GABRIEL MARIANO  DE CASTRO SILVA

# Enhancing impersonation fraud detection on smart buildings physical access control systems: an anomaly-based approach using social groups trajectories data

São Paulo
2022

GABRIEL MARIANO  DE CASTRO SILVA

# Enhancing impersonation fraud detection on smart buildings physical access control systems: an anomaly-based approach using social groups trajectories data

**Corrected Version**

Dissertation submitted for the degree of Master of Science to Escola Politécnica of Universidade de São Paulo.

São Paulo
2022

**GABRIEL MARIANO DE CASTRO SILVA**

# Enhancing impersonation fraud detection on smart buildings physical access control systems: an anomaly-based approach using social groups trajectories data

**Corrected Version**

Dissertation submitted for the degree of Master of Science to Escola Politécnica of Universidade de São Paulo.

Concentration field:

Computer Engineering

Advisor:

Prof. Dr. Jaime Simão Sichman

São Paulo
2022

Catalogação-na-publicação

Silva, Gabriel Mariano de Castro
Enhancing impersonation fraud detection on smart buildings physical access control systems: an anomaly-based approach using social groups trajectories data / G. M. C. Silva -- versão corr. -- São Paulo, 2022.
83 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Edifícios Inteligentes 2.Mineração de Dados 3.Trajetória 4.Reconhecimento de Padrões 5.Grupos Sociais I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

"Every day sees humanity more victorious in the struggle with space and time."

Guglielmo Marconi (born April 25, 1874)

"To every thing there is a season, and a time to every purpose under the heaven: A time to be born, and a time to die; a time to plant, and a time to pluck up that which is planted; A time to kill, and a time to heal; a time to break down, and a time to build up; A time to weep, and a time to laugh; a time to mourn, and a time to dance; A time to cast away stones, and a time to gather stones together; a time to embrace, and a time to refrain from embracing; A time to get, and a time to lose; a time to keep, and a time to cast away; A time to rend, and a time to sew; a time to keep silence, and a time to speak; A time to love, and a time to hate; a time of war, and a time of peace. What profit hath he that worketh in that wherein he laboureth?"

Ecclesiastes 3:1-9

"The FBI has a database consisting of some 200 million fingerprint records... As part of a modernization program, the FBI is digitizing these records as 8-bit grayscale images, with a spatial resolution of 500 dots per inch. This results in some 10 megabytes per card, making the current archive about 2,000 terabytes in size."

C.M. Brislawn (BRISLAWN, 1995)

“Big data isn’t about bits, it’s about talent.”

Douglas Merrill

"After all, a person is herself, and others. Relationships chisel the final shape of one’s being. I am me, and you."

N. K. Jemisin (JEMISIN, 2016)

To my dear wife, Juliana, whose affection, love, and encouragement make me able to accomplish this achievement.

Also, this work is wholeheartedly dedicated to my mother, Rute, for her continuous moral, spiritual, and emotional support during my life.

# ACKNOWLEDGEMENT

I want to thank my advisor Jaime Simão Sichman for having accepted to advise me, and for all the patience and learning he provided me.

Also, thanks to the companies I had the pleasure to work for, INVENZI and ASSA ABLOY, and that helped me providing knowledge, datasets, insights, and the time I needed to work on this research.

To all my friends and family who believed in me.

# RESUMO

Detecção de fraude de identidade baseada em anomalias consiste em construir perfis com base nos comportamentos frequentes dos usuários e compará-los com novos dados. A ideia subjacente é que um comportamento diverso pode indicar uma possível fraude, ou seja, alguém tentando se passar pelo usuário original. A maioria das pesquisas na área visa usar dados espaço-temporais amplamente disponíveis coletados por sensores de localização onipresentes, tais como GPS, telefonia móvel, beacons e sistemas de controle de acesso físico. Por outro lado, muitos estudos alcançaram bom desempenho na descoberta de relações sociais entre os usuários. No presente trabalho, combinamos conceitos de pesquisas anteriores e propusemos um novo modelo de perfis denominado Group-T-Patterns, publicado originalmente em (SILVA; SICHMAN, 2022), que utiliza grupos sociais para construir perfis de mobilidade a fim de melhorar a detecção de anomalias. Em particular, desenvolvemos um algoritmo para minerar padrões de grupos chamado GTPM (Group Trajectory Pattern Mining) e implementamos um detector de fraude de identidade totalmente funcional para sistemas de controle de acesso físico. Conduzimos uma análise empírica usando dois conjuntos de dados do mundo real, e os resultados mostram que adicionar informações de grupos sociais a perfis de mobilidade melhora a detecção de ataques de representação baseados em anomalias.

**Palavras-chave:** Edifícios inteligentes, fraude de identidade, mineração de dados, mineração de padrões de trajetória, controle de acesso físico, detecção de grupos sociais, detecção de anomalias.

# ABSTRACT

Anomaly-based impersonation detection consists of constructing profiles based on users' frequent behaviors and comparing them with new data. The underlying idea is that a diverse behavior may indicate possible fraud, i.e., someone trying to impersonate the user. Most research in the area aims to use spatio-temporal data broadly available from ubiquitous location sensors, like GPS, mobile telephony, beacons, and physical access control systems. On the other hand, many studies achieved good performance in finding social bonds among users. In the present work, we combined concepts from previous research and proposed a new model of profiles called Group-T-Patterns, originally published in (SILVA; SICHMAN, 2022), that uses social groups to construct mobility profiles and enhance anomaly detection. In particular, we developed an algorithm to mine Group-T-Patterns named GTPM (Group Trajectory Pattern Mining) and implemented a fully functional impersonation fraud detector for physical access control systems. We conducted an empirical analysis using data from two real-world datasets, and the results show that adding companion activities information to mobility profiles enhances anomaly-based impersonation attack detection.

**Keywords:** Smart buildings, impersonation fraud, data mining, trajectory pattern mining, physical access control, social groups detection, anomaly detection.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ACS | All Common Subsequences |
| AUC | Area Under the Curve (in Particular, the ROC curve) |
| CDR | phone Call Detail Records |
| COVID-19 | COronaVIrus Disease 2019 |
| CPS | Cyber-Physical Systems |
| CPSs | all Common Pattern Sequences |
| DFM | Deviations from Mean |
| DFS | Depth-first search |
| FAR | False Acceptance Rate |
| FRR | False Rejection Rate |
| FSP | Frequent Sequential Pattern |
| GPS | Global Positioning System |
| GTMP | Group Trajetory Pattern Mining |
| HSP | Hospital |
| LBSN | Location-Based Social Networks |
| LCS | Longest Common Subsequence |
| LFP | Location Frequency Profiles |
| LTCP | Loose Travelling Companion Pattern |
| MiSTA | Mining Sequences with Temporal Annotations |
| MOB | Multi-Office Building |
| MSTP | Maximal Semantic Trajectory Pattern similarity |
| PACS | Physical Access Control Systems |
| PDS | Patterns Distribution-based Similarity measure |

| | |
|---|---|
| POI | Points of Interest |
| Prefix-Span | Prefix-projected sequential pattern |
| REMO | RElative MOtion |
| RFID | Radio-frequency identification |
| ROC | Receiver Operating Characteristic curve |
| SB | Smart Building |
| SCI | Social Connection Inference framework |
| SPADE | Sequential PAttern Discovery using Equivalence classes |
| U.S. | United States (of America) |
| USP | Universidade de São Paulo |
| WCM | Weakly Consistent group Movement pattern |

# LIST OF SYMBOLS

|  |  |
|---|---|
| $\sigma$ | Minimum support threshold[1] |
| $\tau$ | Maximum tolerance time threshold[2]. |
| $\epsilon$ | Minimum similarity threshold[3] |

[1]Minimum number of times that a pattern must be contained among all trajectories to be considered frequent.

[2]Maximum difference between two transition times to be considered similar.

[3]Minimum similarity between a new trajectory to be considered normal (not a threat) and the user Profile.

**CONTENTS**

# 1 INTRODUCTION

Modern organizations' facilities rely on technologies that enhance energy-efficiency, comfort, and security, with various integrated cyber-physical systems that offer close interaction between physical components and computing. These *Smart Buildings* (SB) are increasingly gaining notoriety among companies, governments, and organizations that depend on computer systems to perform their operations (KLEISSL; AGARWAL, 2010). SBs integrate information about energy, heat exchange, airflow, water use, security, and, notably, *Physical Access Control Systems (PACSs)* (CIHOLAS et al., 2019). These later are an essential part of organizations' security as they prevent unauthorized people from getting access to sensitive resources, assets, and information; they provide monitoring, surveillance, and control of people's movements within organizations' facilities. *Impersonation frauds* are constant threats to PACSs, and happen when some person pretends to be another one to get undue access to the facility and may include loan, spoofing, cloning, or theft of credentials.

As a study of human attitudes within organizations, organizational behavior can contribute to the detection of typical behavior profiles in work environments, notably smart buildings. Organizational behavior has two dimensions, namely *individual* and *group behavior* (HUCZYNSKI; BUCHANAN; HUCZYNSKI, 2013). Individual behavior is related to people's schedules and personal attributions, and refers to the activities that the members develop alone in the workplace. On the other hand, work activities are often performed by formally designated work-groups defined by the organizational structure. Informal groups can also emerge when there is a mutual liking of the group members, when people have interactions based on shared interests, value systems, and social bondage they develop, sharing the highest level of sentiments or affinity among them.

In summary, it is a possibility that PACSs can thus capture people's individual and group behaviors within organizations. Human individual and group trajectories show a high degree of temporal and spatial regularity (BARBOSA et al., 2018); people are highly sociable in general and perform certain activities regularly together. In this work, we hypothesize that both individual and group activities can be included in users' profiles to enhance the performance of an anomaly-based impersonation detection system. We can reasonably assume that when a credential is under impersonation attack by a malicious entity, its individual and group behaviors will change significantly from the original behaviors. Monitoring the credential audit data may be useful for detecting anomalous credentials activities in the system. Although the primary goal of a PACS is to anticipate threats, it is humanly impossible to monitor all actions from all the users in real-time, due to the large volume of data. The time needed to manually search and analyze the data, aiming to observe any malicious situations, turns the task impracticable. However, with the power of computers and current data mining algorithms, we believe that such detection can be done automatically.

## 1.1 Motivation

*Impersonation fraud* is the crime of using the personal information or credentials of another person and use his identity to get unauthorized access to resources. In 2018, about 9% of all U.S. residents aging 16 or older reported that they had been victims of identity theft during the prior 12 months (HARRELL; LANGTON, 2018). There are many studies in detecting frauds in other domains, such as finance (TRIPATHI; PAVASKAR, 2012) and mobile networks (YAZJI et al., 2014); however, detecting frauds in PACSs lacks further research. Even though some fraud detection approaches are very efficient in some contexts, they may not apply to mobility data from indoor positioning technologies, since (i) access control audit logs are usually sparse because some users may have more access than others, and (ii) when approaching to doors, users sometimes take advantage of a door already open by a previous user, thus creating missing data.

Although PACSs are vital components to organizations' protection, physical access control systems are aging and becoming vulnerable, which causes an overall lack of awareness, especially when compared to virtual access control technologies in straight development, such as those applied to Blockchains or Internet of Things, to name a few.

*Data mining* is a field of research that consists in discovering useful and interesting information from raw data that helps the understanding of many phenomena and improve decision-making processes. Many techniques emerged from data mining research, among those we may highlight:

- Anomaly detection is a data mining task that aims to discover unusual and suspicions observations that, when applied to PACSs, could raise impersonation fraud alarms;
- Group Pattern Mining techniques output users that are frequently moving together;
- Sequential Pattern Mining aims to find frequent events that are delivered in sequence, e.g., trajectories, that are sequences of locations.

Challenged by the task of detecting impersonation fraud in PACSs, we propose in this work an architecture for an automated impersonation fraud detector that combines these three data mining techniques.

## 1.2 Objectives

This dissertation's primary goal is to present a possible solution to the problem of detecting potential identity theft in physical access control systems.

More specifically, we propose a model for profiling users' behaviors based on their frequent trajectories that include their social groups. We also suggest a similarity measure between profiles and future trajectories to build a classifier that outputs if a trajectory is anomalous and hence a potential theft.

We evaluate the classifier on real-world datasets and compare its performance against existing methods, and perform a sensitivity analysis of the model to its parameters.

After reaching the presented objectives, the following research questions will be answered:

RQ1 Is it possible to enhance impersonation fraud detection in PACSs by incorporating social group information?

RQ2 How the definition of *frequent trajectory* of the model affects its outputs?

## 1.3 Method

In order to reach the proposed objectives and answer the research questions, we first conducted a literature review to identify the state-of-the-art of the related topics and possible gaps. Then, after understanding the underlying subjects, we constructed a novel model of trajectory pattern called *Group-T-Pattern* (Group Trajectory Pattern), that adds the concept of groups to the concept of trajectory patterns. By doing that, we brought together in the same model the notion of trajectory patterns and social groups.

Some existing concepts formed building blocks to our model, as shown in Figure 1:

- Frequent Sequential Patterns study trajectories as *sequences of Locations*, and the algorithm considered is *Prefix-Span* (HAN et al., 2001);
- Trajectory Patterns adds the concept of *transition times* to the frequent sequential patterns, and the algorithm to mine these Trajectory Patterns is called *MiSTA* (GIANNOTTI; NANNI; PEDRESCHI, 2006);
- Group Patterns consider more than one element in the trajectory. The chosen model is *Swarm*, and the mining algorithm is named *ObjectGrowth* (LI et al., 2010).

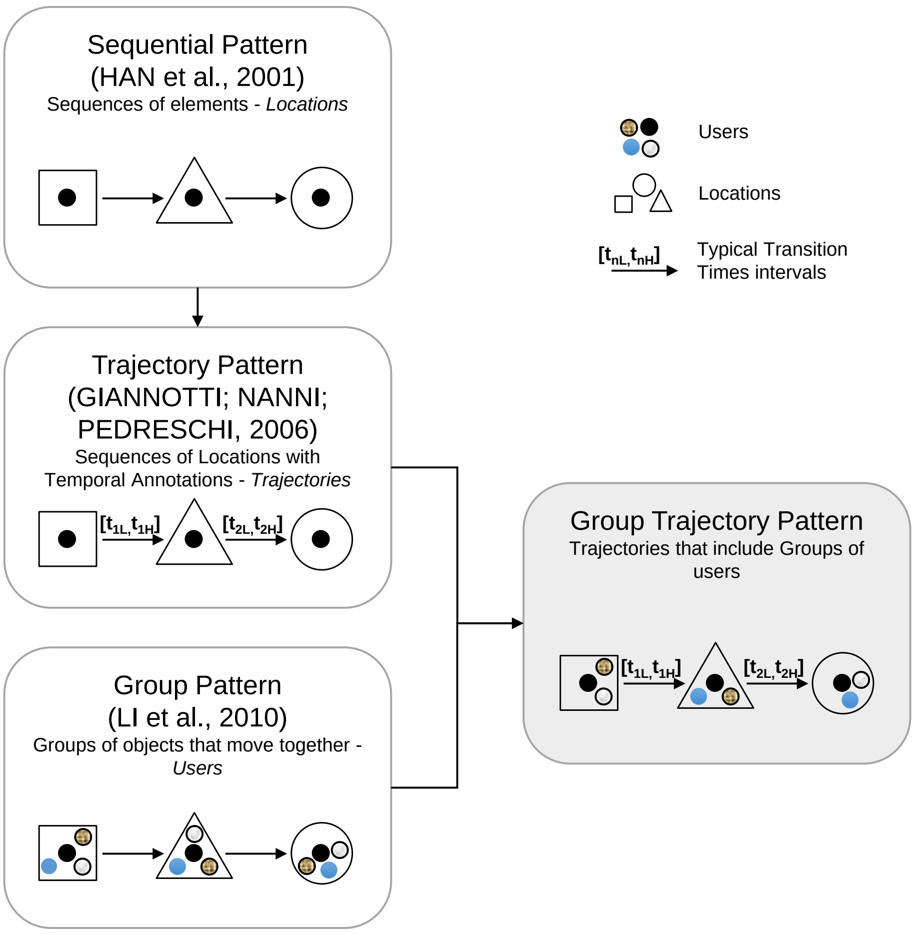

Sequential Pattern
(HAN et al., 2001)
Sequences of elements - *Locations*
Trajectory Pattern
(GIANNOTTI; NANNI; PEDRESCHI, 2006)
Sequences of Locations with Temporal Annotations - *Trajectories*
[$t_{1L}$,$t_{1H}$]
[$t_{2L}$,$t_{2H}$]
Group Pattern
(LI et al., 2010)
Groups of objects that move together - *Users*
Group Trajectory Pattern
Trajectories that include Groups of users
Users
Locations
[$t_{nL}$,$t_{nH}$]
Typical Transition Times intervals

Figure 1: Mobility Patterns Evolution

With the use of this new model, we built users' profiles that are based on their spatio-temporal and social trajectories. We after proposed a Trajectory and Profile similarity measure that includes these added social groups in order to evaluate if new trajectories are consistent with the Profiles.

Additionally, we implemented the model with a new algorithm named GTMP (Group Trajectory Pattern Mining) and performed experiments with two real-world databases, from a multi-office building and from a hospital. We then conducted an extensive sensitivity analysis of the parameters of the model and came up with some conclusions and possible future research work.

## 1.4 Contributions

Our first contribution is a novel model of trajectory pattern model called *Group-T-Pattern*, which adds information about social groups to users' trajectory patterns.

Furthermore, we also proposed a new algorithm denominated GTPM (Group Trajectory Pattern Mining) that implements the concept of Group-T-Pattern. We believe that this notion is easily extendable to other applications that need to find frequent temporally annotated sequences of groups.

We also proposed similarity measures that describe users' regularity in their paths, schedules, and social groups. The proposed similarity measures are based on the concept of all common subsequences (ACS), as originally proposed by (WANG, 2007).

Although the focus of this work is to solve the specific problem of impersonation detection in physical access control systems, the concepts we developed have the potential to solve other real problems and can be easily abstracted for application in other domains that require patterns detection of movement of groups. Some examples of these applications are: detecting the movement of cars in convoys, vessels in the open sea, and groups of people in sports stadiums. The contribution of this work can have wide application not only in the detection of frauds and anomalies, but in urban planning, combating piracy and terrorism, organization of events and consumer relations.

Our fourth contribution is an implementation of a fully functional potential identity theft detector. The open source implementation of the model and detector is available in Github: https://github.com/gabrielmariano1/GTPM

Finally, this work has originated three publications:

(SILVA; SICHMAN, 2018) - SILVA, G. M. de C.; SICHMAN, J. S. Identity theft detection using trajectorypatterns. In: *Anais do VII Workshop de Pós-Graduação em Engenharia de Computação – WPGEC*. PCS/POLI/USP, 2018. p. 69–72.

(SILVA; SICHMAN, 2019)- SILVA, G. M. de C.; SICHMAN, J. S. Using social group trajectories for potential impersonation detection on smart buildings access control. In: *Proc. of the 8th Brazilian Conference on Intelligent Systems (BRACIS)*.

IEEE Computer Society, 2019.p. 389–394.

(SILVA; SICHMAN, 2022)- SILVA, G. M. de C.; SICHMAN, J. S. Impersonation fraud detection on building access control systems: An approach based on anomalous social and spatio-temporal behaviors. *Applied Soft Computing*, vol. 188, pg. 108310, 2022

## 1.5 Text organization

This dissertation is organized as follows. Chapter 2 presents an overview of the state-of-the-art of the related topics, while chapter 3 introduces some basic concepts of the literature; it also provides a more detailed conceptual description of the related work on which we have based our model, as illustrated in Figure 1. In the sequence, Chapter 4 proposes our formalization of the problem and describes our proposed Group-T-Pattern model to solve it. Chapter 5 details the GTPM algorithm that implements our model. Chapter 6 describes experiments conducted with two real-world datasets along with a sensitivity analysis of the method, and a comprehensive evaluation of obtained results. Finally, Chapter 7 brings up our conclusions and indicates the directions for future work.

# 2 LITERATURE REVIEW

This chapter provides an overview of the current knowledge, including substantive findings and theoretical and methodological contributions, related to our research problem and questions. It allowed us to identify relevant theories, methods, and gaps in the existing research. We divided this chapter into four sections related to the main research topics: Section 2.1 describe works that addressed the same problem of detecting intrusion detection in PACS. The following sections, i.e., Sections 2.2, 2.3 and 2.4, present works that addressed the underlying subjects of the main task, respectively trajectory patterns, group pattern mining and anomaly detection.

## 2.1 Intrusion detection in physical access control systems

Detecting intrusion by analyzing physical access control's audit logs is not a new idea, although works published in this area are relatively scarce. This section presents some works that addressed the same problem but from a entirely different perspective.

The pioneering work (MIROWSKI; HARTNETT, 2007) uses statistical methods to look for anomalies in access control events datasets. The authors define users' Location Frequency Profiles (LFP) to look for behaviors that may indicate a change of credential ownership, and the system triggers an alarm if the frequency of use in a determined time-window exceeds a certain threshold that they called Deviations from Mean (DFM).

Garri et al. (2011) use Kohonen's maps are built to profile the normal behavior of the users of an active RFID-enabled access control system. A position is said to be anomalous and may indicate spoofing/cloning or robed tag if it does not belong to any classification defined during the training. In the referred study, users' locations were monitored continuously based on signal strength.

More recently, the work (YIN et al., 2016) presents a real-time intrusion detection system based on a Complex Event Processing (Esper) tool in the RFID middleware. It calculates the probability of a proximity card to be used in each reader at each time-window, and whenever a new event arrives, it checks the likelihood of occurrence of the event. If the probability is lower than a preset threshold, the system triggers an alarm.

Geepalla e Asharif (2020) developed a Graph-based method for analysis of PACS log data to detect normal and abnormal behavior.

Some other works do not use data mining methods to detect intruder attacks. Specifically, regarding RFID proximity cards, card, or reader level may be used to store data in the card to detect cloned tags or analyze the reader signal patterns when the user presents the card. None of these works, however, included social ties in the profiles. Other existing works (SETIAWAN; YAHYA, 2018) (KANG; LIU; QU, 2017) (FOROUGH; MOMTAZI, 2021) mentioned the discovery of human behavior based on sequential pattern mining rules, but aiming to solve different problems and not considering activity companions.

## 2.2 Trajectory patterns

People, when in motion, vary their position in space over time. The path taken by any moving object is called *trajectory*. Trajectory data is collected in a variety of ways, either passively from GPS devices, smartphones, and RFID readers or more actively like when the user sends geo-referenced messages on social networks, makes a phone call, or uses its credit card to pay a bill at a store. With so much data increasingly available, an entirely new research field in trajectories became feasible, and many techniques aiming to extract information from trajectory data have emerged. Mazimpaka e Timpf (2016) review and classifies existing work that used trajectory data to solve real-world problems. Discoveries concluded that human trajectories have a high degree of spatio-temporal regularity and follow simple, reproducible patterns (GONZALEZ; HIDALGO; BARABASI, 2008).

Unlike previous work that tried to solve the same problem of impersonation detection in PACSs, we used user trajectories to model users' behaviors. Profiling users based on their past movements is useful for a series of applications, especially in anomaly detection, location prediction, friendship recommendation, and social ties prediction.

Trajectories are sequences of locations in time. Sequential Pattern Mining helps to extract the sequences which reflect the most frequent behaviors. Frequent Sequential Patterns (FSP) mining was introduced by Agrawal, Srikant et al. (1994) with Apriori-All, AprioriSome, and DynamicSome algorithms. Other approaches of FSP discovery are Sequential PAttern Discovery using Equivalence classes (SPADE) and SPADE-like algorithms (ZAKI, 2000) (ZAKI; J, 2001) that works on databases with a vertical id-list format, where a list of (user-id, transaction-time) pairs are associated with each location, and the pattern candidates are counted by intersecting the id-lists.

The work (HAN et al., 2001) proposed an efficient sequential pattern mining method called PrefixSpan (i.e., **Prefix**-projected **S**equential **pa**tter**n** mining). The main idea is to analyze only the prefix subsequences and project only their corresponding postfix subsequences into the projected database.

Giannotti et al. (2007) introduces the concept of trajectory patterns (*T-Patterns*) as an extension of sequential patterns. They represent a set of users trajectories including the same sequence of places, with similar *transition times*. In their previous work (GIANNOTTI; NANNI; PEDRESCHI, 2006), the authors extend Apriori algorithm to describe *MiSTA*, an algorithm to mine Temporally Annotated Sequences, mainly based in PrefixSpan, described in the work of Han et al. (HAN et al., 2001).

Other works (CAI; LEE; LEE, 2018) (CHEN; PANG; XUE, 2014) (CHEN et al., 2014b) rely on the concept of T-Pattern introduced in (GIANNOTTI et al., 2007) to perform specific tasks without changing the model itself.

