets are introduced in the following.

Bike Sharing Dataset. It collects anonymous information about the usage of the Bay Area docked bike sharing service from August 2013 to August 2015. Data are annotated with the geographical location of all the stations in the system. More specifically, the dataset spans over 5 different cities: San Francisco, San Jose, Redwood City, Mountain View, and Palo Alto, which consist of 35, 16, 7, 7, and 5 stations, respectively. Per-station occupancy levels are recorded every minute, 24 hours a day, seven days a week. This results in a 35 million-row dataset by exclusively considering the San Francisco stations.

To remove noisy measurements, we early pruned all the per-station occupancy levels for which the sum of parked bikes and the free docks exceeded the station capacity by 5 or more.

To characterize the occupancy levels of the stations in the system, we generate sequences of the following event types:

- *Increase* in the station's occupancy level, i.e., at least one occupancy level increase is recorded within the current time slot at the considered station;
- *Decrease* in the station's occupancy level, i.e., at least one occupancy level decrease is recorded within the current time slot at the considered station;
- *Almost full* occupancy level, i.e., the number of available docks at the station takes the value 1 or 2 at least one time in the current time slot;
- *Full* occupancy level: the number of available docks reduces to zero at least once within the current time slot at the considered station.

[2]By enforcing a minsup>>1 the complexity is orders of magnitude lower.

TABLE II: Recommended configuration settings.

	Config 1	Config 2	Config 3
Dataset	Bike Sharing	Bike Sharing	LSTW
Temporal Resolution (td)	10 min	10 min	10 min
Maximum window size (nr. of time slots)	6	6	3
Spatial resolution (r)	100m	100m	100m
Geographical area	Global, City	Global, City	Global, NYC, LA, Boston
Max pattern length	6	6	10
Event types	Full, Almost Full, Increase	Full, Almost Full, Increase, Decrease	Traffic and Weather Events
Trigger event types	Full, Almost Full, Increase	Full, Almost Full, Increase	Traffic Events only

The definition of the event types ($\mathbf{E}$) and the selection of the subset of trigger events ($\mathbf{E}^{tr}$) depend on the application context under analysis. Thus, their definition is driven by domain experts. Based on the common knowledge about bicycle sharing systems, we recommend Configurations 1 and 2 reported in Table II. Configuration 2 differs from 1 in two main aspects: (1) The definition of the event types, which also includes the *Decrease* in the station occupancy level, and (2) The identification of the station neighborhood is solely based on spatial distance in Configuration 1 (r=500m) whereas, in Configuration 2, it depends on the top-N departure stations. In a nutshell, Configuration 2 allows us to investigate the indirect effect of a decrease in a station occupancy, which is likely to induce an increase in the occupancy level of another station.

a) Large-Scale Traffic and Weather Events dataset: The LSTW dataset records salient meteorological (e.g., rain, snow) and traffic (e.g., car accident, traffic congestion) events that occurred in the United States of America from August 2016 to December 2020. Traffic events are geo-localized: the dataset stores specific information about (1) the geographical coordinates in which the event took place, (2) an approximate street address derived from latitude and longitude, (3) the event type (e.g., *Car accident*), (4) the starting and ending timestamps,[3] and (5) the nearest airport code. Weather events are characterized by (1) event type (e.g., *Snow*) and (2) airport code from which the event was acquired. The total number of events stored in the dataset is over 36 million. Additional information about the LSTW dataset can be found in [16].

In compliance with [16], we removed the duplicated events and merged together the traffic events of the same type that co-occurred in a 10-minute time slot within a 100m radius. Similarly, meteorological events of the same type and with the same corresponding airport code are merged using the following time thresholds: (i) *Snow*: 30 minutes, (ii) *Rain*: 15 minutes, and (iii) *Other*: 10 minutes. Temporal gaps are computed based on the event start time for both weather and traffic events. This operation is important to remove the redundancy of some events and prevent the beginning of two identical events within the same temporal span. Thus, if any accident takes place in a specific location, followed by an accident within a 10-minute and 100m distance from the original event, the two events are merged. The resulting event is associated with start time and end time being the minimum and maximum start/end time of the two events, respectively.