## 2.3 Group pattern mining

The task of mining objects that move together is performed in a diverse range of phenomena, such as habitat and migration of animals using satellite radio transponders, vehicles in fleet management, tourists using public transportation, agents simulating people for modeling crowd behavior, and soccer players on a football match.

The conceptual work described in (LAUBE; IMFELD, 2002) aimed to relate one object motion attributes over space and time to all other objects, and called this analysis concept as *RElative MOtion* (REMO). In REMO analysis, motion parameter values, such as direction, speed, and speed change at each timestamp, are grouped into discrete classes and arranged in a matrix and compared qualitatively. The objective was to use the data matrices to find patterns across objects, e.g., concurrence, opposition, dispersion, convergence, and divergence of *flocks*. In their following work (LAUBE; KREVELD; IMFELD, 2005), the authors quantitatively defined *flock* as being a group of objects that have the same direction and are close to each other, within a distance of radius $R$ in two-dimensional space, at *any timestamp*. Benkert et al. (2008) expanded the previous definition of flocks by considering that a minimum number of objects $m$ from a set of object trajectories $O$ have to be within a distance of radius $R$ for *consecutive* $k$ timestamps. One important concept in group pattern mining techniques is a *cluster*. *Clustering* is a task with the final objective of defining if objects are spatially next to each other at the same time. In (KALNIS; MAMOULIS; BAKIRAS, 2005), the notion of *moving cluster* is proposed, which is a sequence of spatial clusters appearing during consecutive timestamps, so the members are allowed to leave the cluster and new members to join the cluster during the lifetime of a cluster. Being $c_t$ a cluster at time t, the portion of common objects in any two consecutive clusters cannot be less than a threshold parameter $\theta$, i.e., $|c_t \cap c_{t+1}|/|c_t \cup c_{t+1}| \geq \theta$.

The definition of *group pattern* in (WANG; LIM; HWANG, 2006) is very similar to flocks, but in the three-dimensional space. In group patterns, the distance between any two moving objects in the cluster is not larger than a predefined variable for at least $k$ consecutive timestamps.

The *convoy* pattern is introduced in (JEUNG; SHEN; ZHOU, 2008) and extended in (JEUNG et al., 2008). With the argument that the chosen radius size in flock definition has substantial effects on the results of the discovery process, the later work employs the notion of density connection (ESTER et al., 1996) between the objects to consider them as part of the same group, so that groups may have different formations, not only circles. Two objects in a cluster are density-connected if exists a sequence of objects that connects the two objects and the distance between consecutive objects is not larger than a user-specified distance. Given a set of object trajectories $O$, a convoy is a set of minimum $m$ objects density-connected with a distance threshold $e$ for $k$ consecutive timestamps.

In (LI et al., 2010), the authors introduced the *swarm* pattern, in which objects may move together not necessarily in consecutive timestamps. The clustering method is not fixed in swarm patterns, which turns it into a more general pattern. The notion of *swarm* is a set of minimum size $m$ of objects that move together for at least $k$ possible nonconsecutive timestamps. Given a set of trajectories, a *closed swarm* is defined as a swarm that cannot be enlarged in size or duration. The authors described *ObjectGrowth*, a Depth-first search (DFS) algorithm with pruning strategies to find closed swarms.

The concept of traveling companion, presented in (TANG et al., 2012), is almost the same concept as a convoy pattern, the main difference being that the pattern discovery algorithm is faster and outputs results incrementally. The sequel (TANG et al., 2013) relaxes the continuity constraint.

Another possible application of group patter mining is discussed in (ZHENG et al., 2014), where the authors introduce *gathering patterns* which represent big events occurring in a city, such as celebrations, parades, protests, and traffic jams.

Weakly Consistent Group Movement Pattern (WCM) (WANG et al., 2015) is another group movement pattern in which each $w$ continuous clusters should contain at least $mC$ persistent members (those who are connected at $w$ continuous timestamps), and also each member can leave the whole for $lC$ time intervals.

The Loose Travelling Companion Pattern (LTCP) (NASERIAN et al., 2018) brings a novel approach by including *subgroups*. They argue that one group, e.g., a family in an airport, may sometimes be divided into subgroups, such as parents and children. An LTCP group is a sequence of cluster-sets at continuous time-slots. Unlike other algorithms that find only subgroups, they solve the problem of partial discovery by considering as subgroups those that do not get mixed up with other subgroups and whose members also stay together for a certain number of time-slots. Another issue is that clustering methods that generate overlapped clusters (one user participates in more than one cluster in the same time-slot) are not applicable for the LTCP model.

In (ZHU et al., 2019), the authors aimed to find groups of tourists from phone call detail records (CDRs). Trajectory similarity was one of the features used to accomplish this task, along with the province of origin of the phone. Trajectory similarity is obtained by counting what they call *matching points*, which are stay points close to each other that users stay for almost the same time. They compare the users two by two to find groups. For example, if in a candidate group $\{a, b, c, d\}$ they find two pairs of traveling companions, namely $a, b$ and $b, c$, they consider the real group as $\{a, b, c\}$.

The first work that aimed to infer friendship from mobility data is (EAGLE; PENTLAND; LAZER, 2009). The authors used data collected from mobile phones and performed 95% of accuracy in discovering friendships, where pairs of friends demonstrate distinctive temporal and spatial patterns in their physical proximity and calling patterns. They confirmed the discovered friendship supported by surveys taken by the users.

Friendship recommendation based in spatio-temporal data is also an extensive field of research, and most of the methods consider context and/or co-occurrence. In works based on co-occurrence, the intuitive idea is that people who have been in the same place at the same time on multiple occasions are likely to know each other. This was proved by the experiments described in (CRANDALL et al., 2010) and performed over a data set of 38 million geotagged photos from the social network Flickr. The authors concluded that the probability of a social bond increases as the number of co-occurrences increases and the temporal range decrease. In particular, we describe in the sequence how recommendation and prediction applications based in Location-Based Social Networks (LBSN) data helped to leverage trajectory-based user profiling research.

## 2.4 Anomaly detection over trajectories

Detecting anomalies in indoor location data is a more recent field of research. The pioneering work (LIU et al., 2012) proposed a stochastic model for context-aware anomaly detection in indoor location traces in the context of a hospital environment, where medical devices are tracked with sensors. The work focus on the missing event detection that may indicate that devices had been stolen. Other applications (DU et al., 2018) (GU et al., 2019) aimed to detect pick-pocketing in public transportation.

In (KONTARINIS et al., 2019), the authors aimed to model indoor trajectories by combining aspects of state-of-the-art semantic outdoor trajectory models. They test their model and have discussions using the Louvre Museum plant.

Detecting anomalous outdoor trajectories is, on the other hand, more explored by literature. Yazji et al. used mobility profiles from mobile networks to detect intrusion detection (YAZJI et al., 2014). They proposed two ways of modeling the behavior: the first model is based on the empirical cumulative probability measure of location and time, while the second is based on the Markov transition property. An anomaly is detected when the probability evolution matching reflecting a normal behavior falls below a threshold. As attacker behavior traces were not presently available, they tested their methods with traces from different users, that is, each user's profile has been compared with trajectories from all other users to calculate the performance indicators, notably False Acceptance Rate (FAR).

Mazumdar et al. propose a common Patterns Distribution-based Similarity measure (PDS) to compute the similarity between any two users and subsequently find the effective K-neighbors (MAZUMDAR et al., 2016). The novelty in the approach is to consider the distribution of check-ins recorded each day on which a common pattern is obtained.

The authors in (TRASARTI et al., 2017) combined three strategies for location prediction: the individual strategy uses only the user individual mobility profile, the collective strategy considers the routines of all users exploiting the possibility that a user could follow a path that is systematic for another user but atypical for the first user and the hybrid strategy that is a combination of the previous two. They also provide an excellent revision of the literature regarding trajectory prediction.

In (NJOO et al., 2017), the authors proposed a framework called SCI (Social Connection Inference) framework, which quantified three key features: diversity, stability, and duration, and aggregated co-occurrence features using machine learning algorithms to predict the friendships between users.

The method described in (XU et al., 2018) considers location context, temporal context, and co-occurrence, combined with periodic mobility. The authors claim that temporal context is included in their method, which is not involved in previous works.

In (LIN et al., 2019), a word embedding technique is used to compute the semantic distance between two stay regions, bringing relative importance to them. They argue that proposed distance metric axioms are satisfied by this technique.

A key for anomaly detection success is the chosen similarity measure. As we consider trajectories as sequences, we were inspired by state-of-art sequence similarity measures from the literature. In (YING et al., 2010), the authors proposed a trajectory similarity measurement, namely, Maximal Semantic Trajectory Pattern Similarity (MSTP Similarity), where the similarity between two trajectory patterns is measured by considering the *longest common subsequence* (LCS) between two patterns. The authors applied frequent sequential pattern mining technologies from (HAN et al., 2001) to extract the sequences of places that a user frequently visits. In (CHEN; PANG; XUE, 2014), the authors improve the method by adding transition times between regions of interest to the calculation of similarity, with the argument that if two users are similar, not only their sequences of regions of interest but also the time elapsed between them should also be similar. In the sequence of their work (CHEN et al., 2014b), they extended this notion of common subsequence: instead of considering just the longest common patterns, they considered all common patterns sets (CPSs) of two users, arguing that the original similarity metrics do not capture all the aspects of users similarity, especially when users' mobility profiles are subsets of other users mobility profiles. The length and support of the common patterns are used to compute their relative importance. In (CHEN et al., 2014a), they provide a tool to manage trajectory datasets and construct and compare users' trajectory patterns.

## 2.5 Discussion

As previously mentioned in Section 1.3, our model Group-T-Pattern uses a combination of the concepts: Frequent Sequential Patterns, Trajectory Patterns, and Group Patterns, as illustrated in Figure 1.

Due to the nature of the data, which is sparse and have missing data, as discussed in Section 1.1, we needed a group pattern that can deal with nonconsecutive time-slots and multiple cluster sets for the same time-slot. In our work, we will use the concept of Swarms (LI et al., 2010) to develop our Impersonation fraud detection because it is the best suitable group pattern given these constraints.

Still considering some characteristics of the datasets, we use the concept of Trajectory Patterns from (GIANNOTTI; NANNI; PEDRESCHI, 2006), as many works in literature, because it is the most established way of representing trajectories as sequences of locations with time annotations. Being the algorithm that mines Trajectory Pattern (MiSTA) an extension of the Frequent Sequential Pattern mining algorithm called Prefix-Span (HAN et al., 2001), we also adopted some ideas of this algorithm.

In a first attempt to solve the problem (SILVA; SICHMAN, 2019), we have proposed the notion of *groups* of users, similarly as proposed by Li et al. (2010)[1], and hypothesized that impersonation frauds could be detected by observing new trajectories of people in a Smart Building and comparing them to users' historical behavior. We expanded the individuals' mobility profiles described in (CHEN; PANG; XUE, 2014) with the insertion of the notion of *groups*, aiming to detect impersonation attacks. We tested our approach with a dataset from a multi-office building and demonstrated that we improved the performance of the detection by adding groups to the mobility profiles. The second attempt advances our research, explains some changes in models and experimental framework, and expands the previous formal description presented in (SILVA; SICHMAN, 2019). This extended model, as well as a second experiment using a hospital dataset, was originally published in (SILVA; SICHMAN, 2022).

In the next chapter, we will show the basic concepts and algorithms that we used to build our model.

[1] In their work, they called this concept as a swarm.

# 3 BACKGROUND

In the next sections, we present the main concepts and vocabulary extensively used in related literature, including this work. We also introduce a detailed description of the models that inspired the construction of Group-T-Pattern model, that we believe will help the understanding of the model. As mentioned in the previous chapters, our model is based in MiSTA (GIANNOTTI; NANNI; PEDRESCHI, 2006), which in turn is based in Prefix-Span (HAN et al., 2001). The novelty is that we added to MiSTA of the concept of Group Pattern, particularly the concept of a pattern called Swarm (LI et al., 2010). In this chapter, we detail the underlying topics that are needed to the comprehension of these base works and consequently of our model.

## 3.1 Basic concepts and definitions

*Timestamps* [1] are an essential concept of our model. In some systems, trajectory data collection may be done in regular steps, that may not be the same for every moving object, so data collected at almost the same time are grouped in timestamps. In other systems (including PACSs), timestamps are recorded when users reach some point of control, whereby they present their credentials to the credential reader. In this manner, data are often sparse, and we have a different number of records from each user.

*Clustering* is a preprocessing task with the final objective of defining if objects are spatially next to each other at the same timestamp.

The concept of "location" varies depending on the domain of application. A common concept is that of *stay points* (ASHBROOK; STARNER, 2003) (a.k.a. stop points), which are geographic regions where users stay for over a time threshold. These stay points are points of interest (POI) in semantic trajectories. The time users spend in a POI and its labels (that can be the user's home, work, a hospital, restaurant, a meeting room) may be featured for some applications. In the present work, we are going to use *Checkpoints* to determine locations. *Checkpoints* are the sensors in points of control, and in our case, they will be groups of credential readers that control the access to the

[1] Sometimes called time-slot.

same room. They may be, for example, a set of turnstiles installed side by side, giving access to the same building. Indoor environment spaces have natural semantics, such as the labels "room B and "building 2", and indoor semantic hierarchy may establish semantic trajectories, as "to reach building 2 from room B, one needs to use lift C".

As trajectories are sequences of checkpoints, the trajectory pattern mining problem is often reduced to a *frequent sequential pattern* (FSP) problem. These frequent checkpoint sequences occurrences are not necessarily consecutive. Consider a dataset $D$ where a user travels the sequence $Home \rightarrow University \rightarrow Hospital$ in one day, and the next day he travels $Home \rightarrow Office \rightarrow University$. We can find the pattern $Home \rightarrow University$ in this trajectory set even if in the second day the user goes to the $Office$ before going to the $University$. On the other hand, if the user travels $Office \rightarrow University \rightarrow Home$ on a third day, we can not find the pattern $Home \rightarrow University$ in this trajectory: even if both the checkpoints appear, they do not appear in the same order.

*Trajectory Pattern* is a checkpoint sequence that frequently appears in users' trajectories, and the task of discovering these frequent occurrences in trajectory data is called *Trajectory Pattern mining*.

Other common concept is that of *pattern containment*. A trajectory may (or may not) contain a pattern if it is observed in the trajectory. The pattern $Home \rightarrow University$ is a sequence of checkpoints that is contained in both first and second trajectory sequences in the previous example. Formally,

**Definition 3.1:** A sequence $s_a = \langle a_1, a_2, ..., a_p \rangle$ is contained in another sequence $s_b = \langle b_1, b_2, ..., b_q \rangle$, defined by $s_a \preceq s_b$, if there are integers $1 \leq j_1 < j_2 < ... < j_p \leq q$ such that $a_1 = b_{j_1}, a_2 = b_{j_2}, ..., a_p = b_{j_p}$. If $s_a$ is contained in $s_b$, then $s_a$ is a *subsequence* of $s_b$ and $s_b$ is a *supersequence* of $s_a$.

A subsequence is obtained from a sequence by removing 0 or more checkpoints, and a *common subsequence* of a set of sequences is a subsequence of every sequence in the set.

*Pattern support* is the number of trajectories that contain the pattern in a trajectory set. It can be presented in terms of relative frequency: the number of trajectories that contain the pattern divided by the number of trajectories in the dataset, represented by

$|D|$. The pattern $Home \rightarrow University$ in the previous example have $support = 2/3 = 67\%$ as it appears in two out of three trajectories from the trajectory set. If the pattern appears more than once in the same trajectory, only one occurrence is counted.

*Minimum Support* $\sigma$ is a user-defined variable the stands for the least number of times that a trajectory must be found in the trajectory set to be considered as a pattern.

*Frequent Pattern* is a pattern that has support above the minimum support.

## 3.2 Prefix-Span

Prefix-Span is an algorithm originally proposed in (HAN et al., 2001) that aims to find frequent sequential patterns in a sequence dataset. The idea of Prefix-Span method is that if a sequence is frequent, if we add any checkpoint at the end of this sequence, thus making a supersequence, we have to search this supersequence only in the sequences where the original sequence appears in the dataset.

Recalling the example in Section 3.1, consider the following dataset $D$:

$$\begin{aligned} &D: \\ &Day\ 1: Home \rightarrow University \rightarrow Hospital \\ &Day\ 2: Home \rightarrow Office \rightarrow University \\ &Day\ 3: Office \rightarrow University \rightarrow Home \end{aligned}$$

In this trajectory dataset, we have four checkpoints: $Home$, $University$, $Office$, and $Hospital$. It is possible to see that $Office$, as a sequence of lenght = 1, occurs only in Day 2 and Day 3, so if we want to find any sequence starting with $Office$, we should search for it only in these trajectories. The strategy is to make a *prefix-projected* dataset, where we remove the occurrence of the *prefix* $Office$ and everything before it from the original dataset. On the other hand, only sequences frequent enough shall be extended. For example, if some define that the minimum support threshold is $\sigma = 50\%$, sequences starting with $Hospital$ will not be extended because it appears only in one sequence of the dataset, thus have $support = 1/3 = 33\% < \sigma$, and there is no point in searching for sepersequences of it. By doing this process of projecting the dataset $D$ with respect to $Office$, we obtain the projection $D|_{Office}$:

$$
\begin{aligned}
D|_{Office} :& \\
& Day\ 2 : University \\
& Day\ 3 : University \rightarrow Home
\end{aligned}
$$

In Prefix-Span, a projection is a simplification of the data which (i) contains only the trajectories of the original dataset that the checkpoint under analysis appears, (ii) contains only frequent checkpoints, and (iii) on each trajectory, the first occurrence of the checkpoint and all the sequence elements that precede it are removed.

Notice that we may obtain many projected datasets $D|_{checkpoint}$ from the original dataset $D$, one for each frequent checkpoint. The fundamental idea is that any pattern starting with $checkpoint$ can be obtained by analyzing only $D|_{checkpoint}$.

Now, the projected dataset $D|_{Office}$, has only two checkpoints: $Home$ and $University$, so we can potentially obtain the prefixes $Office \rightarrow University$ and $Office \rightarrow Home$ from it. As the $support$ value is calculated by considering the size of the original dataset $|D| = 3$, and in this case the checkpoint $Home$ appears in only one trajectory of the projected dataset, $support = 1/3 < \sigma$. Thus, the sequence $Office \rightarrow Home$ is not frequent and does not generate any projected dataset. On the other hand, the sequence $Office \rightarrow University$ have $support = 66\% \geq \sigma$, hence it is frequent, and the projected dataset with respect to the prefix $Office \rightarrow University$ will result in:

$$
\begin{aligned}
D|_{Office \rightarrow University} :& \\
& Day\ 3 : Home
\end{aligned}
$$

Since the occurrence of $Office$ and all checkpoints that precede it are removed, trajectory $Day2$ is empty and removed from the projected dataset. There is no need to store it since it will not produce any longer sequences. There is only one checkpoint

$Home$, and the support $1/3$ for the remaining sequence is smaller than $\sigma$; therefore, there are no more projections to generate. The process shall be repeated to all possible prefixes and projections until there are no more projections to extend. All the discovered prefixes are *frequent sequence patterns* found in the trajectory dataset.

## 3.3 MiSTA

MiSTA (GIANNOTTI; NANNI; PEDRESCHI, 2006) adopts and extends Prefix-Span with the addition of transition times information. Besides having a sequence of checkpoints, it considers a sequence of *transition times* between any two checkpoints. The goal is to find frequent checkpoint sequences that have similar transition times among them.

When added the concept of transition times to the pattern, it is necessary to modify the concept of pattern containment. Suppose that the dataset $D$ from the example in Section 3.2 is now time annotated[2] and have Day 4 added to it, as seen in $D_2$:

$$
\begin{aligned}
&D_2 : \\
&Day\ 1 : Home \xrightarrow{5} University \xrightarrow{3} Hospital \\
&Day\ 2 : Home \xrightarrow{5} \mathbf{Office} \xrightarrow{\mathbf{3}} \mathbf{University} \\
&Day\ 3 : \mathbf{Office} \xrightarrow{\mathbf{4}} \mathbf{University} \xrightarrow{7} Home \\
&Day\ 4 : \mathbf{Office} \xrightarrow{\mathbf{5}} Home \xrightarrow{\mathbf{6}} \mathbf{University}
\end{aligned}
$$

The transition times between checkpoints are annotated above the arrows. We can use Prefix-Span described in Section 3.2 to determine that the sequence of checkpoints $Office \rightarrow University$ is frequent in this dataset.

*Maximum time threshold* $\tau$ is a time tolerance that determines if a trajectory contains a pattern. The pattern will be $\tau$-contained in a trajectory if: (i) the trajectory contains the sequence of checkpoints of the pattern, and (ii) the transition times between the checkpoints similar. Two transition times are considered similar if the difference between the transition time of the pattern and the transition time of the trajectory is

[2]We have stressed the $Office \rightarrow University$ sequence by representing it in red.

smaller than the maximum time threshold $\tau$. If the checkpoints are not consecutive in the checkpoint sequence, the transition times between them must be all summed. In Day 4, for example, the checkpoint sequence $Office \rightarrow University$ has transition time $5 + 6 = 11$, thus $Office \xrightarrow{11} University$ must be considered.

A pattern is now being considered a frequent pattern if it is $\tau$-contained in at least $\sigma$ times the number of trajectories in the dataset. A formal definition of $\tau$-containment is presented in 3.2.

**Definition 3.2:** a sequence $s = (\overline{s}, \overline{\alpha}) = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} s_n$ is $\tau$-contained in a sequence $s' = (\overline{s'}, \overline{\alpha'}) = s'_0 \xrightarrow{\alpha'_1} s'_1 \xrightarrow{\alpha'_2} ... \xrightarrow{\alpha'_m} s'_m$ if

1. $s \leq s'$ ($s$ is a subsequence of $s'$, Definition 3.1)

2. $\forall_{1 \leq k \leq n}$, $|\alpha_k - \alpha_{*k}| \leq \tau$, where $\alpha_{*k} = \sum_{i_{k-1} < j < i_k} \alpha'_j$

In order to illustrate this concept of $\tau$-containment, let us assume that the pattern $Office \rightarrow University$ has transition time equals 5.5, represented by $Office \xrightarrow{5.5} University$, and that the tolerance time $\tau$ is set to 2. This pattern will be contained in trajectory "Day 3", as $Office \rightarrow University$ is a subsequence of the checkpoint sequence of the trajectory, and the difference between pattern transition time and trajectory transition time is smaller than $\tau$ ($|4-5.5| = 1.5 < \tau$). On the other hand, even though the checkpoint sequence in trajectory "Day 2" is also a supersequence of $Office \rightarrow University$, it does not contain the pattern $Office \xrightarrow{5.5} University$ because the difference between pattern transition times and trajectory transition times are higher than the maximum time threshold $|3 - 5.5| = 2.5 > \tau$. Similarly, this pattern is not contained in "Day 4", as $|11 - 5.5| = 5.5 > \tau$.

On the other hand, assuming a second pattern $Office \xrightarrow{3} University$, we can see that it is contained in trajectories of "Day 2" ($|3 - 3| = 0 < \tau$) and "Day 3" ($|4 - 2| = 1 < \tau$), that is 2 out of 4 trajectories in the dataset and if $\sigma$ is set to 50%, this pattern is considered a frequent pattern. Notably, the possible transition times from patterns $Office \rightarrow University$ that satisfy the $\tau$-containment conditions in $D_2$ form transition a time *interval*. Any sequence $Office \rightarrow University$ annotated with transition time between 2 and 5 is $\tau$-contained in trajectories Day 2 and Day 3 from $D_2$ and hence is frequent in this dataset. A *T-Pattern* (Trajectory Pattern) represents all the (infinite) time annotated checkpoint sequences $\tau$-contained in at least $\sigma$ times the number of trajectories in the dataset, and is noted as:

$$s_0 \xrightarrow{[l_1,h_1]} s_1 \xrightarrow{[l_2,h_2]} \ldots \xrightarrow{[l_n,h_n]} s_n$$

Being $\overline{s}$ the sequence of checkpoints and $\overline{[l,h]}$ the sequence of time intervals. In this example:

$$Office \xrightarrow{[2,5]} University$$

In order to find the T-patterns contained by the trajectories, MiSTA adds and subtracts the maximum time threshold $\tau$ to the transition times of all the occurrences of the frequent checkpoint sequences to obtain the edges of the influence region of each trajectory. In the intersection region with support larger than the minimum support threshold $\sigma$, we will find all possible T-Patterns contained in the trajectory dataset. Figure 2 shows this process for the sequence $Office \xrightarrow{\alpha} University$ of the dataset $D_2$ from the example. It is possible to imagine that if we had a larger dataset, more influence regions could occupy the interval [2,5] and the intersection regions would be *denser*. The influence regions shall be dense enough to become a pattern.

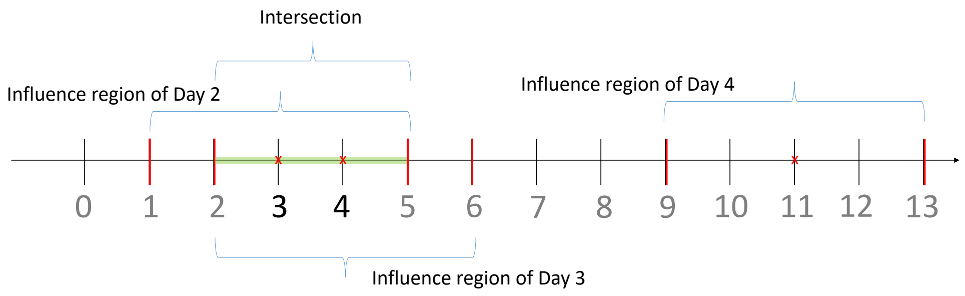

Intersection
Influence region of Day 2
Influence region of Day 4
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Influence region of Day 3

Figure 2: Influence regions of checkpoint subsequence $Office \xrightarrow{\alpha} University$ in MiSTA

If we have a checkpoint sequence one element longer, it will have two transition times and the influence regions will have dimension 2. A similar, more complex example is $D_3$ that is graphically represented in Figure 3, where all occurrences of the checkpoint sequence $Home \xrightarrow{\alpha_1} Office \xrightarrow{\alpha_2} University$ from dataset $D_3$ are plotted.

$D_3$ :

$Day\ 1 : \mathbf{Home} \xrightarrow{\mathbf{20}} \mathbf{Office} \xrightarrow{\mathbf{10}} \mathbf{University} \xrightarrow{3} Hospital$

$Day\ 2 : \mathbf{Home} \xrightarrow{\mathbf{10}} \mathbf{Office} \xrightarrow{\mathbf{20}} Hospital \xrightarrow{\mathbf{15}} \mathbf{University}$

$Day\ 3 : Office \xrightarrow{4} \mathbf{Home} \xrightarrow{\mathbf{22}} \mathbf{Office} \xrightarrow{\mathbf{22}} \mathbf{University}$

$Day\ 4 : \mathbf{Home} \xrightarrow{\mathbf{35}} Hospital \xrightarrow{\mathbf{10}} \mathbf{Office} \xrightarrow{\mathbf{5}} \mathbf{University}$

$Day\ 5 : \mathbf{Home} \xrightarrow{\mathbf{50}} \mathbf{Office} \xrightarrow{\mathbf{40}} \mathbf{University} \xrightarrow{30} Bar$

$Day\ 6 : \mathbf{Home} \xrightarrow{\mathbf{38}} \mathbf{Office} \xrightarrow{\mathbf{50}} \mathbf{University} \xrightarrow{33} Hospital$

$Day\ 7 : \mathbf{Home} \xrightarrow{\mathbf{70}} \mathbf{Office} \xrightarrow{\mathbf{30}} Hospital \xrightarrow{\mathbf{35}} \mathbf{University}$

$Day\ 8 : Hotel \xrightarrow{4} \mathbf{Home} \xrightarrow{\mathbf{65}} \mathbf{Office} \xrightarrow{\mathbf{50}} \mathbf{University}$

$Day\ 9 : \mathbf{Home} \xrightarrow{\mathbf{5}} Hospital \xrightarrow{\mathbf{5}} \mathbf{Office} \xrightarrow{\mathbf{70}} \mathbf{University}$

$Day\ 10 : \mathbf{Home} \xrightarrow{\mathbf{59}} \mathbf{Office} \xrightarrow{\mathbf{67}} \mathbf{University} \xrightarrow{30} Club$

The dataset $D_3$ contains ten trajectories, and each one presents exactly one occurrence of the checkpoint sequence $Home \xrightarrow{\alpha_1} Office \xrightarrow{\alpha_2} University$. If we map these occurrences to *dataset points* $(\alpha_1, \alpha_2)$ in $\mathbb{R}^2$, we will obtain $\{(20, 10), (10, 35), (22, 22), (35, 15), (50, 40), (38, 50), (70, 65), (65, 50), (10, 70), (59, 67)\}$. These points will be the center of of the squares in figure 3. The influence regions of these dataset points will be squares of edges $2\tau$ centered in the dataset points. Figure 3 graphically shows the influence regions, which are limited by the red squares, while the colored regions are dense, regions, that is, where the influence regions overlap. The darkest (blue) regions correspond to the infinitely many transition times $\alpha_1$ and $\alpha_2$ that make the checkpoint sequence a frequent pattern for $\sigma = 0.3$ and $\tau = 10$. Similarly, the lighter (green) regions (plus the darkest/blue ones, implicitly) represent frequent transition times for $\sigma = 0.2$, and outlined regions correspond to frequent transition times for $\sigma = 0.1$. It is possible to see that small values of $\sigma$ result in more and larger regions that correspond to frequent patterns transition times. On the other hand, larger values of $\sigma$ produce fewer and smaller regions.

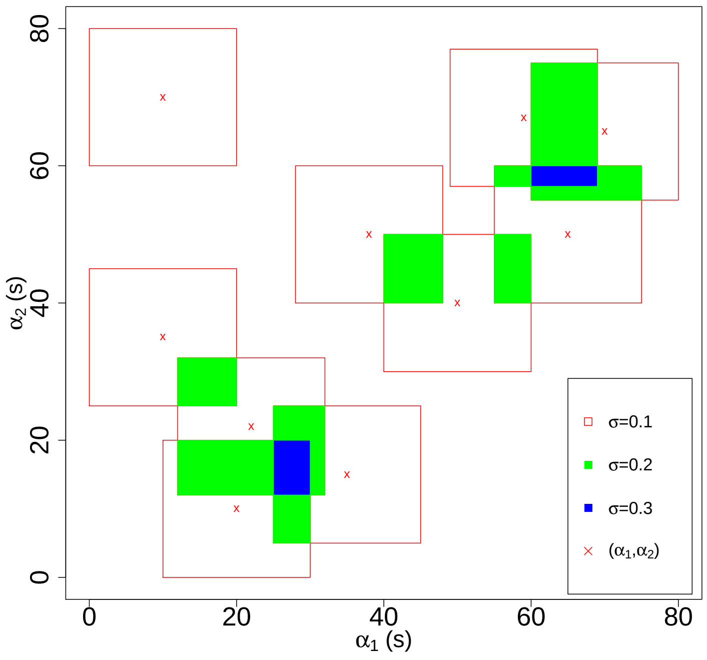


Figure 3: Influence regions of checkpoint subsequence $Home \xrightarrow{\alpha_1} Office \xrightarrow{\alpha_2} University$ in MiSTA
Source: Adapted from (GIANNOTTI; NANNI; PEDRESCHI, 2006)

While MiSTA algorithm uses prefix-span to find checkpoint sequences that are one checkpoint longer at each step, dense regions get one dimension higher. Thus, what are areas in $\mathbb{R}^2$ become cubes in $\mathbb{R}^3$ that turns into hypercubes in the hyperspace $\mathbb{R}^N$. Therefore, MiSTA reduces a trajectory pattern mining task to a geometry problem in the hyperspace. The algorithm splits this task of finding dense regions in the hyperspace into smaller tasks:

1. Extract Annotation Blocks: this task builds the influence regions, i.e., the hypercubes centered in coordinates given by the transition times from the dataset with edges $2\tau$;

2. Compute Density Blocks: splits regions of homogeneous density in hyperrectangles and selects those with a sufficiently high density. The output of the

task is a set of hyper-rectangles of different densities that may cover large dense regions;

3. Coalesce Density Blocks: merges hyper-rectangles that cover adjacent regions with different densities but still have a density higher than the minimum support. The output is a smaller set of hyper-rectangles that covers the same volume as the original set so that the output of the whole process is a more concise set of patterns;

4. Annotation-Based Prune: if some trajectory did not contribute to the generation of dense regions for a given prefix, it will not contribute to any extension of a prefix by the addition of one checkpoint. Hence, this trajectory is removed from the projected dataset to save computational effort in the next iterations.

The model proposed in the present work is essentially based in MiSTA, and these tasks and the corresponding algorithms are further described in detail in Chapter 5, where we explain our proposed algorithm.

## 3.4 ObjectGrowth

In our model, we use the concept of *Swarm*, a Group Pattern introduced in (LI et al., 2010), that also proposes an algorithm to mine Swarms called ObjectGrowth.

*Swarm* as a Group Pattern definition, relaxes the constraint of the clusters to remain for consecutive timestamped access events, so cluster members move together for certain timestamps that are possibly non-consecutive. Figure 4 shows an example of input database extracted from (LI et al., 2010), with clusters and associated timestamps.

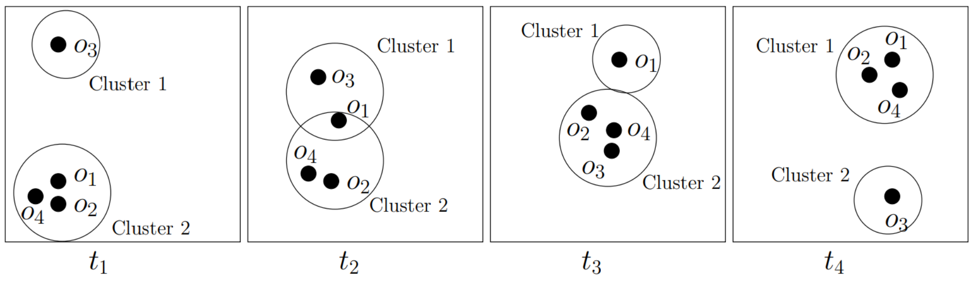


Figure 4: Example of moving clusters
Source: (LI et al., 2010)

There are 4 users and 4 timestamps ($U_{DB} = \{o_1, o_2, o_3, o_4\}$, $T_{DB} = \{t_1, t_2, t_3, t_4\}$). Each sub-figure is a snapshot of user clusters at each timestamp. We can observe that $o_1$, $o_2$, and $o_4$ travel together for most of the time, and $o_2$ and $o_4$ form an even more stable swarm since they are close to each other in the whole time span. Given $min_o = 2$ and $min_t = 2$, there are totally 15 swarms: $(\{o1, o2\}, \{t1, t2\})$, $(\{o_1, o_4\}, \{t_1, t_2\})$, $(\{o_2, o_4\}, \{t_1, t_3, t_4\})$, and so on. Some swarms are redundant. For example: $(\{o_2, o_4\}, \{t_1, t_2\})$ and $(\{o_2, o_4\}, \{t_2, t_3, t_4\})$ are not time-closed since both of them can be enlarged to form another swarm: $(\{o2, o4\}, \{t_1, t_2, t_3, t_4\})$. Similarly, $(\{o_1, o_2\}, \{t_1, t_2, t_4\})$ and $(\{o_2, o_4\}, \{t_1, t_2, t_4\})$ are redundant (not object-closed) since both of them can be enlarged as $(\{o_1, o_2, o_4\}, \{t_1, t_2, t_4\})$. There are only two closed swarms in this example: $(\{o_2, o_4\}, \{t_1, t_2, t_3, t_4\})$ and $(\{o_1, o_2, o_4\}, \{t_1, t_2, t_4\})$.

A pair $(O, T)$ is said to be a swarm if all objects in $O$ are in the same cluster at any timestamp in $T$.

**Definition 3.3:** Given two minimum thresholds $min_o$ and $min_t$, for $(O, T)$ to be a swarm, where $O = \{o_{i_1}, o_{i_2}, ..., o_{i_p}\} \subseteq U_{DB}$ and $T \subseteq T_{DB}$, it needs to satisfy three requirements:

1. $|O| \leq min_o$: There should be at least $min_o$ users;
2. $|T| \leq min_t$: Users in O are in the same cluster for at least $min_t$ timestamps;
3. $\forall t_i \in T$, $C_{t_1}(o_{i_1}) \cap C_{t_2}(o_{i_2}) \cap ... \cap C_{t_n}(o_{i_n}) \neq \emptyset$ : there is at least one cluster containing all the users in $O$ at each timestamp in $T$. $C_{t_i}(o_j)$ denotes the set of cluster that user $o_j$ is in timestamp $t_i$.

A swarm $(O, T)$ is object-closed if fixing $T$, $O$ cannot be enlarged ($\nexists O'$ s.t. $(O', T)$ is a swarm and $O \subsetneq O'$). Similarly, a swarm $(O, T)$ is time-closed if fixing $O$, $T$ cannot be enlarged ($\nexists T'$ s.t. $(O, T')$ is a swarm and $T \subsetneq T'$) Finally, a swarm $(O, T)$ is a closed swarm if it is both object-closed and time-closed.

In (LI et al., 2010), the authors describe an efficient algorithm to mine closed swarms called *ObjectGrowth*, that is basically a depth-first search algorithm that starts with one cluster of size 1 (in this example, the clusters $\{o_1\}]$, $\{o_2\}]$, $\{o_3\}]$, $\{o_4\}]$), and checks if it lasts for at least $min_t$ timestamps. If so, the algorithm then adds one user to that cluster in order to have clusters of size 2. Expanding the node $\{o_1\}]$ would give the branches $\{o_1, o_2\}$, $\{o_1, o_3\}$, $\{o_1, o_4\}$. The algorithm then checks if each new branch lasts for $min_t$ and continues expanding each branch and testing until all possibilities are exhausted. There are two pruning strategies in ObjectGrowth: (i) *Apriori pruning* rule considers that if a cluster with $n$ users does not last for $min_t$, there is no superset of this cluster by adding one more user that is a swarm, and (ii) *Backward pruning* prevents the DPS algorithm from revisiting a cluster that has already been found as a swarm.

We have been inspired by and modified the ObjectGrowth algorithm to extend projections in the Prefix-Span algorithm. The new algorithm will be explained in detail in Chapter 5.

## 3.5 All Common Subsequences

The concept of All Common Subsequences (ACS) as a similarity measure is firstly described in the work (WANG, 2007). ACS similarity measure is applied by counting the number of all common subsequences, so two sequences are more similar to each other if they have more common subsequences. In order to illustrate ACS, consider two sequences:

$$
\begin{aligned}
&D_4 : \\
&\quad Day\ 1 : Home \rightarrow University \rightarrow Hospital \\
&\quad Day\ 2 : Home \rightarrow Office \rightarrow University
\end{aligned}
$$

The set of all common subsequences of "$Day1$" and "$Day2$" is (ignoring the empty sequence) is $Home$, $University$, $Home \rightarrow University$. Therefore these two sequences have a similarity equals to 3.

We can normalize the similarity according to the length of the sequences. For a sequence of length $|\alpha|$, the number of its subsequences is $\sum_{i=1}^{|\alpha|} \binom{i}{|\alpha|} = 2^{|\alpha|} - 1$, which does not include the empty sequence. If we compare sequences of different lengths, the length of the smaller sequence shall be considered. In the case of the example, the sequences have length equals 3. Being the number of common subsequences equals

3, the normalized ACS score is: $3/(2^3 - 1) = 3/7 = 0.43$.

## 3.6 Anomaly detection

Supervised *anomaly-based detection* (also known as outlier detection) consists in observing the regular operation of a system and alarm if some abnormal activity that may suggest an attack is observed (BADDAR; MERLO; MIGLIARDI, 2014). In many applications, *behavioral profiles* model the regular use of a system to further compare them with future users' behavior and detect possible anomalies.

Two main concepts have to be defined when adopting an anomaly-based detection approach:

- *Profiles* model the regular use of the system by each user. We adopted a profile model that summarizes frequent trajectories: sequences of checkpoints, transition times, and groups.
- *Similarity* measures are then taken to compare if new captured behaviors are consistent with previous behavior summarized by the profile. The output of a similarity measure is a single score. Lower scores mean a high probability of the trajectory being an anomaly. It is necessary to set a *threshold* to define the minimum score for a trajectory to be rejected as an anomaly.

*Effectiveness* of an anomaly detection systems is usually measured by two error rates, namely False Acceptance Rate (FAR), in which the system say that a trajectory is an anomaly when it is not, therefore generating a false alarm, and False Rejection Rate (FRR), when the system accepts an intruder as a legitimate user. *Accuracy* is employed to quantify the detection performance in a single number. Another method employed to quantify classification performance in terms of FAR and FRR is the area under the *Receiver Operating Characteristic Curve* (ROC). Area Under the Curve (AUC) is scale-invariant: it measures how well predictions are ranked, rather than their absolute values. AUC larger than 0.5 means that the classifier under evaluation is better than a random classifier. AUC equals 1 means a perfect classifier.

## 3.7 Discussion

Having introduced the main concepts, we can now present the proposed model of Group-T-Pattern. The ideas of T-Pattern and Swarm are incorporated into our model, as well as the ideas from the algorithms MiSTA and ObjectGrowth, respectively designed to mine these two patterns.

In this work, we adopted the concept of All Common Subsequences (ACS) described in the work (WANG, 2007) to measure similarities. We extended this concept of ACS to consider similarities regarding checkpoint sequences, transition times, and group sequences.

Figure 5 shows the evolution of the models and how Group-T-Pattern incorporate the already presented concepts.

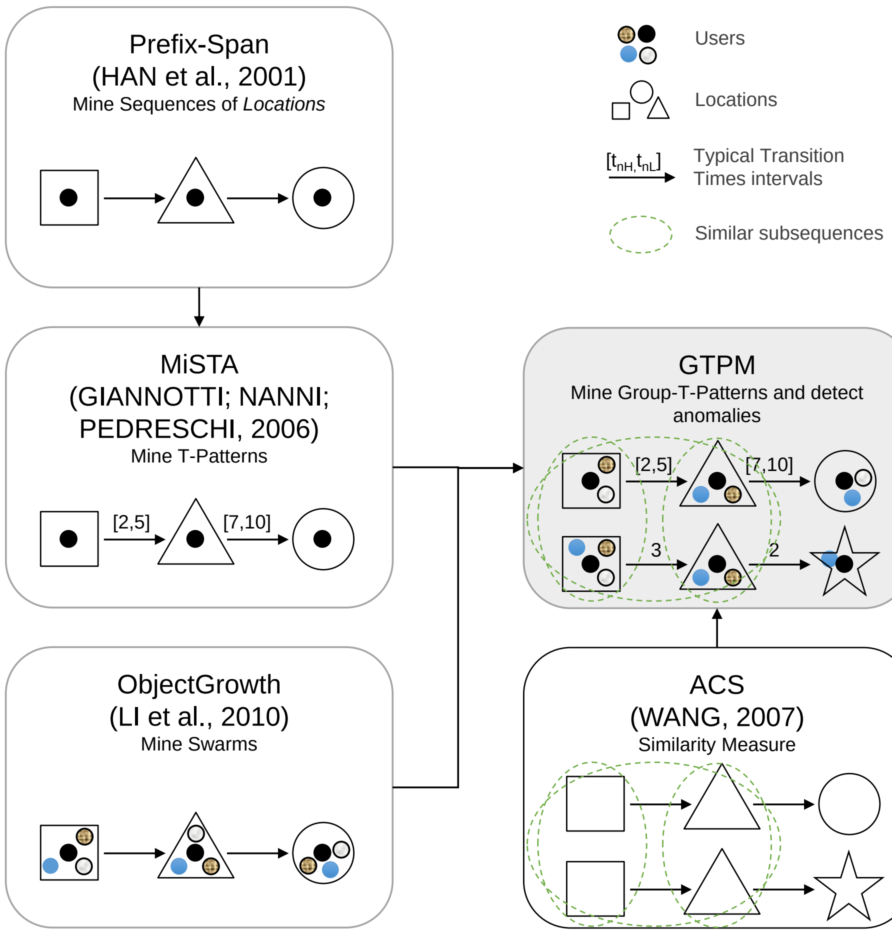


Figure 5: Algorithms Evolution

In Chapter, 4, we formally describe the Group-T-Pattern model, as well as the new proposed similarity measure. Chapter 5 details the Group Trajectory Pattern Mining (GTPM), the algorithm that implements Group-Trajectory-Pattern model.

# 4 GROUP-T-PATTERNS MODEL

In this chapter, we formally introduce our Mobility Profile model called Group-Trajectory-Pattern. We have used some ideas of the works described in Chapter 3 regarding the construction of mobility profiles, in particular (GIANNOTTI; NANNI; PEDRESCHI, 2006) and (LI et al., 2010), but the novelty in our approach is the insertion of *groups* to individuals' mobility profiles. Groups describe an abstraction of systematic companions that can let us infer social bonds.

## 4.1 Preprocessing

The proposed model uses Audit Logs from smart buildings access control systems to construct mobility profiles. Typically, these datasets are composed by *events* that have at least three attributes, which are: *who* performed the access request, *when*, and to *where*. Users are often enrolled with many pieces of information: their full name, social security or driver license numbers, address, and other personal data. We worked with anonymized datasets, so only the user id, a unique and random numeric identifier for each user, is used for identification. In order to request access, users present their credentials to credential *readers*, so we know the exact location of the users by looking at the name of the readers. The system also stores a timestamp, indicating when the credential was presented to the reader. Table 1 shows an example of how these data are stored, with each line corresponding to an event: the moment when the user presents a credential to a credential reader.

The main challenge in using such datasets for identity theft is that we have few data to explore. Other application domains, such as credit cards (TRIPATHI; PAVASKAR, 2012), have much more data to work with, for example: users' home addresses, delivery address, network IP address, and amount of purchasing. The work (ZHU et al., 2019) uses call duration, origin and destination provinces. In order to make possible the use of groups in the presented model, some *preprocessing* steps are taken to build new features: *checkpoints* and *clusters*. The result of this preprocessing step is shown in Table 2.

Table 1: PACS raw events data.

| User | Timestamp | Credential Reader |
|---|---|---|
| $u_1$ | 8:30:04 | Turnstile 1 |
| $u_2$ | 8:30:22 | Turnstile 2 |
| $u_3$ | 8:30:48 | Turnstile 2 |
| $u_4$ | 8:31:57 | Turnstile 3 |
| $u_5$ | 8:32:01 | Turnstile 1 |
| $u_6$ | 8:38:12 | Turnstile 2 |
| ⋮ | ⋮ | ⋮ |
| $u_2$ | 12:13:55 | Warehouse |
| ⋮ | ⋮ | ⋮ |
| $u_2$ | 13:07:15 | Back gate |
| ⋮ | ⋮ | ⋮ |
| $u_2$ | 17:30:38 | Main exit |
| ⋮ | ⋮ | ⋮ |

### 4.1.1 Checkpoints

In the presented model, firstly, we have to aggregate readers that are physically very close to each other and provide access to the same room. In our experiments, experts with in-depth knowledge about the environment did this task. Checkpoint is a group of readers that has the same semantic context, for example, a set of turnstiles installed side by side that control access to the same hall.

**Definition 4.1:** Checkpoint is a set of credential readers that give access to the same room.

In the example dataset from Table 1, *Turnstile 1*, *Turnstile 2*, and *Turnstile 3* provide access to the same entrance hall and can be aggregated together as "Entrance". The other credential readers are the only ones that control access to the rooms they are installed in, and were not aggregated. The capital letters E, W, B, and X respectively designate the Entrance, Warehouse, Back gate, and main eXit checkpoints in Table 2.

### 4.1.2 Users clustering

The second task is finding *clusters*. Clustering is a preprocessing step that aims to find users that are moving together. We have adopted the concept of *time window* to

Table 2: PACS events after preprocessing for $t\_window = 30s$

| User | Timestamp | Checkpoint | Cluster |
|---|---|---|---|
| $u_1$ | $t_1$ = 8:30:04 | E | $c_1 = \{u_1, u_2\}$ |
| $u_2$ | $t_2$ = 8:30:22 | E | $c_1 = \{u_1, u_2\}$, $c_2 = \{u_2, u_3\}$ |
| $u_3$ | $t_3$ = 8:30:48 | E | $c_2 = \{u_2, u_3\}$ |
| $u_4$ | $t_4$ = 8:31:57 | E | $c_3 = \{u_4, u_5\}$ |
| $u_5$ | $t_5$ = 8:32:01 | E | $c_3 = \{u_4, u_5\}$ |
| $u_6$ | $t_6$ = 8:38:12 | E | $c_4 = \{u_6\}$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| $u_2$ | $t_7$ = 12:13:55 | W | $c_5 = \{u_2\}$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| $u_2$ | $t_8$ = 13:07:15 | B | $c_6 = \{u_2\}$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| $u_2$ | $t_9$ = 17:30:38 | X | $c_7 = \{u_2\}$ |
| ⋮ | ⋮ | ⋮ | ⋮ |

build clusters: Cluster is a set of users that reached the same checkpoint within a time window $t\_window$.

Let $D$ denote the whole events dataset from a PACS. $T_{DB}$ denote the set of all timestamps in $D$, $U_{DB}$ denote the set of all users in $D$, and $S_{DB}$ denote the set of all checkpoints in $D$.