[3]Note that the information about the duration of the historical events stored in this dataset has been provided by the authors of the dataset, who collected the information about the events and their occurrence time.

B. Reproducibility

To make our experiments fully reproducible, we release the project code (written in Spark) and the full scripts used for testing.[4]

C. Pattern extraction and analysis

Here we summarize the results achieved on the bike sharing and LSTW datasets with various configuration settings. Whenever not otherwise specified, the *minsup* threshold used in the experiments is equal to 1.[5]

1) Bike Sharing Dataset: We perform several pattern extractions on geographical areas of different sizes, i.e., global-sized (i.e., the entire dataset) and city areas.

We explore the outcomes achieved with Configuration 1 with the twofold aiming at (1) mining local patterns in which users gradually fill stations and (2) understanding how neighbor stations interact with each other. Furthermore, we also explore Configuration 2 to identify long-term spatial departure-arrival patterns. In the latter configuration, the most typical case involves users picking up their bicycles from distant stations and reaching the arrival station within one hour.

a) Statistics on events and STInv patterns: We analyze the distribution of the single event types. Specifically, Table III reports the percentages of event occurrences per type. 5% of the overall events in San Francisco (Configuration 1) are critical, as they are associated with the trigger event type *Full*. Conversely, the trigger event *Almost full* is associated with roughly 30% of the events in Palo Alto (Configuration 1).

Table IV summarizes the number and characteristics of mined STInvs. By considering the entire geographical area, more than 45 million patterns are mined using Configuration 1, whereas a few more than 10 million STInvs are extracted with Configuration 2. Notably, this goes against the prevailing trend because the overall number of events in Configuration 1 is smaller than those in Configuration 2, where departure-arrival relations are explicitly considered.

The average number of event occurrences per STInv is close to 6 (see Table V). Since they are likely to show different spatial/temporal gaps, they potentially represent non-trivial spatiotemporally invariant patterns that a human expert is unlikely to detect during a manual inspection of the analyzed data.

[4]https://github.com/lccol/stinv-miner Latest access: May 2025

[5]The distributed STInvMiner algorithm succeeded in extracting relatively infrequent STInvs as well.

TABLE III: Number of generated events for the bike sharing dataset.

		Full (%)	Almost Full (%)	Increase (%)	Decrease (%)	Total events
Global	Conf 1	3.94	22.12	73.94	N/A	1075347
	Conf 2	2.26	12.67	42.35	42.72	1877101
San Francisco	Conf 1	5.03	23.45	71.52	N/A	707572
	Conf 2	2.92	13.60	41.47	42.01	1220209
San Jose	Conf 1	2.25	19.82	77.93	N/A	175037
	Conf 2	1.26	11.13	43.78	43.83	311608
Redwood City	Conf 1	0.29	7.41	92.30	N/A	62340
	Conf 2	0.15	3.85	47.94	48.06	120024
Mountain View	Conf 1	1.71	18.16	80.13	N/A	56283
	Conf 2	0.95	10.12	44.64	44.29	101023
Palo Alto	Conf 1	2.39	30.26	67.35	N/A	74115
	Conf 2	1.42	18.05	40.18	40.35	124237

TABLE IV: Number of STInv patterns extracted from the bike sharing dataset.

	Config 1	Config 2
Global area	45298359	10975205
San Francisco	45289660	6235864
San Jose	2809205	2847177
Redwood City	82961	498463
Mountain View	178478	783465
Palo Alto	110591	344150

b) Comparison with traditional event sequences: We compute the percentage of STInvs including at least one $\delta s > 0$ (i.e., STInvs including also events that do not occur in the same location). Similarly, we also compute the number of patterns with at least one $\delta t > 0$ (i.e., patterns that disregard variations in the temporal dimension). In both cases, the computed values are above 99.9% (see Table V) thus confirming the added value of exploring STInvs rather than simpler event sequences. Most of the STInvs (99.2%) show both non-zero temporal and spatial gaps.

c) Comparison with non-spatially invariant sequences: STInvs are spatially and temporally invariant. Hence, they summarize event correlations regardless of the absolute event location and timestamp. We compare the number of STInvs with the number of traditional sequences mined by considering the absolute location of the events (see Table VI), i.e., we relax the spatial invariance constraint to explore its impact on the mined patterns. In the bike sharing context, the mined sequences simply represent temporal correlations among stations.