**Definition 4.2:** Given a maximum time window threshold $t\_window$, $U'_{DB} = \{u_{i_1}, u_{i_2}, ..., u_{i_n}\}$, $U'_{DB} \subseteq U_{DB}$, is a cluster if there is $T'_{DB} = \{t_{i_1}, t_{i_2}, ..., t_{i_n}\}$, $T'_{DB} \subseteq T_{DB}$, being $t_{i_q}$ a timestamp from an event of user $u_{i_q}$, such that $\forall t_{i_o}, t_{i_p} \in T'_{DB}, |t_o - t_p| \leq t\_window$

Notice that users may belong to more than one cluster at the same event timestamp. If we consider $t\_window = 30s$, we can see in Table 2 that user $u_2$ in $t_2$ belongs to cluster $c_1 = \{u_1, u_2\}$ because $|t_2 - t_1| = 18 \leq 30s$, and $t_1$ is a timestamp from an event performed by $u_1$. Similarly, $u_2$ in $t_2$ also belongs to cluster $c_2 = \{u_2, u_3\}$ because $|t_2 - t_3| = 26 \leq 30s$. On the other hand, $u_1$ and $u_3$ doesn't belong to the same cluster, since $|t_1 - t_3| = 44 > 30s$. Clusters mined for the same event will form a *cluster set* [1].

[1]We adopt lowercase letters to represent elements of a sequence, even though sometimes these elements represent sets of single elements. Hence, we will use the lowercase letter $c$ to represent a cluster, which is a set of users $\{u_0, ..., u_n\}$, $cs$ to represent a cluster set $\{\{u_a, ..., u_b\}, ..., \{u_c, ..., u_d\}\}$, and $g$ to represent a group, which is also a user set $\{u_e, ..., u_f\}$. On the other hand, we are going to adopt capital letters to represent *sequences* of these elements, in particular $CS$ for a sequence of Cluster Sets and $G$ for a sequence of Groups.

## 4.2 Constructing profiles

The mobility profile is the key concept of our framework. It summarizes all users' systematic movements regarding space, time, and travel companions (groups). Users' movements are represented by a set of trajectories that describe their paths, i.e., their routines.

### 4.2.1 Trajectory

Let $T_{DB} = \{t_1, t_2, ..., t_t\}$ be the set of all timestamps, $S_{DB} = \{s_1, s_2, ..., s_m\}$ be the set of all checkpoints, $U_{DB} = \{u_1, u_2, ..., u_n\}$ be the set of all users, and $CS_{DB} = \{cs_1, cs_2, ..., cs_t\}$ is the collection of all user cluster sets in a PACS dataset.

A *temporal annotation* $\alpha_n$ represents the elapsed time that a user has taken to navigate between checkpoints $s_n$ and $s_{n+k}$, $k \in \mathbb{N}$: if the user was respectively detected in $s_n$ and $s_{n+k}$ at timestamps $t_n$ and $t_{n+k}$, the corresponding temporal annotation is $\alpha_m = t_{n+k} - t_n$.

**Definition 4.3:** A Trajectory $T = \langle S, A, CS \rangle$ is a temporally-annotated sequence of length $n > 0$, where $S = (s_0, ..., s_i)$ is called the checkpoint sequence, $A = (\alpha_1, ..., \alpha_n) \in \mathbb{R}^n_+$ is called the (temporal) annotation and $CS = (cs_0, ..., cs_i)$ is called the cluster set sequence, and $\forall_{0 \leq i \leq n}, s_i \in S_{DB}, cs_i \in CS_{DB}$.

A *Trajectory* can also be represented as follows:

$$(s_0, cs_0) \xrightarrow{\alpha_1} (s_1, cs_1) \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_m} (s_i, cs_i)$$

Considering again the dataset presented in Table 2, a more concrete example of a trajectory for user $u_2$ is shown in Equation 4.1. The temporal annotation is expressed in minutes.

$$T_1^{u_2} = (E, \{\{u_1, u_2\}, \{u_2, u_3\}\}) \xrightarrow{223} (W, \{\{u_2\}\}) \xrightarrow{53} (B, \{\{u_2\}\}) \xrightarrow{263} (X, \{\{u_2\}\}) \tag{4.1}$$

### 4.2.2 Group-T-Pattern

As mentioned before, the contribution of our work is to extend the mobility profiles with the addition of social context, i.e, considering *group* profiles. In our model, a Group $g = \{u_1, u_2, ..., u_n\}, u_i \in U_{DB}$ is a set of users. In short, the notion of cluster represents users in a checkpoint in a specific moment in time, while the notion of group captures whether these users are frequently together. The final outputs of the method are *frequent* Group-T-Pattern, i.e., Group-T-Patterns contained in many trajectories.

**Definition 4.4:** For a user $u_j$, a Group-T-Pattern is defined as a triple $\langle S, A, G\rangle$ where $S = (s_0, ..., s_n)$ is a sequence of checkpoints, $A = (\alpha_1, ..., \alpha_n)$ is the sequence of temporal annotations between checkpoints, and $G = (g_0, ..., g_n)$ is the sequence of groups $g = \{u_0, ..., u_j, ..., u_m\}$ to which the user $u_j$ belongs to at each checkpoint of the sequence.

A Group-T-Pattern (we will sometimes call *pattern*, for short) can be contained in a trajectory. This is ultimately the main contribution of our work.

**Definition 4.5:** Given a maximum tolerance time threshold $\tau$, a Group-T-Pattern $\langle S, A, G\rangle = (s_0, g_0) \xrightarrow{\alpha_1} (s_1, g_1) \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} (s_n, g_n)$ is fully contained in the trajectory $T = (s'_0, cs_0) \xrightarrow{\alpha'_1} (s'_1, cs_1) \xrightarrow{\alpha'_2} ... \xrightarrow{\alpha'_m} (s'_m, cs_m)$ (denoted by $\langle S, A, G\rangle \preceq_F T$ ) if and only if exists a sequence of integers $0 \leq i_0 < ... < i_n \leq m$ such that:

1. $\forall_{0 \leq k \leq n},\ s_k = s'_{i_k}$: the sequence of checkpoints from the pattern is a subsequence of those from the trajectory, i.e, the trajectory is a supersequence of the pattern;

2. $\forall_{0 \leq k \leq n},\ \exists\ c \in cs_{i_k}\ s.t.\ g_k \subseteq c$: for all the groups of the pattern, there is at least one cluster $c$ from the corresponding cluster set $cs_{i_k}$ from the trajectory that contains all the users of the corresponding group $g_k$;

3. $\forall_{1 \leq k \leq n},\ |\alpha_k - \alpha_{*k}| \leq \tau$, where $\alpha_{*k} = \sum_{i_{k-1} < j < i_k} \alpha'_j$: transition times differences are not greater than the maximum time threshold $\tau$.

By definition, in the particular case of the length of $\langle S, A, G\rangle$ is equals 1, that is, $A = \emptyset$, being the conditions 1 e 2 satisfied, $\langle S, A, G\rangle$ is contained in the trajectory.

For instance, given $\tau = 10$, we can say that the Group-T-Pattern $\langle S, A, G\rangle$ presented in Equation 4.2 is contained in the trajectory $T$ introduced in Equation 4.1.

$$\langle S,A,G\rangle = (E,\{u_2,u_3\}) \xrightarrow{216} (W,\{u_2\}) \xrightarrow{55} (B,\{u_2\}) \xrightarrow{272} (X,\{u_2\}) \tag{4.2}$$

We can remark that a trajectory may contain infinitely many Group-T-patterns. For instance, considering the trajectory $T_1^{u_2}$ in Equation 4.1 and assuming the checkpoint sequence $E \rightarrow W$, and $\tau = 10$, for any $t$ such that $213 \le t \le 233$, $g \in \{\{u_2\},\{u_1,u_2\},\{u_2,u_3\}\}$, the pattern $(E,g) \xrightarrow{t} (W,\{u_2\})$ will be contained in $T_1^{u_2}$. The time interval $213 \le t \le 233$ is obtained by subtracting $\tau$ from the transition time 223 to get the lower limit and adding $\tau$ to it to get the upper boundary.

In order to better represent this situation, we use $[t_L,t_H]$ to respectively represent the lower and upper bounds of the time interval. Thus, $\{(E,\{u_2\}) \xrightarrow{[213,233]} (W,\{u_2\})$, $(E,\{u_1,u_2\}) \xrightarrow{[213,233]} (W,\{u_2\})$, $(E,\{u_2,u_3\}) \xrightarrow{[213,233]} (W,\{u_2\})\}$ represent the set of all Group-T-patterns with the checkpoint sequence $E \rightarrow W$ contained in $T_1^{u_2}$ given $\tau = 10$.

In our model, we adopt the concept of *closed* Group-T-Patterns. The strategy is to consider only Group-T-Patterns containing the larger groups (removing those containing group subsets), aiming to have a more concise set of Group-T-Patterns. In the case of the example, only the patterns $(E,\{u_1,u_2\}) \xrightarrow{[213,233]} (W,\{u_2\})$, $(E,\{u_2,u_3\}) \xrightarrow{[213,233]} (W,\{u_2\})\}$ would be considered. Further, we are going to describe this concept in detail.

### 4.2.3 Mobility Profile

Having introduced the concepts of Trajectory and Group-T-Pattern, we can now introduce the notion of *Mobility Profile*, as shown in Equation 4.3.

**Definition 4.6:** Given a maximum tolerance time threshold $\tau$ and a minimum support threshold $\sigma$, the mobility profile $PS^{Tu}_{\tau,\sigma}$ of a user $u$ is the set of frequent closed Group-T-patterns $\langle S,A,G\rangle$ contained in user's trajectories.

$$PS^{Tu}_{\tau,\sigma} = \{\langle S,A,G\rangle | \mathit{support}^{Tu}_{\tau,\sigma}\langle S,A,G\rangle \ge \sigma\}. \tag{4.3}$$

A mobility profile (hereafter referred simply as *profile* for conciseness) is a set of all *frequent* Group-T-Patterns contained in a set of trajectories from a user. It represents a digest of the user's social and spatio-temporal behavior. The frequency of occurrences of patterns in a set of trajectories can be quantified by the ratio of trajectories in which the pattern is contained, which is called the *support* value denoted by $\sigma$. The support function returns the number of trajectories in which a given pattern $\langle S, A, G\rangle$ is contained in relation to all user's trajectories from a trajectory set $T_u$.

Occasionally, many patterns made of subsets of groups from $\langle S, A, G\rangle$, $G = (g_{i_0}, ..., g_{i_n})$ satisfy the containment restriction, leading to many redundant patterns. Technically, if $\langle S, A, G\rangle$ is contained in the trajectory T, any $\langle S, A, G'\rangle$, $G' = (g'_{i_0}, ..., g'_{i_n})$ is also contained in T if for all $k$, $0 \leq k \leq n$, $g'_{i_k} \subseteq g_{i_k}$. In order to avoid finding redundant patterns, our method will output only closed Group-T-Patterns, i.e., by removing any user from any group from the sequence, support does not change, and by adding any user to the groups, support will be reduced. The conceptual idea is having a more concise set of Group-T-Patterns by keeping only the largest groups. This concept is derived from the idea of *closed swarm* from (LI et al., 2010) and helped the creation of a pruning rule for GMTP algorithm, showed in Chapter 5.

For simplification of notation, in the rest of the manuscript, we denote the profile $PS^{Tu}_{\tau,\sigma}$ simply as $PS^u$.

## 4.3 Mobility Profile Extraction

As we adopted Prefix-Span as the base of our method of extracting the profiles, both share many similarities. Prefix-Span was detailed in Section 3.2. In this section, we are going to explain the new prefix-projected sequential pattern that was developed, named GTMP (Group-T-Pattern Mining).

In the following we present an example of profile extraction from a trajectory dataset. Consider the trajectory dataset from the user $u_1$ illustrated in Figure 6. For simplicity of explanation, cluster sets in these trajectories have only one cluster each. The set of trajectories of the user to build his profile is $Tu = \{T_1, T_2, T_3\}$.

Figure 6: Example Trajectories $Tu$.

$$\begin{aligned}
T_1 &= (A, \{\{u_1, u_2, u_3\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_4\}\}) \xrightarrow{4} (C, \{\{u_1, u_2, u_3\}\}) \\
T_2 &= (A, \{\{u_1, u_2, u_9\}\}) \xrightarrow{4} (B, \{\{u_1, u_3, u_6\}\}) \xrightarrow{9} (C, \{\{u_1, u_3, u_4\}\}) \\
T_3 &= (A, \{\{u_1, u_2, u_5\}\}) \xrightarrow{2} (D, \{\{u_1, u_3, u_4\}\}) \xrightarrow{6} (C, \{\{u_1, u_2, u_4\}\})
\end{aligned}$$

Let us define $\tau = 2$ and $\sigma = 0.5$ as inputs, and $PS^{Tu}_{\tau,\sigma} = \emptyset$ as the output of the method.

The mining process starts by checking if checkpoint sequences $(s_0, g_0)$ of length [equals 1 are frequent. It considers a random checkpoint from $S_{DB}$, lets say $A$, and the group containing the user $u_1$ itself, thus forming the *checkpoint-group sequence* $(A, \{u_1\})$. This checkpoint-group sequence is contained in all the trajectories (have $support = 1$), therefore it is frequent, so it is added to $PS^{Tu}_{\tau,\sigma}$.

Finding Groups is a depth-first search task inspired in ObjectGrowth described in Section 3.4. The next step is to add a new user to $(A, u_1)$ and check if the new the checkpoint-group sequence is still frequent. In the example, if we add $u_2$ to the group, thus forming the checkpoint-group sequence $(A, \{u_1, u_2\})$, it is still frequent and still have $support = 1$. If we add any other user from $U_{DB}$, $\sigma$ will be reduced, so we conclude that $(A, \{u_1, u_2\})$ is a closed Group-T-Pattern and remove the originating sequence $(A, \{u_1\})$ from the output $PS^{Tu}_{\tau,\sigma}$, avoiding redundant checkpoint-group sequences. Therefore, we can define the Group-T-Pattern P1:

$$P_1 = (A, \{u_1, u_2\})$$

For each frequent checkpoint-group sequence $(s_0, g_0)$, starting with $(A, \{u_1, u_2\})$ in this example, a projection of the initial trajectory dataset $T^u$ can be created, denoted as $T^u|_{(A,\{u_1,u_2\})}$. A projection is a simplification of the data which (i) contains only the trajectories from $T^u$ where $(A, \{u_1, u_2\})$ is contained, that is, checkpoint is $A$ and $\{u_1, u_2\}$ is a subset of at least one of the clusters in the cluster set, (ii) contains only frequent checkpoints and groups, and (iii) on each trajectory, the first occurrence of $(A, \{u_1, u_2\})$ and all the sequence elements that precede it are removed. In this case, the single-element sequence $(A, \{u_1, u_2\})$ is called the prefix of $T^u|_{(A,\{u_1,u_2\})}$. The projected dataset is shown in Figure 7.

Figure 7: Projections of $Tu$ with the prefix $(A, \{u_1, u_2\})$: $T^u|_{(A,\{u_1,u_2\})}$

$$\begin{gathered}(B, \{u_1, u_3, u_4\}) \overset{4}{\rightarrow} (C, \{u_1, u_2, u_3\}) \\ (B, \{u_1, u_3, u_6\}) \overset{9}{\rightarrow} (C, \{u_1, u_3, u_4\}) \\ (D, \{u_1, u_3, u_4\}) \overset{6}{\rightarrow} (C, \{u_1, u_2, u_4\})\end{gathered}$$

Any pattern starting with $(A, \{u_1, u_2\})$ can be obtained by analyzing only $T^u|_{(A,\{u_1,u_2\})}$, which is smaller than $T^u$. Then, each new pair $(s_1, g_1)$ that is frequent in $T^u|_{(A,\{u_1,u_2\})}$, can be found by executing the same explained search. We can say $(B, \{u_1, u_3\})$, for example, will correspond to a frequent pattern $(A,\{u_1,u_2\}) \rightarrow (B,\{u_1,u_3\})$ in $T^u$, and a new, smaller projection $T^u|_{(A,\{u_1,u_2\})\rightarrow(B,\{u_1,u_3\})}$ can be recursively computed and used for finding longer patterns starting with $T^u|_{(A,\{u_1,u_2\})\rightarrow(B,\{u_1,u_3\})}$.

Figure 8: Projection $T^u|_{(A,\{u_1,u_2\})\rightarrow(B,\{u_1,u_3\})}$

$$(C,\{u_1,u_2,u_3\})$$
$$(C,\{u_1,u_3,u_4\})$$

The prefix $(A,\{u_1,u_2\}) \rightarrow (B,\{u_1,u_3\})$ is a frequent checkpoint-group sequence ($support = 0.66 > \sigma$), but it is still not a Group-T-Pattern. Outputting sequences with length > 1 to $PS^{Tu}_{\tau,\sigma} = \emptyset$ requires them to have typical *transition times*. The steps to calculate typical transition times are inspired in MiSTA, detailed in Section 3.3.

For each of the occurrences of $(A,\{u_1,u_2\}) \rightarrow (B,\{u_1,u_3\})$ in the dataset, we take their lower and upper bounds by adding $\pm\tau$ to the transition times and then find the intersection regions among them.

$$From\ T_1 : [1,5]$$

$$From\ T_2 : [2,6]$$

The lower limit of the intersection region is given by the highest lower boundary (2), and the higher limit is given by the lowest upper limit (5), so we have:

$$P_2 = (A,\{u_1,u_2\}) \xrightarrow{[2,5]} (B,\{u_1,u_3\})$$

, that is contained in $T_1$ and $T_2$. After keep running the method, we will also find:

$$P_3 = (B,\{u_1,u_3\})$$

, that is contained in $T_1$ and $T_2$.

$$P_4 = (A,\{u_1,u_2\}) \xrightarrow{[6,9]} (C,\{u_1,u_4\}),$$

that is contained in $T_2$ and $T_3$.

$$P_5 = (C,\{u_1,u_4\}),$$

that is contained in $T_2$ and $T_3$.

$$P_6 = (C, \{u_1, u_3\}),$$

that is contained in $T_1$ and $T_2$.

$$P_7 = (C, \{u_1, u_2\}),$$

that is contained in $T_1$ and $T_3$.

The user profile will be hence $PS^{Tu}_{\tau=2,\sigma=0.5} = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$. Notice that there is not any pattern with the checkpoint sequence $A \rightarrow B \rightarrow C$, neither $B \rightarrow C$, as there is not any transition time that match simultaneously $B \xrightarrow{4} C$ and $B \xrightarrow{9} C$ in $T_1$ and $T_2$, considering the maximum tolerance time threshold $\tau = 2$.

## 4.4 Similarity measures

After constructing mobility profiles based on users' historical data, we propose an anomaly detector that will identify users' trajectories that do not correspond to their profiles. The anomaly detection method can be classified as supervised since it incorporates a training step in which all existing trajectories will be pre-classified as normal. The similarity of a new trajectory and the user profile will include four different notions of similarity: *checkpoint* sequences, *transition times* between checkpoints, *groups* with whom the user was at each checkpoint, and *full similarity*, when checkpoints, transition times and groups match the profile.

Given that the trajectory and the Group-T-pattern both consist of a sequence of checkpoints, in order to calculate the similarity between these sequences, we argue that the more common checkpoints the sequences have, the more similar they are. In our first work (SILVA; SICHMAN, 2019), similarity calculation considered only the maximum sequences of the profiles, i.e., patterns that had no supersequences. However, this approach would not give good results if some patterns were contained in several different larger ones. For instance, consider the case when a user always arrives at and leaves the building at the same time with the same groups, but during the day, he travels different trajectories. If a new trajectory is considered, in which the user arrives and leaves according to the pattern, but during the day travels a trajectory that does not contain the maximum pattern, this new trajectory would not be evaluated as similar, although it could fit other smaller patterns from the profile. Hence, in the sequence, we will enhance our definition of similarity to deal with this issue, also described in the second work (SILVA; SICHMAN, 2022).

A new trajectory $T_N$ may contain many patterns from the profile. As we could show in Definition 4.5, a Group-T-Pattern must be a subsequence of the trajectory. In order to maximally capture the common information between sequences $T_N$ and $PS^u$, we assumed a new measure of sequence similarity, the number of *all common subsequences* (ACS) described in (WANG, 2007). The use of all possible common information between a new observed trajectory and all the profile trajectories is intuitively appealing and may provide a fuller picture of the sequences' similarity relationship. As the profile $PS^u$ is a set of sequences, we will use the concept of ACS as the set of common subsequences between the profile $PS^u$ and the trajectory $T_N$.

A trajectory $T = \langle S, A, CS \rangle$, as we have shown in Definition 4.3, is formed by three sequences, composed of checkpoints, annotations and cluster sets. However, the groups and annotations depend on the checkpoint sequence. Consequently, a new trajectory $T$ will have similarity with a profile $\langle S, A, G \rangle \in PS^u$ if they share any common checkpoint subsequence.

However, to propose a richer measure of similarity between trajectories and profiles, we also want to represent separately other similarity measures concerning groups and transition times. To this end, we consider the four separated containment definitions, derived from Definition 4.5.

**Definition 4.7:** A trajectory *checkpoint-contains* a pattern $\langle S, A, G \rangle \preceq_C T_N$ if the pattern is contained in the trajectory when ignoring groups and transition times, that is, meets only the requirements of rule 1 of Definition 4.5.

**Definition 4.8:** A trajectory *group-contains* a pattern $\langle S, A, G \rangle \preceq_G T_N$ if the pattern is contained in the trajectory when ignoring transition times, that is, meets only the requirements of rules 1 and 2 of Definition 4.5.

**Definition 4.9:** A trajectory *time-contains* a pattern $\langle S, A, G \rangle \preceq_T T_N$ if the pattern is contained in the trajectory when ignoring groups, that is, meets only the requirements of rules 1 and 3 of Definition 4.5.

**Definition 4.10:** A trajectory *fully-contains* a pattern $\langle S, A, G\rangle \preceq_F T_N$ if the pattern is contained in the trajectory according to Definition 4.5.

Lets call $AUS(PS^u)$ the set all unique subsequences of all patterns in the profile. If a sequence $\langle S', A', G'\rangle$ is a subsequence of more than one pattern in the profile, it will appear only once in $AUS(PS^u)$. For a sequence of length $|T_N|$, the number of its subsequences is $\sum_{i=1}^{|T_N|} \binom{i}{|T_N|} = 2^{|T_N|} - 1$, which does not include the empty sequence.

**Definition 4.11:** The similarity of checkpoints $simC(T_N, PS^u)$ between a new trajectory $T_N$ and the profile $PS^u$ is defined in Equation 4.4.

$$simC(T_N, PS^u) = \frac{|\{\langle S', A', G'\rangle \preceq_C T_N | \langle S', A', G'\rangle \in AUS(PS^u)\}|}{2^{|T_N|} - 1} \tag{4.4}$$

where $|T_N|$ is the length of the trajectory..

Intuitively, the more unique subsequences from the profile match those from the trajectory, the more the trajectory is similar to the profile. Notably, if a sequence from the profile is contained in the trajectory, all its subsequences will also be contained.

**Definition 4.12:** The similarity of groups $simG(T_N, PS^u)$ between a new trajectory $T_N$ and the pattern $\langle S, A, G\rangle$ is defined in Equation 4.5.

$$simG(T_N, PS^u) = \frac{|\{\langle S', A', G'\rangle \preceq_G T_N | \langle S', A', G'\rangle \in AUS(PS^u)\}|}{2^{|T_N|} - 1} \tag{4.5}$$

We can analogously define $simT$ and $simF$ using the same concept but in terms of time containment and full (group and time) containment.

**Definition 4.13:** The similarity of Transition Times $simT(T_N, PS^u)$ between a new trajectory $T_N$ and the user profile $PS^u$ is defined in Equation 4.6.

$$simT(T_N, PS^u) = \frac{|\{\langle S', A', G'\rangle \preceq_T T_N | \langle S', A', G'\rangle \in AUS(PS^u)\}|}{2^{|T_N|} - 1} \tag{4.6}$$

**Definition 4.14:** The full similarity $simF(T_N, PS^u)$ between a new trajectory $T_N$ and the user profile $PS^u$ is defined in Equation 4.7.

$$simF(T_N, PS^u) = \frac{|\{\langle S', A', G' \rangle \preceq_F T_N | \langle S', A', G' \rangle \in AUS(PS^u)\}|}{2^{|T_N|} - 1} \tag{4.7}$$

Notice that, if a subsequence of the pattern is entirely contained in the trajectory, it will also be time-contained, group-contained and group-and-time-contained in the trajectory as well.

Finally, the similarity of the new trajectory to the user's profile will be the average of the similarities $simC(T_N, PS^u)$, $simG(T_N, PS^u)$, $simT(T_N, PS^u)$ and $simF(T_N, PS^u)$, which we will write respectively $simC$, $simG$, $simT$ and $simF$, for short.

**Definition 4.15:** The similarity $sim(T_N, PS^u)$ between a new trajectory $T_N$ and the user profile $PS^u$ is defined in Equation 4.8.

$$sim(T_N, PS^u) = \frac{simC + simG + simT + simF}{4} \tag{4.8}$$

## 4.5 Detecting anomalous trajectories

Having defined a measure of similarity, we are able now to define whether a new trajectory $T_N$ may be considered a threat.

**Definition 4.16:** A new trajectory $T_N$ is considered a threat if its similarity with the user profile $PS^u$ is less than a certain minumum support threshold $\epsilon$, as depicted in Equation 4.9.

$$\text{threat}(T_N, PS^u) = \begin{cases} 1, & \text{if } sim(T_N, PS^u) < \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

## 4.6 Example

Let us consider the dataset $Tu$ from the example of Section 4.3, and a new trajectory $T_N$ to be compared to the user profile.

$$T_N = (A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_8\}\}) \xrightarrow{8} (C, \{\{u_1\}\})$$

This trajectory has 7 checkpoint-clusterset subsequences:

$$
\begin{aligned}
&(A, \{\{u_1, u_2, u_4\}\}),\\
&(B, \{\{u_1, u_3, u_8\}\}),\\
&(C, \{\{u_1\}\}),\\
&(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_8\}\}),\\
&(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{11} (C, \{\{u_1\}\}),\\
&(B, \{\{u_1, u_3, u_8\}\}) \xrightarrow{8} (C, \{\{u_1\}\}),\ and\\
&(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_8\}\}) \xrightarrow{8} (C, \{\{u_1\}\}).
\end{aligned}
$$

And the mobility profile $PS^{Tu}_{\tau=2,\sigma=0.5}$ is formed by:

$$
\begin{aligned}
P_1 &= (A, \{u_1, u_2\}),\\
P_2 &= ((A, \{u_1, u_2\}) \xrightarrow{[2,5]} (B, \{u_1, u_3\}),\\
P_3 &= ((B, \{u_1, u_3\}),\\
P_4 &= ((A, \{u_1, u_2\}) \xrightarrow{[6,9]} (C, \{u_1, u_4\}),\\
P_5 &= ((C, \{u_1, u_4\}),\\
P_6 &= ((C, \{u_1, u_3\}),\ and\\
P_7 &= ((C, \{u_1, u_2\}).
\end{aligned}
$$

By analyzing each one of the trajectory's subsequences, we notice that the subsequence $(A, \{\{u_1, u_2, u_4\}\})$ of the new trajectory $T_N$ fully-contains the pattern $P_1$, the subsequence $(B\{\{u_1, u_3, u_8\}\})$ fully-contains the pattern $P_3$, and the subsequence $(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_8\}\})$ fully-contains the pattern $P_2$. Therefore, 3 out of 7 subsequences of the trajectory contain one or more Group-T-Patterns from $PS^{Tu}_{\tau=2,\sigma=0.5}$, leading to $simF = 3/7$

Similarly, the subsequence $(A, \{\{u_1, u_2, u_4\}\})$ group-contains the pattern $P_1$, the subsequence $(B, \{\{u_1, u_3, u_8\}\})$ group-contains the pattern $P_3$, and the subsequence

$(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_8\}\})$ group-contains the pattern $P_2$. Hence, $simG = 3/7$.

Likewise, the subsequence $(A, \{\{u_1, u_2, u_4\}\})$ of the new trajectory $T_N$ time-contains the pattern $P_1$, the subsequence $B\{\{u_1, u_3, u_8\}\}$ time-contains the pattern $P_3$, the subsequence $C\{\{u_1\}\}$ time-contains the pattern $P_5$, and the subsequence $(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_8\}\})$ time-contains the pattern $P_2$, thus $simT = 4/7$.

Finally, the subsequence $(A, \{\{u_1, u_2, u_4\}\})$ checkpoint-contains the pattern $P_1$, the subsequence $(B, \{\{u_1, u_3, u_8\}\})$ checkpoint-contains the pattern $P_3$, the subsequence $(C, \{\{u_1\}\}$ checkpoint-contains the pattern $P_5$, the subsequence $(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{3} (B, \{\{u_1, u_3, u_8\}\})$ checkpoint-contains the pattern $P_2$, andhe subsequence $(A, \{\{u_1, u_2, u_4\}\}) \xrightarrow{11} (C, \{\{u_1\}\})$ checkpoint-contains the pattern $P_4$. Accordingly, $simC = 5/7$.

There is no pattern contained by neither the subsequence $(B, \{\{u_1, u_3, u_8\}\}) \xrightarrow{8} (C, \{\{u_1, u_4\}\})$ nor the one identical to $T_N$. Thus, we can assume $simC = 5/7$, $simT = 4/7$, $simG = 3/7$ and , $simF = 3/7$ that would give us $sim(T_N, PS^u) = (5/7 + 4/7 + 3/7 + 3/7)/4 = 0.536$.

If this trajectory is anomalous or not, it depends on the value of $\epsilon$. If $\epsilon$ is set to 0.6, this results possibly indicates an anomalous trajectory $(0.536 < 0.6)$.

In the sequence, we detail the algorithms that implement this model.

# 5 GTPM ALGORITHM

This chapter details the algorithms that implement the model described in Chapter 4. The algorithm that we created to extract mobility profiles, named GTPM, is presented in Section 5.1. Its first part is essentially based in MiSTA (GIANNOTTI; NANNI; PEDRESCHI, 2006), with modifications to support the concept of *groups*, while the second part is inspired by PrefixSpan (HAN et al., 2001), to which we added a concept of group, similar as the concept of swarm proposed by the ObjectGrowth algorithm (LI et al., 2010), thus creating a new algorithm named ProjectionGrowth, presented in Algorithm 2.

We also developed an algorithm to calculate similarity between a new trajectory and a profile based in the concept of All Common Subsequences (WANG, 2007), that is described in Section 5.4

## 5.1 Main algorithm

The main algorithm (Algorithm 1) is composed of two complementary parts: searching checkpoints-group *frequent sequences* (handled by Steps 16–20), and handling *temporal annotations* (described in Steps 5–15). Each of these two parts are respectively detailed in Sections 5.2 and 5.3. However, we have developed a *ProjectionGrowth* procedure, which incorporates our notion of groups, detailed in the sequence.

ProjectionGrowth in line is a central part of GTPM algorithm. It is responsible for outputting frequent checkpoint-group pairs from a projection, thus creating a list of prefixes 1 element longer.

**Algorithm 1:** Group Trajectory Pattern Mining (GTPM).

```
Input: $Tu$: a set of trajectories from the user $u$, $\sigma$: minimum support threshold, $\tau$: maximum
       tolerance time threshold, $U_{DB}$: users in the database
Output: user profile $PS^{Tu}_{\tau,\sigma}$
$L \leftarrow 0$, $P_0 \leftarrow \{Tu \times \{\langle\rangle\}\}$, $s_{min} \leftarrow \sigma \times |Tu|$;
while $P_L \neq \emptyset$ do
    $P_{L+1} \leftarrow \emptyset$;
    foreach $P \in P_L$ do
        if $length(P.prefix) \geq 2$ then
            $A \leftarrow Extract\_annotation\_blocks(P)$;
            $D \leftarrow Compute\_density\_blocks(A)$;
            $D^* \leftarrow Coalesce\_density\_blocks(D)$;
            $P^* \leftarrow Annotation\text{-}Based\_prune(P, D^*)$;
            $PS^{Tu}_{\tau,\sigma} \leftarrow PS^{Tu}_{\tau,\sigma} \cup (P.prefix, D*)$;
        end
        else
            $P^* \leftarrow P$;
            $P.prefix.A \leftarrow \emptyset$ $PS^{Tu}_{\tau,\sigma} \leftarrow PS^{Tu}_{\tau,\sigma} \cup (P.prefix, D*)$;
        end
        foreach checkpoint $s \in P^*$ do
            $last \leftarrow 0$, $P' \leftarrow \{\}$, $P^L \leftarrow \{\}$, $G < -\{\}$;
            $P_{L+1} \leftarrow P_{L+1} \cup \{ProjectionGrowth(P, G, s, P', P^L, last, s_{min}, U_{DB})\}$;
            $L \leftarrow L + 1$;
        end
    end
end
return $PS^{Tu}_{\tau,\sigma}$
```

## 5.2 Mining frequent checkpoints-group sequences

The task of mining frequent checkpoint-group sequences is performed by ProjectionGrowth Algorithm. Given a projected, time-stamped sequence $S = \langle(s_1, t_1, cs_1), ..., (s_n, t_n, cs_1)\rangle$, obtained as projection of sequence $S_0$ with respect to the prefix $s^*$ (i.e., $S = S_0|_{s^*}$ ), we define a temporal checkpoint-group sequence for $S$ as the triple (S, A, CS), where $(S, G)$ will be called the checkpoint-group sequence, and $A = \langle(a_1, e_1), ..., (a_m, e_m)$ is the annotation sequence: each couple $(a_i, e_i)$ represents an occurrence of the prefix $s^*$ in the original sequence $S_0$, $a_i$ being the sequence of timestamps of such an occurrence, and $e_i$ being a pointer to the element of $S$ where the occurrence terminates, or the symbol $\emptyset$ if such element is not in $S$.

Notice that annotation sequences contain timestamps and not transition times, in contrast with Definition 4.3. When needed, the latter are simply computed on-the-fly from the former.

Steps 16–20 of Algorithm 1 generate all sub-projections of the current projection, that add the new checkpoint and all possible groups (swarms) as the last element of the prefix. Steps 5–15 handle *temporal annotations*.

**Algorithm 2:** $ProjectionGrowth(P, G, s, P', P^L, last, s_{min}, U_{DB})$

```
Input: $P$: projection to be extended, $G$: current group, $s$: checkpoint , $P'$: projection $P$ extended
       w.r.t. $s$ and $G$, $P^L$: list of projections to be outputted, $last$: index of the latest user added
       into $G$, $s_{min}$: minimum threshold parameter, $U_{DB}$: users in the database
Output: $P^L$: a list of Projections extended from $P$
/* Apriori Prunning */
1  if $|P'| < s_{min}$ then return;
/* Backward Prunning */
2  $backward\_prunning \leftarrow false$;
3  foreach $P^{prev} \in P^L$ do
4      if $P^{prev}.prefix \preceq P'.prefix \wedge |P^{prev}| = |P'|$ then
5          $backward\_prunning \leftarrow true$;;
6          break;
7      end
8  end
9  if $backward\_prunning$ then return;
10 $forward\_closure \leftarrow true$;
11 forall $u \in \{u_{last+1}, ..., u_{|U_{DB}|}\}$ do
12     $G' \leftarrow G \cup \{u\}$;
13     $P'' \leftarrow extend\_proj(P, s, G')$;
14     $last \leftarrow last + 1$;
       /* Forward Closure Checking */
15     if $P'' = P'$ then $forward\_closure \leftarrow false$;
       /* Recursive call */
16     $P^L \leftarrow ProjectionGrowth(P, G \leftarrow G', s, P' \leftarrow P'', P^L, last, s_{min}, U_{DB})$;
17 end
18 if forward_closure then $P^L \leftarrow P^L \cup \{P'\}$;
19 return $P^L$;
```

When visiting the node with with the checkpoint-group pair $P'.prefix = (s, G)$, Step 1 checks whether $(s, G)$ can pass the Apriori Pruning. Next, we check whether the current node can pass the Backward Pruning (Steps 2-8). This is made by checking if any of the previously mined prefixes $P^{prev}.prefix = (s^{prev}, G^{prev})$ contain the current prefix $P' = (s, G)$, denoted by $P^{prev}.prefix \preceq P'.prefix$, that is, $s^{prev} = s$ and $G \subseteq G^{prev}$, and checks simultaneously if the projected datasets have the same number of projections $|P^{prev}| = |P'|$.

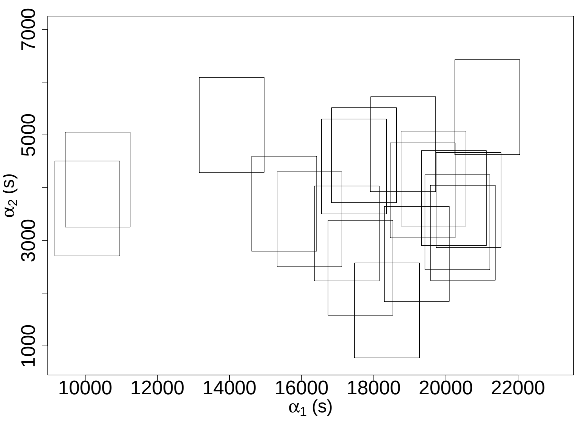
After both pruning, we will visit all the child nodes in the DFS order. Step 13 calls *extend_proj* algorithm, that returns all the projections that contains $(s, G')$. Finally, after visiting the subtree under the current node, if the node passes Forward Closure Checking we can output $P'$ as an extended projection.

**Algorithm 3:** *extend_proj*

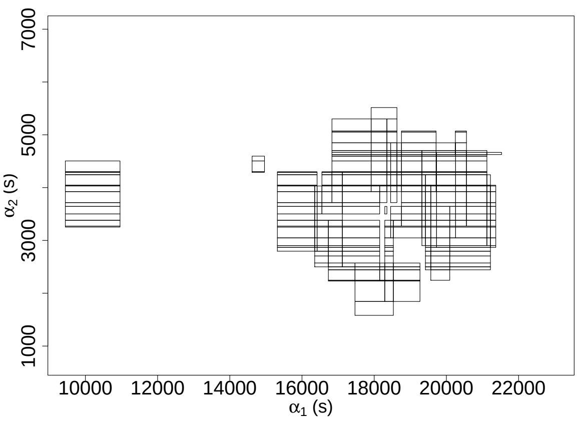
```
Input: A projection $P$, a checkpoint $s$, and a group of users $G$
Output: A list of Projections $P''$ w.r.t. $s$ and $G$
foreach $sequence\ T = (S, A, CS) \in P, s \in S, U \subset CS$ do
    $(S', CS') = (S, CS)|_{(s,U)}\ and\ A' = \langle\rangle$;
    foreach $annotation(a, e) \in A$ do
        foreach $(s, t, cs) \in (S, CS)\ s.t.\ i = s \ \wedge\ \exists c \in cs|U \subseteq c \ \wedge\ t > e$ do
            $A' = append(A', (append(a, t), \rightarrow t))$;
            $P' = P' \cup \{(S', A', CS')\}$;
        end
    end
end
return $P'$;
```

## 5.3 Handling temporal annotations

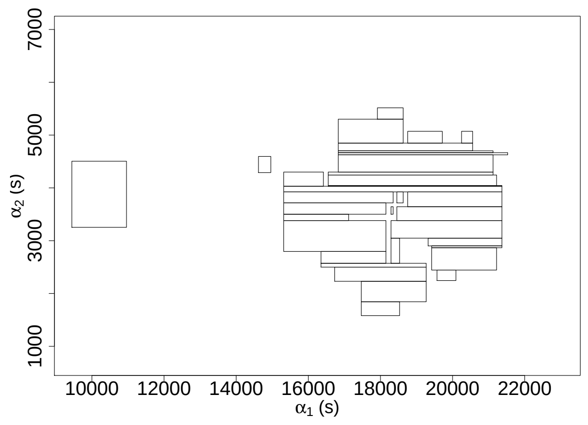
Steps 16–20 are originally described in MiSTA (GIANNOTTI; NANNI; PEDRESCHI, 2006) that we previously described in Section 3.3, and they handle temporal annotations. *Mine Profiles* algorithms process temporal annotation if a previously mined checkpoint-group sequence has length 2, obviously because you have a temporal annotation that represents the transition time between at least 2 checkpoints $(s_0, g_0) \xrightarrow{\alpha_1} (s_1, g_1)$. The whole process can be seen in Figure 9 for the prefix $(s_0, g_0) \xrightarrow{\alpha_1} (s_1, g_1) \xrightarrow{\alpha_2} (s_2, g_2)$ that yields 2-dimensional graphs for ease of understanding.



(a) Annotation Blocks



(b) Density Blocks



(c) Coalesced Density Blocks

Figure 9: Handling Temporal Annotations steps
Source: (SILVA; SICHMAN, 2022)

*Extracting Annotation Blocks* is the task of the procedure called in line 6 of Algorithm 1. Algorithm 4 first draw the influence areas for the transition times of the prefix, hyper-cubes with center $\overline{\alpha} = (\alpha_1, \alpha_2)$ and edge $2\tau$. If a trajectory has many occurrences of the same pattern, its influence areas will overlap, and this overlapped region would be incorrectly considered dense. That is the reason why their influence areas are divided into non-overlapping hyper-rectangles called *Annotation Blocks* that are illustrated in Figure 9a.

After then, we can find the dense regions where hyper-rectangles overlap (line 7 in *Compute Density Blocks*) that calls Algorithm 5). In order to do that, we first divide the annotation space in regions of homogeneous density by collecting the extreme coordinates of each annotation block along some dimension $d$, then split the space in correspondence of such values, and recursively re-apply the same process on the result for all the other dimensions. We then remove hyper-rectangles that are not dense enough and would result in patterns under the minimum support. The result can be seen in Figure 9b, where the $\sigma$ value was adjusted to result a minimum density equal to two.

Notice that this computing density blocks process can yield excessively many density blocks that cover the a large area. When we *Coalesece density blocks*, we merge these hyper-rectangles, resulting in a more concise set of patterns. The process can be seen in Figure 9c.

For coalescing density blocks we used a different approach than described in (GIANNOTTI; NANNI; PEDRESCHI, 2006). Whereas there is a randomness in the original algorithm, we preferred a simple approach of ordering the set of dense blocks $\mathcal{D}$ picking the first hyper-rectangle from the density blocks set $\mathcal{D}$, checking if there are neighbor blocks adjacent to it, and if so, merging blocks into a new hyper-rectangle. The result is shown in Algorithm 6.

The process's output is an (infinite) set of frequent annotations where any annotation having components within the hyper-rectangles is frequent.

**Algorithm 4:** Extract_annotation_blocks(P)

```
Input: A projection $P$
Output: A set of hyper-rectangles, representing the influence areas of each sequence in $P$
1  $\mathcal{A} \leftarrow \emptyset$;
2  foreach $(S, A, G) \in P$ do
3      $H \leftarrow \emptyset$;
4      foreach annotation $(a, e) \in A$ do
5          Derive annotation $\overline{\alpha}$ from time-stamps $a$;
6          $h \leftarrow$ hyper-cube with center $\overline{\alpha}$ and edge $2\tau$;
7          Merge $h$ with $H$;
8      end
9      Partition $H$ into a set of hyper-rectangles $H'$;
10     $\mathcal{A} \leftarrow \mathcal{A} \cup H'$;
11 end
12 return $\mathcal{A}$;
```

**Algorithm 5:** Compute_density_blocks($\mathcal{A}$)

```
Input: A set of hyper-rectangles in $\mathbb{R}^d$
Output: A set of hyper-rectangles $\mathcal{D}$ and their density.
$\mathcal{D} = \emptyset$;
Recursive_density($\mathcal{A}$,$d$,$\langle\rangle$, $\mathcal{D}$);
return $\mathcal{D}$;
Def Recursive_density($\mathcal{A}$,$d$,$\langle\rangle$, $\mathcal{D}$):
    $B \leftarrow \{x|[l_1,h_1] \times ... \times [l_n,h_n] \in \mathcal{A}, x \in \{l_d,h_d\}\}$;
    $\hat{B} \leftarrow sorted\_sequence(B)$
    for $i = 1; i < |\hat{B}|; i++$ do
        $\mathcal{A}_i \leftarrow \{[l_1,h_1] \times ... \times [l_n,h_n] \in \mathcal{A}|[l_d,h_d] \cap [\hat{B}_i, \hat{B}_{i+1}] \neq \emptyset$;
        if $|\mathcal{A}_i|$ then
            $\hat{H}' \leftarrow append((l_d,h_d),\hat{H})$;
            if $d = 1$ then
                $\hat{h} \leftarrow [l_1,h_1] \times ... \times [l_n,h_n]$ given that
                $\hat{H}' = \langle(l_1,h_1),...,(l_n,h)\rangle$ ;
                $\hat{h}.density \leftarrow |\mathcal{A}_i|$;
                $\mathcal{D} \leftarrow \mathcal{D} \cup \{\hat{h}\}$;
            end
            Recursive_density($\mathcal{A}_i$,$d-1$,$\hat{H}'$, $\mathcal{D}$);
        end
    end
    return $\mathcal{D}$;
```

**Algorithm 6:** Coalesce_density_blocks($\mathcal{D}$)

```
Input: A set of dense hyper-rectangles $\mathcal{D}$
Output: A sequence of hyper-rectangles, covering the same volume as $\mathcal{D}$ but yielding a better
        approximation series
$\hat{H} \leftarrow sorted\_sequence(\mathcal{D})$;
$\hat{H}' \leftarrow \emptyset$
for $i = 1; i < |\mathcal{H}| - 1; i++$ do
    $\hat{h}_1 \leftarrow \mathcal{H}_i$;
    for $j = i+1; j < |\mathcal{H}|; j++$ do
        $\hat{h}_2 \leftarrow \mathcal{H}_j$;
        if $adjacent(\hat{h}_1,\hat{h}_2)$ then
            $\hat{h}' \leftarrow merge(\hat{h}_1,\hat{h}_2)$;
            $\hat{H}' \leftarrow \hat{H}' \cup \hat{h}'$;
        end
    end
end
return $\hat{H}'$;
```

As implicitly shown in Algorithm 5 (lines 8 and 12 of the recursive procedure), the n-dimensional hyper-rectangles obtained by the algorithm can be written as the cartesian product of n 1-dimensional intervals: $h = [l_1,h_1] \times ... \times [l_n,h_n]$. Therefore, each output sequence $(s_n, g_n) \rightarrow ... \xrightarrow{\alpha_1} (s_n, g_n)$ with any of its dense hyper-rectangles is essentially ready for presentation to the user in a compact, written form as follows:

$$(s_n, g_n) \xrightarrow{[l_1,h_1]} ... \xrightarrow{[l_n,h_n]} (s_n, g_n)$$

## 5.4 Calculating similarity

The algorithm for calculating similarity is presented in Algorithm 7. The algorithm's input is a profile $PS^{Tu}_{\tau,\sigma}$ that we are going to write as $PS$ for simplicity of notation, and a trajectory $T$. The algorithm has a projection-based approach that starts seeking common subsequences of size 1 and is recursively called to seek more extensive sequences in a projected Profile.

**Algorithm 7:** Compute_similarity($PS$, $T$)

```
   Input:
   Output:
1  $r \leftarrow (simS \leftarrow 0,\ simT \leftarrow 0,\ simG \leftarrow 0, simF \leftarrow 0)$;
2  $r \leftarrow Similarity\_ACS(T, PS, r)$;
3  return $(r.simS + r.simG + r.simT + r.simF)/4$;
4  Def Similarity_ACS($T, PS, r$):
5      $found \leftarrow (simS \leftarrow 0, simT \leftarrow 0, simG \leftarrow 0, simF \leftarrow 0)$;
       /* Given that $T = \{(s^T_0, , cs^T_0), (s^T_1, \alpha^T_1, cs^T_1), ..., (s^T_{|T|}, \alpha^T_{|T|}, cs^T_{|T|})\}$ */
6      foreach $i = 1;i \leq |T|; i++$ do
7          foreach pattern $Pt \in PS$ do
               /* Given that $Pt = \{(s^{Pt}_0, [], g_0), (s^{Pt}_1, [l_1, h_1], g_1), ..., (s^{Pt}_{|Pt|}, [l_{|Pt|}, h_{|Pt|}], g_{|Pt|})\}$ */
8              foreach $j = 1;i \leq |Pt|; j++$ do
9                  $[l_j, h_j] \leftarrow Reduce([l_1, h_1], ..., [l_j, h_j])$;
10                 if $s^T_i = s^{Pt}_j$ then
11                     $found.simS \leftarrow 1$;
12                     if $l_j \leq \alpha_i \leq h_j$ then $found.simT \leftarrow 1$;
13                     if $\exists\ c \in cs_i \mid g_j \subseteq c$ then $found.simG \leftarrow 1$;
14                     if $found.simT = 1 \wedge found.simG = 1$ then $found.simF \leftarrow 1$;
15                     $Pt' \leftarrow Pt - \{(s^{Pt}_0, [], g_0), ..., (s^{Pt}_j, [l_j, h_j], g_j)\}$;
16                     $PS' \leftarrow PS \cup Pt'$;
17                 end
18             end
19         end
20         $r \leftarrow r + found$;
           /* Recursive call if any pattern is found */
21         if $found.simS = 1$ then
22             $T' \leftarrow T - \{(s^T_0, , cs^T_0), ..., (s^T_i, \alpha_i, cs^T_i)\}$;
23             $r \leftarrow r + Similarity\_ACS(T', PS', r)$;
24         end
25     end
26     return $r$;
```

The return object $r$ stores $simS$, $simT$, $simG$, and $simF$ similarities. In Step 9, the function $Reduce()$ sums the transition times from the beginning of the sequence to the checkpoint under analysis to have a proper transition time. Depending on the common subsequence found in the Profile, if checkpoints, transition time, groups or all, the pattern the object $found$ is filled with the respective flag $simS$, $simT$, $simG$, or $simF$, and a scalar sum between $r$ and $found$ is made, finally returning the count of all common subsequences found.