TABLE V: Statistics on the bicycle sharing dataset. Global area.

	Config 1	Config 2
Total num. of STInvs	45298359	10975205
Mean #triplets per sequence	5.90	5.77
Mean discrete spatial distance	2.40	8.31
Mean discrete temporal distance	2.13	2.17
#sequences with at least one $\delta s > 0$ (%)	99.95	99.89
#sequences with at least one $\delta t > 0$ (%)	99.99	99.99
#sequences with at least one $\delta s > 0$ *and at least one* $\delta t > 0$ (%)	99.29	98.36

TABLE VI: Spatial Invariance Compression Ratio (*SICR*) of the STInv patterns and comparison with the absolute non-spatially invariant patterns (namely Absolute).

	Config 1		
	STInv	Absolute	Average SICR(STInv)
San Francisco	45289660	2350326927	51.90
San Jose	2809205	7865362	2.80
Redwood City	82961	157216	1.90
Mountain View	178478	238923	1.34
Palo Alto	110591	142679	1.29
	Config 2		
	STInv	Absolute	Average SICR(STInv)
San Francisco	6235864	28241600	4.53
San Jose	2847177	6116185	2.15
Redwood City	498463	1130338	2.27
Mountain View	783465	1353856	1.73
Palo Alto	344150	591598	1.72

Let us consider an example to clarify the advantage of STInvs compared to traditional non-spatial invariant sequences. For example, the STInv pattern $\{\text{Almost Full}(\delta s\text{=0m})\} \xrightarrow{15 \text{ mins}} \{\text{Full}(\delta S\text{=0m})\}$ is a summarized version of the set of non-spatially invariant patterns $\{\text{Almost Full}(Station_x)\} \xrightarrow{15 \text{ mins}} \{\text{Full}(Station_x)\}$, where $Station_x$ is an arbitrary station. According to the mined pattern, if $Station_x$ is almost full then $Station_x$ will become full within the next 15 minutes. The compactness of STInv provides a clear advantage compared to simpler pattern versions as it facilitates domain experts' decisions reducing the number of patterns to analyze. This compression ratio is measured by means of the newly proposed Spatial Invariance Compression Ratio (SICR) index.

In San Francisco (i.e., the largest city of the bicycle sharing dataset), the average SICR is 51.9 under Configuration 1 (see Table VI). This means that, beyond the additional information inherently provided by the newly proposed patterns, each STInv represents a group of almost 52 non-spatially invariant patterns and the corresponding almost 52 location sequences on average.

d) Qualitative analysis: Table VII reports some representative STInv patterns mined from San Francisco using Configuration 1. Table VIII reports similar information using Configuration 2. All the selected patterns are in the top-100 most frequent STInvs.

The first three STInv patterns in Table VII are the three most frequent ones in the analyzed data. They model the inherently stationary behavior of the bike sharing stations. Specifically, they indicate that the types of events occurring in a station will likely remain unchanged in the short term (i.e., in the next time slots). For instance, according to the first STInv, if a station is almost full then within the next 10 minutes it is still almost full in 187546 cases. The fourth and the fifth STInvs represent highly frequent correlations between the trigger event and the occupancy level of the stations in the neighborhood within the next 10 minutes. Specifically, the fourth STInv indicates that an increasing occupancy trend of the station associated with the trigger event is frequently associated with an increasing occupancy level of the stations at a distance between 400m

TABLE VII: Examples of frequent STInv patterns. San Francisco. Configuration 1.