In the sequence, we present the algorithm's obtained results in two experiments using real-world datasets.

# 6 EXPERIMENTS

This chapter describes the experiments that aimed to evaluate our method's performance when applied on datasets from two real-world PACSs: (i) from a Multi-office Building and (ii) from a Hospital. All data were anonymized to protect the privacy of the users.

## 6.1 Description

The experimental framework is illustrated by Figure 10.

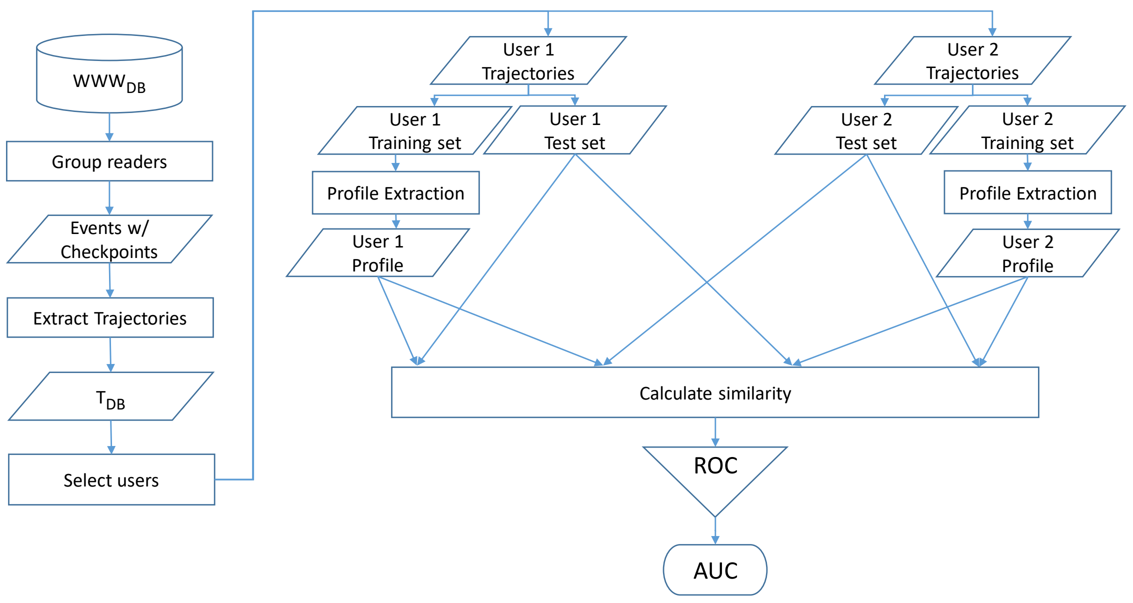


Figure 10: Experimental framework
Source: (SILVA; SICHMAN, 2022)

First, readers were grouped by their location and direction (entry or exit). Then, we clustered users who accessed the same checkpoint at almost the same time, according to the time window preset, and extracted their trajectories, a sequence of chronologically ordered and annotated checkpoints with clusters associated with them. Then, each user trajectory set was split by five (5-fold validation), so that one part was selected as the test set, and the other four were used as the training set. The choice $k = 5$

was made considering the literature (JAMES et al., 2013), that have shown empirically that this value of $k$ yields test error rate estimates that do not suffer from excessively high bias. In time series cross-validation, the corresponding training set generally consists only of observations that occurred before the observation that forms the test set so that no future observations should be used in profile construction. This is the best approach to conduct our tests due to our method's proposed application (detecting future anomalies).

After splitting each user's trajectory set, profiles were extracted for all of them, and each of the users' profiles (represented as "User 1" in Figure 10) was tested against trajectories from their own test set and against trajectories from the test set of one random user (represented by "User 2"). This approach enables balancing between positive and negatives instances of data (anomalous and regular trajectories, respectively), so we had approximately 50% instances of each class. Balancing datasets in machine learning experiments is necessary to avoid bias in the results. For example, if we test each user profile against all other users' trajectories, we would have a large number (larger than 99.9%) of positive instances, which means that if our system classifies all trajectories as anomalies, it will present 99.9% of accuracy. We used the results to build ROC curves and then to calculate the Area Under the Curve AUC. Results were used to construct the ROC curve and then to calculate AUC.

Some trajectories were not useful for building profiles because: (i) the user had only one trajectory in his training set, (ii) the trajectory had only one checkpoint, or (iii) the training set did not have any frequent pattern, so the number of profiles is smaller than the number of users. Each trajectory from the test set was tested against the user and a random user, so the number of tests is twice the number of trajectories in the test sets.

All the experiments were implemented using language R 3.4 and run on RStudio Version 1.1.463 on an Intel Core CPU i7-6700K 4.00GHz machine with 16GB of RAM running Microsoft Windows 10. The following libraries were used: (i) Matrix (ii) arules (iii) arulesSequences (iv) qualV (v) pROC (vi) foreach (vii) doParallel (viii) parallel (ix) doSNOW (x) doMPI (xi) R.utils.

## 6.2 Multi-office building (MOB) scenario

We empirically evaluated the proposed system's performance with a real-world dataset from a PACS of a smart building that hosts several companies and organizations. The PACS from which this data was collected has proximity cards, readers, and a database. Each user is assigned a proximity card that contains a unique serial number. A card is a passive Mifare ISO 1K in conformity with ISO/IEC 14443 transponder. Any user has to present their proximity card at the reader mounted next to the door to unlock the door. The database records the activity of requesting access to a doorway with a proximity card. The raw dataset has 84k access events occurring in 30 days, and more details are shown in Table 3.

The first step was finding trajectories. Each trajectory is a sequence of access events generated by one user within one day, from 12:00 AM to 11:59 PM, counting 20641 sequences from 1324 users. Parameters were set to $\sigma = 0.1$ and $\tau = 1200$, considered as the best values as we explain in more details further in this section. The resulting ROC curve is shown in Figure 11. This curve has Area Under the Curve (AUC) 0.79, and the best threshold may be defined by the user considering wanted FAR and FRR. In the figure, we highlighted the threshold 0.154 that maximizes the distance to the identity (diagonal) line, a method known as Youden's method (YOUDEN, 1950), and maximizes $FAR - FRR$. In this point of the ROC curve, $FAR = 0.685$ and $1 - FRR = 0.753$, hence $FRR = 0.247$.

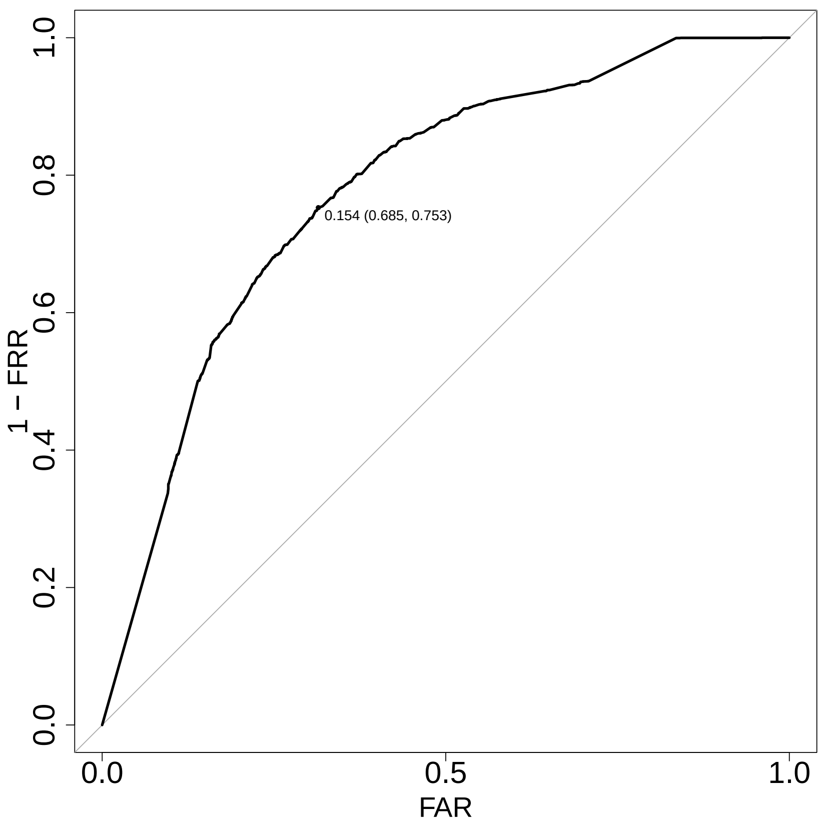


Figure 11: ROC Curve for Multi-Office Building
Source: (SILVA; SICHMAN, 2022)

## 6.3 Hospital (HSP) scenario

We also evaluated our method with data from a PACS of a Hospital that uses the same type of cards as the Multi-Office Building. More details about the dataset are found in Table 3, and Figure 12 shows the resulting ROC curve. Different from the MOB, this Hospital works 24/7, and people work overnight. According to the local regulations, workers have to rest at least 12 hours between work journeys, so we considered a new trajectory only if events intervals were larger than this gap. Data were collected from March $1^{st}$ to March $31^{st}$ of 2020.

This ROC has AUC 0.90, which indicates that our method applied to the hospital performs better than when applied to the multi-office building. We can reasonably explain that by observing that the hospital has more checkpoints than the MOB, so the sequences are more extended, making trajectories more distinguishable.

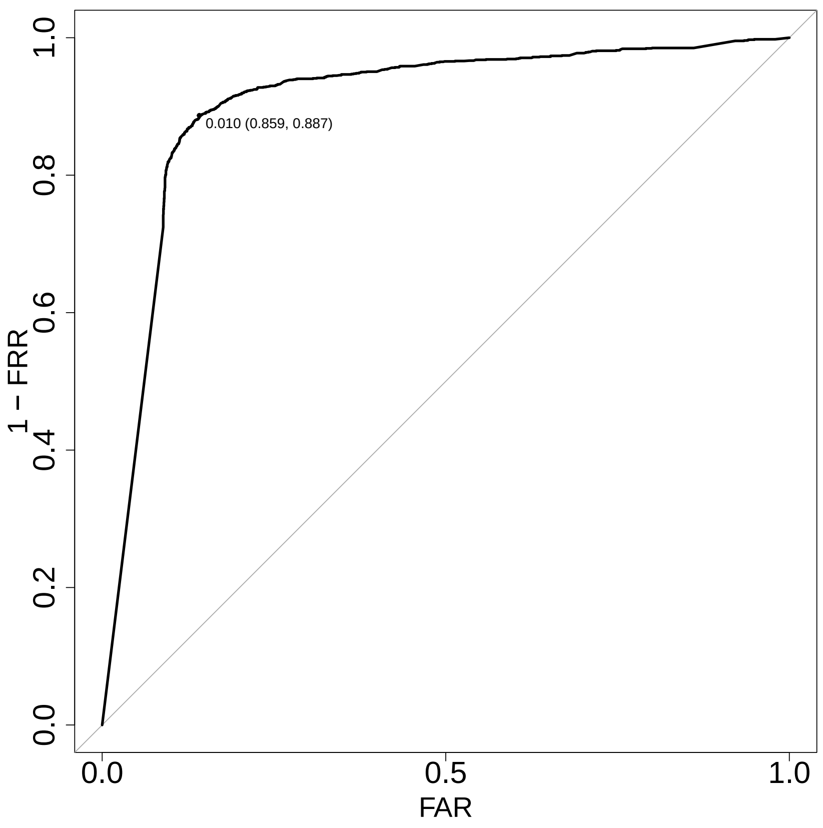


Figure 12: ROC Curve for Hospital
Source: (SILVA; SICHMAN, 2022)

Table 3: Multi-Office Building (MOB) vs. Hospital (HOSP) datasets.

| | **MOB** | **HOSP** |
|---|---|---|
| Users | 1324 | 3307 |
| Checkpoints | 15 | 207 |
| Events | 83988 | 358822 |
| Trajectories | 20641 | 23271 |
| Avg. Trajectory length | 4.1 | 15.41 |
| Profiles | 1150 | 928 |
| $t\_window$ | 30 | 30 |
| $\sigma$ | 10% | 10% |
| $\tau$ | 1200 | 60 |
| Tests | 8526 | 4803 |
| AUC | 0.79 | 0.90 |

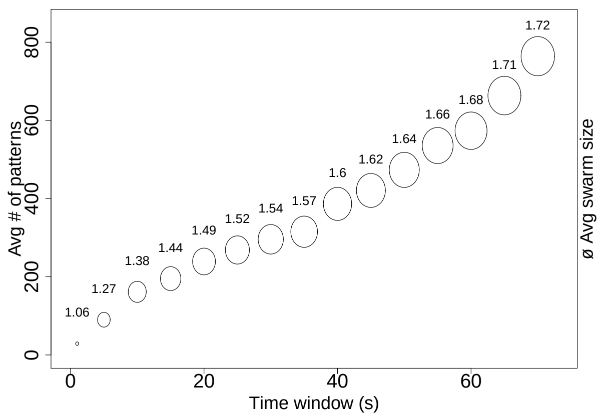
Source: (SILVA; SICHMAN, 2022)

## 6.4 Sensitivity analysis

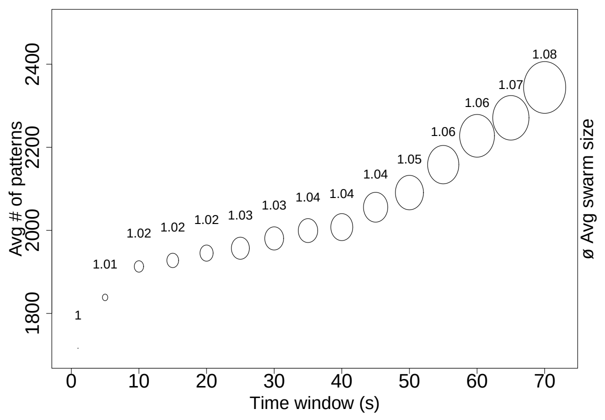
The following described experiments will evaluate our model's sensitivity to its three key parameters: time window used for clustering $t\_window$, minimum support threshold $\sigma$, and maximum time threshold $\tau$. In order to carry out the sensitivity analysis, we estimated a sample size of 300 profiles that leads to a 95% confidence interval with a 5% margin of error, and the tests were always performed with the same profiles. Sometimes we plotted results only for MOB because graphs would appear very alike for Hospital, thus avoid providing redundant information.

### 6.4.1 Maximum time window threshold $t\_window$

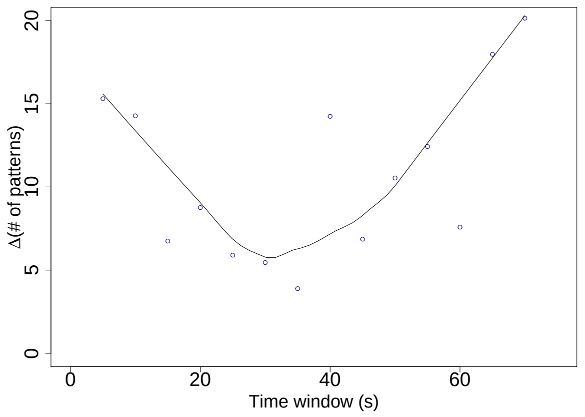
Parameter $t\_window$ defines that users that reached the same checkpoint within the time window are clustered together. This parameter influences how many groups will be mined and their sizes, and consequently, the number of patterns that form the profile. A too-small $t\_window$ will result only in groups of size 1 (the user himself) and patterns regardless of companions, while a too-big $t\_window$ may incorrectly result in many groups that are not effectively frequent companions, but were together by coincidence. In order to confirm this predicted behavior for $t\_window$, we fixed $\sigma = 0$ and $\tau = \infty$ (any pattern should appear in at least two trajectories to be considered frequent, regardless of its transition times) and tested many values for $t\_window$. The more groups are found, the more patterns the profiles will have. The results are presented in Figure 13.



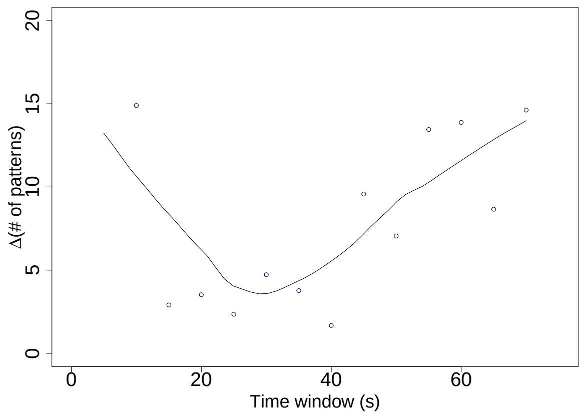
(a) MOB: Avg. number of patterns and avg. group sizes for different *t_window* (in seconds)



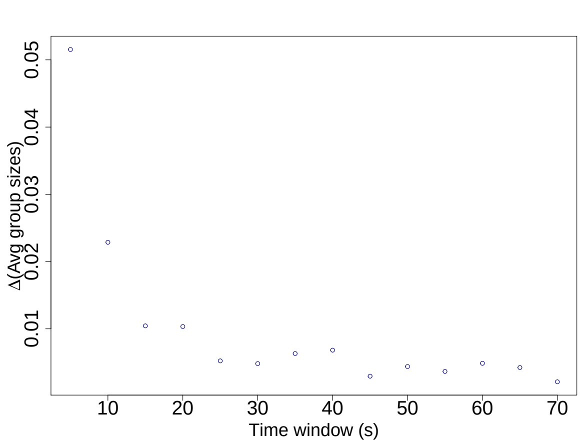
(b) HSP: Avg. number of patterns and avg. group sizes for different *t_window* (in seconds)



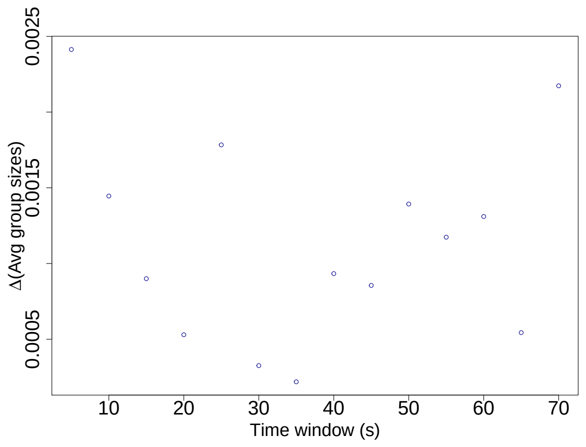
(c) MOB: Variation in the avg. number of patterns when increasing *t_window* (per second)



(d) HSP: Variation in the avg. number of patterns when increasing *t_window* (per second)



(e) MOB: Variation in the average group size when increasing *t_window* (per second)



(f) HSP: Variation in the average group size when increasing *t_window* (per second)

Figure 13: Impact of *t_window* in patterns extraction
Source: (SILVA; SICHMAN, 2022)

Figures 13a and 13b plot the average number of patterns for many *t_window*, where the disc diameter represents the average group size. We can see that, for $t\_window \approx 30s$ in the MOB experiment, the number of patterns and group sizes almost do not change. Figure 13c and Figure 13d show the variation in the number of patterns while Figure 13e and Figure 13f show the variation of average group size. We assume that this inflection point in Figures 13c and 13d reveal the optimum $t\_window$ for each experiment.

### 6.4.2 Minimum support threshold $\sigma$

The choice of the minimum support threshold $\sigma$ affects the performance of the anomaly detector, thus the resulting score AUC. For instance, if we consider $\sigma$ as 100%, a new pattern would only be derived if it has appeared in all the trajectories. Clearly, restrictively big threshold values of $\sigma$ ignore important patterns that could distinguish users. In Figures 14 and 15, we first explored this behavior of $\sigma$ by ignoring the $\tau$ value. As we expected, we can see that with $\sigma \approx 100\%$, the overall performance is close to 0.6, slightly better than a random detector that would score 0.5. In sequence, we explored in detail the behavior of $\sigma$ for many different $\tau$ values, as presented in Section 6.4.3. Since we considered that a minimum frequent pattern is contained in at least two trajectories, the fact that the results are almost the same in MOB for the $\sigma$ interval between 0 and 10% is justified by the size of our MOB dataset, which contains data from only one month and the average number of trajectories $\approx$ 20. For HSP scenario, when ignoring $\tau$, the best performance is achieved when $\sigma$ is between 20% and 50%. The order of magnitude that we have obtained for $\sigma$ is similar on related work (CHEN; PANG; XUE, 2014) (GIANNOTTI et al., 2007).

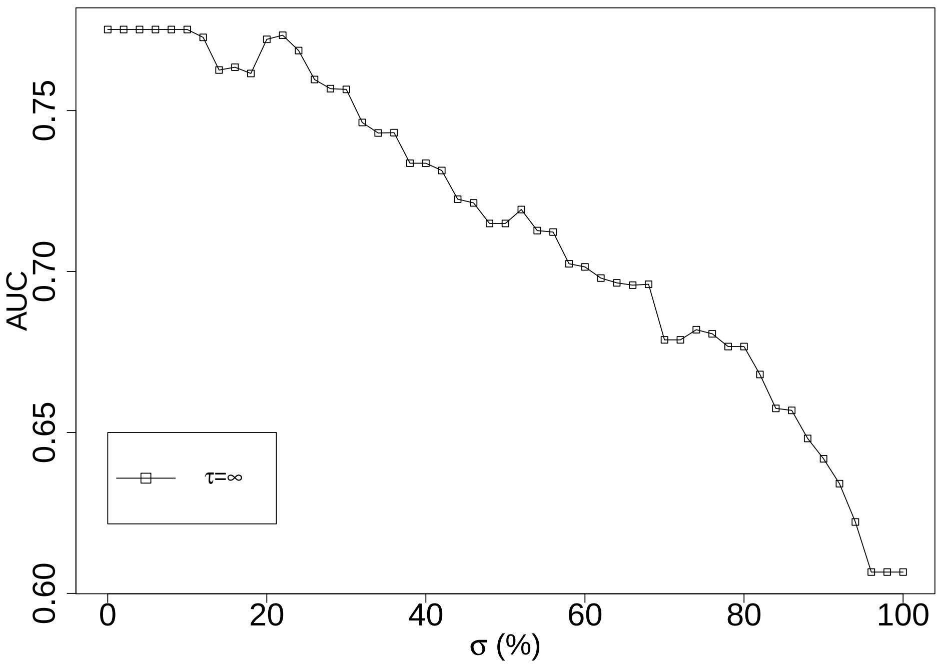


Figure 14: AUC for different $\sigma$ and ignoring $\tau$ for MOB
Source: (SILVA; SICHMAN, 2022)

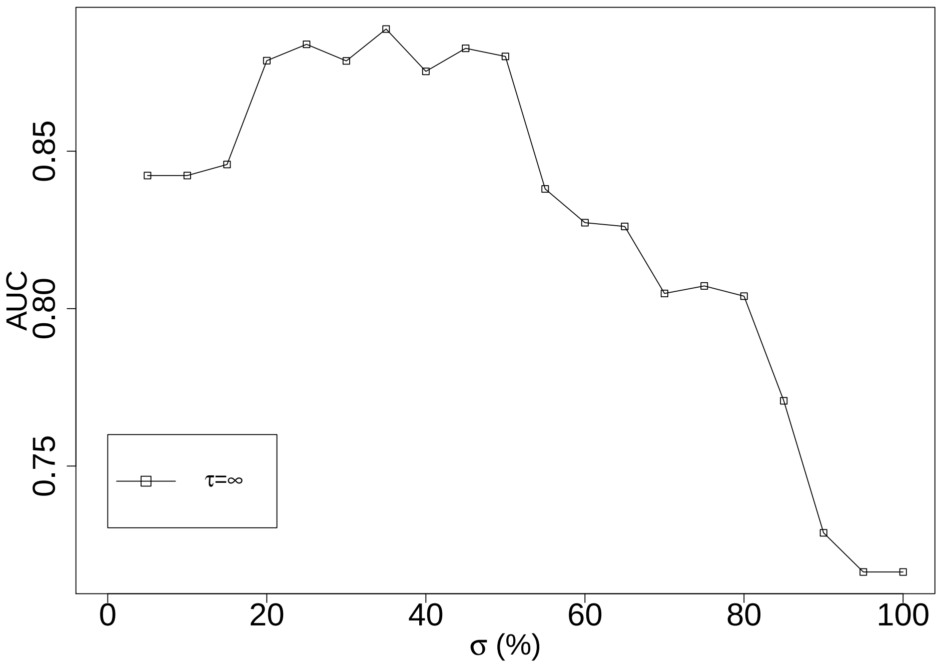


Figure 15: AUC for different $\sigma$ and ignoring $\tau$ for HSP

### 6.4.3 Maximum tolerance time threshold $\tau$

The maximum tolerance time threashold $\tau$ is a key concept of the model. Low values of $\tau$ will produce a very few number of patterns. Figure 16 shows annotation blocks for $\tau = 100$ from 2-dimensional patterns of an example dataset for the sequence $A \xrightarrow{\alpha_1} B \xrightarrow{\alpha_2} C$. If the minimum support threshold is set to $\sigma = 0.1$, at least 2 annotation blocks (squares centered in the transition times from the original dataset) need to cover the same region, i.e., they need to overlap. It is possible to see that none of the annotation blocks overlap. By increasing the value of $\tau$ to 300, some blocks start to overlap (Figure 17a), so they will produce blocks with sufficiently high density (Figure 17c) that can be coalesced (Figure 17e). The whole process outcome is a set of Group T-patterns in the form $A \xrightarrow{[l_1,h_1]} B \xrightarrow{[l_2,h_2]} C$, where $l_1$, $h_1$, $l_2$ and $h_2$ are given by the edges of the coalesced density blocks. If we increase the maximum time threashold even more to $\tau = 600$, we will see much more annotation blocks, density blocks and coalesced density blocks, hence more patterns that cover larger areas.