STInv pattern	Support	SICR
{Almost Full(δs=0m)} $\xrightarrow{10\text{ mins}}$ {Almost Full(δs=0m)}	187546	35
{Increase(δs=0m)} $\xrightarrow{20\text{ mins}}$ {Increase(δs=0m)}	173616	35
{Increase(δs=0m)} $\xrightarrow{10\text{ mins}}$ {Increase(δs=0m)}	173082	35
{Increase(δs=0m)} $\xrightarrow{10\text{ mins}}$ {Increase(δs=(400m, 500m])}	114241	48
{Increase(δs=0m), Increase(δs=(400m,500m])} $\xrightarrow{10\text{ mins}}$ {Increase(δs=(400m, 500m])}	59047	71

TABLE VIII: Examples of frequent STInv patterns. San Francisco. Configuration 2.

STInv pattern	Support	SICR
{Increase(δs=0m), Decrease(δs=(1.3km, 1.4km])} $\xrightarrow{10\text{ mins}}$ {Increase(δs=0m)}	8752	2
{Increase(δs=0m)} $\xrightarrow{10\text{ mins}}$ {Decrease(δs=(1.3km, 1.4km])} $\xrightarrow{10\text{ mins}}$ {Increase(δs=0m)}	8453	2
{Increase(δs=0m), Decrease(δs=(1.6km,1.7km])} $\xrightarrow{10\text{ mins}}$ {Increase(δs=0m)}	8335	2

and 500m ($5 \cdot 100$m), i.e., the increasing occupancy level of a station is often spread across its neighborhood. The SICR of this STInv is 71. This means that domain experts can make a decision by analyzing one single STInv instead of 71 non-spatial invariant patterns representing the same sequence of events.

Table VIII reports some representative STInv patterns obtained using Configuration 2. Since the most frequent STInvs are the same as in Table VII, we omit them for the sake of brevity. In Configuration 2 we can also observe patterns that are associated with the decreasing status of the departure stations. The reported STInvs show how a decreasing occupancy level in the departure stations affects the increasing occupancy levels of the stations associated with the trigger event (i.e., the destination of the trip). All three examples of patterns show the inverse relationship between departure and arrival stations: an increase in the occupancy level of the arrival stations yields a decrease in the departure ones and vice versa.

e) Decision-making based on STInvs for bike sharing system management: Domain experts can leverage the mined STInvs to study the spatial and temporal relations between the bike sharing stations' occupancy levels. For instance, STInvs can be used to predict in advance if the occupancy level of some stations near a reference station will increase in a medium/short time. The pattern {Increase(δs=0m)} $\xrightarrow{10\text{ mins}}$ {Increase(δs=(400m,500m])} (see Table VII) is an example of an STInv that can potentially support occupancy level predictions and trigger specific actions. If the occupancy of a (reference) station is increasing, then it frequently happens that the occupancy level of some of its neighbor stations will also increase in the next 10 minutes. This information can be used to trigger a rebalance operation in the neighborhood of the reference station, prevent an overload situation of the stations, and, hence, a disservice. It is worth noticing that, thanks to their spatial invariance property, STInv patterns can be applied to new stations for which the location is known even in the absence of historical data. Conversely, traditional non-spatially invariant patterns (e.g.,[16]) cannot be applied to new stations.

Discovering spatio-temporally invariant relations between station occupancy levels can be potentially helpful to either plan the creation of new stations in poorly served areas or reshape the service provision in the whole urban area. For instance, to prevent service disruption, the discovery of an STInv pattern that represents a frequent correlation between the almost full status of a set of neighbor stations can trigger the construction of new stations in the areas matching the pattern. The Spatial Invariance Compression Ratio (*SICR*) index can be helpful to understand if the critical pattern is generally occurring in many sequences of locations and, hence, if there is a potential criticality affecting a large majority of the areas where the stations are currently installed. The higher the value of the SICR index, the higher the pattern's generality/spatial invariance and impact.

2) LSTW dataset: We perform four separate tests: an STInv extraction performed over the entire dataset and 3 separate extractions on data acquired from the large U.S. cities available in the benchmark dataset, i.e., New York City, Boston, and Los Angeles, which are among the largest cities available in the dataset. The aim of this analysis is to evaluate the informativeness of the extracted patterns in the context of traffic management and possible associations between different types of event, including the weather information. Finally, we compare the outcome of our analysis with a state-of-the-art methodology on the same dataset [16].

a) Statistics on events and STInv patterns: The number of mined STInv patterns increases with both the city size and the number of considered events (see Table IX for the main STInv characteristics). Even on the LSTW dataset, the majority of the mined patterns are characterized by non-zero spatial or temporal gaps. In the global geographical scenario, more than 98% of the patterns provide complementary information.

b) Comparison with propagation patterns: Propagation patterns [16] are recently proposed spatiotemporal patterns which describe sequences of co-located and co-occurring geospatial entities. Unlike STInvs, they are extracted by visiting ad hoc tree structures. Since propagation patterns are, to the best of our knowledge, the most similar spatiotemporal patterns available in the literature, we quantitatively estimate their content overlap with the STInvs under multiple aspects.

To our aim, we initially formalize the matching function

TABLE IX: Statistics about the STInv extractions on the LSTW dataset.

	Global	Boston	LA	NYC
Total patterns	19260983	138807	2153122	10076293
Mean #triplets per sequence	8.87	6.72	8.54	8.92
Mean discrete spatial distance	1.79	1.64	1.89	1.82
Mean discrete temporal distance	0.85	0.78	0.84	0.88
#sequences with at least one $\delta s > 0$ (%)	99.97	99.69	99.97	99.99
#sequences with at least one $\delta t > 0$ (%)	99.21	96.61	99.00	99.68
#sequences with at least one $\delta s > 0$ *and at least one* $\delta t > 0$ (%)	98.60	94.87	98.60	99.41

used to specify when a propagation pattern and an STInv are similar in terms of underlying information. We compared our patterns with propagation patterns on this dataset because it is the same dataset used in the paper that introduced the propagation patterns [16].

Definition 13: Let T_i be a tree associated with a propagation pattern and let $STInv_j$ be an STInv. T_i matches $STInv_j$, and vice versa, if and only if the following conditions hold:

- The number of nodes of T_i is equal to the number of event occurrences in $STInv_j$.
- For each node n_x in T_i such that n_x is parent of node n_y in T_i then there exist two event occurrences tr_z and tr_w in $STInv_j$ such that $\delta t(tr_w) \geq \delta t(tr_z)$
- The event type of tr_w is the same of n_y.
- The event type of tr_z is the same of n_x.

□

A match between an STInv and a propagation pattern indicates a correlation between the same set of event types in the same temporal order.

Based on the matching function, we can formalize the concept of *coverage* of a set of STInv patterns on a set of propagation patterns (and vice versa).

Definition 14: Let $PropPat$ be the set of frequent propagation patterns and let $STInvPat$ be the set of frequent STInv patterns mined from the same set of trigger events. The coverage of $PropPat$ on $STInvPat$ indicates the level of representativeness of the set of (existing) patterns $PropPat$ with respect to the newly proposed $STInvPat$ set. It is defined as follows:

$$Cov(PropPat \rightarrow STInvPat) = \frac{|STInvs \in STInvPat \text{ matched by any } T_i \in PropPat|}{|STInvPat|}$$

Conversely, the coverage of $STInvPat$ on $PropPat$ indicates the level of representativeness of the new patterns with respect to the old ones and is defined as follows.

$$Cov(STInvPat \rightarrow PropPat) = \frac{|\text{pp} \in PropPat \text{ matching any } STInv_j \in STInvPat|}{|PropPat|}$$

□

Notice that the aforesaid coverage measures are not symmetric because they respectively indicate the fraction of *known* patterns that are retrieved by the *new* mining task ($Cov(STInvPat \rightarrow PropPat)$) and the fraction of *new* patterns that are retrieved by reusing the patterns extracted using an existing approach [16].