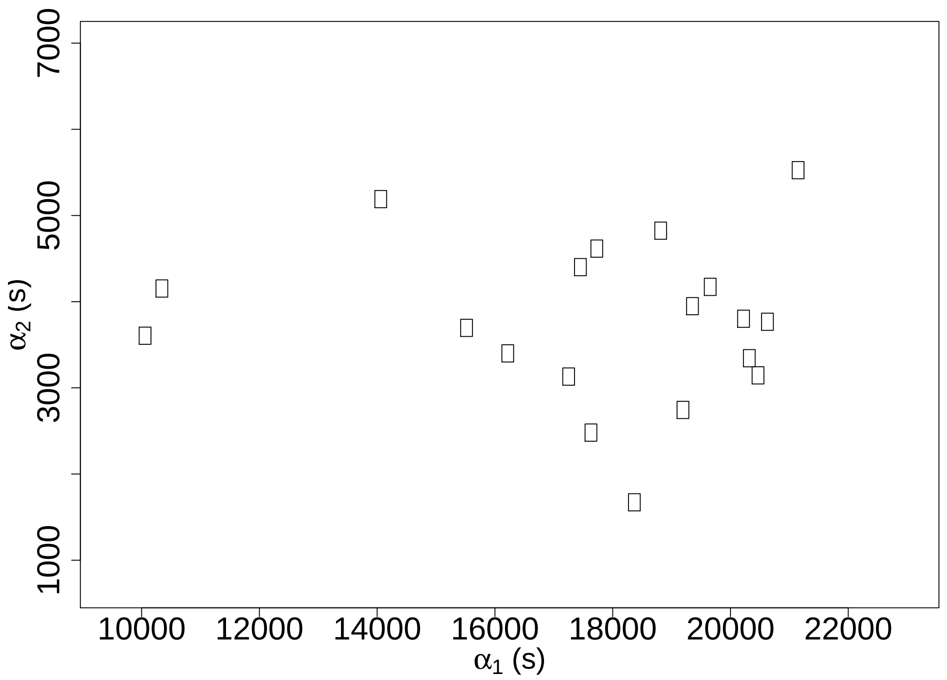
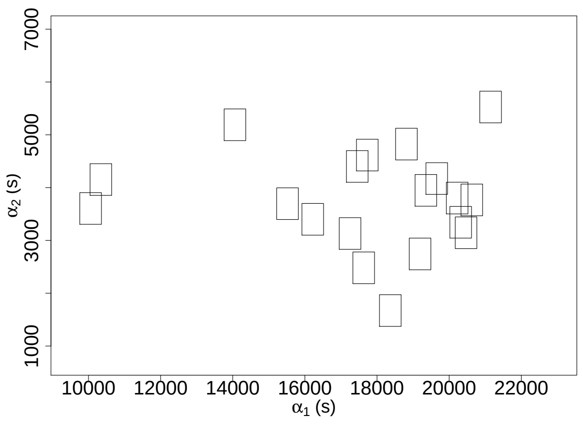
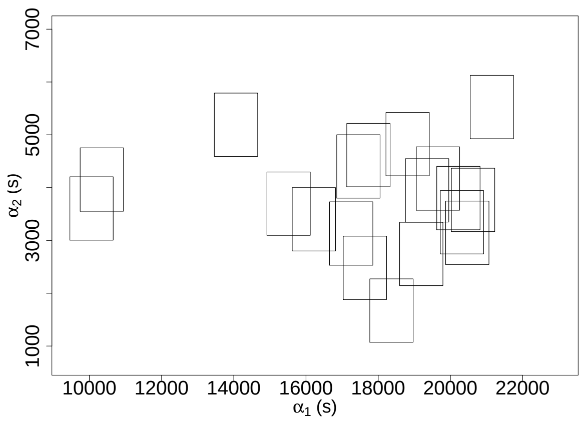


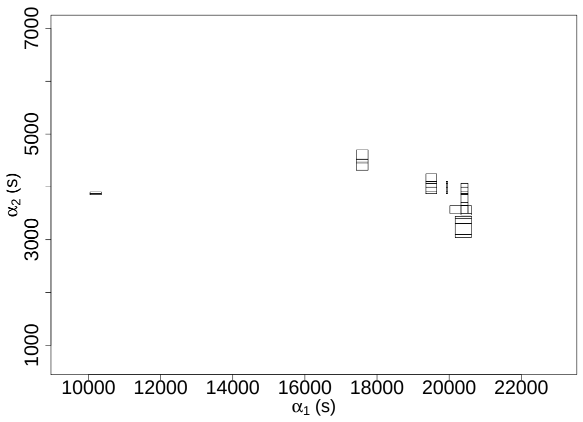
Figure 16: Annotation Blocks for $\tau = 100$



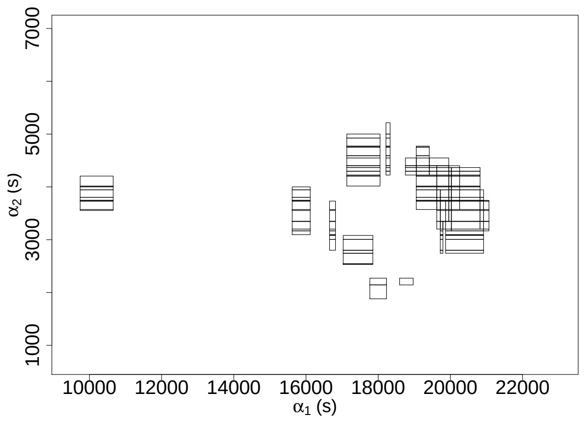
(a) Annotation Block for $\tau = 300$



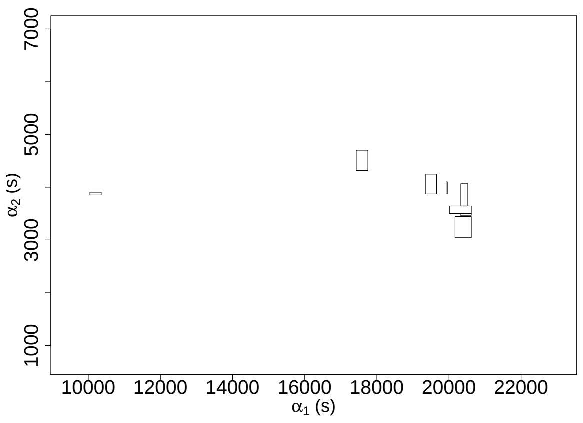
(b) Annotation Block for $\tau = 600$



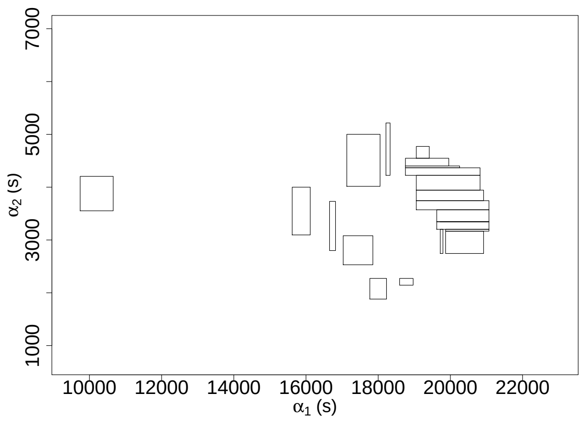
(c) Density Blocks for $\tau = 300$



(d) Density Blocks for $\tau = 600$



(e) Coalesced Density Blocks for $\tau = 300$

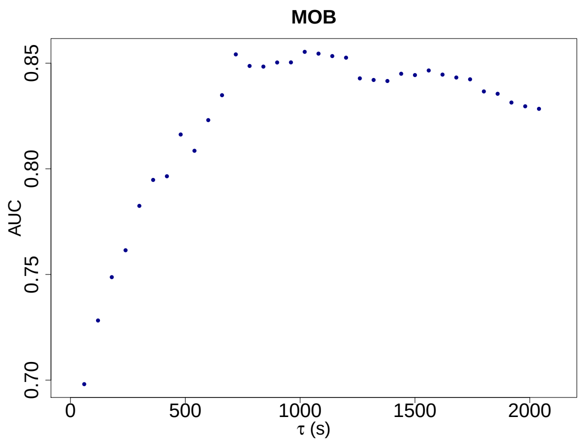


(f) Coalesced Density Blocks for $\tau = 600$

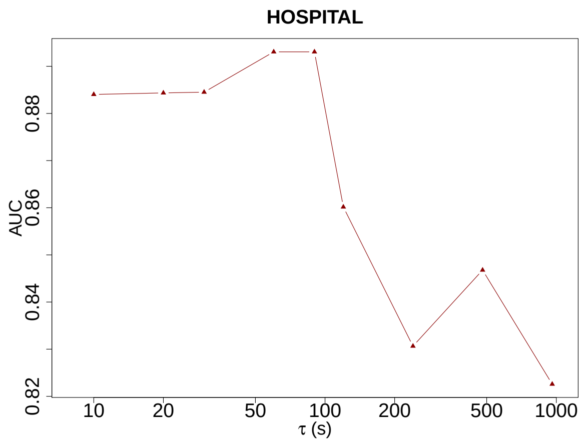
Figure 17: Impact of the maximum time threshold $\tau$ in patterns extraction

#### 6.4.3.1 Effects of $\tau$ parameter

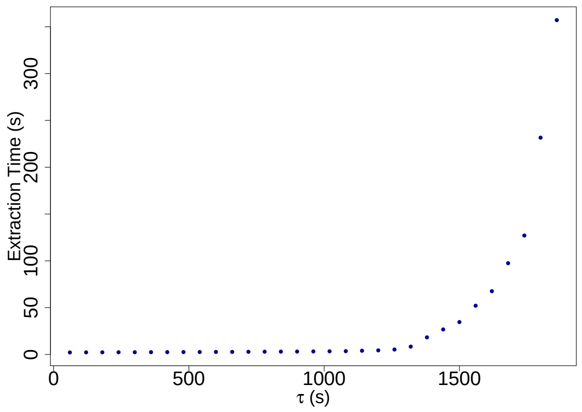
This influence of the maximum time threshold $\tau$ in the number of patterns will influence the overall performance of the detector. We conducted a sensitivity analysis of the system with respect to $\tau$ for the Multi-Office Building environment and saw how it influences in Area Under the ROC Curve of the whole system. As we can see in Figure 18, for both scenarios, raising $\tau$ improves the performance of the anomaly detector: the larger $\tau$ is, the more patterns are found, and more accurate the system is in terms of AUC. In a further analysis, we demonstrate that at some point, increasing $\tau$ does not improve the system's performance because all relevant patterns have already been discovered. If we increase $\tau$ even more, the detector's overall performance decreases as the tolerance is so big that the system cannot accurately distinguish groups from users that were coincidentally together. Figure 22a shows this behavior, where we fixed $\sigma = 10\%$ and variate $\tau$ in steps of 60 seconds until AUC starts to decrease. On the other hand, Figure 18a shows that as we increase $\tau$, profile extraction times increase exponentially, so it is better to set $\tau$ as small as possible.



(a) $\tau$ vs. AUC for MOB



(b) $\tau$ (logarithmic scale) vs. AUC for HSP



(c) $\tau$ vs. avarage Extraction Time for MOB

Figure 18: Sensitivity to $\tau$
Source: (SILVA; SICHMAN, 2022)

For the hospital environment, the best $\tau$ value is 60 seconds, as seen in Figure 18b. As this environment has more checkpoints and longer trajectories, transition times between checkpoints are shorter, and, naturally, time tolerance should be shorter. We also can see a peak in the curve in $\tau = 500s$. It means that, in some situations, there are new groups discovered when using a larger maximum time threshold (that is still under the maximum performance in 50s). The order of magnitude of transition time in some routes are larger than others.

By observing this behavior, we also conducted a sensitivity analysis with the change in the maximum threshold $\tau$ definition to be time-variant, and this analysis is presented next.

In order to graphically illustrate this analysis, we highlight the behavior of the AUC vs. $\sigma$ for the MOB scenario, where the score increases until $\tau = 1200$: the larger $\tau$ is, the more patterns are found, and more accurate the system is in terms of AUC. After, if we increase $\tau$ even more, AUC starts to decrease, until a threshold defined by $\tau = \infty$ (ignored) is reached, as we can see in Figure 19.

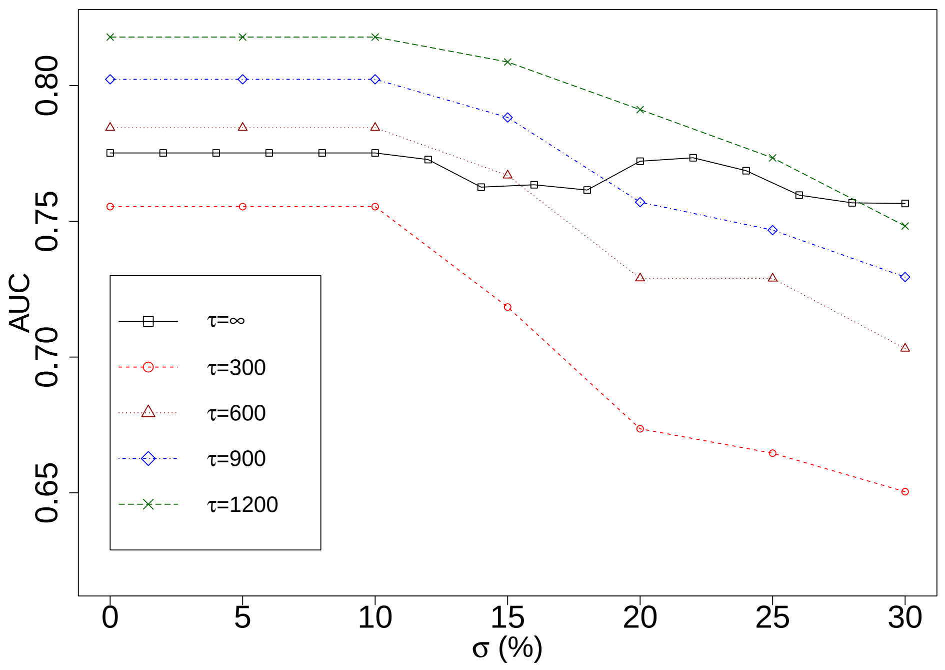


Figure 19: AUC for different $\sigma$ and $\tau$
Source: (SILVA; SICHMAN, 2022)

### 6.4.3.2 Constant $\tau$ vs. relative $\tau$

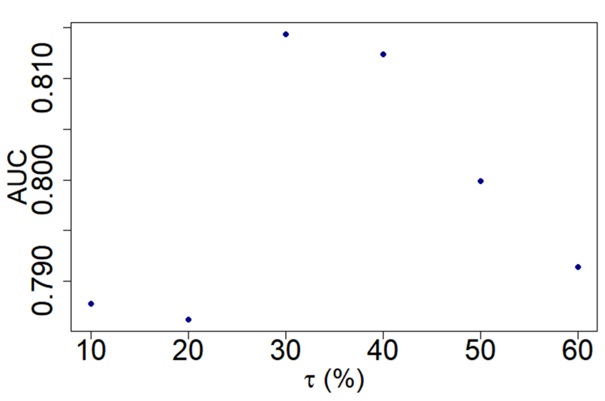
We also experimented a relative value of $\tau$ instead of a constant $\tau$, so the maximum tolerance time was not fixed, but a percent of each transition time from the trajectories from the test dataset. In order to do that, we modified the definition of containment (Definition 4.5) in a way that the maximum time threshold given by a relative parameter $\tau$

**Definition 6.1:** Given a maximum time threshold $\tau$, a Group T-Pattern $\langle S, A, G\rangle = (s_0, g_0) \xrightarrow{\alpha_1} (s_1, g_1) \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} (s_n, g_n)$ is fully contained in the trajectory $T = (s'_0, cs_0) \xrightarrow{\alpha'_1} (s'_1, cs_1) \xrightarrow{\alpha'_2} ... \xrightarrow{\alpha'_m} (s'_m, cs_m)$ (denoted by $\langle S, A, G\rangle \leq_F T$ ) if and only if exists a sequence of integers $0 \leq i_0 < ... < i_n \leq m$ such that:

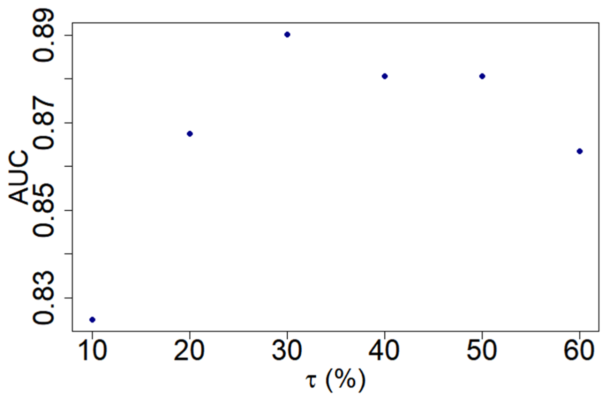
1. $\forall_{0\leq k\leq n}, s_k = s'_{i_k}$: the sequence of checkpoints from the pattern is a subsequence of those from the trajectory, i.e, the trajectory is a supersequence of the pattern;
2. $\forall_{0\leq k\leq n}, \exists\ c \in cs_{i_k} | g_k \subseteq c$: for all the groups of the pattern, there is at least one cluster $c$ from the corresponding cluster set $cs_{i_k}$ from the trajectory containing all users of the group;
3. **$\forall_{1\leq k\leq n}, |\alpha_k - \alpha_{*k}| \leq \tau \times \alpha_{*k}$, where $\alpha_{*k} = \sum_{i_{k-1}<j<i_k} \alpha'_j$: transition times differences are not greater than a maximum time threshold calculated by multiplying the trajectory transition time $\alpha'_k$ by $\tau$.**

Accordingly, as Definition 4.9 of time similarity relies on Definition 4.5, this first is also changed.

The results in Figure 20b show that there are not significant differences in the maximum detector's performance for the Hospital environment when compared to Figure 18b (approximately 0.89). For the MOB environment, the overall performance is lower than when using the original $\tau$. We assume that it happens because MOB is much more controlled regarding schedules, so people hardly change the arrival and attendance times, while for HSP environment, workers usually have different and changing work shifts, and the arrival time and transition times (intervals for lunch, etc.) may change according to predefined schedules.



(a) $\tau$ (%) vs. AUC for MOB



(b) $\tau$ (%) vs. AUC for Hospital

Figure 20: Sensitivity to $\tau$ variated in percent

### 6.4.4 Relevance of the similarity measures

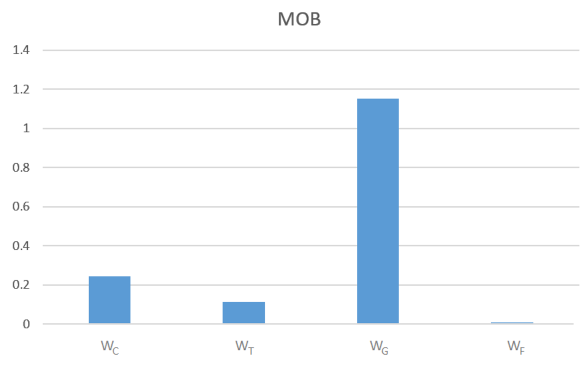
One approach for feature importance analysis is Linear Regression approach. It assigns the score of each feature based on their importance to predict the output. More the features will be responsible to predict the output more will be their score.

We conducted a linear regression with the feature under analysis ($simS$, $simT$, $simG$, and $simF$) an the results are presented in Figure 21. In order find the features, we assigned weight coefficients to the equation 4.8 thus having the equation 6.1, where $w$ are the coefficients.

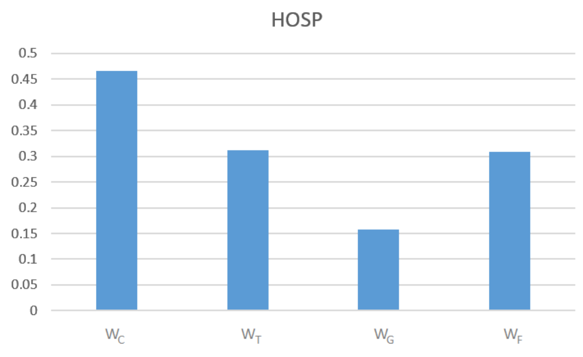
$$sim(T_N, PS^u) = \frac{w_C \cdot simC + w_G \cdot simG + w_T \cdot simT + w_F \cdot simF}{4} \tag{6.1}$$

We then fed the dataset built with the process described in Figure 10 to a linear model, with the targeted value of prediction $sim(T_N, PS^u)$ being 1 when users' profiles are compared to their own trajectories and equals 0 when compared to someone else's. We then trained the model and made the prediction to get the values of the coefficients.

These values of coefficients were tested as weights for the similarities and the AUC score were 0.8 for MOB and 0.92 for HSP.



(a) $\tau$ (%) vs. AUC for MOB



(b) $\tau$ (%) vs. AUC for Hospital

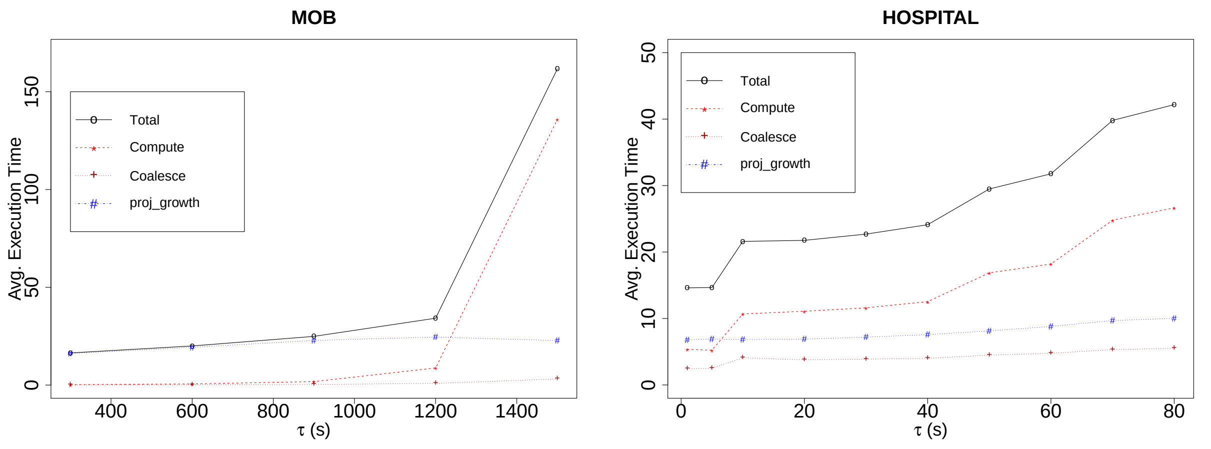
Figure 21: Relevance of similarity measures when $\tau$ variates in percent

It is possible to see that $simG$ is the most relevant feature for the MOB environment but is not so distinctive for HSP. We hypothesize that this occurs because there are fewer social bonds in this kind of environment.