To perform a fair comparison, for both algorithms we set the minimum support threshold $minsup$ to 10, the temporal resolution td to 10 minutes, and the spatial threshold r to 0.2 miles.

The results, which are reported below, confirm the expressiveness, representativeness, and novelty of STInvs compared to propagation patterns:

- $Cov(STInvPat, PropPat)$=88.29%, i.e., STInvs mostly cover the information provided by the propagation patterns.
- $Cov(PropPat, STInvPat)$=43.71%. i.e., Propagation patterns are not sufficient to express the same knowledge provided by spatiotemporally invariant patterns.

Some examples of STInvs that are not matched by the propagation patterns follow.

{Accident(δs=0m)} $\xrightarrow{20 \text{ mins}}$ {Flow-Incident(δs=(0.4miles, 0.6miles])}

{Accident(δs=0m)} $\xrightarrow{20 \text{ mins}}$ {Congestion(δs=(0.2miles, 0.4miles]), Congestion(δs=(0.4miles, 0.6miles])}

Both STInvs indicate correlations between events whose spatial and temporal distances exceed the temporal and spatial resolutions. STInv patterns do not require that the events in the sequence are temporally and spatially contiguous. Conversely, propagation patterns are negatively influenced by the presence of "large" spatial/temporal gaps that break the event sequence. In fact, the chain of parent-child relationships is broken and the connection between far events is missed.

Notice also that the few propagation patterns that are not matched by STInvs are related to events that are not always at the same relative distance. They are split in STInvs associated with the same events but different spatial and temporal relative distances. Those STInvs are typically infrequent. Hence, they are not extracted by our algorithm using the current minimum support threshold. Therefore, by lowering the $minsup$ value of the STInv mining process the coverage value is likely to further increase.

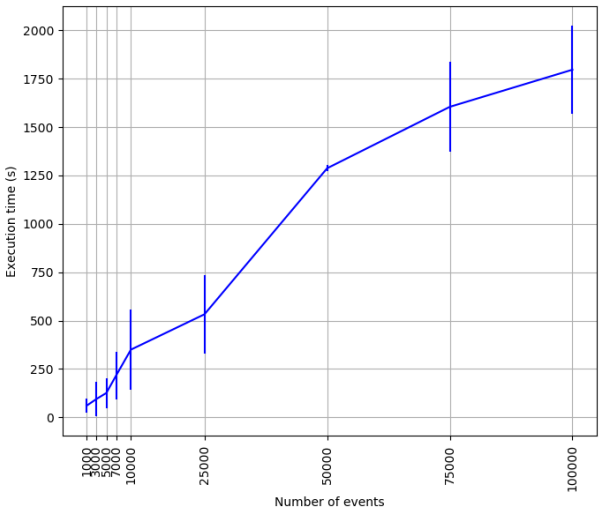
D. Execution time and scalability test

We test the scalability of STInvMiner on the bike sharing dataset, varying the number of input events and the maximum number of events for each mined sequence. Each test is

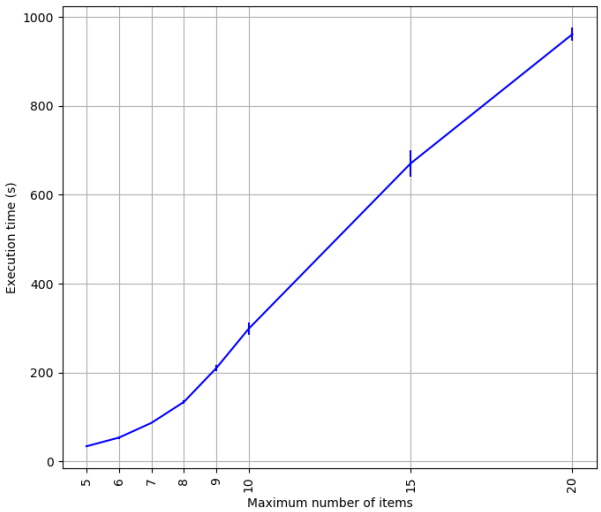
repeated three times using the following configuration setting: $td = 10$min, $r = 100m$, 6 time slots, maximum distance of 500m and Full, Almost Full, Increase events on the city of San Francisco. We set the $minsup$ value to 1 and the number of executors to 2 for all the experiments.

Figure 3 reports the average execution time and its standard deviation[6]. Specifically, Figure 3a reports the execution time with a fixed maximum mined sequence length of 6 and a number of input events varying from 1.000 to 100.000 and with a maximum pattern length of 6, whereas Figure 3b illustrates the execution time with a fixed number of 5.000 input events as the maximum mined sequence length varies in the range [5, 20].

The algorithm performs well despite the setting of the absolute minimum support threshold to 1, with a maximum execution time that is less than 2000s for all the scalability experiments.



(a) Scalability with the number of events.



(b) Scalability with the maximum number of items per sequence.

Fig. 3: Scalability of the STInvMiner algorithm. 2 executors and $minsup$=1.

VI. Conclusions and Future Works

In this paper, we proposed a new type of spatiotemporally correlated pattern called STInv. STInvs model the *relative* spatiotemporal event correlations regardless of the absolute location and time values of the corresponding instance. To the best of our knowledge, this is the first attempt to tailor pattern analysis to spatial/temporal invariance.

The takeaways from the experiments conducted on two real-world datasets are enumerated below:

- STInvs averagely summarize the temporal correlations described by tens of traditional sequences.
- STInv patterns show non-zero spatial and temporal gaps in most cases.
- STInvs achieve almost 90% coverage of the propagation patterns [16], whereas their underlying information is covered by propagation patterns only in a limited manner.
- STInvs represent a valid support for modeling correlations of mobility data and triggering decisions.

As future work, we plan to deeply push new spatial and temporal constraints into the STInv mining process and explore the applicability of the proposed patterns in new application scenarios (e.g., emergency management, predictive maintenance, cybersecurity). Moreover, we aim to design and test a predictive model, based on STInv patterns, for traffic congestion forecasting. Finally, we plan to use a traffic flow theory-based approach to assess the information mined from the Large-Scale Traffic and Weather Events dataset. However, this last activity needs a dataset with information that is not available in the Large-Scale Traffic and Weather Events dataset. Hence, a new specific traffic dataset must be collected.

VII. Acknowledgements

This work was partially funded by the SmartData@PoliTO center. We thank the SmartData@PoliTO center and HPC@PoliTO for providing the computational resources. We thank Giuseppe Moscarelli, Lorenzo Ottino, and Martina Toma for their preliminary work. Within the FAIR (Future Artificial Intelligence Research) this work received funding from the European Union Next-GenerationEU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

[6]The computing facilities adopted in this paper are shared with other researchers within our organization, with variable workload.

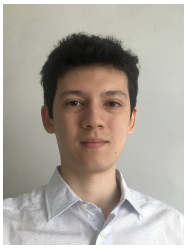
References

[1] S. Paiva, M. A. Ahad, G. Tripathi, N. Feroz, and G. Casalino, "Enabling technologies for urban smart mobility: Recent trends, opportunities and challenges," *Sensors*, vol. 21, no. 6, 2021.

[2] D. Huang, S. Wang, and Z. Liu, "A systematic review of prediction methods for emergency management," *Int. J. Disaster Risk Reduction*, vol. 62, p. 102412, 2021.

[3] A. Corallo, M. Lazoi, M. Lezzi, and A. Luperto, "Cybersecurity awareness in the context of the industrial internet of things: A systematic literature review," *Computers in Industry*, vol. 137, p. 103614, 2022.