## 6.5 Comparison with previous methods

As our approach is an extension of MiSTA (GIANNOTTI et al., 2007), we are driven to compare the results of our algorithm with it. The original MiSTA does not address the problem of the context where the trajectories occurred, i.e., the *companions*. Adding group information to the model enable the use of a new feature that can be used for a broad variety of applications. For example, by adding group information to the task of impersonation in access control, we capture if the impersonating user is alone and not with her regular companions. This situation would be ignored if only MiSTA was considered.

This new approach brings also differences to performance. The main difference is that, while MiSTA seeks only for sequences of checkpoints and transition times, the addition of groups in our method will lead to many more patterns, as each of the checkpoints unfolds different tuples considering the user in the same checkpoint but with different groups. Besides that, when the sequences are big, the time taken to compute density blocks is significantly higher than to the other steps, making the time take for group extraction task by *ProjectionGrowth* algorithm (Algorithm 2) less relevant. Compute density blocks task has complexity $O(n^d)$, exponential to the length of the pattern $(d)$ in the worst case; therefore, as raising $\tau$ turns the model more tolerant and makes it discover more and more lengthy patterns, the total extraction time raises exponentially as well. The Hospital environment has longer trajectories, and even for small $\tau$, computing density blocks is the most significant tasks in terms of execution time.



(a) Average execution time vs. $\tau$ for MOB (b) Average execution time vs. $\tau$ for Hospital

Figure 22: Relative performance vs. MiSTA: ProjectionGrowth performance
Source: (SILVA; SICHMAN, 2022)

We also explored how would MiSTA and the PrefixSpan perform in comparison to the proposed method. For doing that, we have implemented both algorithms, which the main differences between them are that:

1. PrefixSpan profiles have only for sequences of checkpoints;
2. MiSTA ones have sequences of checkpoints and transition times; and
3. our method GTPM is MiSTA enriched with group information, i.e., considers sequences of checkpoints, transition times and groups.

When the feature does not exist, similarities $simT$ or $simG$ are set to zero, so AUC shows how well ranked are the trajectories in terms of $simC$ only for PrefixSpan or $simC + simT$ for MiSTA. Results for many $\sigma$ and $\tau = 1200$ can be seen in Figure 23. We can notice that our method performs better for $\sigma$ values lower than 40%; this is an excellent results, since typical $\sigma$ adopted values are around 10%. By these results, we can argue that the profiles constructed with our method, incorporating group information, present more useful information for this particular task.

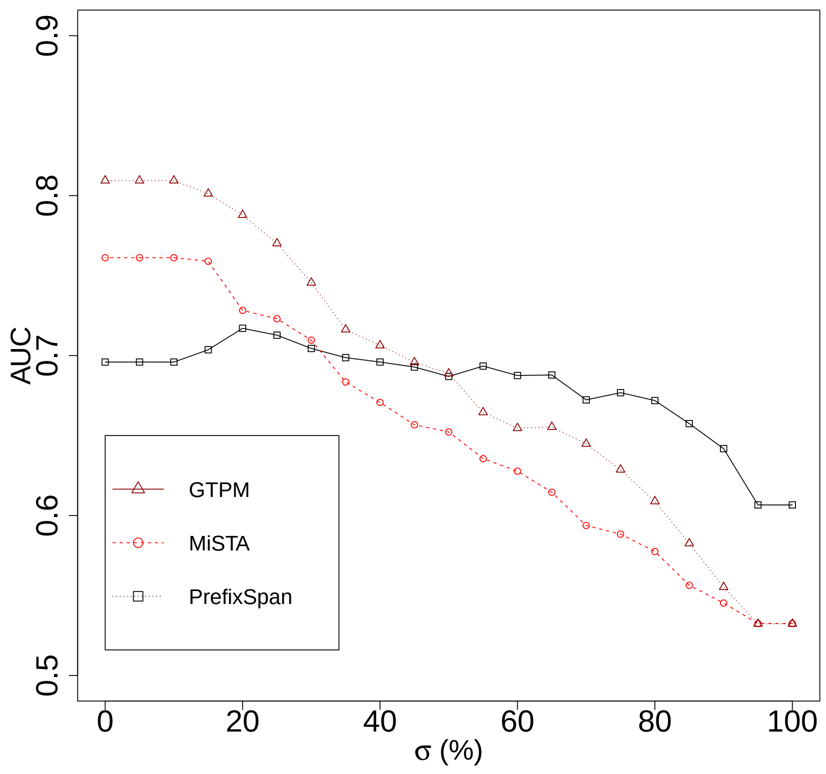


Figure 23: Comparison vs. MiSTA and PrefixSpan for MOB
Source: (SILVA; SICHMAN, 2022)

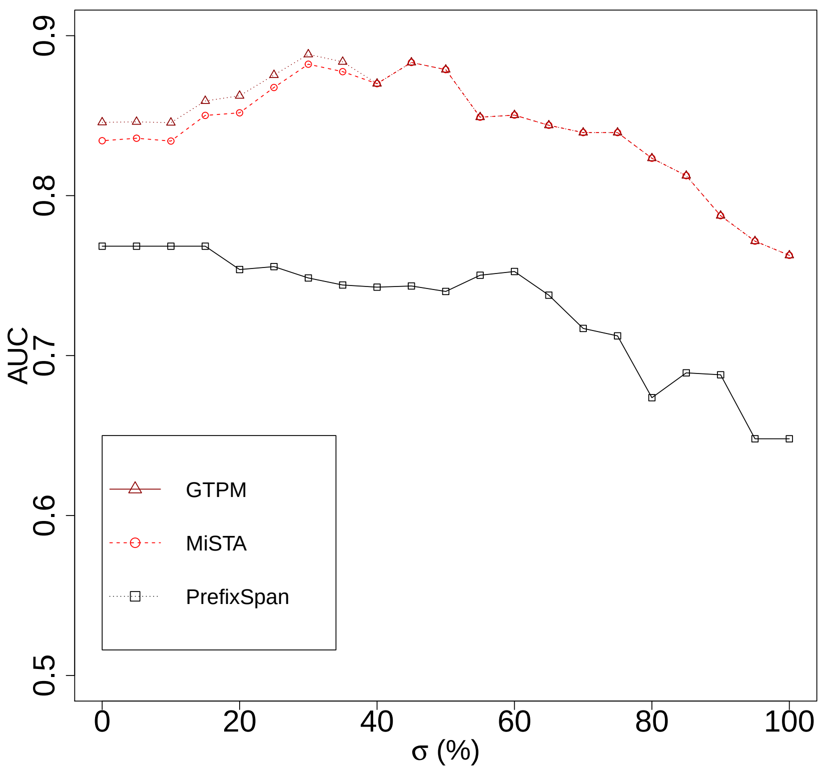


Figure 24: Comparison vs. MiSTA and PrefixSpan for HSP
Source: (SILVA; SICHMAN, 2022)

By analyzing the performance of our GTPM Algorithm in comparison with the existing algorithms, PrefixSpan and MiSTA, for MOB scenario, shown in Figure 23, we notice that the performance curve for GTPM and MiSTA seems linear and decreasing because if we increase the restrictions to consider the patterns contained in the trajectories (in this case, more restrictive $\sigma$), a pattern needs to appear more often within the trajectories, with the same social group and with the same time intervals. Therefore, it is natural that there are fewer patterns to compare, and that new trajectories will be incorrectly classified. PrefixSpan is more stable as it has fewer restrictions (only checkpoint sequences). On the other hand, the HSP scenario has more checkpoints than MOB, so checkpoint sequences are more relevant than MOB scenario and AUCs are higher for all the three models, considering the same $\sigma$ (Figure 15. Also and as a consequence of this largest number of checkpoints, the transition times for HSP are shorter, and hence more regular than in MOB scenario, so that both MiSTA and GTPM, that incorporate transition times to the models, always perform better than PrefixSpan. Other observation is that in the Hospital environment, employees usually work in different and changing shifts, have less social bonds, and this is a possible explanation why incorporating groups does not improve so much the results when comparing to MiSTA.

# 7 CONCLUSIONS AND FURTHER WORK

Over the last half-century, data mining techniques have been developed to find patterns, connections, correlations, or anomalies in large amounts of data, allowing you to find problems, hypotheses and opportunities more quickly.

Especially when modeling human behaviors that fit in widely available data, such as trajectory data, research in this field can help to solve real-life problems. In the present work we, for the first time, included the notion of traveling companions to users' trajectory profiles aiming to detect anomalies. The central concept is that social groups are a human attribute and can be noticed in users' trajectories to form a new feature that comes from relations of all trajectory observations and not only features from isolated data instances.

Although the focus of this work has been detecting anomalies that could indicate potential identity fraud in physical access control systems, the concept of social group information in trajectory profiles can be easily extended to other real problems. For example, during the year 2020, the COVID-19 pandemic rage killing millions of people, and the best way to prevent the virus dissemination was to isolate infected people, and the companions they had contact in the period of the disease. Knowing these companions could help to control isolation more effectively.

Another possible extension of our model can be monitoring people's movements in large cities and helping authorities to improve transport systems. If they know which people go to the same place at the same approximate time, they can suggest users take the same cab, reducing traffic and pollution. Many other applications can take advantage of the concepts developed, such as fraud in mobile networks, detecting anomalies in vessels to prevent theft and piracy, and profiling tourist patterns, to name a few.

On the other hand, the addition of social group information leverage the potential of the technology developed. Our experiments with real-world datasets from a smart commercial multi-office building and a hospital showed that our group similarity measure enhanced the detector's performance.

After performing all two experiments, the research questions presented in this dissertation introduction were answered as follows:

RQ1 Is it possible to enhance impersonation fraud detection in PACSs by incorporating social group information?

In both scenarios, the performance was increased with the addition of group information and the maximum score of the detector was achieved only when using GTPM. We conclude that in our tested scenarios, adding social groups information to mobility profiles enhances anomaly-based impersonation attack detection.

RQ2 How the definition of *frequent trajectory* of the model affects its outputs?

In Section 6.4, we conducted a sensitivity analysis of the parameters of the model. In particular, the concepts of maximum tolerance time threshold $\tau$ and minimum support threshold $\sigma$ were extensively stressed and have led us to came up with some conclusions. The higher the $\tau$ value, the fewer are the restrictions for considering the trajectory as frequent and the better the results. If we increase $\tau$ even further, a maximum threshold is reached so that the results start to get worse, and the anomaly detector starts to classify incorrectly the trajectories. Inversely, the higher the $\sigma$ value, the more are the restrictions, and in both studied scenarios, there is an optimum $\sigma$ such that the score AUC is maximum.

In this work, we have considered a system for preventing a certain type of crime that relies on users' personal data, an approach with some ethical issues that have to be addressed. One meaningful discussion is the use of the data for purposes other than the ones specified. Although experiments were conducted with anonymized data, in a real-life system, important information about users' behavior may be exposed, to cite a few: the typical time the user is out of his workstation, frequent companions, and work absences. The misuse of such information may lead to biased decisions about the user specifically or even compromise the whole organization.

Another dangerous premise is that an anomalous trajectory is a potential impersonation fraud, which may lead to further, maybe intrusive investigation if a crime is under development. Anomalies can happen anywhere at any time, and they are not necessarily related to someone doing something wrong. The presumption of guilt is against the major criminal justice system assumptions.

In the future, we will extend our feature analysis by considering the following issues:

- Implement a preprocessing step to handle missing data in Profiles extraction. Usually, users take advantage of an open door and enter a room without presenting their cards, thus creating missing data. Pattern information may help to fill out these missing data;
- We foresee the use of optimization methods such as evolutionary algorithms and neural networks, to find the best adjustment and combinations of the proposed similarity measures (checkpoints, transition times, and groups);
- We want to associate with every user a different maximum tolerance time threshold $\tau$ and a minimum support threshold $\sigma$ that strikes a right balance between the FAR and FRR values, similar to the approach proposed in (YAZJI et al., 2014).
- We plan to explore other group definitions different than Swarm.

## REFERENCES

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES VLDB, 20., 1994, Santiago. **Proceedings [...]**. Santiago: Morgan Kaufmann Publishers, 1994. p. 487–499.

ASHBROOK, D.; STARNER, T. Using gps to learn significant locations and predict movement across multiple users. **Personal and Ubiquitous computing**, Springer-Verlag, v. 7, n. 5, p. 275–286, 2003.

BADDAR, S. W. A.-H.; MERLO, A.; MIGLIARDI, M. Anomaly detection in computer networks: a state-of-the-art review. **Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications**, v. 5, n. 4, p. 29–64, 2014.

BARBOSA, H. et al. Human mobility: Models and applications. **Physics Reports**, Elsevier, v. 734, p. 1–74, 2018.

BENKERT, M. et al. Reporting flock patterns. **Computational Geometry**, Elsevier, v. 41, n. 3, p. 111–125, 2008.

BRISLAWN, C. M. Fingerprints go digital. **Notices of the AMS**, v. 42, n. 11, p. 1278–1283, 1995.

CAI, G.; LEE, K.; LEE, I. Itinerary recommender system with semantic trajectory pattern mining from geo-tagged photos. **Expert Systems with Applications**, v. 94, p. 32–40, 2018.

CHEN, X. et al. Minus: Mining user similarity with trajectory patterns. In: JOINT EUROPEAN CONFERENCE ON MACHINE LEARNING AND KNOWLEDGE DISCOVERY IN DATABASES, 1994, Santiago. **Proceedings [...]**. Nancy: Springer, 2014. p. 436–439.

CHEN, X. et al. Measuring user similarity with trajectory patterns: Principles and new metrics. In: ASIA-PACIFIC WEB CONFERENCE, 16., 2014, Changsha. **Proceedings [...]**. Changsha: Springer, 2014. p. 437–448.

CHEN, X.; PANG, J.; XUE, R. Constructing and comparing user mobility profiles. **ACM Transactions on the Web**, ACM, v. 8, n. 4, p. 21, 2014.

CIHOLAS, P. et al. The security of smart buildings: a systematic literature review. **arXiv preprint**, arXiv:1901.05837, 2019. Ahead of print.

CRANDALL, D. J. et al. Inferring social ties from geographic coincidences. **Proceedings of the National Academy of Sciences**, United States National Academy of Sciences, v. 107, n. 52, p. 22436–22441, 2010.

DU, B. et al. Detecting pickpocket suspects from large-scale public transit records. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 31, n. 3, p. 465–478, 2018.

EAGLE, N.; PENTLAND, A. S.; LAZER, D. Inferring friendship network structure by using mobile phone data. **Proceedings of the National Academy of Sciences**, United States National Academy of Sciences, v. 106, n. 36, p. 15274–15278, 2009.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING KDD, 2., 1996, Portland. **Proceedings [...]**. Portland: AAAI Press, 1996. p. 226–231.

FOROUGH, J.; MOMTAZI, S. Ensemble of deep sequential models for credit card fraud detection. **Applied Soft Computing**, Elsevier, v. 99, p. 106883, February 2021.

GARRI, K. et al. Anomaly detection in rfid systems. **International Journal of Radio Frequency Identification Technology and Applications**, Inderscience Publishers Ltd., v. 3, n. 1-2, p. 31–46, 2011.

GEEPALLA, E.; ASHARIF, S. Analysis of physical access control system for understanding users behavior and anomaly detection using neo4j. In: INTERNATIONAL CONFERENCE ON ENGINEERING & MIS, 6., 2020, Almaty Kazakhstan. **Proceedings [...]**. Almaty Kazakhstan: ACM Press, 2020. p. 1–6.

GIANNOTTI, F.; NANNI, M.; PEDRESCHI, D. Efficient mining of temporally annotated sequences. In: SIAM INTERNATIONAL CONFERENCE ON DATA MINING, 6., 2006, Bethesda. **Proceedings [...]**. Bethesda: SIAM, 2006. p. 348–359.

GIANNOTTI, F. et al. Trajectory pattern mining. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING ACM SIGKDD, 13., 2007, San Jose. **Proceedings [...]**. San Jose: ACM Press, 2007. p. 330–339.

GONZALEZ, M. C.; HIDALGO, C. A.; BARABASI, A.-L. Understanding individual human mobility patterns. **Nature**, Nature publishing group, v. 453, n. 7196, p. 779, 2008.

GU, H. et al. Detecting pickpocketing offenders by analyzing beijing metro subway data. In: INTERNATIONAL CONFERENCE ON BIG DATA ANALYTICS, 4., 2019, Suzhou. **Proceedings [...]**. Suzhou: IEEE, 2019. p. 62–66.

HAN, J. et al. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 17., 2001, Heidelberg. **Proceedings [...]**. Heidelberg: IEEE, 2001. p. 215–224.

HARRELL, E.; LANGTON, L. **Victims of identity theft, 2018**. [S.l.]: US Department of Justice, Office of Justice Programs, Bureau of Justice Statistics, 2018.

HUCZYNSKI, A.; BUCHANAN, D. A.; HUCZYNSKI, A. A. **Organizational behaviour**. London: Pearson, 2013.

JAMES, G. et al. **An introduction to statistical learning**. New York: Springer, 2013.

JEMISIN, N. K. **The obelisk gate**. [S.l.]: Orbit, 2016.

JEUNG, H.; SHEN, H. T.; ZHOU, X. Convoy queries in spatio-temporal databases. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 24., 2008, Cancun. **Proceedings [...]**. Cancun: IEEE, 2008. p. 1457–1459.

JEUNG, H. et al. Discovery of convoys in trajectory databases. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 1, n. 1, p. 1068–1080, 2008.

KALNIS, P.; MAMOULIS, N.; BAKIRAS, S. On discovering moving clusters in spatio-temporal data. In: INTERNATIONAL SYMPOSIUM ON SPATIAL AND TEMPORAL DATABASES, 20., 2005, Berlin, Heidelberg. **Proceedings [...]**. Berlin, Heidelberg: Springer, 2005. p. 364–381.

KANG, J.; LIU, M.; QU, W. Using gameplay data to examine learning behavior patterns in a serious game. **Computers in Human Behavior**, Elsevier, v. 72, p. 757–770, 2017.

KLEISSL, J.; AGARWAL, Y. Cyber-physical energy systems: Focus on smart buildings. In: DESIGN AUTOMATION CONFERENCE, 47., 2010, Anaheim. **Proceedings [...]**. Anaheim: IEEE, 2010. p. 749–754.

KONTARINIS, A. et al. Towards a semantic indoor trajectory model. In: WORKSHOPS OF THE EDBT/ICDT JOINT CONFERENCE, 2019, Lisbon. **Proceedings [...]**. Lisbon: Springer, 2019.

LAUBE, P.; IMFELD, S. Analyzing relative motion within groups of trackable moving point objects. In: INTERNATIONAL CONFERENCE ON GEOGRAPHIC INFORMATION SCIENCE, 2002, Berlin, Heidelberg. **Proceedings [...]**. Berlin, Heidelberg: Springer, 2002. p. 132–144.

LAUBE, P.; KREVELD, M. van; IMFELD, S. Finding remo — detecting relative motion patterns in geospatial lifelines. In: FISHER, P. F. (Org.). **Developments in spatial data handling**: 11th international symposium on spatial data handling. Berlin: Springer, 2005. p. 201–215.

LI, Z. et al. Swarm: Mining relaxed temporal moving object clusters. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 3, n. 1-2, p. 723–734, 2010.

LIN, Z. et al. A semantic user distance metric using gps trajectory data. **IEEE Access**, IEEE, v. 7, p. 30185–30196, 2019.

LIU, C. et al. A stochastic model for context-aware anomaly detection in indoor location traces. In: INTERNATIONAL CONFERENCE ON DATA MINING, 12., 2012, Brussels. **Proceedings [...]**. Brussels: IEEE, 2012. p. 449–458.

MAZIMPAKA, J. D.; TIMPF, S. Trajectory data mining: A review of methods and applications. **Journal of Spatial Information Science**, v. 2016, n. 13, p. 61–99, 2016.

MAZUMDAR, P. et al. An approach to compute user similarity for gps applications. **Knowledge-Based Systems**, Elsevier, v. 113, p. 125–142, 2016.

MIROWSKI, L.; HARTNETT, J. Deckard: A system to detect change of rfid tag ownership. **International Journal of Computer Science and Network Security**, v. 7, n. 7, p. 89–98, 2007.

NASERIAN, E. et al. A framework of loose travelling companion discovery from human trajectories. **IEEE Transactions on Mobile Computing**, IEEE, v. 17, n. 11, p. 2497–2511, 2018.

NJOO, G. S. et al. Exploring check-in data to infer social ties in location based social networks. In: PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2017, Jeju. **Proceedings [...]**. Jeju: Springer, 2017. p. 460–471.

SETIAWAN, F.; YAHYA, B. N. Improved behavior model based on sequential rule mining. **Applied Soft Computing**, Elsevier, v. 68, p. 944–960, July 2018.

SILVA, G. M. de C.; SICHMAN, J. S. Identity theft detection using trajectory patterns. In: WORKSHOP DE PÓS-GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO, 7., 2018, São Paulo. **Anais [...]**. São Paulo: PCS/POLI/USP, 2018. p. 69–72.

SILVA, G. M. de C.; SICHMAN, J. S. Using social group trajectories for potential impersonation detection on smart buildings access control. In: BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS, 8., 2019, Salvador. **Proceedings [...]**. Salvador: IEEE, 2019. p. 389–394.

SILVA, G. M. de C.; SICHMAN, J. S. Impersonation fraud detection on building access control systems: An approach based on anomalous social and spatio-temporal behaviors. **Applied Soft Computing**, v. 118, p. 108310, 2022.

TANG, L.-A. et al. On discovery of traveling companions from streaming trajectories. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 28., 2012, Arlington. **Proceedings [...]**. Arlington: IEEE, 2012. p. 186–197.

TANG, L.-A. et al. A framework of traveling companion discovery on trajectory data streams. **ACM Transactions on Intelligent Systems and Technology**, ACM, v. 5, n. 1, p. 3, 2013.

TRASARTI, R. et al. Myway: Location prediction via mobility profiling. **Information Systems**, Elsevier, v. 64, p. 350–367, 2017.

TRIPATHI, K. K.; PAVASKAR, M. A. Survey on credit card fraud detection methods. **International Journal of Emerging Technology and Advanced Engineering**, v. 2, n. 11, p. 721–726, 2012.

WANG, H. All common subsequences. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICAL INTELLIGENCE, 20., 2007, Hyderabad. **Proceedings [...]**. Hyderabad: Morgan Kaufmann Publishers Inc., 2007. p. 635–640.

WANG, Y.; LIM, E.-P.; HWANG, S.-Y. Efficient mining of group patterns from user movement data. **Data & Knowledge Engineering**, Elsevier, v. 57, n. 3, p. 240–282, 2006.

WANG, Y. et al. Discovering loose group movement patterns from animal trajectories. In: INTERNATIONAL CONFERENCE ON E-SCIENCES, 11., 2015, Munich. **Proceedings [...]**. Munich: IEEE, 2015. p. 196–206.

XU, C. et al. Context co-occurrence based relationship prediction in spatiotemporal data. In: INTERNATIONAL CONFERENCE ON COMPUTER MODELING, SIMULATION AND ALGORITHM, 2018, Beijing. **Proceedings [...]**. Beijing: Atlantis Press, 2018. p. 281–287.

YAZJI, S. et al. Efficient location aware intrusion detection to protect mobile devices. **Personal and Ubiquitous Computing**, Springer-Verlag, v. 18, n. 1, p. 143–162, 2014.

YIN, S.-N. et al. Intrusion detection system based on complex event processing in rfid middleware. In: INTERNATIONAL CONFERENCE ON RESEARCH IN ADAPTIVE AND CONVERGENT SYSTEMS, 2016, Odense. **Proceedings [...]**. New York: ACM, 2016. p. 125–129.

YING, J. J.-C. et al. Mining user similarity from semantic trajectories. In: INTERNATIONAL WORKSHOP ON LOCATION BASED SOCIAL NETWORKS, 2., 2010, Seattle. **Proceedings [...]**. New York: ACM, 2010. p. 19–26.

YOUDEN, W. J. Index for rating diagnostic tests. **Cancer**, Wiley Online Library, v. 3, n. 1, p. 32–35, 1950.

ZAKI; J, M. Spade: An efficient algorithm for mining frequent sequences. **Machine learning**, Springer, v. 42, n. 1-2, p. 31–60, 2001.

ZAKI, M. J. Sequence mining in categorical domains: incorporating constraints. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 9., 2000, McLean. **Proceedings [...]**. New York: ACM, 2000. p. 422–429.

ZHENG, K. et al. Online discovery of gathering patterns over trajectories. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 26, n. 8, p. 1974–1988, 2014.

ZHU, X. et al. Exploring group movement pattern through cellular data: A case study of tourists in hainan. **ISPRS International Journal of Geo-Information**, Multidisciplinary Digital Publishing Institute, v. 8, n. 2, p. 74, 2019.