[4] X. Chen, K. Huang, and H. Jiang, "Detecting changes in the spatiotemporal pattern of bike sharing: A change-point topic model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 361–18 377, 2022.
[5] S. Cheng, F. Lu, and P. Peng, "Short-term traffic forecasting by mining the non-stationarity of spatiotemporal patterns," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6365–6383, 2021.
[6] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "FreeSpan: frequent pattern-projected sequential pattern mining," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 355–359.
[7] M. J. Zaki, "SPADE: an efficient algorithm for mining frequent sequences," *Mach. Learn.*, vol. 42, no. 1/2, pp. 31–60, 2001.
[8] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *ICDE'01*, 2001, pp. 215–224.
[9] J. J. Harris, C. Chen, and M. J. Zaki, "A framework for generating summaries from temporal personal health data," *ACM Trans. Comput. Heal.*, vol. 2, no. 3, pp. 21:1–21:43, 2021.
[10] T. Niyazmand and I. Izadi, "Pattern mining in alarm flood sequences using a modified prefixspan algorithm," *ISA transactions*, vol. 90, pp. 287–293, 2019.
[11] F. Giannotti, M. Nanni, and D. Pedreschi, "Efficient mining of temporally annotated sequences," in *SDM'06*, 2006, pp. 348–359.
[12] P. Yang, L. Wang, X. Wang, L. Zhou, and H. Chen, "Parallel co-location pattern mining based on neighbor-dependency partition and column calculation," in *ACM SIGSPATIAL'21*, 2021, pp. 365–374.
[13] Y. Huang, S. Shekhar, and H. Xiong, "Discovering colocation patterns from spatial data sets: a general approach," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 12, pp. 1472–1485, 2004.
[14] J. S. Yoo and S. Shekhar, "A joinless approach for mining spatial colocation patterns," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1323–1337, 2006.
[15] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *ACM SIGKDD'07*, 2007, p. 330–339.
[16] S. Moosavi, M. H. Samavatian, A. Nandi, S. Parthasarathy, and R. Ramnath, "Short and long-term pattern discovery over large-scale geo-spatiotemporal data," in *ACM SIGKDD'19*, 2019, pp. 2905–2913.
[17] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, p. 56–65, 2016.
[18] F. Verhein, "Mining complex spatio-temporal sequence patterns," in *SDM'09*, 2009, pp. 605–616.
[19] M. J. Zaki, "Efficiently mining frequent embedded unordered trees," *Fundamenta Informaticae*, vol. 66, no. 1-2, pp. 33–52, 2005.
[20] L. Colomba, L. Cagliero, and P. Garza, "Mining spatiotemporally invariant patterns," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1–4. [Online]. Available: https://doi.org/10.1145/3557915.3560998
[21] L. Cagliero, T. Cerquitelli, S. Chiusano, P. Garza, G. Ricupero, and E. Baralis, "Characterizing situations of dock overload in bicycle sharing stations," *Applied Sciences*, vol. 8, no. 12, 2018.
[22] C. Herff and D. J. Krusienski, *Extracting Features from Time Series*. Cham: Springer International Publishing, 2019, pp. 85–100. [Online]. Available: https://doi.org/10.1007/978-3-319-99713-1_7
[23] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann Publishers Inc., 2011.



Luca Colomba received the master's degree in Computer Engineering from Politecnico di Torino in 2019, where he is currently a Ph.D. student in the Department of Control and Computer Engineering (DAUIN). His major research interests include big data analytics, scalable algorithms, data mining and machine learning applied to spatio-temporal data.

Luca Cagliero has been associate professor at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2020 and coordination of the SmartData@Polito interdepartmental center since January 2024. His current research interests are in the fields of pattern mining and Deep Natural Language Processing. Specifically, he has worked on text summarization, classification and association rule mining. He has published 100+ papers in international journals, book chapters, and conference proceedings.

Paolo Garza received the master's and Ph.D. degrees in Computer Engineering from Politecnico di Torino in 2001 and 2005, respectively. Since December 2018, he has been an associate professor at the Dipartimento di Automatica e Informatica, Politecnico di Torino. His research interests include data mining algorithms, big data analytics, and data science. He has worked on classification, clustering, itemset mining, and scalable algorithms.